
MC1322x

Advanced ZigBee™ - Compliant SoC Platform
for the 2.4 GHz IEEE® 802.15.4 Standard
Reference Manual

Document Number: MC1322xRM
Rev. 0.0
06/2008



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 or registered trademarks of Freescale Semiconductor, Inc. in the U.S. and other countries. All other product or service names are the property of their respective owners. ARM is the registered trademark of ARM Limited. ARM7TDMI-S is the trademark of ARM Limited. © Freescale Semiconductor, Inc. 2007, 2008

Contents

About This Book

Audience	xvii
Organization	xvii
Revision History	xviii
References	xix

Chapter 1 MC1322x Introduction

1.1	Features	1-2
1.1.1	Block Diagram	1-2
1.1.2	Features Summary	1-3
1.2	High Density, Low Component Count, Integrated IEEE 802.15.4 Solution	1-4
1.3	Integrated IEEE 802.15.4 Transceiver (Radio and Modem)	1-5
1.3.1	RF Interface and Usage.	1-5
1.3.2	Modem	1-5
1.4	High Performance, Low Power 32-Bit ARM7 Processor.	1-6
1.5	Low Power Operation and Power Management	1-7
1.5.1	Operating Current	1-7
1.5.2	Power Management	1-8
1.5.3	Optional Buck Regulator	1-9
1.6	Battery Detect	1-10
1.7	IEEE 802.15.4 Acceleration Hardware.	1-10
1.7.1	802.15.4 MAC Accelerator (MACA) Overview	1-10
1.8	Advanced Security Module (ASM)	1-12
1.9	Memory	1-12
1.9.1	RAM and ROM	1-12
1.9.2	Serial FLASH (NVM)	1-13
1.10	MCU Peripherals	1-14
1.11	Parallel IO (GPIO)	1-14
1.12	Keyboard Interface (KBI) Interrupts	1-15
1.13	Timer (TMR) Module.	1-15
1.14	UART Modules.	1-17
1.15	Inter-Integrated Circuit (I2C) Module	1-18
1.16	Serial Peripheral Interface (SPI) Modules	1-18
1.16.1	External SPI Module.	1-18
1.16.2	SPI FLASH Module (SPIF)	1-19
1.17	Synchronous Serial Interface (SSI) Module	1-19
1.18	Analog-to-Digital Converter (ADC) Module	1-20

Chapter 2 Pins and Connections

2.1	Pin Assignments and Connections	2-1
2.2	Pin Definitions	2-2

2.3	Hardware Development Interface Interconnects	2-7
2.3.1	ARM JTAG Interface Connector	2-7
2.3.2	Nexus Mictor Interface Connector	2-7

Chapter 3 MC1322x System Considerations

3.1	Introduction.	3-1
3.2	Power Connections and Design	3-1
3.2.1	Power Pin Descriptions.	3-1
3.2.2	Varying VDD from 2.0 - 3.6 VDC Using Onboard Regulation.	3-3
3.2.3	Varying VDD from 2.1 - 3.6 VDC with Optional Onboard Buck Regulator.	3-4
3.2.4	Fixed Regulated 1.8 VDC VDD Voltage	3-8
3.3	System Reset and Low Power GPIO Signal Connections	3-8
3.3.1	GPIO Pin State During Reset, Default, and Low Power Modes	3-8
3.3.2	Using GPIO in Low Power Modes.	3-10
3.4	External Clock Connections	3-11
3.5	Reference Oscillator	3-12
3.5.1	Description	3-12
3.5.2	Reference Oscillator 24 MHz Crystal Specifications	3-14
3.5.3	Crystal Trimming	3-14
3.5.4	Using a Reference Frequency Other Than 24 MHz	3-16
3.5.5	Application Requirements for Reference Oscillator Performance	3-16
3.5.6	Using an External Reference Source	3-16
3.6	32.768 kHz Crystal Oscillator (use optional)	3-16
3.6.1	Description	3-17
3.6.2	32.768 kHz Crystal Specifications	3-18
3.7	Transceiver RF Operation.	3-18
3.7.1	Standard Single-Ended RF Connection	3-18
3.7.2	Dual-Port RF Operation	3-19
3.8	Low Power Considerations.	3-21
3.8.1	Controlling Low Power Current	3-21
3.8.2	Wakeup or Recovery from Low Power Modes	3-24
3.8.3	Controlling Run-time Operating Current	3-26
3.9	Bootloader.	3-27
3.9.1	Overview.	3-27
3.9.2	Exception Vectors.	3-28
3.9.3	Bootstrap Flow	3-28
3.9.4	Booting from FLASH.	3-32
3.9.5	FLASH Erase or Recovery Mode.	3-32
3.9.6	Secure Mode	3-32
3.9.7	GPIO Function Upon Boot Exit	3-32
3.9.8	Bootstrap version.	3-33

Chapter 4

Memory

4.1	Introduction	4-1
4.2	Features	4-1
4.3	Memory Map	4-2
4.4	Exception Vectors	4-3
4.5	ROM and RAM	4-4
4.6	Serial FLASH (NVM)	4-5
4.6.1	FLASH Access	4-5
4.6.2	FLASH Security	4-5
4.6.3	FLASH Recovery (Erase)	4-6
4.7	Peripheral Register Access	4-6

Chapter 5 System Management (Including CRM)

5.1	Management Overview	5-1
5.1.1	Management Features	5-1
5.1.2	System Reset	5-2
5.1.3	System Clocks	5-2
5.1.4	System Power Distribution	5-4
5.1.5	KBI Interface Signals	5-5
5.2	Clock/Reset/Power Module (CRM) Overview	5-5
5.2.1	CRM Block Diagram	5-6
5.2.2	Signal Descriptions	5-6
5.2.3	Sleep Module	5-7
5.3	Managing Low Power Mode	5-18
5.3.1	Entering Low Power Mode	5-18
5.3.2	Recovery from Low Power Mode	5-18
5.3.3	Auto ADC Mode	5-18
5.4	Managing Reference Oscillator	5-20
5.5	Bus Steal Function	5-21
5.6	Computer Operating Properly (COP) or Watchdog Timer Module	5-22
5.7	Interrupts	5-23
5.7.1	Wake-up Timer Time-out	5-24
5.7.2	Real Time Clock Time-out	5-24
5.7.3	External wake-up Signals KBI_7:KBI_4	5-24
5.7.4	Ring Oscillator Calibration Done	5-24
5.7.5	COP Time-out	5-24
5.8	CRM Register Memory Map	5-25
5.9	CRM Registers and Control Bits	5-26
5.9.1	System Control (SYS_CNTL)	5-26
5.9.2	Wake-up Control (WU_CNTL)	5-28
5.9.3	Sleep Control (SLEEP_CNTL)	5-32
5.9.4	Bus Stealing Control (BS_CNTL)	5-34
5.9.5	COP Control (COP_CNTL)	5-36

5.9.6	COP Service (COP_SERVICE)	5-38
5.9.7	Status (STATUS)	5-39
5.9.8	Module Enable Status (MOD_STATUS)	5-43
5.9.9	Wake-up Count (WU_COUNT)	5-45
5.9.10	Wake-up Time-out (WU_TIMEOUT)	5-46
5.9.11	RTC Count (RTC_COUNT)	5-47
5.9.12	RTC Time-out (RTC_TIMEOUT)	5-48
5.9.13	Calibration Control Register (CAL_CNTL)	5-49
5.9.14	Calibration XTAL Count (CAL_COUNT)	5-50
5.9.15	Ring Oscillator Control (RINGOSC_CNTL)	5-51
5.9.16	Reference XTAL Control (XTAL_CNTL)	5-52
5.9.17	32kHz XTAL Control (XTAL32_CNTL)	5-53
5.9.18	Voltage Regulator Control (VREG_CNTL)	5-54
5.9.19	Software Reset (SW_RST)	5-58

Chapter 6 MC1322x 2.4 GHz ISM Band Transceiver

6.1	Introduction	6-1
6.2	Transceiver Overview	6-1
6.2.1	IEEE 802.15.4 Packet Structure	6-2
6.2.2	Transmit Path	6-2
6.2.3	Receive Path	6-2
6.2.4	Independent Transceiver Operation	6-3
6.2.5	Modem Overview	6-3
6.3	Transceiver Register Memory Map Base Addresses	6-3
6.4	Modem Synthesizer	6-4
6.4.1	Synthesizer Description	6-4
6.5	Modem Synthesizer Register Map	6-5
6.5.1	Enable and Override Register	6-6
6.5.2	Reference Loop Divider Register	6-7
6.5.3	VCO Loop Divider Register	6-8
6.5.4	VCO Lock Register	6-9
6.5.5	Frequency Programming Example	6-9
6.6	Modem Transceiver Sequence Manager (TSM)	6-10
6.6.1	TSM Register Map	6-12
6.7	Modem Radio Receiver Module	6-12
6.7.1	SFD Control Register	6-13
6.7.2	AGC CCA and ED Control Register	6-14
6.7.3	AGC RSSI Parameters Register	6-15
6.8	Modem Radio Transmitter Module	6-15
6.9	Modem Radio Frequency Synthesizer	6-16
6.9.1	RF Synthesizer Register Map	6-17
6.9.2	VCO Loop Divider (Integer) Register	6-18
6.9.3	VCO Loop Divider (Fractional) Register	6-19

6.9.4	VCO Lock Register	6-20
6.9.5	Frequency Programming Example	6-20
6.10	Modem Tracking Oscillator Module (TOC).....	6-22
6.11	Radio Analog Function Control	6-22
6.11.1	Transmit Power Amp Control.....	6-23
6.11.2	Transmit Power Adjust.....	6-24

Chapter 7 Central Processing Unit (CPU)

7.1	ARM7TDMI-S Processor Overview	7-1
7.1.1	Memory Access	7-2
7.1.2	Memory Interface	7-2
7.1.3	Instruction Pipeline.....	7-2
7.2	ARM7TDMI-S Architecture.....	7-3
7.2.1	Instruction Compression.....	7-3
7.2.2	The Thumb Instruction Set.....	7-3
7.3	About the Programmer's Model.....	7-4
7.3.1	Switching State.....	7-4
7.3.2	Memory Formats.....	7-4
7.4	Instruction Length.....	7-5
7.5	Data Types	7-5
7.6	Operating Modes.....	7-5
7.7	Registers	7-6
7.7.1	The ARM State Register Set.....	7-6
7.7.2	The Thumb State Register Set	7-8
7.7.3	The Relationship Between ARM State and Thumb State Registers.....	7-8
7.7.4	Accessing High Registers in Thumb State	7-9
7.8	The Program Status Registers.....	7-10
7.8.1	The Condition Code Flags	7-10
7.8.2	The Control Bits	7-10
7.8.3	Interrupt Disable Bits	7-11
7.8.4	T Bit	7-11
7.8.5	Mode Bits	7-11
7.8.6	Reserved Bits	7-11
7.9	Exceptions.....	7-12
7.9.1	Exception Entry/Exit Summary	7-12
7.9.2	Entering an Exception.....	7-12
7.9.3	Leaving an Exception	7-13
7.9.4	Fast Interrupt Request (FIQ).....	7-13
7.9.5	Interrupt Request (IRQ)	7-14
7.9.6	Abort.....	7-14
7.9.7	Software Interrupt Instruction.....	7-15
7.9.8	Undefined Instruction	7-15
7.9.9	Exception Vectors.....	7-16

7.9.10	Exception Priorities	7-16
7.10	Interrupt Latencies	7-17
7.10.1	Maximum Interrupt Latencies	7-17
7.10.2	Minimum Interrupt Latencies	7-17
7.11	Reset	7-17

Chapter 8 Interrupt Controller (ITC)

8.1	MC1322x MCU Interrupt Operation Overview	8-1
8.1.1	CPU Request Service	8-2
8.1.2	Interrupt Request Generation	8-3
8.2	Interrupt Controller Overview	8-4
8.3	ITC Features	8-4
8.4	Interrupt Controller Operation	8-5
8.4.1	ITC Prioritization of Interrupt Sources	8-6
8.4.2	Assigning and Enabling Interrupt Sources	8-7
8.5	Interrupt Controller Memory Map	8-8
8.6	ITC Registers	8-9
8.6.1	Interrupt Control Register (INTCNTL)	8-9
8.6.2	Normal Interrupt Mask Register (NIMASK)	8-10
8.6.3	Interrupt Enable Number Register (INTENNUM)	8-11
8.6.4	Interrupt Disable Number Register (INTDISNUM)	8-11
8.6.5	Interrupt Enable Register (INTENABLE)	8-13
8.6.6	Interrupt Type Register (INTTYPE)	8-14
8.6.7	Normal Interrupt Vector (NIVECTOR)	8-15
8.6.8	Fast Interrupt Vector (FIVECTOR)	8-16
8.6.9	Interrupt Source Register (INTSRC)	8-17
8.6.10	Interrupt Force Register (INTFRC)	8-18
8.6.11	Normal Interrupt Pending Register (NIPEND)	8-19
8.6.12	Fast Interrupt Pending Register (FIPEND)	8-20

Chapter 9 MAC Accelerator (MACA)

9.1	Overview	9-1
9.2	Features	9-1
9.3	Primary Functionality	9-2
9.4	Block Diagram	9-3
9.5	Module descriptions	9-4
9.5.1	Sequencer	9-4
9.5.2	Dedicated Direct Memory Access (DDMA)	9-17
9.5.3	Random Generator	9-17
9.5.4	Radio Modem Control	9-18
9.5.5	Beacon Mode Support	9-18
9.5.6	FIFO Buffer Interrupts	9-19

9.6	MACA Register Memory Map	9-19
9.7	MACA Register Description	9-22
9.7.1	MACA_VERSION	9-22
9.7.2	MACA_RESET	9-23
9.7.3	MACA_RANDOM	9-24
9.7.4	MACA_CONTROL	9-25
9.7.5	MACA_STATUS	9-27
9.7.6	MACA_FRMPND	9-29
9.7.7	MACA_FREQ	9-30
9.7.8	MACA_EDVALUE	9-30
9.7.9	MACA_TMREN	9-31
9.7.10	MACA_TMRDIS	9-32
9.7.11	MACA_CLK	9-33
9.7.12	MACA_STARTCLK	9-33
9.7.13	MACA_CPLCLK	9-34
9.7.14	MACA_SFTCLK	9-35
9.7.15	MACA_CLKOFFSET	9-36
9.7.16	MACA_RELCLK	9-36
9.7.17	MACA_CPLTIM	9-37
9.7.18	MACA_SLOTOFFSET	9-38
9.7.19	MACA_TIMESTAMP	9-39
9.7.20	MACA_DMARX	9-40
9.7.21	MACA_DMATX	9-41
9.7.22	MACA_DMAPOLL	9-41
9.7.23	MACA_TXLEN	9-42
9.7.24	MACA_TXSEQNR	9-43
9.7.25	MACA_SETRXLVL	9-44
9.7.26	MACA_GETRXLVL	9-45
9.7.27	MACA_IRQ	9-46
9.7.28	MACA_CLRIRQ	9-47
9.7.29	MACA_SETIRQ	9-48
9.7.30	MACA_MASKIRQ	9-50
9.7.31	MACA_MACPANID	9-51
9.7.32	MACA_MAC16ADDR	9-51
9.7.33	MACA_MAC64HI	9-52
9.7.34	MACA_MAC64LO	9-52
9.7.35	MACA_FLTREJ	9-53
9.7.36	MACA_CLKDIV	9-54
9.7.37	MACA_WARMUP	9-54
9.7.38	MACA_PREAMBLE	9-55
9.7.39	MACA_FRAMESYNC0	9-55
9.7.40	MACA_FRAMESYNC1	9-56
9.7.41	MACA_TXACKDELAY	9-57
9.7.42	MACA_RXACKDELAY	9-58
9.7.43	MACA_EOFDELAY	9-59

9.7.44	MACA_CCADELAY	9-60
9.7.45	MACA_RXEND	9-61
9.7.46	MACA_TXCCADELAY	9-62
9.7.47	MACA_KEY3	9-62
9.7.48	MACA_KEY2	9-63
9.7.49	MACA_KEY1	9-63
9.7.50	MACA_KEY0	9-64
9.7.51	MACA_OPTIONS	9-65

Chapter 10 Advanced Security Module (ASM)

10.1	Overview	10-1
10.1.1	Features	10-1
10.2	ASM Module Block Diagram	10-1
10.3	Mode of Operation	10-2
10.3.1	CTR mode Block Diagram	10-3
10.3.2	CBC mode block diagram	10-3
10.3.3	CCM mode	10-4
10.3.4	Boot mode	10-4
10.3.5	Bypass mode	10-4
10.4	ASM module Block Diagram	10-5
10.4.1	AES Encryption Engine Algorithm	10-6
10.5	Counter mode encryption	10-7
10.6	Message Authentication Code generation (MAC)	10-7
10.7	Counter mode and Authentication mode encryption combined	10-8
10.8	Signal Description	10-8
10.9	ASM Registers	10-9
10.9.1	key0, key1, key2, key3 Registers	10-10
10.9.2	data0,data1,data2,data3 Registers	10-10
10.9.3	ctr0, ctr1, ctr2, ctr3 Registers	10-11
10.9.4	ctr_result0, ctr_result1, ctr_result2, ctr_result3 Registers	10-11
10.9.5	cbc_result0, cbc_result1, cbc_result2, cbc_result3 Registers	10-12
10.9.6	Control 0 Register	10-12
10.9.7	Control 1 Register	10-13
10.9.8	Status Register	10-14
10.9.9	mac0, mac1, mac2, mac3 Registers	10-15

Chapter 11 General Purpose I/O Module

11.1	Introduction	11-1
11.2	Reset and Low Power GPIO Operation	11-1
11.3	Features	11-2
11.4	Pin Descriptions	11-2
11.4.1	Pin Functionality	11-2

11.4.2	Pin Register Mappings	11-3
11.5	Functional Description	11-5
11.5.1	GPIO Function Select	11-6
11.5.2	Pull-up/Pull-down Resistors	11-8
11.5.3	Input Hysteresis	11-8
11.5.4	Pad Keeper Function	11-8
11.5.5	Port I/O Control	11-9
11.5.6	Special Signals and Conditions	11-10
11.6	GPIO Module Register Memory Map	11-11
11.7	GPIO Module Registers and Control Bits	11-12
11.7.1	GPIO Pad Direction Registers (GPIO_PAD_DIR0 and GPIO_PAD_DIR1)	11-12
11.7.2	GPIO Data Registers (GPIO_DATA1 and GPIO_DATA0)	11-13
11.7.3	GPIO Pull-up Enable Registers (GPIO_PAD_PU_EN1 and GPIO_PAD_PU_EN0) ...	11-14
11.7.4	GPIO Function Select Registers (GPIO_FUNC_SEL3, GPIO_FUNC_SEL2, GPIO_FUNC_SEL1, and GPIO_FUNC_SEL0)11-15	
11.7.5	GPIO Data Select Registers (GPIO_DATA_SEL1 and GPIO_DATA_SEL0)	11-19
11.7.6	GPIO Pad Pull-up Select (GPIO_PAD_PU_SEL1 and GPIO_PAD_PU_SEL0)	11-20
11.7.7	GPIO Pad Hysteresis Enable Registers (GPIO_PAD_HYST_EN1 and GPIO_PAD_HYST_EN0)11-21	
11.7.8	GPIO Pad Keeper Enable Registers (GPIO_PAD_KEEP1 and GPIO_PAD_KEEP0) ..	11-22
11.7.9	GPIO Data Set Registers (GPIO_DATA_SET1 and GPIO_DATA_SET0)	11-23
11.7.10	GPIO Data Reset Registers (GPIO_DATA_RESET1 and GPIO_DATA_RESET0) ...	11-24
11.7.11	GPIO Pad Direction Set Registers (GPIO_PAD_DIR_SET1 and GPIO_PAD_DIR_SET0) ..	11-26
11.7.12	GPIO Pad Direction Reset Registers (GPIO_PAD_DIR_RESET1 and GPIO_PAD_DIR_RESET0)11-27	

Chapter 12 Timer Module (TMR)

12.1	Overview	12-1
12.2	Features	12-1
12.3	Customization	12-2
12.4	Modes of Operation	12-2
12.5	Block Diagram	12-2
12.6	Signal Descriptions	12-2
12.7	Overview	12-2
12.8	External Signal Descriptions	12-4
12.8.1	TIO3-TIO0 - Timer Input/Outputs	12-4
12.9	Memory Map and Registers	12-4
12.10	Overview	12-4
12.11	Module Memory Map	12-4
12.12	Register Descriptions	12-5
12.12.1	TMR Compare Register 1 (COMP1)	12-5
12.12.2	TMR Compare Register 2 (COMP2)	12-6

12.12.3	TMR Capture Register (CAPT)	12-6
12.12.4	TMR Load Register (LOAD)	12-7
12.12.5	TMR Hold Register (HOLD)	12-7
12.12.6	TMR Counter Register (CNTR)	12-7
12.12.7	TMR Control Registers (CTRL)	12-8
12.12.8	TMR Status and Control Registers (SCTRL)	12-10
12.12.9	TMR Comparator Load Register 1 (CMPLD1)	12-12
12.12.10	TMR Comparator Load Register 2 (CMPLD2)	12-13
12.12.11	TMR Comparator Status and Control Register (CSCTRL)	12-13
12.12.12	TMR Channel Enable Register (ENBL)	12-15
12.13	Functional Description	12-15
12.14	General	12-15
12.15	Functional Modes	12-16
12.15.1	STOP Mode	12-16
12.15.2	COUNT Mode	12-16
12.15.3	EDGE-COUNT Mode	12-17
12.15.4	GATED-COUNT Mode	12-18
12.15.5	QUADRATURE-COUNT Mode	12-18
12.15.6	SIGNED-COUNT Mode	12-19
12.15.7	TRIGGERED-COUNT Mode	12-20
12.15.8	One-Shot Mode	12-21
12.15.9	CASCADE-COUNT Mode	12-21
12.15.10	PULSE-OUTPUT Mode	12-23
12.15.11	FIXED-FREQUENCY PWM Mode	12-25
12.15.12	VARIABLE-FREQUENCY PWM Mode	12-25
12.15.13	Usage of Compare Registers	12-28
12.15.14	Usage of Compare Load Registers	12-29
12.15.15	Usage of Capture Register	12-29
12.16	Resets	12-30
12.17	General	12-30
12.18	Clocks	12-30
12.19	General	12-30
12.20	Interrupts	12-30
12.21	General	12-30
12.22	Description of Interrupt Operation	12-31
12.22.1	Timer Compare Interrupts	12-31
12.22.2	Timer Overflow Interrupts	12-31
12.22.3	Timer Input Edge Interrupts	12-31
12.23	Timing Specifications	12-32

Chapter 13 Universal Asynchronous Receiver/Transmitter Module (UART)

13.1	Overview	13-1
13.2	Features	13-1

13.3	Block Diagram	13-2
13.4	Functional Description	13-2
13.4.1	Signal Descriptions	13-2
13.4.2	Baud Rate Generation (Fractional Divider)	13-3
13.4.3	Basic Operation	13-6
13.4.4	FIFO Operation	13-7
13.5	Flow Control	13-8
13.6	RX Timeout Counter	13-9
13.7	Interrupts	13-9
13.7.1	Transmitter Sources	13-10
13.7.2	Receiver Sources	13-11
13.8	UART Register Memory Map	13-11
13.9	UART Registers	13-12
13.9.1	UART Control Register (UCON)	13-12
13.9.2	UART Status Register (USTAT)	13-14
13.9.3	UART Data Register (UDATA)	13-15
13.9.4	UART Buffer Control Registers	13-16
13.9.5	UART Baud Rate Divider Register (UBR)	13-19

Chapter 14 I2C Module

14.1	Overview	14-1
14.2	Features	14-1
14.3	Block Diagram	14-2
14.4	Functional Description	14-2
14.4.1	Signal Description	14-2
14.4.2	Modes of Operation	14-3
14.4.3	I ² C Protocol Description	14-3
14.4.4	Description of I ² C Functional Blocks	14-7
14.5	Reset	14-9
14.6	Interrupts	14-9
14.7	I2C Register Memory Map	14-10
14.8	I2C Registers	14-10
14.8.1	I2C Address Register (I2CADR)	14-10
14.8.2	I2C Frequency Divider Register (I2CFDR)	14-11
14.8.3	I2C Control Register (I2CCR)	14-13
14.8.4	I2C Status Register (I2CSR)	14-14
14.8.5	I2C Data Register (I2CDR)	14-16
14.8.6	Digital Filter Sampling Rate Register (I2CDFSRR)	14-16
14.8.7	Clock Enable Register (I2CCKER)	14-17
14.9	Initialization/Application Information	14-17
14.9.1	Initialization Sequence	14-18
14.9.2	Generation of START	14-18
14.9.3	Post-Transfer Software Response	14-18

14.9.4	Generation of STOP	14-19
14.9.5	Generation of Repeated START	14-19
14.10	Generation of SCL When SDA Low	14-19
14.10.1	Slave Mode Interrupt Service Routine	14-20
14.10.2	Interrupt Service Routine Flow Chart	14-20

Chapter 15 Serial Peripheral Interface Module (SPI)

15.1	Overview	15-1
15.2	Features	15-1
15.3	SPI Module Used for FLASH Interface (SPIF)	15-2
15.4	Block Diagrams	15-2
15.4.1	SPI System Block Diagram	15-2
15.4.2	SPI Module Block Diagram	15-3
15.5	Functional Description	15-3
15.5.1	Signal Descriptions	15-3
15.5.2	Clock Generation	15-4
15.5.3	Basic Operation	15-4
15.5.4	SPI Shift Clock Formats	15-5
15.5.5	SPI Interrupts	15-7
15.6	SPI Register Memory Map	15-8
15.7	SPI Registers and Control Bits	15-9
15.7.1	Transmit Data Register (SPI_TX_DATA)	15-9
15.7.2	SPI Received Data Register (SPI_RX_DATA)	15-10
15.7.3	SPI Clock Control Register (SPI_CLK_CTRL)	15-11
15.7.4	SPI Setup Register (SPI_SETUP)	15-13
15.7.5	SPI Status Register (SPI_STATUS)	15-17
15.8	Timing Information	15-18

Chapter 16 Synchronous Serial Interface Module (SSI)

16.1	Overview	16-1
16.1.1	Features	16-1
16.2	Block Diagram	16-2
16.3	Modes of Operation	16-3
16.3.1	Normal Mode	16-4
16.3.2	Network Mode	16-8
16.3.3	I2S Mode	16-13
16.3.4	External Frame and Clock Operation	16-15
16.3.5	Data Alignment Formats Supported	16-15
16.4	External Signal Description	16-17
16.5	Functional Description	16-17
16.5.1	Architecture	16-17
16.5.2	Clocking	16-17

16.5.3	Receive Interrupt Enable Bit Description	16-22
16.5.4	Transmit Interrupt Enable Bit Description	16-22
16.5.5	Internal Frame and Clock Shutdown	16-23
16.5.6	IP Bus Interface	16-24
16.6	Initialization/Application Information	16-25
16.7	Memory Map and Register Definition	16-27
16.7.1	SSI Memory Map	16-27
16.7.2	Register Summary	16-28
16.7.3	Register Descriptions	16-32

Chapter 17

Analog to Digital Converter Module (ADC)

17.1	Overview	17-1
17.2	Features	17-1
17.3	Block Diagram	17-2
17.4	ADC Registers	17-2
17.5	Register Access	17-4
17.5.1	adc_comp_0	17-4
17.5.2	adc_comp_1	17-5
17.5.3	adc_comp_2	17-6
17.5.4	adc_comp_3	17-7
17.5.5	adc_comp_4	17-8
17.5.6	adc_comp_5	17-9
17.5.7	adc_comp_6	17-10
17.5.8	adc_comp_7	17-11
17.5.9	adc_bat_comp_over	17-11
17.5.10	adc_bat_comp_under	17-12
17.5.11	adc_seq_1	17-12
17.5.12	adc_seq_2	17-13
17.5.13	adc_control	17-13
17.5.14	adc_triggers	17-15
17.5.15	adc_prescale	17-15
17.5.16	adc_fifo_read	17-16
17.5.17	adc_fifo_control	17-17
17.5.18	adc_fifo_status	17-17
17.5.19	adc_sr_1_high	17-18
17.5.20	adc_sr_1_low	17-18
17.5.21	adc_sr_2_high	17-18
17.5.22	adc_sr_2_low	17-19
17.5.23	adc_on_time	17-19
17.5.24	adc_convert_time	17-19
17.5.25	adc_clock_divider	17-20
17.5.26	adc_override	17-20
17.5.27	adc_irq	17-21

17.5.28	adc_mode	17-22
17.5.29	adc_ad1_result	17-22
17.5.30	adc_ad2_result	17-22
17.6	Detailed description of Digital ADC functionality	17-23
17.6.1	Prescale Clock	17-23
17.6.2	Clock Divider	17-23
17.6.3	Sequencers	17-23
17.6.4	Comparators	17-23
17.6.5	Fifo	17-24
17.6.6	Timing	17-24

Chapter 18 Development and Debug Support

18.1	Hardware Development Interfaces	18-1
18.1.1	JTAG Hardware Debug Port	18-1
18.1.2	A7S Nexus3 (NEX) ARM7 Core Development Interface	18-1
18.2	Software Development Tools	18-2
18.3	Development Hardware	18-2

Appendix A MC1322x Register Address Map

Appendix B MC1322x Software Driver Utilities

B.1	Overview	2-1
B.2	Driver Summary	2-1
B.3	Using the Drivers	2-1

Appendix C Bootloader Reference

C.1	Overview	3-1
C.2	Alternative Load Ports	3-1
C.3	Bootling from UART1	3-1
C.3.1	Procedure	3-1
C.3.2	Data Format	3-2
C.4	Bootling using SPI on the MC1322x	3-3
C.4.1	MC1322x Is SPI Slave	3-3
C.4.2	MC1322x Is SPI Slave Data Format	3-4
C.4.3	MC1322x Is SPI Master (Boot from External FLASH)	3-5
C.4.4	MC1322x Is SPI Master Data Format	3-6
C.5	Bootling from I2C (Boot from External Serial EEPROM)	3-6
C.5.1	Procedure	3-6
C.5.2	MC1322x Is I2C Master Data Format	3-7
C.6	Information Available to the User Program	3-7

About This Book

This manual describes Freescale's MC1322x, third-generation, ZigBee platform which incorporates a complete, low power, 2.4 GHz radio frequency transceiver, 32-bit ARM7 core based MCU, hardware acceleration for both the IEEE Standard 802.15.4 MAC and AES security, and a full set of MCU peripherals into an 80-pin LGA System on a Chip (SoC).

Audience

This manual is intended for system designers.

Organization

This document is organized into 18 chapters and 3 appendices.

Chapter 1	Introduction — This chapter introduces the MC1322x features and functionality.
Chapter 2	Pins and Connections — Describes device pinout and functionality.
Chapter 3	System Considerations — Describes system level considerations of the MC1322x modem and MCU.
Chapter 4	Memory — This chapter details the MC1322x ARM7 core which has a register-based instruction set and CPU registers that are not located in the memory map.
Chapter 5	System Management — This chapter details the management of the PiP including reset, power management, wake-up from low power, clock control, and KBI interface.
Chapter 6	IEEE 802.15.4 Transceiver — This chapter provides the reference for the radio and the modem. The MC1322x transceiver consists of the 2.4 GHz radio, modem, and MAC Accelerator.(MACA).
Chapter 7	CPU — Describes the MC1322x ARM7TDMI-S processor which is a member of the ARM family of general-purpose 32-bit microprocessors.
Chapter 8	Interrupts — Shows how interrupts provide a way for the MC1322x to inform the MCU of onboard events without requiring the MCU to constantly query MC1322x status.
Chapter 9	MACA MAC Accelerator — This chapter describes the low-level MAC and PHY link controller.
Chapter 10	Advanced Security Module (ASM) — The ASM block is a hardware engine that encrypts/decrypts using the Advanced Encryption Standard (AES).
Chapter 11	Parallel IO (GPIO) — Details the 64 GPIOs.
Chapter 12	Timer (TMR) — This chapter describes each Timer module (TMR) that contains four identical counter/timer groups.
Chapter 13	Universal Asynchronous Receiver/Transmitter Module (UART) — This chapter describes the Universal Asynchronous Receiver Transmitter (UART) module.

Chapter 14	I2C Module — The inter-IC (IIC or I2C) bus is a two-wire—serial data (SDA) and serial clock (SCL)— bidirectional serial bus.
Chapter 15	Serial Peripheral Interface (SPI) — This chapter describes the Serial Peripheral Interface (SPI) module for external use.
Chapter 16	Synchronous Serial Interface (SSI) — This chapter describes the Synchronous Serial Interface (SSI) architecture, programming model, operating modes, and initialization.
Chapter 17	Analog to Digital Converter (ADC) — The ADC module manages the interface to external sensors, scans multiple channels, and controls the warm-up of the analog portions of the analog to digital system.
Chapter 18	Development Debug and Support — This chapter describes the full set of hardware/software evaluation and development tools.
Appendix A	MC1322x Register Address Map — Provides a single table layout of the memory map.
Appendix B	Software Utilities in ROM — This appendix describes the extensive set of driver utilities for the platform.
Appendix C	Bootloader Reference — This appendix describes the MC1322x ROM-based bootloader in more detail.

Revision History

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

Revision History

Location	Revision
Entire Document	First public release of this document.

Definitions, Acronyms, and Abbreviations

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


ACK	Acknowledgement Frame
API	Application Programming Interface
BB	Baseband
CCA	Clear Channel Assessment
CRC	Cyclical Redundancy Check
DCD	Differential Chip Decoding
DME	Device Management Entity
FCS	Frame Check Sequence
FFD	Full Function Device
FFD-C	Full Function Device Coordinator

FLI	Frame Length Indicator
GTS	Guaranteed Time Slot
HW	Hardware
IRQ	Interrupt Request
ISR	Interrupt Service Routine
LO	Local Oscillator
MAC	Medium Access Control
MCPS	MAC Common Part Sublayer
MCU	Microcontroller Unit
MLME	MAC Sublayer Management Entity
MSDU	MAC Service Data Unit
NWK	Network
PA	Power Amplifier
PAN	Personal Area Network
PANID	PAN Identification
PHY	PHYSical Layer
PIB	PAN Information Base
PPDU	PHY Protocol Data Unit
PSDU	PHY Service Data Unit
RF	Radio Frequency
RFD	Reduced Function Device
SAP	Service Access Point
SFD	Start of Frame Delimiter
SPI	Serial Peripheral Interface
SSCS	Service Specific Convergence Layer
SW	Software
VCO	Voltage Controlled Oscillator

References

The following sources were referenced to produce this book:

1. IEEE 802.15.4 Standard
2. Freescale MC1319x Data Sheet
3. Freescale MC9S08GB/GT60 Data Sheet
4. Freescale MC1321x Data Sheet

- 
5. Standard for Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low rate wireless personal area networks (WPAN). IEEE Std 802.15.4-2006, IEEE, New York, NY, 2006.

Chapter 1

MC1322x Introduction

The MC1322x is Freescale's third-generation ZigBee platform which incorporates a complete, low power, 2.4 GHz radio frequency transceiver, 32-bit ARM7 core based MCU, hardware acceleration for both the IEEE 802.15.4 MAC and AES security, and a full set of MCU peripherals into a 99-pin LGA Platform-in-Package (PiP).

The RF radio interface provides for low cost and the high density as shown in [Figure 1-1](#). An onboard balun along with a TX/RX switch allows direct connection to a single-ended 50-Ω antenna. The integrated PA provides programmable output power typically from -30 dBm to +4 dBm, and the RX LNA provides -95 dBm sensitivity. In addition, separate complementary PA outputs allow use of an external PA and/or LNA for extended range applications. This solution also has onboard bypass capacitors and crystal load capacitors for the smallest footprint in the industry. All components are integrated into the package except the crystal and antenna.

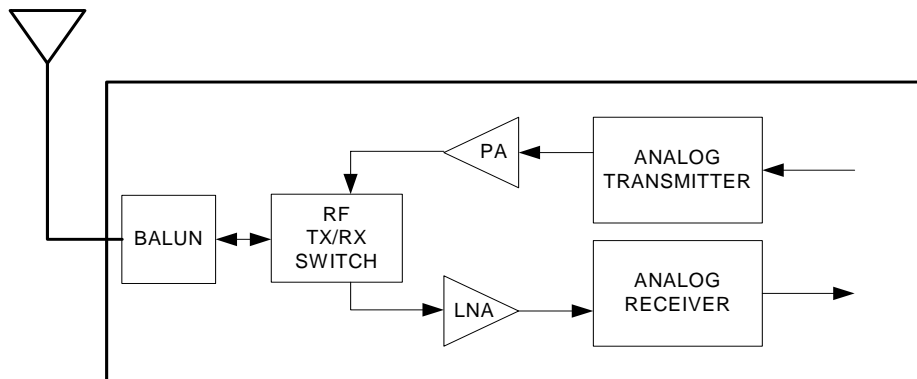


Figure 1-1. MC1322x RF Radio Interface

In addition to the best-in-class MCU performance and power, the MC1322x also provides best-in-class power savings. Typical transmit current is 28 mA and typical receive current is 21 with the CPU at 2 MHz operation. Onboard power supply regulation is provided for source voltages from 2.0 Vdc to 3.6 Vdc, and the source voltage can be as low as a regulated 1.8 Vdc if the non-volatile memory is powered directly from the source. Numerous low current modes are available to maximize battery life including sleep or restricted performance operation.

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

- Residential and commercial automation
 - Lighting control
 - Security
 - Access control
 - Heating, ventilation, air-conditioning (HVAC)

- Automated meter reading (AMR)
- Industrial Control
 - Asset tracking and monitoring
 - Homeland security
 - Process management
 - Environmental monitoring and control
 - HVAC
 - Automated meter reading
- Health Care
 - Patient monitoring
 - Fitness monitoring
- Consumer
 - Remote control
 - Entertainment systems
 - Cellular phone attach

1.1 Features

This section provides a simplified block diagram and highlights MC1322x features.

1.1.1 Block Diagram

Figure 1-2 shows a simplified block diagram of the MC1322x.

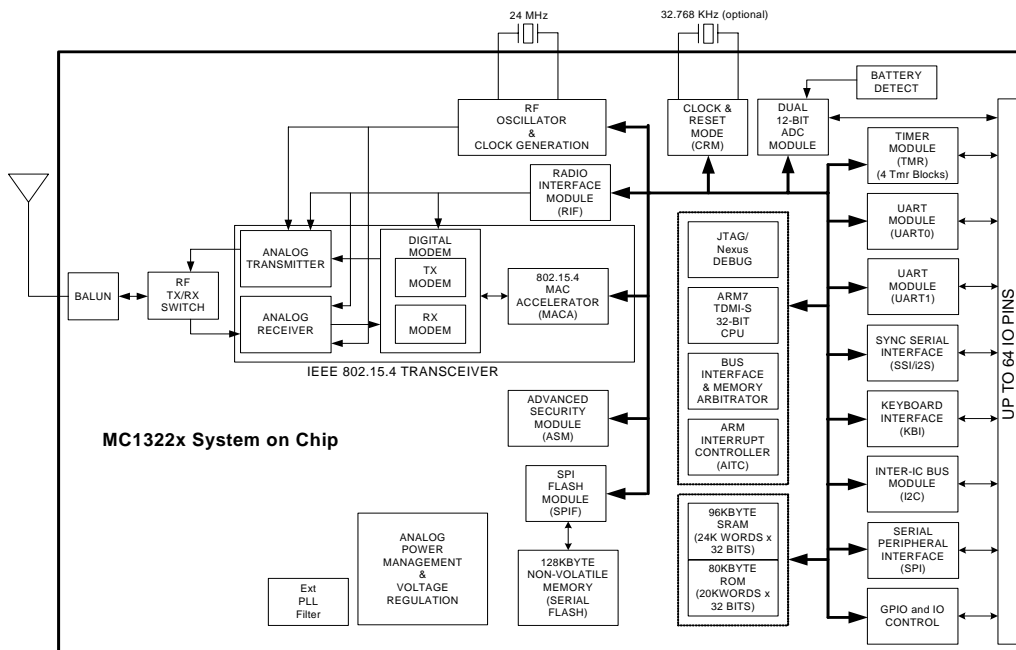


Figure 1-2. MC1322x Simplified Block Diagram

1.1.2 Features Summary

- IEEE 802.15.4 standard compliant on-chip transceiver/modem
 - 2.4GHz
 - 16 selectable channels
 - Programmable transmitter output power (-30 dBm to +4 dBm typical)
 - World-class receiver sensitivity
 - < -96 dBm typical receiver sensitivity using DCD mode (<1% PER, 20-byte packets)
 - < -100 dBm typical receiver sensitivity using NCD mode (<1% PER, 20-byte packets)
- Hardware acceleration for IEEE 802.15.4 applications
 - MAC accelerator (sequencer and DMA interface)
 - Advanced encryption/decryption hardware engine (AES 128-bit)
- Supports standard IEEE 802.15.4 signalling with 250 kbps data rate
- 32-bit ARM7TDMI-S CPU core with programmable performance up to 26 MHz (24 MHz typical)
- Extensive on-board memory resources
 - 128 Kbyte serial FLASH memory (can be mirrored into RAM)
 - 96 Kbyte SRAM
 - 80 Kbyte ROM
- Best-in-class power dissipation
 - 20-24mA maximum RX current draw (DCD mode) with radio and MCU active
 - 16-40mA maximum TX current draw with radio and MCU active (coin cell capable)
 - 5mA maximum current draw with MCU active (radio off)
 - 0.9mA maximum current with MCU idle (radio off)
 - 1.1 μ A maximum Hibernate current (retain 8 Kbyte SRAM contents)
 - 0.3 μ A maximum Off current (device in reset)
- Extensive sleep mode control and variation
 - Hibernate and Doze low power modes
 - Programmable degree of power down
 - Clock management
 - Onboard 2kHz oscillator for wake-up timer.
 - Optional 32.768 kHz crystal oscillator for accurate real-time sleep mode timing and wake-up with a possible sleep period greater than 36.4 hours
 - wake-up through programmable timer, external real-time interrupts, or ADC timer
- Extensive MCU peripherals set
 - Dedicated 802.15.4 modem/radio interface module (RIF)
 - Dedicated NVM SPI interface for managing FLASH memory
 - Two dedicated UART modules capable of 2Mbps with CTS/RTS support
 - SPI port with programmable master and slave operation

- 8-pin keyboard interface (KBI) supports up to a 4x4 matrix. Also, provides up to 4 asynchronous interrupt inputs for wake-up
- Two 12-bit analog-to-digital converters (ADCs) share 8 input channels
- Four independent 16-bit timers with PWM capability. These can cascade in combinations up to 64-bit operation
- Inter-integrated circuit (I2C) interface
- Synchronous Serial Interface (SSI) with I2S and SPI capability and FIFO data buffering
- Up to 64 programmable I/O shared by peripherals and GPIO
- Powerful In-circuit debug and FLASH programming available via on-chip JTAG and/or Nexus debug ports
- System protection features
 - Low battery detect
 - Watchdog timer (COP)
 - Sleep mode timer
- Low external component count
 - Only antenna needed for single-ended 50- Ω RF interface (balun in package)
 - Only a crystal is required for the main oscillator; programmable crystal load capacitors are on-chip
 - All bypass capacitors in package
- Supports single crystal reference clock source (typical 24 MHz crystal with 13 - 26 MHz usable) with on-chip programmable crystal load capacitance or external frequency source. Also provides onboard 2kHz oscillator for wake-up timing or an optional 32.768 kHz crystal for accurate low power timing.
- 2V to 3.6V operating voltage with on-chip voltage regulators; down to 1.8V with off-chip regulation. Optional buck converter for better battery life.
- -40°C to +105°C temperature range
- RoHS-compliant 9.5mm x 9.5mm x 1.2mm 99-pin LGA package

1.2 High Density, Low Component Count, Integrated IEEE 802.15.4 Solution

The MC1322x is more than a high performance, low power system-on-chip IEEE 802.15.4 solution. Not only are the transceiver (radio) and MCU on an SoC, the packaged solution contains a 128 Kbyte serial FLASH memory, onboard bypass capacitors for critical nodes, and RF components that present a single-ended 50- Ω interface for an external antenna. The radio is a full differential design with an on-chip transmit/receive (TX/RX) switch, and the PiP also has an onboard balun for differential to singled-ended conversion that is wired to present the single-ended interface to the application. On-chip RF matching is also provided to present the proper impedance to the antenna.

To further simplify the application, single crystal operation (typically a 24 MHz crystal) is supported for full radio and MCU operation. The load capacitance to the crystal oscillator is supplied on-chip to eliminate the need for the otherwise required external capacitors.

1.3 Integrated IEEE 802.15.4 Transceiver (Radio and Modem)

The MC1322x IEEE 802.15.4 fully-compliant transceiver provides a complete 2.4 GHz radio with 250 Kbps Offset-Quadrature Phase Shift Keying (O-QPSK) data in 5.0 MHz channels and full spread-spectrum encode and decode. The modem supports transmit, receive, clear channel assessment (CCA), Energy Detect (ED), and Link Quality Indication (LQI) as required by the 802.15.4 Standard.

1.3.1 RF Interface and Usage

The MC1322x RF interface provides for a single-ended, 50- Ω port that connects directly to an antenna. There is an onboard balun that converts the single-ended interface to a full differential, bi-directional, on-chip interface with transmit/receive switch, LNA, and complementary PA outputs. The required port impedance matching is also onboard. This combination allows for a very small footprint and a very low cost RF solution.

For longer range applications where external amplification may be desired (LNA and/or PA), additional ports are provided for secondary complementary PA outputs. These can be used as a separate PA interface while the single-ended port through the balun is used as an input only. Also, 4 control pins and a regulated 20 mA voltage source (VREG_ANA) are provided to control external components and supply power to the PA outputs.

The RF Interface functionality can be summarized as follows:

- Programmable output power — 0 dBm nominal output power, programmable to from -30 to +4 dBm max
- Receive sensitivity (at 1% PER, 20-byte packet) -
 - < -96 dBm (typical) DCD receive (well above IEEE 802.15.4 specification of -85 dBm)
 - < -100 dBm (typical) NCD receive (higher current)
- Single-ended 50- Ω port — Uses integrated transmit/receive (T/R) switch, LNA, and onboard balun. Impedance matching onboard on the chip side of the balun
- Maximum flexibility — Optionally, single-ended port becomes RF input only and a separate set of full differential PA outputs are provided. Separate input and outputs allow for a variety of RF configurations including external LNA and PA for increased range
- Four programmable control signals for external components
- Regulated voltage source for PA biasing and powering external components

1.3.2 Modem

The modem supports the full requirement of the IEEE 802.15.4 Standard to transmit and receive data packets. In addition, the mechanism is present to measure received signal level to provide CCA, ED, and LQI as required by the 802.15.4 Standard.

1.4 High Performance, Low Power 32-Bit ARM7 Processor

- The ARM7TDMI-S processor is a member of the 32-bit ARM family of general-purpose 32-bit microprocessors that offers high performance with very low-power consumption
- A three stage instruction pipeline (fetch, decode, execute) increases the speed of the flow of instructions to the processor
- Data access can be 8-bit bytes, 16-bit half words, or 32-bit words. Words must be aligned to 4-byte boundaries. Half words must be aligned to 2-byte boundaries
- The ARM7TDMI-S processor supports two instruction sets, i.e., the 32-bit ARM instruction set and the 16-bit Thumb instruction set. The Thumb mode incorporates 16-bit instructions for higher code density while retaining all the benefits of a 32-bit architecture, i.e., full 32-bit registers, 32-bit operations, and 32-bit memory transfer. The use of the instruction sets can be intermixed for maximizing performance while retaining higher code density

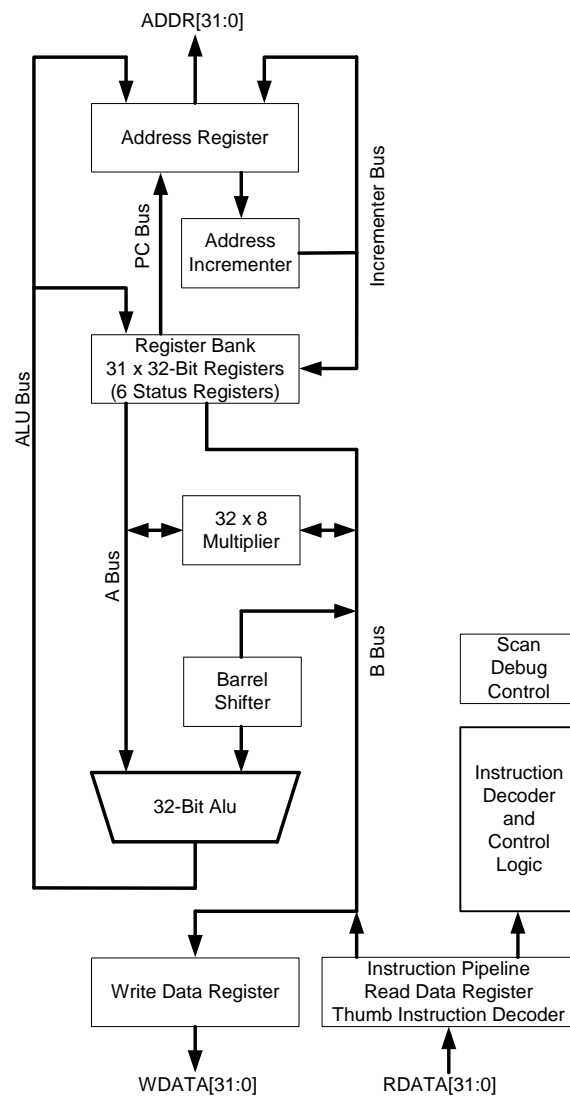


Figure 1-3. ARM7TDMI-S 32-Bit CPU Core

1.5 Low Power Operation and Power Management

The MC1322x is inherently a very low power device, but it also has extensive power management and an onboard buck regulator option to maximize battery life.

1.5.1 Operating Current

Table shows MC1322x currents as a function of operating mode. There are two basic low power modes of Hibernate and Doze, and both have options of how much RAM contents are retained. The primary difference between Hibernate and Doze is that Doze mode keeps the primary reference oscillator running.

Table 1-1. MC1322x Operating Currents @ 25°C

Mode	Description	Max	Unit
Off	Device is in reset condition (held in reset)	0.3	μA
Hibernate w/RTI	RAM retained (8k, 64k, or 96k)		
	Reference oscillator on (low frequency accuracy)		
	CPU off (stop mode)		
	wake-up from internal timer, 32.768 oscillator, ADC timer, or external request		
	Radio off		
	ADCs not available		
	8 Kbyte RAM retention	1.1	μA
	64 Kbyte RAM retention	3	μA
	96 Kbyte RAM retention	5	μA
Doze	RAM retained (8k, 64k, or 96k)		
	Onboard 2 kHz oscillator on (high frequency accuracy)		
	CPU off (stop mode)		
	wake-up from internal timer, 32.768 oscillator, ADC timer, or external request		
	Peripheral registers retained		
	Radio off		
	ADCs available, but inactive		
	8 Kbyte RAM retention	60	μA
	64 Kbyte RAM retention	65	μA
	96 Kbyte RAM retention	70	μA
Idle	All RAM active Reference oscillator on (24 MHz) at 1.2 VDC CPU on at reference frequency RF registers on Reference clock available to all peripherals ADCs available, but inactive	0.9	mA
Run	All RAM active Reference oscillator on (24 MHz) at 1.2 VDC CPU on at reference frequency Radio off Reference clock available to all peripherals ADCs available, but inactive	5	mA

Table 1-1. MC1322x Operating Currents @ 25°C

Mode	Description	Max	Unit
Receive	All RAM active Reference oscillator on (24MHz) at 1.2 VDC Radio RX on (receiving data) Reference clock available to all peripherals ADC1 available, but inactive		
	CPU on at 2 MHz (DCD) CPU on at 2 MHz (NCD)	24 27	mA mA
Transmit	All RAM active Reference oscillator on (24MHz) at 1.2 VDC Radio TX on (sending data) Reference clock available to all peripherals ADCs available, but inactive		
	CPU clock at 2 MHz (< 20% EVM)	33	mA

1.5.2 Power Management

The MC1322x power management is controlled through the Clock and Reset Module (CRM). The CRM is a dedicated module to handle MCU clock, reset, and power management functions which includes control of the power regulators. All these functions have impact on attaining lowest power.

1.5.2.1 CRM Features

The CRM features include:

- Control system reset
- Control clock gating for power savings
- Sleep mode (Hibernate and Doze) management
 - Degree of chip power down
 - Retention of programmed parameters
 - Programmable retention of RAM contents
 - Clock management
- Wake-up management
 - Graceful power-up
 - Clock management
 - Wake-up via programmable timer or external interrupts.
- Wake-up timer
 - Based on onboard 2 kHz oscillator
 - Optional 32.768 kHz crystal oscillator
- Controls reference clocks based on default 24 MHz crystal oscillator or optional 13-26 MHz oscillator with PLL for 24 MHz frequency synthesis.
- MCU watchdog timer (COP)
- Software initiated reset

- Management control of onboard linear regulators and optional buck regulator

1.5.2.2 CRM Operation

The CRM has primary control of the entire system:

- Reset and power up — After release of the hardware RESETB signal, the CRM will perform a power up sequence of the MCU after the POR. The linear regulators and clock sources are managed for a graceful start-up of the MCU and its resources. The radio is not powered until needed for an operation
- Normal operation of MCU — The clock management of the MCU and its resources are controlled by the CRM. The processor clock is programmable from low frequencies up to the maximum reference frequency (13-26 MHz optional w/24 MHz standard) to allow the application to trade-off processing speed versus power savings
- Sleep modes and recovery — There are two sleep modes of Hibernate and Doze. The primary difference is that Doze mode keeps the reference oscillator running. Both modes retain critical programmed parameters and have selectable sizes of RAM retention. Hibernate has lowest power, but Doze allows useful clock options with peripherals
- The CRM manages the recovery from low power, similar to power-up from reset, providing regulator and clock management.
 - Wake-up can be based on external interrupts through 4 KBI inputs
 - Wake-up can be from internal interrupts
 - Wake-up can be based on an RTI (wake-up) timer.
- The RTI timer with 2 possible frequency sources provides a very low power wake-up option from sleep
 - One option is a onboard, low accuracy 2 kHz oscillator
 - A second option is to add an external 32.768 kHz crystal for the RTI clock source
 - A 32-bit timer allows greater than a 36.4 hour wake-up delay with the 32.768 crystal oscillator
- Other features of the CRM:
 - An optional COP watchdog timer to monitor CPU program activity
 - A programmable software reset

1.5.3 Optional Buck Regulator

For battery based applications, an optional buck regulator is provided to maximize battery life. [Figure 1-4](#) shows the configuration of the buck regulator versus the normal connection. An onboard MOSFET is used as a switch with an external 100 μ H inductor and 10 μ F capacitor when the buck regulator is enabled. The buck regulator drops the higher battery voltage to 1.8 - 2.0 Vdc that is applied to the onboard linear regulators. This allows lower net current from the battery to maximize the life of the battery. The battery voltage can drop as low as 2.1 VDC and still provide power to the buck regulator.

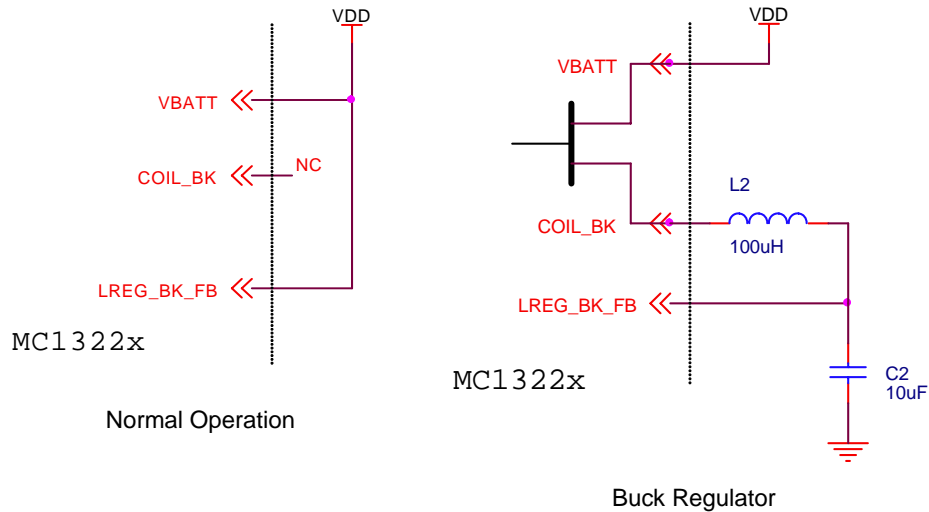


Figure 1-4. Optional Buck Regulator

1.6 Battery Detect

An optional feature of the ADC module is battery voltage detect. Programmable thresholds are provided for an ADC analog sample channel to monitor the battery high voltage and battery low voltage. This feature can be used as a trigger to provide low battery indication, protection for data that may be lost due to end-of-life for the battery, or monitoring charging.

1.7 IEEE 802.15.4 Acceleration Hardware

The MC1322x contains two blocks that provide acceleration for IEEE 802.15.4 applications that use the 802.15.4 MAC and AES encryption/decryption.

1.7.1 802.15.4 MAC Accelerator (MACA) Overview

The MC1322x contains a hardware block that provides a low-level MAC and PHY link controller, which together with software running on the ARM core, implements the baseband protocols and other low-level link routine control and link control. Components of the MACA include a sequencer/controller (with timers), TX and RX packet buffers, DMA block, frame check sequence (FCS) generator/checker, and control registers. [Figure 1-5](#) shows a MACA simplified block diagram.

As part of the 802.15.4 protocol, packets are generated and transmitted, packets are received and verified, and channel energy is measured via a clear channel assessment (CCA). Also, combinations or sequences of events are required as part of the protocol such as an ACK response following a received packet. The MACA facilitates these activities via control of the transceiver and off loads the functions from the CPU. A dedicated DMA function moves data between the MACA buffers and RAM on a cycle steal basis and does not require intervention from the CPU.

The MACA is responsible for construction of packets for TX including FCS, and for parsing the received packets. The MACA will also handle ACKs and TxPoll sequences independent of the ARM processor. During TX the MACA will construct the entire packet. This includes preamble and SFD (start of frame

delimiter). During receive, the modem will recognize preamble and SFD, then begin receiving the packet with the first bit of frame length, and finally, will check the FCS.

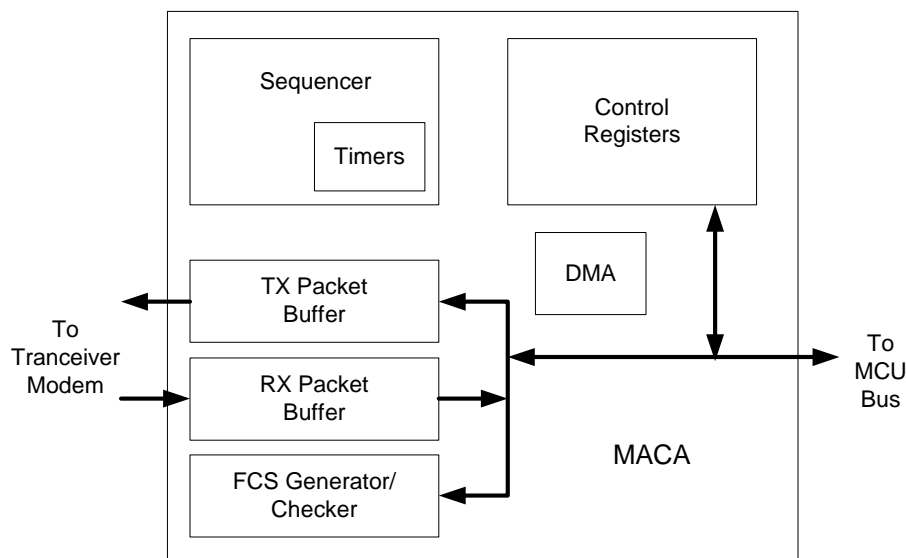


Figure 1-5. MAC Accelerator Simplified Block Diagram

1.7.1.1 MACA Features

In order to reduce CPU load, the MACA module has embedded features for controlling parts of the IEEE 802.15.4 PHY and MAC layer requirements. The MACA core features include:

- Sequence manager auto sequences
 - Automatic acknowledgment frame reception on transmitted packets
 - Automatic acknowledgment frame transmission on received packets
 - Auto-RX for continuous reception as coordinator
 - Auto sequence for transmitted MAC data.request
 - Assist for efficient response to MAC data.requests
 - Embedded channel assessment in sequence
 - Support for sequences with slotted mode access
 - Timer triggered and immediately executed actions
 - Support for extended RX for reception in random backoff and battery life extension
 - Support for promiscuous mode
- Dedicated DMA for transfer of TX/RX data from/to RAM
- Maskable, event-driven interrupt generation
- Address header filtering for received packets. A promiscuous mode allows bypass of the filtering for monitoring network traffic
- Packet manager
 - Handles preamble data
 - Handles frame check sequence (FCS) a.k.a CRC

- Embedded header filter for received packets
- Beacon Support Mode
- Control/status registers mapped into CPU memory map
- 32-Bit random number generator — Runs at the bus clock rate, a 32-bit Linear Feedback Shift Register (LFSR) can be set with a seed value and uses a 32-bit primitive polynomial. A 32-bit random number is fetched with every read of the proper control register

1.8 Advanced Security Module (ASM)

The IEEE 802.15.4 Standard and the ZigBee Standard both provide for optional use of data encryption. The ASM engine is a hardware block that accelerates encryption/decryption using the Advanced Encryption Standard (AES). The engine can perform “Counter” (CTR) and Cipher Block Chaining (CBC) encryption. The combination of these two modes of encryption are known as CCM mode encryption. CCM is short for Counter with CBC-MAC. CCM is a generic authenticate and encrypt block cipher mode. CCM is only defined for use with 128 bit block ciphers, such as AES. The definition of CCM mode encryption is documented in the NIST publication SP800-38C.

The ASM has the following features:

- 32-Bit wide bus interface
- CTR encryption in 13 clock cycles
- CBC encryption in 13 clock cycles
- Encrypts 128 bits as a unit
- The 128-bit registers are aligned on quad word boundaries (16 byte)
- Self-test mode
- Maskable “action complete” interrupt

1.9 Memory

The MC1322x memory resources consist of RAM, ROM, and serial FLASH.

1.9.1 RAM and ROM

The static RAM and ROM resources are located on the primary chip. Features include:

- 96 Kbytes RAM.
 - RAM0: 8 Kbytes, 2 Kwords (2048 x 32 bits)
 - RAM1: 24 Kbytes, 6 Kwords (6144 x 32 bits)
 - RAM2: 32 Kbytes, 8 Kwords (8192 x 32 bits)
 - RAM3: 32 Kbytes, 8 Kwords (8192 x 32 bits)
- All read or write accesses require a minimum of two system clock cycles
- Stall signal generated for read after write cycles
- Clock is enabled only on the accessed memory device for low power consumption

- RAMs have been divided to allow for power savings. While sleeping, combinations of the 4 RAMs can be turned off and the RAM remainder can be placed in a low voltage mode for data retention. If more RAMs are turned on, then less battery life will be achieved. Depending on the amount of RAM powered during sleep, the boot time may be longer with less RAM as the non-powered RAM must be reloaded from FLASH
- 80 Kbytes ROM
 - 20 Kwords (20480 x 32 bits)
 - Initially contains bootstrap code, 802.15.4 MAC (no security), UART driver, and SPI driver. The MAC software builds on the lower level hardware capability of the transceiver and MACA. All code except the bootstrap is “patchable”
 - Can be extended later to extend communications stack software and NVM services (erase, program, and read routines)

1.9.2 Serial FLASH (NVM)

The MC1322x also contains a 128 Kbyte serial FLASH memory that can be mirrored into the 96 Kbyte RAM. The serial FLASH is accessed via a dedicated SPI module designated as the SPIF. The FLASH erase, program, and read capability are programmed through the SPIF port. The FLASH is accessed at boot time to load/initialize RAM. All actual CPU program and data access is typically from RAM or ROM.

1.10 MCU Peripherals

The MC1322x has a rich set of MCU peripherals. Figure 1-6 shows the peripheral modules.

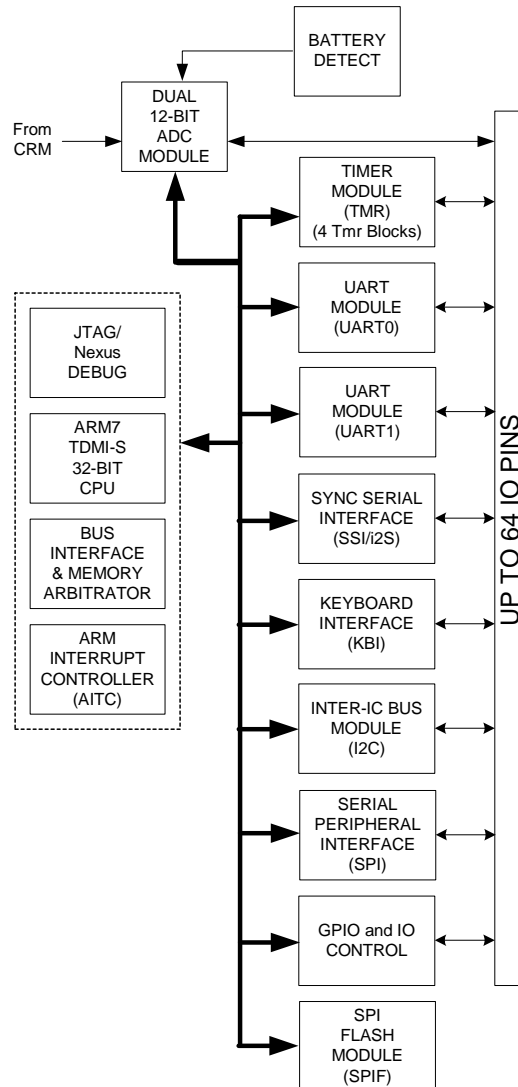


Figure 1-6. MCU Peripherals

1.11 Parallel IO (GPIO)

In addition to general purpose usage the parallel IO provide:

- Analog inputs for the ADC channels
- Control for external RF components such as an LNA and/or PA
- Debug ports for JTAG and Nexus modules

The basic parallel I/O features include:

- A total of 64 general-purpose I/O pins
- Port assignments TBD

- Hysteresis input buffers
- Individual control (direction and output state) for each pin when in GPIO mode
- Software-controlled pull-ups on each input pin
- Eight pins shared with ADC input channels
- Eight pins shared with UART1 and UART2
- Two pins shared with I2C
- Four pins shared with timer block
- Four pins shared with SPI block
- Four pins shared with SSI block
- Eight pins shared with KBI block
- Four pins shared with RF external component control
- Four pins shared with ADC reference voltages
- Four pins shared with JTAG port
- Fourteen pins shared with Nexus port
- Pin states are captured and retained when entering low power (sleep) modes
- KBI pins (4 total) can be used as wake-up interrupts

1.12 Keyboard Interface (KBI) Interrupts

The MC1322x designates 8 pins (KBI_0 to KBI_7) as typically used for a keyboard interface, where four of these signals are outputs and four are inputs (KBI_4 to KBI_7) that support asynchronous interrupts for wake-up (during sleep modes). These 8 pins can be used as a matrix interface to support up to 16 switches or buttons, such as a keypad. These signals can also be used as general purpose IO if a keyboard is not present.

1.13 Timer (TMR) Module

The MC1322x provides a timer module (TMR) that contains four identical counter/timer groups. Each group is capable of many variants of input capture, output compare and pulse-width modulation. The wide range of operational modes is useful for many control and sensor applications.

[Figure 1-7](#) shows a block diagram of an individual timer group.

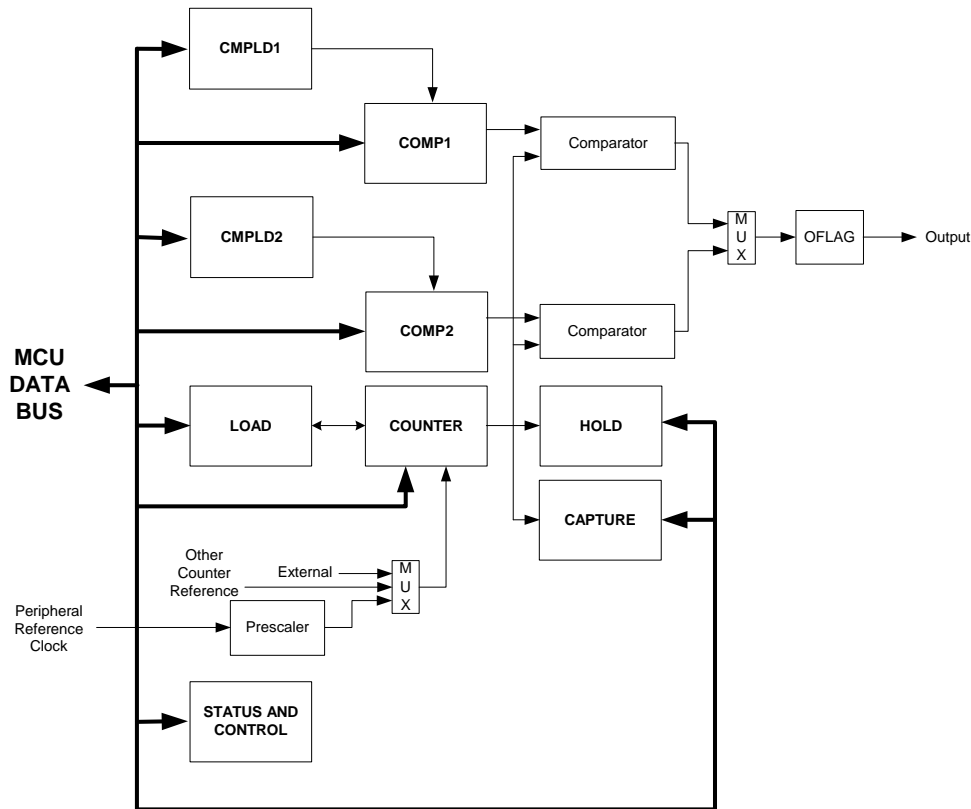


Figure 1-7. Timer Group Block Diagram

Each 16-bit counter/timer group contains a prescaler, a counter, a load register, a hold register, a capture register, two compare registers, and status and control registers.

- Load Register — Provides the initialization value to the counter when the counter's terminal value has been reached
- Hold Register — Captures the counter's value when other counters are being read. This feature supports the reading of cascaded counters
- Capture Register — Enables an external signal to take a snap shot of the counter's current value
- COMP1 and COMP2 Registers — Provides the values to which the counter is compared. If a match occurs, the OFLAG signal can be set, cleared, or toggled. At match time, an interrupt is generated (if enabled), and the new compare value is loaded into the COMP1 or COMP2 registers from CMPLD1 and CMPLD2 if enabled
- The Prescaler provides different time bases useful for clocking the counter/timer
- The Counter provides the ability to count internal or external events
- Control and Status Registers — Provides operational mode control of the counter, status, clock source control, interrupt control, and external interface control

Within a Timer module (set of 4 timer/counters) the input pins are shareable.

The TMR module feature include:

- Four - 16 bit counters/timers groups

- Up/down count
- Counters can be cascaded for up to 64 bit delay counter
- Programmable count modulo.
- Peripheral reference clock equates to reference oscillator frequency
- External clock max count rate equals peripheral clock divided by 2
- Internal clock max count rate equals peripheral clock.
- Count once or repeatedly
- Counters can be preloaded
- Compare registers can be preloaded
- Counters share available 4 IO pins with programmable use as inputs or outputs and falling or rising edge
- Separate prescaler for each counter
- Each counter has capture and compare capability
- Optional input glitch filter
- Functional modes include stop, count, edge-count, gated-count, quadrature-count, signed-count, triggered-count, one-shot, cascade-count, pulse-output, fixed frequency PWM, and variable-frequency PWM

1.14 UART Modules

The MC1322x has two universal asynchronous receiver/transmitter (UART) modules. Each UART has an independent fractional divider, baud rate generator that is clocked by the peripheral bus clock (typically 24 MHz) which enables a broad range of baud rates up to 1,843.2 Kbaud. Transmit and receive use a common baud rate for each module.

Each UART provides the following features:

- 8-Bit only data
- One or two stop bits
- Programmable parity (even, odd, and none)
- Four-wire serial interface (RXD, TXD, RTS, and CTS)
- Hardware flow control support for RTS and CTS signals
- 32-byte receive FIFO and 32-byte transmit FIFO
- Programmable sense for RTS/CTS pins (high true/low true)
- Status flags for various flow control and FIFO states
- Receiver detects framing errors, start bit error, break characters, parity errors, and overrun errors.
- Voting logic for improved noise immunity (16X/8X oversampling)
- Maskable interrupt request
- Time-out counter, which times out after eight non-present characters
- Receiver and transmitter enable/disable
- Low-power modes

- Baud rate generator to provide any multiple-of-2 baud rate between 1.2 kbaud and 1,843.2 kbaud

1.15 Inter-Integrated Circuit (I2C) Module

The MC1322x provides an Inter-Integrated Circuit (I2C) module for the I2C which is a two-wire, serial data (SDA) and serial clock (SCL), bidirectional serial bus. The I2C allows for data exchange between the MC1322x and other devices such as MCUs, serial EEPROM, serial ADC and DAC devices, and LCDs. The I2C minimizes interconnections between devices and is a synchronous, multi-master bus that allows additional devices to be connected and still handle system expansion and development. The bus includes collision detection and arbitration to prevent data corruption if two or more masters attempt to simultaneously control the I2C.

The I2C module is driven by the peripheral bus clock (typically 24 MHz) and the SCL bit clock is generated from a prescaler. The prescaler divide ratio can be programmed from 61,440 to 160 (decimal) which gives a maximum bit clock of 150 kbps.

The I2C module supports the following features:

- Two-wire (SDA and SCL) interface
- Multi-master operation
- Master or slave mode
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Acknowledge bit generation/detection
- Bus busy detection
- Software-programmable bit clock frequency up to 150 kbps
- Software-selectable acknowledge bit
- On-chip filtering for spikes on the bus

1.16 Serial Peripheral Interface (SPI) Modules

The MC1322x has two SPI modules that use a common architecture

1.16.1 External SPI Module

The MC1322x offers a dedicated Serial Peripheral Interface (SPI) module for external use. The SPI is a high-speed synchronous serial data input/output port used for interfacing with serial memories, peripheral devices, or other processors. The SPI allows a serial bit stream of a programmed length (1 to 32 bits) to be shifted simultaneously into and out of the device at a programmed bit-transfer rate (called 4-wire mode). There are four pins associated with the SPI port (SPI_SCK, SPI_MOSI, SPI_MISO, and SPI_SS).

The SPI module can be programmed for master or slave operation. It also supports a 3-wire mode where for master mode the MOSI becomes MOMI, a bidirectional data pin, and for slave mode the MISO becomes SISO, a bidirectional data pin. In 3-wire mode, data is only transferred in one direction at a time.

The SPI bit clock is derived from the peripheral reference clock (typically 24 MHz with a maximum of 26 MHz). A prescaler divides the peripheral reference clock with a programmed divide ratio from 2 to 256. Typical bit clock range will be from 12 MHz to 93.75 kHz.

The SPI has the following features:

- Master or slave mode operation
- Transfer length programmable from 1 to 32 bits
- MSB-first shifting
- Data buffer is 4 bytes (32 bits) in length
- Programmable transmit bit rate (typically 12 MHz max)
- Serial clock phase and polarity options
- Full-duplex (4-wire) or bidirectional data (3-wire) operation
- SPI transaction can be polled or interrupt driven
- Slave select output
- Low Power (SPI Master uses gated clocks. SPI Slave clock derived completely from SPI_SCK.)
- SPI Slave does not need system clock to be active

1.16.2 SPI FLASH Module (SPIF)

The SPIF is an internal SPI block dedicated to control, reading, and writing of the serial FLASH memory (NVM). It uses the same architecture as the general SPI block, but will be limited by the characteristics of the FLASH SPI interface.

1.17 Synchronous Serial Interface (SSI) Module

The MC1322x provides a versatile Synchronous Serial Interface (SSI) which is a full-duplex, serial port that allows communication with a variety of serial devices. These serial devices can be standard CODer-DECoder (CODECs), digital signal processors (DSPs), MCUs, peripherals, and popular industry audio CODECs that implement the Inter-Integrated Circuit sound bus standard (I2S).

The SSI typically transfers samples in a periodic manner and it consists of independent transmitter and receiver sections with common clock generation and frame synchronization. The external signals include the bit clock (SSI_BITCK), frame sync (SSI_FSYN), RX data (SSI_RX), and TX data (SSI_TX). The SSI has the following basic operating modes all with synchronous protocol:

- Normal mode — The simplest SSI mode transfers data in one time slot per frame
- Network mode — Creates a Time Division Multiplexed (TDM) network, such as a TDM CODEC network or a network of DSPs
- Gated Clock mode — Hooks up to SPI-type interfaces on MCUs or external peripheral chips in addition to providing communication

With its multi-modes, the SSI can be programmed for two very useful functions:

- A second SPI port augmenting the MC1322x SPI module

- I2S interface - the SSI is capable of generating the required clock frequencies and data format to drive a serial stereo audio DAC

The SSI includes the following features:

- Synchronous transmit and receive sections with shared internal/external clocks and frame syncs operating in Master or Slave mode
- Normal mode operation using frame sync
- Network mode operation allowing multiple devices to share the port with as many as thirty-two time slots
- Gated Clock mode operation requiring no frame sync
- Transmit and Receive FIFOs. Each of the FIFOs is 8x24 bits. The TX/RX FIFOs can be used in Network mode to provide 2 independent channels for transmission and reception
- Programmable data interface modes such as I2S, LSB, MSB aligned
- Programmable word length (8, 10, 12, 16, 18, 20, 22 or 24 bits)
- Program options for frame sync and clock generation
- Programmable I2S modes (Master, Slave or Normal). Oversampling clock available as output from SRCK in I2S Master mode
- External SSI_CLK input for use in I2S Master mode. Programmable oversampling clock of the sampling frequency available as output in master mode at SRCK, when operated in sync mode
- Programmable internal clock divider
- Time Slot Mask Registers for reduced CPU overhead (for both TX and RX)
- SSI power-down feature

1.18 Analog-to-Digital Converter (ADC) Module

The MC1322x ADC module provides two 12-bit analog-to-digital converters (ADC1 and ADC2) with 8 external channels (ADC7 - ADC0) that can be multiplexed to either ADC. ADC1 can also sample the battery voltage for monitoring purposes. External pins (ADC2_VREFH, ADC2_VREFL, ADC1_VREFH, and ADC1_VREFL) are provided for independent ADC reference voltages. The minimum sample time is 20 μ s. [Figure 1-8](#) shows a block diagram of the ADC module.

Each ADC can be programmed to scan multiple selected channels on a timed basis. The primary clock to the ADC module is the peripheral reference clock (typically 24 MHz). For the time period between scan sequences, the primary clock is first divided by an 8-bit prescale (1-255), and the derived clock drives both the 32-bit delay timer and the ADC sequencer. Each ADC has its own delay timer and sequencer.

Once a scan sequence has been initiated, all selected channels can be sampled. Registers are provided to define thresholds that can be enabled for the sampled channels. A threshold can be assigned to a specific channel and can be programmed to be a less-than or greater-than threshold. Multiple thresholds can be assigned to a single channel. Warm-up of the analog portion of the circuitry is provided (for power management), and a separate 300 kHz ADC clock must be programmed via its own divider.

The battery monitor has two (2) dedicated threshold registers to set the high and low limits of the battery sample channel.

Sample values are stored in a 8x16-bit FIFO. The FIFO accumulates samples from both ADCs, and the 12-bit sample value and a 4-bit channel tag are saved for each sample. The FIFO is read by the CPU from a register address.

The module can be programmed to interrupt the processor based on the timed sample activity. Sample activity, sequencer activity, or FIFO “fullness” can all be enabled to generate an interrupt.

The ADCs can also be overridden to sample on command as opposed to sequencer, time-based activity.

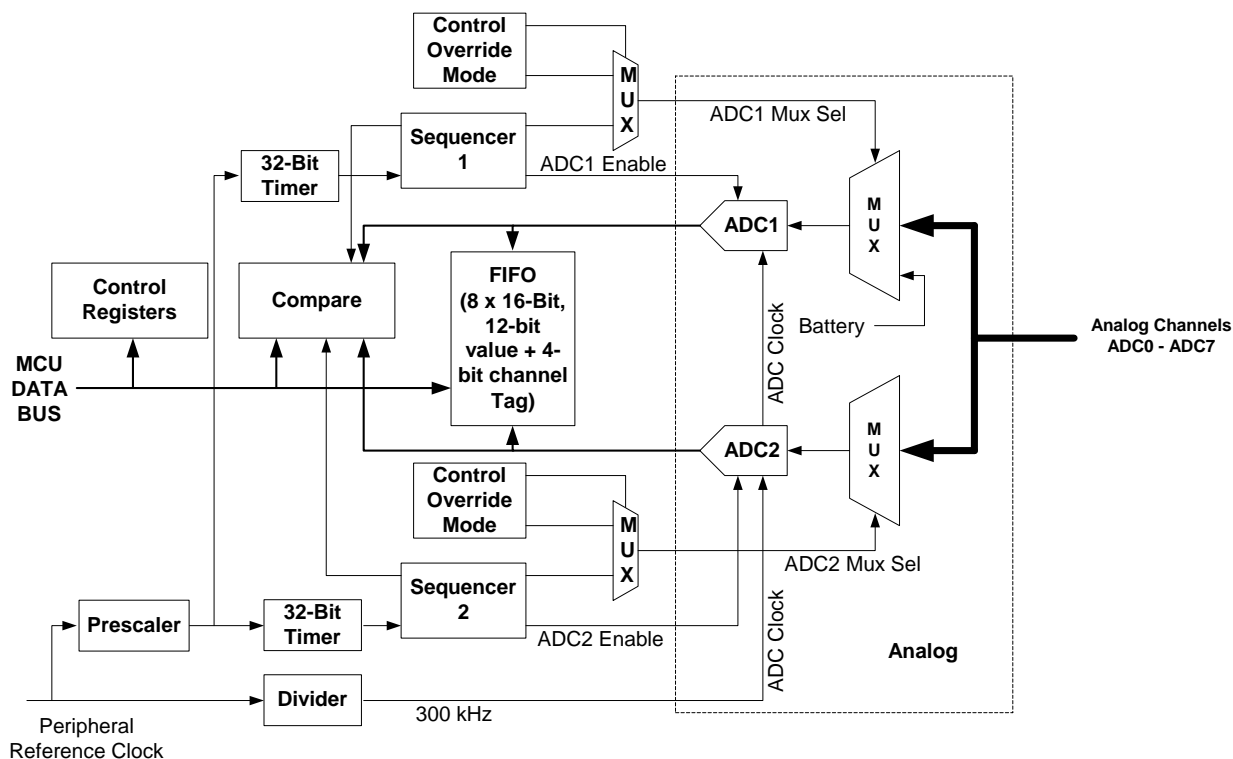


Figure 1-8. ADC Module Block Diagram

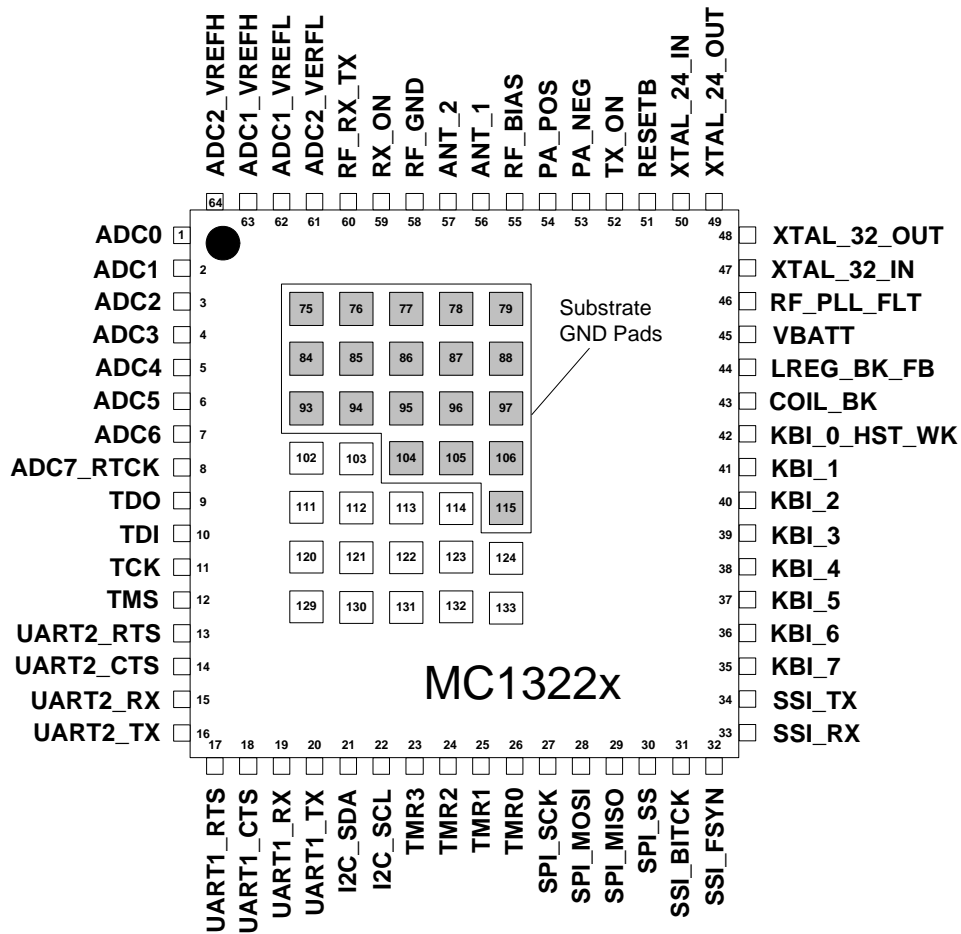
The ADC has the following features:

- 12 bit resolution
- Separate input voltage ranges: V_{ref_high} to V_{ref_low} . Maximum input of 3.6 VDC. Minimum input of 0.0 VDC
- Conversion rate has a minimum sample time of 20 μ s. Effective number of bits varies with sample rate.
- 8-Bit prescaler to provide the time base for the 32-bit timers
- Two independent channels, each with a 32-bit timer
- Simultaneous channel sampling or sequential channel sampling with dual ADCs
- On-board battery detect sample channel
- Primary ADC has 9 channels. 8 external channels plus battery detect
- Secondary ADC has 8 external channels
- Active channels for each ADC are programmable

- A maximum of 8 active monitors can generate a IRQ trigger
- A 8 deep FIFO for recording data (12-bit sample plus 4-bit channel tag)
- Interrupt requests can be generated by the channel compare values, end-of-scan sequence, out-of-range, FIFO status, and 32-bit timers
- Independent soft reset

Chapter 2 Pins and Connections

2.1 Pin Assignments and Connections



Notes:

1. Only active substrate bottom pads are shown.
2. Additional substrate pads (see mechanical drawings) are no connect and are for mechanical attach only.

Figure 2-1. MC1322x Pinout (Top View; active bottom pads shown)

2.2 Pin Definitions

Table 2-1 details the MC1322x pinout and functionality.

Table 2-1. Pin Function Description

Pin #	Pin Name	Type	Description ¹	Functionality
1	ADC0	Analog Input or Digital Input/Output	ADC analog input Channel 0 / GPIO30	ADC sample channel can be used by either ADC1 or ADC2.
2	ADC1	Analog Input or Digital Input/Output	ADC analog input Channel 1 / GPIO31	ADC sample channel can be used by either ADC1 or ADC2.
3	ADC2	Analog Input or Digital Input/Output	ADC analog input Channel 2 / GPIO32	ADC sample channel can be used by either ADC1 or ADC2.
4	ADC3	Analog Input or Digital Input/Output	ADC analog input Channel 3 / GPIO33	ADC sample channel can be used by either ADC1 or ADC2.
5	ADC4	Analog Input or Digital Input/Output	ADC analog input Channel 4 / GPIO34	ADC sample channel can be used by either ADC1 or ADC2.
6	ADC5	Analog Input or Digital Input/Output	ADC analog input Channel 5 / GPIO35	ADC sample channel can be used by either ADC1 or ADC2.
7	ADC6	Analog Input or Digital Input/Output	ADC analog input Channel 6 / GPIO36	ADC sample channel can be used by either ADC1 or ADC2.
8	ADC7_RTCK	Analog Input or Digital Input/Output	ADC analog input Channel 7 / ReTurn Clock / GPIO37	ADC sample channel can be used by either ADC1 or ADC2. Alternately, the signal returns TCK for JTAG to support adaptive clocking.
9	TDO	Digital Input/Output	JTAG Test Data Output / GPIO49	JTAG debug port serial data output.
10	TDI	Digital Input/Output	JTAG Test Data Input / GPIO48	JTAG debug port serial data input. This pin has an internal pullup resistor.
11	TCK	Digital Input/Output	JTAG Test Clock Input / GPIO47	JTAG debug port clock input.
12	TMS	Digital Input/Output	JTAG Test Mode Select Input / GPIO46	JTAG debug port test mode select input. This pin has an internal pullup resistor.
13	UART2_RTS	Digital Input/Output	UART2 Request to Send input / GPIO21	UART2 RTS control input.
14	UART2_CTS	Digital Input/Output	UART2 Clear to Send output / GPIO20	UART2 CTS control output.
15	UART2_RX	Digital Input/Output	UART2 RX data input / GPIO19	UART2 receive data input.
16	UART2_TX	Digital Input/Output	UART2 TX data output / GPIO18	UART2 transmit data output.

Pin #	Pin Name	Type	Description ¹	Functionality
17	UART1_RTS	Digital Input/Output	UART1 Request to Send input / GPIO17	UART1 RTS control input.
18	UART1_CTS	Digital Input/Output	UART1 Clear to Send output / GPIO16	UART1 CTS control output.
19	UART1_RX	Digital Input/Output	UART1 RX data input / GPIO15	UART1 receive data input.
20	UART1_TX	Digital Input/Output	UART1 TX data output / GPIO14	UART1 transmit data output.
21	I2C_SDA	Digital Input/Output	I ² C Bus data / GPIO13	I ² C bus signal SDA
22	I2C_SCL	Digital Input/Output	I ² C Bus clock / GPIO12	I ² C bus signal SCL
23	TMR3	Digital Input/Output	Timer 3 IO signal / GPIO11	Pin is used as counter output or counter input clock.
24	TMR2	Digital Input/Output	Timer 2 IO signal / GPIO10	Pin is used as counter output or counter input clock.
25	TMR1	Digital Input/Output	Timer 1 IO signal / GPIO9	Pin is used as counter output or counter input clock.
26	TMR0	Digital Input/Output	Timer 0 IO signal / GPIO8	Pin is used as counter output or counter input clock.
27	SPI_SCK	Digital Input/Output	SPI Port clock / GPIO7	SPI port clock.
28	SPI_MOSI	Digital Input/Output	SPI Port MOSI/ GPIO6	SPI Port Master Out Slave In (MOSI) data signal.
29	SPI_MISO	Digital Input/Output	SPI Port MISO / GPIO5	SPI Port Master In Slave Out (MISO) data signal.
30	SPI_SS	Digital Input/Output	SPI Port SS / GPIO4	SPI Port Slave Select (SS) signal.
31	SSI_BITCK	Digital Input/Output	SSI Bit Clock / GPIO3	SSI serial TX/RX clock and is bi-directional.
32	SSI_FSYN	Digital Input/Output	SSI Frame Sync / GPIO2	SSI frame sync for data (RX or TX) and is bi-directional.
33	SSI_RX	Digital Input/Output	SSI RX data input / GPIO1	SSI serial RX data input.
34	SSI_TX	Digital Input/Output	SSI TX data output / GPIO0	SSI serial TX data output.
35	KBI_7	Digital Input/Output	Keyboard Interface Bit 7 / GPIO29	Asynchronous interrupt input.
36	KBI_6	Digital Input/Output	Keyboard Interface Bit 6 / GPIO28	Asynchronous interrupt input.
37	KBI_5	Digital Input/Output	Keyboard Interface Bit 5 / GPIO27	Asynchronous interrupt input.

Pins and Connections

Pin #	Pin Name	Type	Description ¹	Functionality
38	KBI_4	Digital Input/Output	Keyboard Interface Bit 4 / GPIO26	Asynchronous interrupt input.
39	KBI_3	Digital Input/Output	Keyboard Interface Bit 3 / GPIO25	Used as output for keyboard interface.
40	KBI_2	Digital Input/Output	Keyboard Interface Bit 2 / GPIO24	Used as output for keyboard interface.
41	KBI_1	Digital Input/Output	Keyboard Interface Bit 1 / GPIO23	Used as output for keyboard interface.
42	KBI_0_HST_WK	Digital Input/Output	Keyboard Interface Bit 0 / HoST wake-up output / GPIO22	Used as output for keyboard interface / Alternative function as a wake-up output (based on a timer) to external device.
43	COIL_BK	Power Switch Output	Buck converter coil drive output	Onboard buck converter connection to external coil, driven by onboard MOSFET.
44	LREG_BK_FB	Power Input	Voltage input to onboard regulators, buck regulator feedback voltage	<ul style="list-style-type: none"> When using onboard buck converter, connect to load side of coil. When not using buck converter, connect to VBATT.
45	VBATT	Power Input	High side supply voltage to buck regulator switching MOSFET and IO buffers	Connect to battery.
46	RF_PLL_FLT	Analog Voltage	PLL filter connection	<ul style="list-style-type: none"> Connection for PLL filter (Type 2, 2nd Order) when using primary crystal with frequency other than 24 MHz (13-26 MHz). No Connect for 24 MHz crystal.
47	XTAL_32_IN	Analog Input	Optional 32.768 kHz crystal oscillator input	Connect to 32.768 kHz crystal
48	XTAL_32_OUT	Analog Output	Optional 32.768 kHz crystal oscillator output	Connect to 32.768 kHz crystal
49	XTAL_24_OUT	Analog Output	Primary 24 MHz crystal oscillator output	<ul style="list-style-type: none"> Connect to 13-26 MHz crystal (24 MHz default). No load capacitor required Do not load with any capacitance.
50	XTAL_24_IN	Analog Input	Primary 24 MHz crystal oscillator input	<ul style="list-style-type: none"> Connect to 13-26 MHz crystal (24 MHz default). No load capacitor required Do not load with any capacitance.
51	RESETB	Digital Input	System reset input	Active low, asynchronous reset
52	TX_ON	Digital Input/Output	Control output for external RF component / GPIO44	Programmable control pin
53	PA_NEG	RF Output	RF power amplifier (PA) output negative	<ul style="list-style-type: none"> Open drain. Must be connected to RF_BIAS through a bias network. Only used for external dual port operation. Do not use for single port operation. No Connect.

Pin #	Pin Name	Type	Description ¹	Functionality
54	PA_POS	RF Output	RF power amplifier (PA) output positive	<ul style="list-style-type: none"> Open drain. Must be connected to RF_BIAS through a bias network. Only used for external dual port operation. Do not use for single port operation. No Connect.
55	RF_BIAS	Analog Power Output	Analog VDD regulator output	When using dual port operation, tie to PA_POS and PA_NEG through bias networks.
56	ANT_1	Digital input / Output	Control output for external RF component / GPIO42	Programmable control pin.
57	ANT_2	Digital input / Output	Control output for external RF component / GPIO43	Programmable control pin.
58	RF_GND	Power Input	RF ground.	Connect to ground VSS.
59	RX_ON	Digital input / Output	Control output for external RF component / GPIO45	Programmable control pin.
60	RF_RX_TX	RF Input/Output	RF single-ended, single port input and output	<ul style="list-style-type: none"> Interfaces to onboard balun. 50 Ω impedance Full bidirectional port with onboard T/R switch. Used as single-ended RF input port for dual port operation with PA_NEG and PA_POS PA outputs.
61	ADC2_VREFL	Analog Input or Digital Input / Output	Low reference voltage for ADC2 / GPIO39	VREFL for ADC2.
62	ADC1_VREFL	Analog Input or Digital Input / Output	Low reference voltage for ADC1 / GPIO41	VREFL for ADC1.
63	ADC1_VREFH	Analog Input or Digital Input / Output	High reference voltage for ADC1 / GPIO40	VREFH for ADC1.
64	ADC2_VREFH	Analog Input or Digital Input / Output	Low reference voltage for ADC2 / GPIO38	VREFH for ADC2.
75-79	VSS	Power input	External package GND pads. Common VSS.	Connect to ground.
84-88	VSS	Power input	External package GND pads. Common VSS.	Connect to ground.
93-97	VSS	Power input	External package GND pads. Common VSS.	Connect to ground.
102	MDO01	Digital Input/Output	Message Data Out Bit 1 output / GPIO52	Nexus debug port message data output Bit 1.
103	MDO00	Digital Input/Output	Message Data Out Bit 0 output / GPIO51	Nexus debug port message data output Bit 0.

Pins and Connections

Pin #	Pin Name	Type	Description ¹	Functionality
104-106	VSS	Power input	External package GND pads. Common VSS.	Connect to ground.
111	MDO03	Digital Input/Output	Message Data Out Bit 3 output / GPIO54	Nexus debug port message data output Bit 3.
112	MDO02	Digital Input/Output	Message Data Out Bit 2 output / GPIO53	Nexus debug port message data output Bit 2.
113	MSEO1_B	Digital Input/Output	Message Start / End Out Bit 1 output / GPIO60	Nexus debug port message start / end output Bit 1. Signal is active low.
114	MSEO0_B	Digital Input/Output	Message Start / End Out Bit 0 output / GPIO59	Nexus debug port message start / end output Bit 0. Signal is active low.
115	VSS	Power input	External package GND pads. Common VSS.	Connect to ground.
120	MDO05	Digital Input/Output	Message Data Out Bit 5 output / GPIO56	Nexus debug port message data output Bit 5.
121	MDO04	Digital Input/Output	Message Data Out Bit 4 output / GPIO55	Nexus debug port message data output Bit 4.
122	RDY_B	Digital Input/Output	Ready output / GPIO61	Nexus debug port ready output. Signal is active low.
123	EVTO_B	Digital Input/Output	Event Out output / GPIO62	Nexus debug port event out output. Signal is active low.
124	DIG_REG	Digital Power Output	Digital core logic VDD supply.	1.2 VDC internally regulated VDD supply to digital logic core. <u>No Connect.</u> For test only
129	MDO07	Digital Input/Output	Message Data Out Bit 7 output / GPIO58	Nexus debug port message data output Bit 7.
130	MDO06	Digital Input/Output	Message Data Out Bit 6 output / GPIO57	Nexus debug port message data output Bit 6.
131	MCKO	Digital Input/Output	Message Clock Out output / GPIO50	Nexus debug port message clock output.
132	EVTI_B	Digital Input/Output	Event In input / GPIO63	Nexus debug port event in input. Signal is active low.
133	NVM_REG	NVM Power Output	FLASH (NVM) VDD supply.	VDD supply to FLASH. <u>Typically No Connect.</u> Can be connected to VDD when regulated 1.8Vdc mode is used.
65-74, 80-83, 89-92, 98-101, 107-110, 116-119, 125-128, 134-145	NC		No Connect	These pads are provided for extra mechanical attach strength to meet demanding requirements of drop tests.

¹ Pins described as GPIO have an alternative general purpose IO function.

2.3 Hardware Development Interface Interconnects

The MC1322x supports two development hardware interfaces.

2.3.1 ARM JTAG Interface Connector

The MC1322x supports connection to a subset of the ARM JTAG connector. The JTAG hardware interface is a standard 0.1 inch spacing, 20-pin header. [Table 2-2](#) shows the device pins that are connected to the associated JTAG header pinouts if the JTAG connector is used.

Table 2-2. ARM JTAG 20-Pin Connector Assignments

Name ¹	Pin #	Pin #	Name
VBATT	1	2	VBATT
NC ²	3	4	GND
TDI	5	6	GND
TMS	7	8	GND
TCK	9	10	GND
RTCK	11	12	GND
TDO	13	14	GND
VBATT (pullup) ³	15	16	GND
NC	17	18	GND
NC	19	20	GND

¹ NC = No Connect.

² MC1322x does not support separate JTAG reset TRST.

³ VBATT through a 100k- Ω pullup.

2.3.2 Nexus Mictor Interface Connector

The MC1322x also supports connection to a subset of the defined Nexus Mictor connector. The hardware interface is a 38-pin Mictor target connector. [Table 2-3](#) shows the device pins that are connected to the associated Mictor pin outs if the Mictor connector is used.

Table 2-3. Nexus 38-Pin Mictor Connector Assignments

Name ¹	Pin #	Pin #	Name
NC	1	2	NC
NC	3	4	NC
NC	5	6	RTCK
NC	7	8	NC
VBATT (pullup) ²	9	10	EVTL_B
TDO	11	12	VBATT ³

Table 2-3. Nexus 38-Pin Mictor Connector Assignments (continued)

Name ¹	Pin #	Pin #	Name
NC	13	14	RDY_B
TCK	15	16	MDO07
TMS	17	18	MDO06
TDI	19	20	MDO05
VBATT (pullup) ⁴	21	22	MDO04
NC	23	24	MDO03
NC	25	26	MDO02
NC	27	28	MDO01
NC	29	30	MDO00
NC	31	32	EVTO_B
NC	33	34	MCKO
NC	35	36	MSEO1_B
NC	37	38	MSEO0_B

¹ NC means No Connect.

² VBATT through a 100k- Ω pullup.

³ VBATT isolated by a 1k- Ω resistor.

⁴ VBATT through a 100k- Ω pullup.

Chapter 3

MC1322x System Considerations

3.1 Introduction

The MC1322x is the embodiment of a IEEE 802.15.4 node in a single Platform in a Package (PiP) which can provide solutions to proprietary nets, IEEE 802.15.4 MAC-compatible nets, or full ZigBee-compatible nets. This chapter presents information on application and operation of the node from a system level. The areas considered here are also covered in greater detail in other sections of the book.

3.2 Power Connections and Design

The MC1322x can be used in three different power supply configurations that are discussed in the following sections:

- Varying VDD voltage from 2.0 - 3.6 VDC using onboard regulation
- Varying VDD voltage from 2.1 - 3.6 VDC with optional onboard buck regulator
- Fixed regulated VDD voltage at 1.8 VDC.

3.2.1 Power Pin Descriptions

The power pin connections for the PiP are shown in [Table 3-1](#).

Table 3-1. Power Pin Descriptions

Pin #	Pin Name	Type	Description	Functionality
45	VBATT	Power Input	High side supply voltage to: <ul style="list-style-type: none"> • Digital logic regulators • Oscillator regulators • Buck regulator logic and switching MOSFETs • KBI pads • IO buffers 	Connect to battery.
43	COIL_BK	Power Switch Output	Buck converter coil drive output	Onboard buck converter connection to external coil, driven by onboard MOSFETs.
44	LREG_BK_FB	Power Input	Voltage input to onboard regulators, buck regulator feedback voltage, analog regulators, and NVM	<ul style="list-style-type: none"> • When using onboard buck converter, connect to load side of coil. • When not using buck converter, connect to VBATT.

Table 3-1. Power Pin Descriptions

Pin #	Pin Name	Type	Description	Functionality
55	RF_BIAS	Analog Power Output	Analog VDD regulator output. Use only to supply analog voltage to RF PA outputs.	When using dual port RF operation, tie to PA_POS and PA_NEG through bias networks.
58	RF_GND	Power Input	RF ground.	Connect to ground (VSS).
75-79	VSS	Power input	External package GND pads. Common VSS.	Connect to ground.
84-88	VSS	Power input	External package GND pads. Common VSS.	Connect to ground.
93-97	VSS	Power input	External package GND pads. Common VSS.	Connect to ground.
104-106	VSS	Power input	External package GND pads. Common VSS.	Connect to ground.
115	VSS	Power input	External package GND pads. Common VSS.	Connect to ground.
124	DIG_REG	Digital Power Output	Digital core logic VDD supply.	1.2 VDC internally regulated VDD supply to digital logic core. <u>No Connect</u> . For test purposes only.
133	NVM_REG	NVM Power Output	FLASH (NVM) VDD supply.	VDD supply to FLASH. <u>Typically No Connect</u> . Must be connected to VDD when regulated 1.8Vdc mode is used.

When designing power to the MC1322x PiP, the following points need to be considered:

- The PiP LGA package has a single common ground flag (VSS) with multiple solder pads. The multiple pad approach provides better solderability.

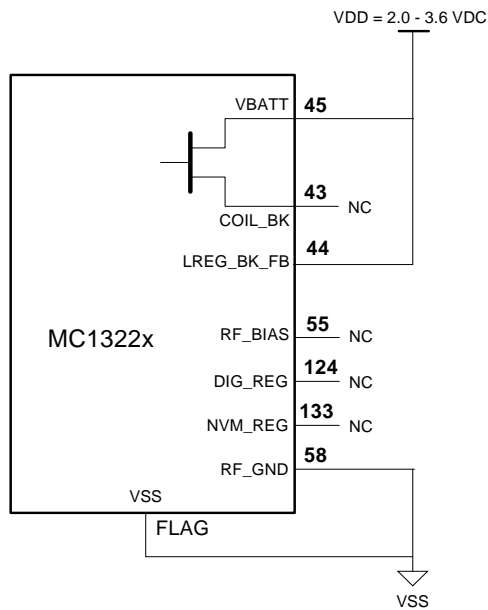
NOTE

Extra pads are provided on the package bottom simply for greater mechanical attach strength. These are “No Connect” and need not be connected to ground.

- The VBATT pin provides the high side supply to the P-channel MOSFET for the optional buck regulator and to the IO pads.
- For logic level compatibility between the MC1322x and external logic devices, VBATT, must be connected to a common logic supply with the external devices. The MC1322x IO are not 5 VDC tolerant.
- LREG_BK_FB feeds the common supply to the analog and NVM circuitry regulators. Bypass capacitance for voltage regulators is provided onboard the PiP.

3.2.2 Varying VDD from 2.0 - 3.6 VDC Using Onboard Regulation

The most common power connection is shown in [Figure 3-1](#). This connection allows VDD to vary from 2.0 - 3.6 VDC and uses a minimum of components.



NOTE:
RF_BIAS is used with dual-port RF operation.

Figure 3-1. Varying VDD from 2.0 - 3.6 VDC Using Onboard Regulation

3.2.3 Varying VDD from 2.1 - 3.6 VDC with Optional Onboard Buck Regulator

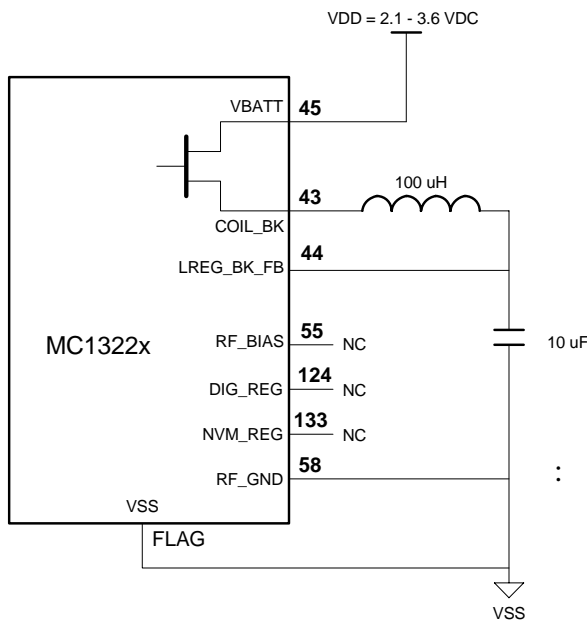


Figure 3-2. Varying VDD from 2.1 - 3.6 VDC Using Buck Regulator

For some battery-based applications, an optional, onboard buck regulator is provided to help maximize battery life. Figure 3-2 shows the configuration of the buck regulator. Onboard MOSFETs are used to switch an external inductor and capacitor as a synchronous-rectifier regulator when enabled. The buck regulator drops the higher battery voltage to about 2.0Vdc that is applied to the onboard linear regulators. This allows lower net current from the battery to maximize the life of the battery.

3.2.3.1 Buck Regulator Hardware Considerations

For use of the buck regulator consider the following:

- The switch-mode regulator is a synchronous rectifier design. As a consequence, an external clamping Schottky diode is not required.
- The maximum “ON”-resistance of the P-channel MOSFET switch is 4 ohms. This impedance impacts maximum current load for the switcher output.
- External inductor
 - 100 μ H \pm 20% inductance
 - 0.4-0.5 ohm maximum series resistance
 - Suggested part TDK #SLF6028T-101MR42
- 10 μ F ceramic capacitor

3.2.3.2 Buck Regulator Usage/Software Considerations

3.2.3.2.1 Buck Control and Status Mechanisms

Control of the buck regulator is managed in the CRM (see [Chapter 5, “System Management \(Including CRM\)”](#)) primarily through the VREG_CNTL register (see [Section 5.9.18, “Voltage Regulator Control \(VREG_CNTL\)”](#))

- The buck design switch frequency is nominally 1.6 MHz. This controlled by the BUCK_CLKDIV[3:0] field of the VREG_CNTL register. If the normal 24 MHz crystal is used, the default value for this field can be used; if not, it must be programmed as required.
- Enabling the buck is accomplished by -
 - Set the BUCK_EN and BUCK_SYNC_REC_EN bits of the VREG_CNTL register
 - This can be done simultaneously; the default condition are both disabled
 - The buck ready status can be observed via the VREG_BUCK_RDY status bit in the CRM STATUS register (see [Section 5.9.7, “Status \(STATUS\)”](#))
 - Proper start-up procedure must be followed
- Enabling the buck regulator from a cold start (POR) also requires buck enable in the PWR_SOURCE[1:0] field of the SYS_CNTL register (see [Section 5.9.1, “System Control \(SYS_CNTL\)”](#)). Default is NOT buck mode.
- The voltage output from the buck (supplied back to the series regulators via pin LREG_BK_FB) drives only the 1.8V NVM regulator and the 1.5V Analog regulator for the radio/modem.
 - These regulators get enabled by the VREG_1P5V_EN[1:0] and the VREG_1P8V_EN fields of the VREG_CNTL register
 - Enable is possible after warmup of the switcher
 - The regulator ready status can be observed via the VREG_1P5V_RDY and VREG_1P8V_RDY status bits in the CRM STATUS register
 - Proper start-up procedure must be followed
- After the battery voltage drops to $\leq 2.5V$, the buck should be put into “bypass” mode
 - At 2.5V and below, the switcher efficiency drops and the switcher function is shut-off; the P-channel transistor then is fully turned-on and the P-channel and inductor become a simple series circuit to supply power to the series regulators
 - The application software must monitor the VBATT voltage via the ADC and cause the switch-over
 - Buck bypass is enabled by the BUCK_BYPASS_EN bit in the VREG_CNTL register

3.2.3.2.2 Buck Control Procedures

Use of the buck regulator must be managed properly to get best efficiency and allow proper operation with varying battery voltage

- Device start-up from reset/POR - The circuit configuration of [Figure 3-2](#) presents a conundrum for start-up: the NVM must be powered to allow the boot process to read the NVM, load RAM, and start executing from RAM. However, the default for power supply configuration is VBATT

selected (PWR_SOURCE[1:0] = 0b00), where LREG_BK_FB is normally tied directly to the NVM regulator input. This allows the regulator to be enabled to access the NVM.

With use of the buck, the P-channel MOSFET switch must be turned-on (BUCK_BYPASS_EN bit set) BEFORE the NVM can be ceased (the NVM regulator input has to be enabled through the switch). THE ROM BOOT CODE PROVIDES THIS FUNCTION.

NOTE

The user application code need not provide this function, as it resides in ROM-based boot code. The above information is provided for user understanding.

- Buck start-up after Boot - Once the device has booted (executing from RAM), the application can start-up the buck regulator (the boot code has disabled the bypass function). The following procedure must be followed:
 - Set PWR_SOURCE[1:0] field of SYS_CNTL Register = 2'b01 (enable buck mode)
 - Write the VREG_CNTL Register:
 - BUCK_CLKDIV[3:0] = 0xF (default), for 24MHz, otherwise as required by the reference oscillator
 - Set BUCK_SYNC_REC_EN, enable sync rectifier
 - Set BUCK_EN, enable buck regulator
 - Poll CRM STATUS Register: wait for VREG_BUCK_RDY status to be set (indicates buck is ready)
 - Write the VREG_CNTL Register:
 - These conditions remain - BUCK_CLKDIV[3:0] = 0xF, BUCK_SYNC_REC_EN = 1, BUCK_EN = 1
 - Write VREG_1P5V_EN[1:0] as required to enable radio (typically 2'b11)
 - Write VREG_1P5V_SEL[1:0] as required to set radio current (recommended 2'b11)
 - Set VREG_1P8V_EN as required to enable NVM regulator
 - Poll CRM STATUS Register: wait for VREG_1P5V_RDY and/or VREG_1P8V_RDY status to be set (indicates regulated voltage is ready)
 - Continue with application
- Monitor battery voltage (VBATT) for 2.5V and enable bypass - The application code must periodically sample VBATT via ADC1. Once VBATT has dropped below 2.5V, it is recommended the buck mode be changed to bypass (the switcher disabled and bypass mode enabled).

NOTE

A suggested means to enable ADC1 sampling is use of the RTC interrupt request on a periodic basis.

- Write the VREG_CNTL Register:
 - Clear BUCK_SYNC_REC_EN = 0, BUCK_EN = 0 to disable switcher
 - Write VREG_1P5V_EN[1:0] to disable radio (2'b00)

- Clear VREG_1P8V_EN to disable NVM regulator
- Set BUCK_BYPASS_EN to enable bypass transistor
- Poll CRM STATUS Register: wait for VREG_BUCK_RDY status to be set (indicates buck bypass is ready)
- Write the VREG_CNTL Register:
 - These conditions remain - BUCK_SYNC_REC_EN = 0, BUCK_EN = 0
 - Write VREG_1P5V_EN[1:0] as required to enable radio (typically 2'b11)
 - Write VREG_1P5V_SEL[1:0] as required to set radio current (recommended 2'b11)
 - Set VREG_1P8V_EN as required to enable NVM regulator
- Poll CRM STATUS Register: wait for VREG_1P5V_RDY and/or VREG_1P8V_RDY status to be set (indicates regulated voltage is ready)
- Continue with application

3.2.3.2.3 Buck Efficiency

The buck regulator can be used with a VBATT voltage range of 3.6 - 2.5 Vdc. [Figure 3-3](#) shows typical buck efficiency versus load current at VBATT = 2.5 V.

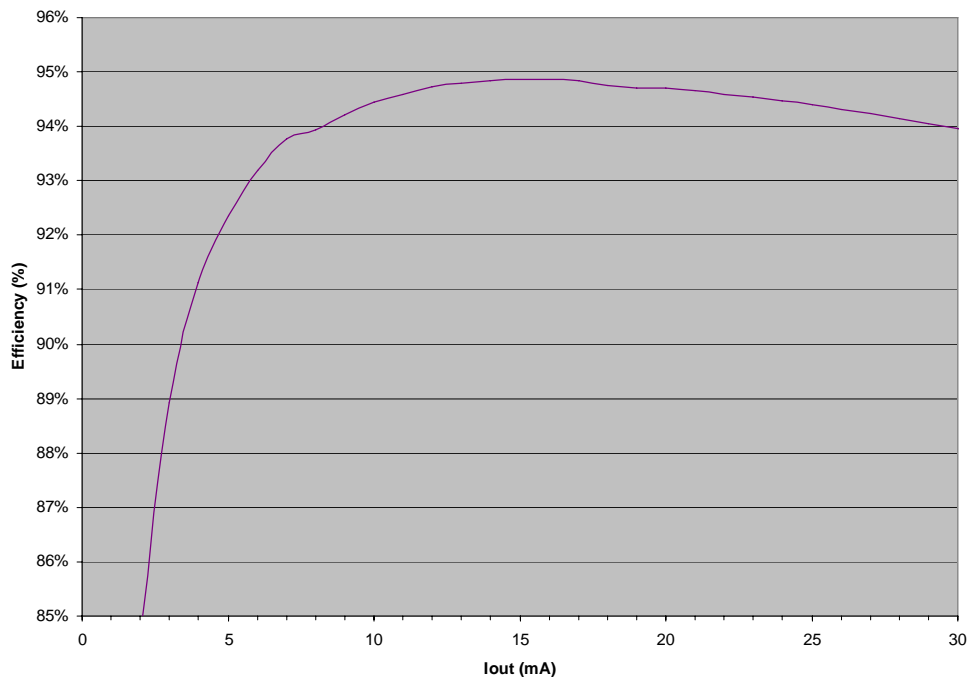
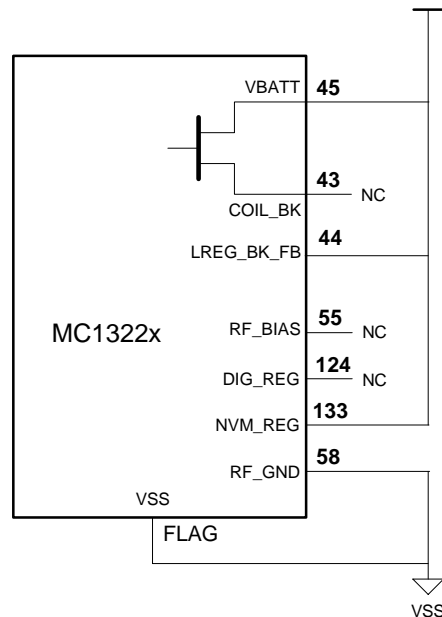


Figure 3-3. Typical Buck Efficiency vs. Load Current (25°C and VBATT = 2.5V)

3.2.4 Fixed Regulated 1.8 VDC VDD Voltage

The very lowest power consumption connection is shown in Figure 3-4. This connection requires that VDD be an external fully regulated 1.8 VDC. The NVM_REG pin is connected in this configuration to supply a full 1.8 V to the FLASH memory; normally an onboard regulator supplies the FLASH supply. The NVM regulator must stay disabled.



NOTE:
RF_BIAS is used with dual-port RF operation.

Figure 3-4. Fixed Regulated 1.8 VDC VDD Voltage

3.3 System Reset and Low Power GPIO Signal Connections

The MC1322x has three basic low power modes:

- Reset (off) - Master reset input RESETB is held asserted low and completely disables the device. Current is the absolute lowest with less than 0.3 μA dissipated. There is no onboard resistor to contribute to this current.
- Hibernate - This state shuts down all clocks with the exception a low power onboard oscillator or an optional 32.768 kHz oscillator for a wakeup timer. Depending on program options current dissipation is can be as low as 1 to 5 μA .
- Doze - This state shuts down most clocks but allows the reference oscillator to run. Depending on program options current dissipation is can be as low as 20 to 28 μA .

The use of the RESETB and GPIO signals during low power modes are discussed in this section.

3.3.1 GPIO Pin State During Reset, Default, and Low Power Modes

Table 3-2 shows the state of the GPIO signals versus mode of the device.

Table 3-2. GPIO Pin States vs. Mode

Item	Category	Pin Function	Reset Active	Default (Upon exiting reset, before Boot)	Hibernate & Doze
1	Standard	Standard GPIO	Unpowered	1) GPIO as inputs w / pull-downs enabled 2) ADC6:ADC0 have pad keepers enabled	Unpowered
2	Exception	I2C_SDA & I2C_SCL	Unpowered	GPIO as input w / pull-ups enabled	Unpowered
3	Exception	ADC1_VREFH & ADC1_VREFL	Unpowered	Analog Inputs (no pull-up/pull-down)	Unpowered
4	Exception	ADC2_VREFH & ADC2_VREFL	Unpowered	1) ADC2_VREFH - GPIO as input w / pull-up enabled 2) ADC2_VREFL - GPIO as input w / pull-down enabled ¹ 3) Pad keepers enabled for both pins	Unpowered
5	Exception	ADC7_RTC	Unpowered	JTAG RTCK output	Unpowered
6	Exception	JTAG	Unpowered	1) JTAG Inputs - TDI, TMS, & TCK w / pull-ups enabled 2) JTAG Output - TDO	Unpowered
7	Exception	Nexus	Unpowered	All Nexus signals are outputs with the exception of the input EVT1_B w / pull-up enabled	Unpowered
8	Exception	KBI	Powered (revert to default CRM KBI control) 1) KBI_3:KBI_0 - outputs high 2) KBI_7:KBI_4 - inputs w / pulldowns enabled	1) KBI_3:KBI_0 - outputs high 2) KBI_7:KBI_4 - inputs w / put-downs enabled	Powered (revert to CRM KBI control)

¹ ADC2_VREFL exits reset with pull-up enabled, but boot code changes to pull-down enabled for “clear FLASH” test. See [Section 3.9.5, “FLASH Erase or Recovery Mode”](#)

Reset and low power operation must consider the following points:

- With RESETB active or in Hibernate or Doze, all GPIO except the KBI signals are unpowered (disconnected from VBATT).
- It is best practice not to drive a GPIO high or tie a GPIO high through a resistor when the GPIO is unpowered.

NOTE

Driving a GPIO high when not powered can forward bias the ESD diode and cause excess current. No damage will be done to the device, but excess leakage current can be high and defeat the desire for very low current while in a low power mode.

- The KBI signals are powered during reset, Hibernate, and Doze to disable external circuitry and/or to allow an external asynchronous interrupt to wakeup the device from Doze or Hibernate modes.

3.3.2 Using GPIO in Low Power Modes

The MC1322x was designed to have extremely low current during low power modes. Because of the large number of GPIO on the device, the GPIO are generally disconnected from VBATT when a low power mode is enabled, with the exception of the KBI signals as described in [Section 3.3.1, “GPIO Pin State During Reset, Default, and Low Power Modes”](#).

It is envisioned that the MC1322x will be generally used in two basic configurations:

- As a peripheral communication channel for a primary device such as a cell phone.
- Standalone (perhaps with its own peripheral functions), where the MC1322x is the primary device.

3.3.2.1 Peripheral Communication Channel

In a battery operated device such as a cell phone, the peripheral function must use as little power as possible, especially when disabled and not in use. For this scenario, the reset condition is a desired low power state because of absolute lowest current. [Figure 3-5](#) shows a simple interconnect diagram for this application.

- The MC1322x RESETB should be driven from a main MCU GPIO that is always active. When RESETB is active all MC1322x GPIO (except KBI) will be unpowered and need not be tied to any voltage.
- It is best that other peripheral devices not be tied to the MC1322x.
- The control and data interface between the devices would typically be a SPI, SSI, UART, or other serial interface.
 - MC1322x normal outputs (when reset active) revert to no power and will not drive the main MCU inputs. The main MCU may either provide a pull-down under this condition or be over-programmed as outputs in the low condition to hold the MC1322x signals low.
 - The MC1322x inputs should be driven low by the main MCU outputs.
 - If the I2C is used as the data interface, any required external pull-up resistors must be disabled when the RESETB is active.
- The disadvantage to using reset as a low power condition is the recovery time of the MC1322x. The device must recover from reset, start its clocks, and load its RAM to be able to be active.

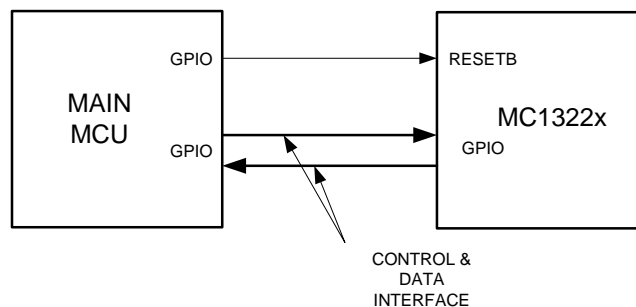


Figure 3-5. MC1322x as a Peripheral Function

3.3.2.2 MC1322x as Primary Device

The more common scenario is where the MC1322x is the primary device and may have peripheral devices such as sensors tied to it. In this situation reset is not typically used as a low power condition. Figure 3-6 shows examples of how external connections can be controlled.

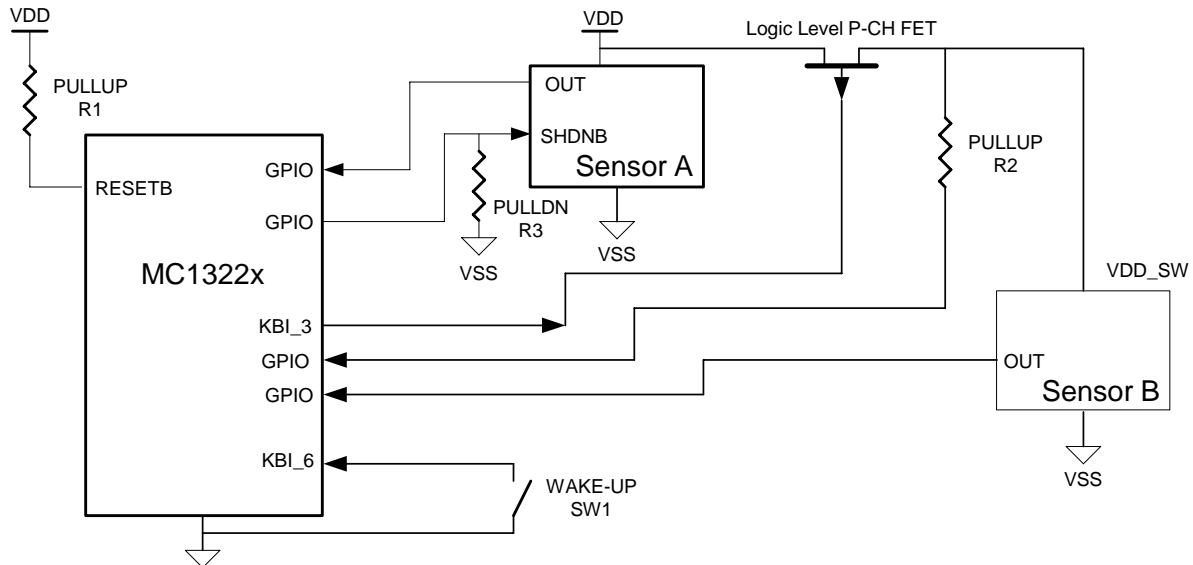


Figure 3-6. MC1322x as a Primary Device

- A pull-up resistor should be connected to RESETB.
- If the peripheral device has a shutdown input (active low) such as Sensor A, then a standard GPIO can be used to disable the device. The external pull-down is required to be sure the SHDNB input is pulled low in the reset or low power state.
- Other external connections such as pull-up resistors or a device without a shutdown option (Sensor B) can be controlled by an external P-channel MOSFET. A logic level gate voltage device is recommended, and the gate can be driven from KBI_3 through KBI_0. These outputs retain their state under low power states and default to a high condition to keep the FET off during reset (and Hibernate and Doze if desired).
- For a Hibernate or Doze wakeup option, an external switch can be connected to KBI_4 through KBI_7.

3.4 External Clock Connections

The MC1322x uses clock sources for two primary functions:

- Reference clock oscillator - all active clocks for the radio, CPU, and peripherals are derived from the reference oscillator.
- Real Time Clock (RTC)- wakeup timer for Hibernate or Doze modes. The RTC input clock can be derived from three sources:
 - 2 kHz internal ring oscillator
 - Reference oscillator divided by 128 (nominally 24MHz/128)

— Optional external 32 kHz crystal oscillator

The reference clock typically uses an external 24 MHz crystal, although it can be driven from an external source. The reference oscillator can also use a crystal ranging from 13 to 26 MHz in place of the 24 MHz standard, but there is an onboard PLL that must be used with the non-standard crystal.

A second totally optional 32.768 KHz crystal oscillator is available for the RTC. Again an external crystal is required. The advantage of the 32.768 KHz oscillator is that it is very low power while retaining excellent long term accuracy.

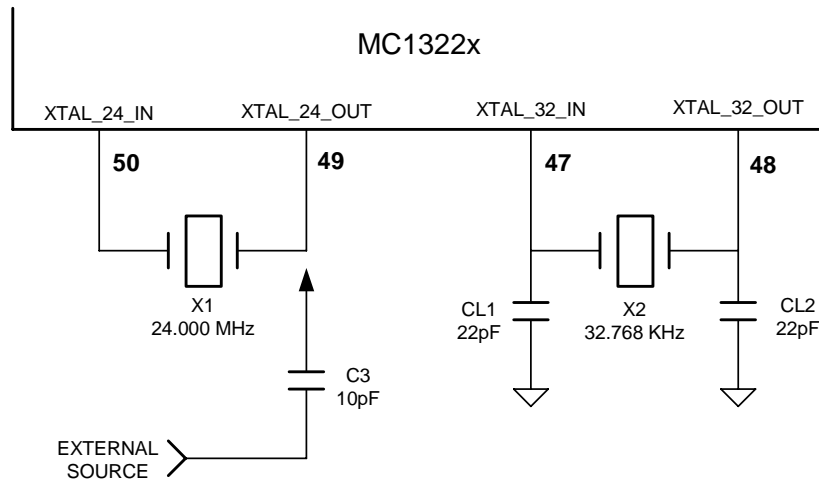


Figure 3-7. MC1322x Clock Connections

The MC1322x external clock connections are shown in [Figure 3-7](#).

3.5 Reference Oscillator

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

3.5.1 Description

The MC1322x has a built-in Pierce reference crystal oscillator. It uses an off-chip crystal with frequency selection range from 13 MHz to 26 MHz with 24 MHz the default frequency. The oscillator is powered by a separate on-chip regulator and all the load capacitors are internal and programmable.

The reference crystal oscillator is able to drive a variety of crystals with a typical load of 9pF. The integrated load capacitors on each leg include a separately enabled course tune 4pF capacitor, a coarse tune capacitor array of 1, 2, 4 and 8pF (4 program bits), and fine tune capacitors of 5pF in 160 fF steps (5 program bits). The crystal load capacitance on each crystal leg is a total of $24\text{pF} + C_{\text{stray}}$.

[Figure 3-8](#) shows the user oscillator model.

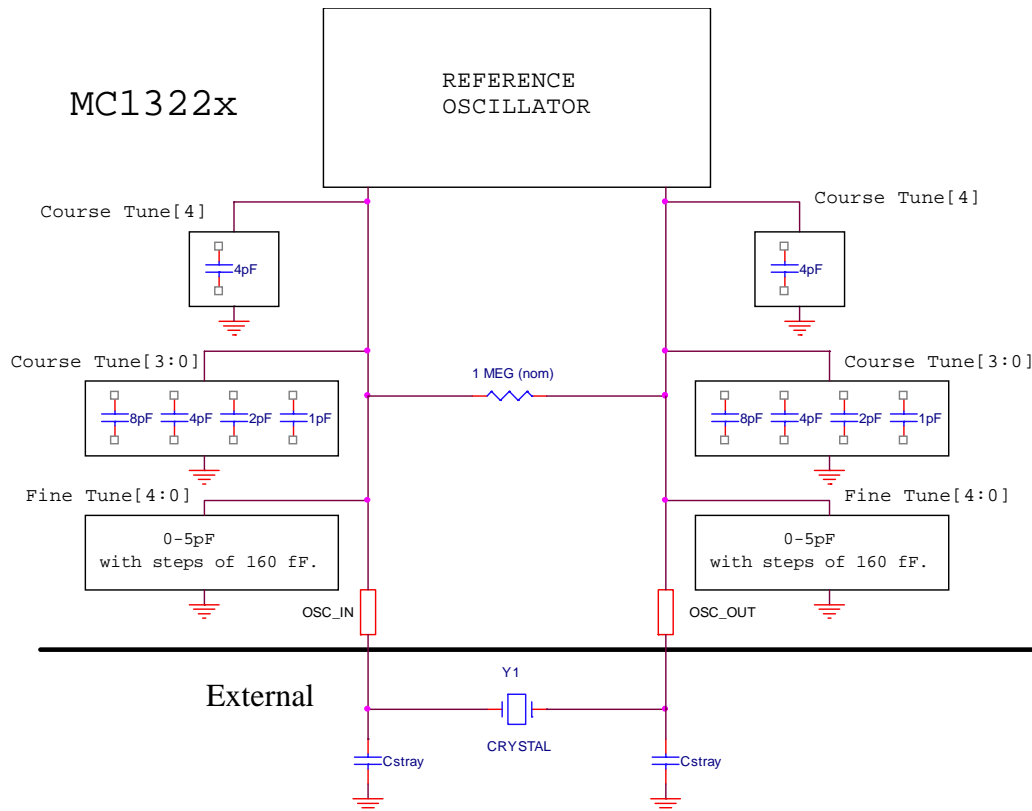


Figure 3-8. MC1322x Reference Oscillator Model

The model shows that with the onboard capacitor array, external load capacitors for the oscillator crystal are not necessary. Stray capacitance (sum of pad, package, and trace capacitance) is typically ~1.5pF.

The load capacitance C_L to the crystal is symmetric and equals $1/2 (C_{\text{course}} + C_{\text{fine}} + C_{\text{stray}})$. Target load capacitance is 9pF.

An onboard feedback resistor (1 Megohm, nominal) provides dc-biasing for the oscillator amplifier.

3.5.2 Reference Oscillator 24 MHz Crystal Specifications

The recommended 24 MHz crystal specifications are shown in [Table 3-3](#).

Table 3-3. Recommended 24 MHz Crystal Specifications

Parameter Name	Minimum	Typical	Maximum	Units
Crystal frequency		24.000		MHz
Operating temperature range				°C
Industrial	-40		+85	
Commercial	0		+70	
Extended	-40		+105	
Frequency tolerance @ 25 °C ± 3 °C			± 10	ppm
Frequency stability with temperature (full temp range, reference 25 °C)			± 16	ppm
Load capacitance		9		pF
Equivalent series resistance (ESR)			40	Ω
Shunt capacitance		1.3		pF
Tolerated drive level	100			μW
Mode of vibration	AT-cut / fundamental			

3.5.3 Crystal Trimming

The MC1322x reference oscillator frequency can be trimmed (via changing the load capacitance) through programming the CRM register Reference XTAL Control (XTAL_CNTL) as described in [Section 5.9.16, “Reference XTAL Control \(XTAL_CNTL\)”](#). The Reference XTAL Control register of the CRM has two control fields that control the trimming.

- XTAL_CTUNE[4:0] - selects course load capacitance
 - XTAL_CTUNE[4] - 4 pF load to crystal
 - XTAL_CTUNE[3:0] - values 0-F select load capacitance equal to programmed number, i.e., value 7 selects 7 pF
- XTAL_FTUNE[4:0] - selects fine tuning for load capacitance; selects from 0-5 pF in 32 steps of approximately 156 fF

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

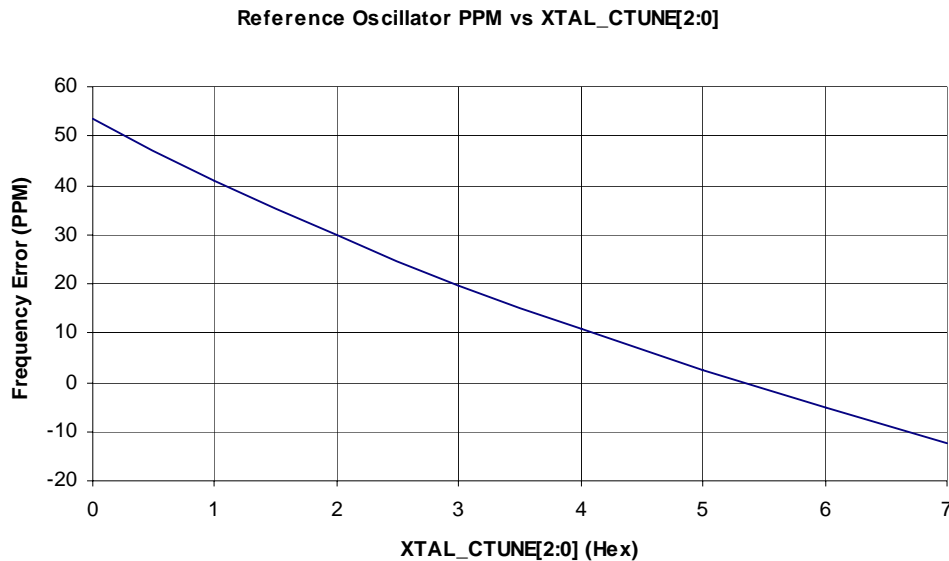


Figure 3-9. MC1322x Reference Oscillator Variation vs. XTAL_CTUNE[2:0] (XTAL_CTUNE[4] = 1, XTAL_FTUNE[4:0] = 0x10, XTAL C_L = 9pF)

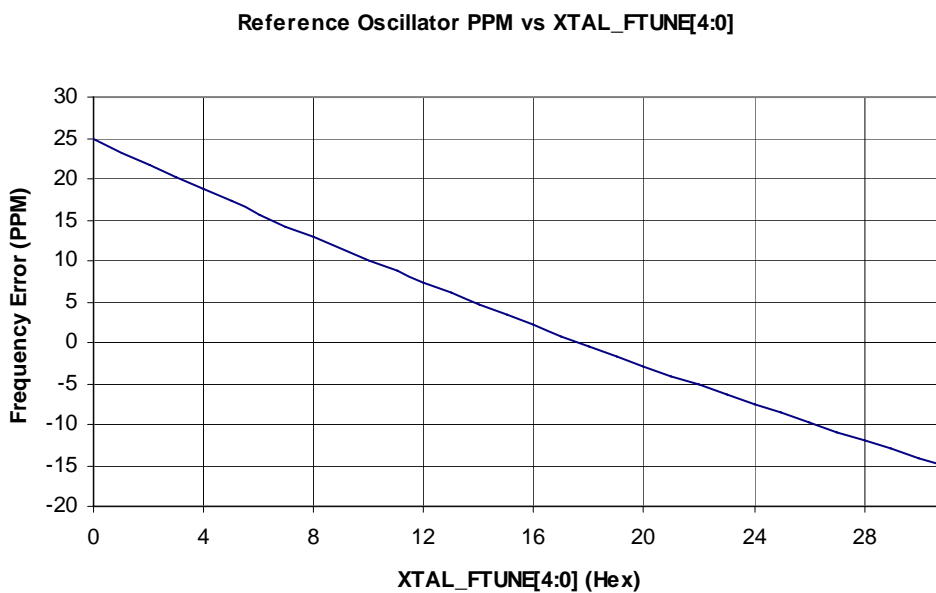


Figure 3-10. MC1322x Reference Oscillator Variation vs. XTAL_CTUNE[2:0] (XTAL_CTUNE[4] = 1, XTAL_CTUNE[2:0] = 0x5, XTAL C_L = 9pF)

Figure 3-9 and Figure 3-10 illustrate the variation in oscillator frequency vs. capacitive loading for a 9pF C_L crystal. For Figure 3-9 the separate 4pF course load is enabled, the fine tune is centered and the course tune array is varied. The result is that the frequency was reasonably centered with the XTAL_CTUNE[2:0] = 0x5. From this result, the XTAL_CTUNE[2:0] was set equal to 0x5 and the fine tune varied in Figure 3-10. The result is that for a 9pF load crystal suggested field settings are:

- XTAL_CTUNE[4:0] = 0x15
- XTAL_FTUNE[4:0] = 0x18

3.5.4 Using a Reference Frequency Other Than 24 MHz

The reference oscillator can use a crystal from 13 to 26 MHz or use an external source in the same frequency range. If the default frequency of 24 MHz is NOT used, then the Modem Synthesizer (see [Section 6.4, “Modem Synthesizer”](#)) must be activated and programmed for proper use of the radio. The synthesizer requires use of an external loop filter for the PLL. The external loop filter is a second order RC filter and is an integral part of the synthesizer loop, and the charge pump and VCO are connected to the loop filter. [Figure 3-11](#) shows the external components for the required external loop filter.

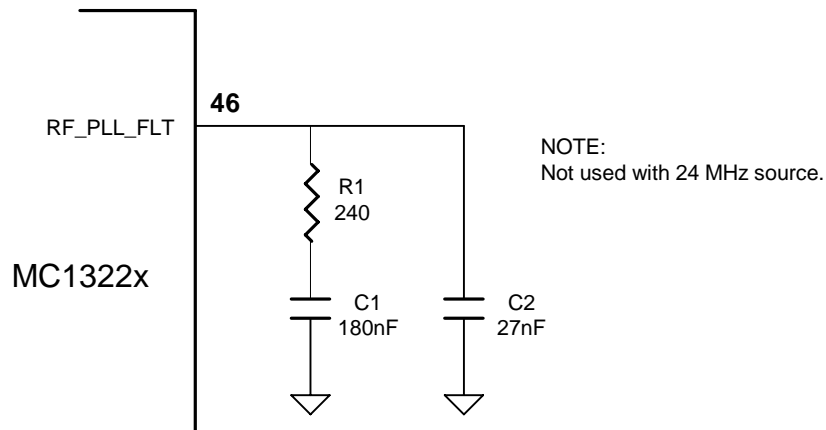


Figure 3-11. External PLL Loop Filter

3.5.5 Application Requirements for Reference Oscillator Performance

The IEEE 802.15.4 has a required reference frequency tolerance of ± 40 ppm max.

3.5.6 Using an External Reference Source

If an external source is desired in place of the reference oscillator crystal, the source can be connected as shown in [Figure 3-7](#). The crystal is not installed and the reference source is injected at XTAL_24_OUT via a 10 pF coupling capacitor

- The source must have the same ± 40 ppm frequency accuracy as an onboard reference.
- The source amplitude should be approximately $3 V_{p-p}$ with the 10 pF capacitor. A lower amplitude may require a larger coupling capacitor.

3.6 32.768 kHz Crystal Oscillator (use optional)

The 32.768 KHz oscillator is strictly optional as the RTC can also be supplied by a low accuracy onboard 2 KHz oscillatory or by keeping the reference oscillator alive during sleep (Doze mode). The 32 KHz oscillator has the disadvantage of extra cost, but is much lower in power than doze mode and much more accurate than the onboard 2 KHz oscillator.

The use of the 32.768 KHz oscillator versus the onboard 2 KHz oscillator is controlled via a register in the CRM (see [Section 5.9.17, “32kHz XTAL Control \(XTAL32_CNTL\)”](#)).

3.6.1 Description

The MC1322x has a built-in 32kHz Pierce crystal oscillator used to clock the RTC and the sleep state machine in the CRM module. The oscillator is powered by a 1 VDC power supply. It uses an external 32.768 kHz crystal with a typical load capacitance of 12.5 pF. The load capacitors are external.

Figure 3-12 shows the user oscillator model. The model shows that there are external load capacitors for the oscillator crystal. On each leg of the oscillator (**osc_in** and **osc_out**) there is:

- A fixed C_L capacitor
- Stray capacitance - sum of pad, package, and trace capacitance (typically $\sim 1.5\text{pF}$)

The load capacitance C_L to the crystal is symmetric and equals $1/2 (C_L + C_{\text{stray}})$. Target load capacitance is 12.5pF.

An active onboard feedback resistor provides dc-biasing for the oscillator amplifier.

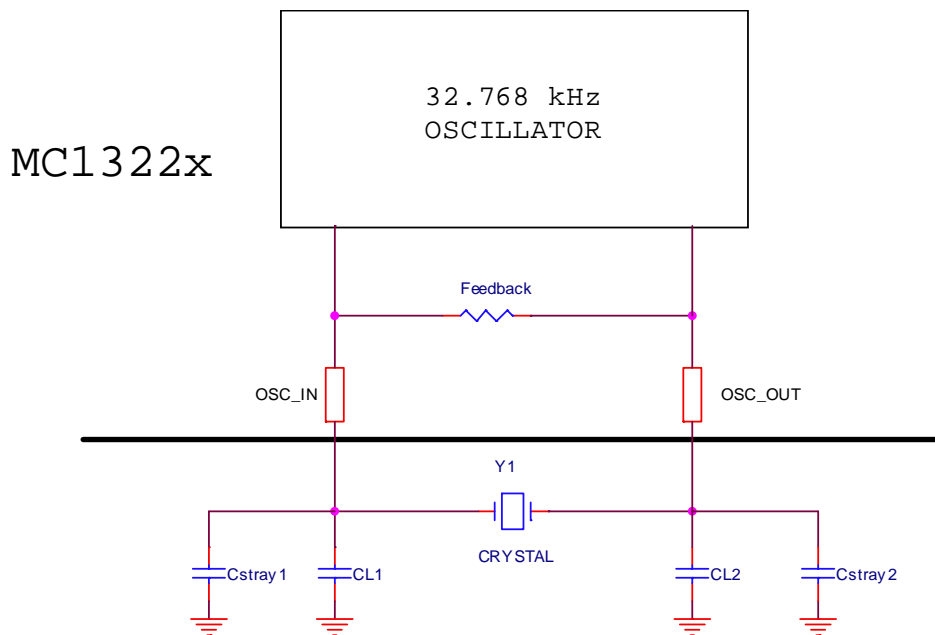


Figure 3-12. MC1322x 32.768 kHz Oscillator Model

3.6.2 32.768 kHz Crystal Specifications

Table lists the recommended crystal specifications.

Table 3-4. 32.768 kHz Crystal Specification

Parameter Name	Minimum	Typical	Maximum	Units
Crystal frequency		32.768		kHz
Storage temperature range	-55		+125	°C
Operating temperature range Industrial	-40		+85	°C
Frequency tolerance @ 25 °C			± 20	ppm
Frequency tolerance with temperature	-0.034 ±0.006ppm / (25-T)			ppm
Load capacitance	11	12.5	13	pF
Equivalent series resistance (ESR)			60	kΩ
Shunt capacitance			1.35	pF
Tolerated drive level			1	μW

This specification is consistent with the Abracon Corporation crystal part number ABS25-32.768-12.5-B

3.7 Transceiver RF Operation

For lowest cost and simplicity, the MC1322x RF operation requires only an external antenna. Additionally, a second TX PA port and dedicated control signals are available for utilization of an external PA and/or LNA for greater output power and/or sensitivity.

3.7.1 Standard Single-Ended RF Connection

The MC1322x radio interface provides for lowest cost and highest integration as shown in [Figure 3-13](#). The PiP concept in the LGA package contains components interconnected with the IC to provide the integrated RF network.

- A single-ended 50-Ω antenna connection (RF_RX_TX pin) is provided via an onboard balun. The balun provides the conversion from single-ended external antenna node to the internal differential interface.
- The IC contains a fully integrated RX switch, RX LNA, differential TX PA and balun bias supply.
- Programmable output power ranges typically from -30 dBm to +6 dBm with 0 dBm typical default.
- <-94 dBm (typical) receive sensitivity - At 1% PER, 20-byte packet (well above IEEE 802.15.4 specification of -85 dBm).
- The RX LNA provides -95 dBm sensitivity.
- Onboard bypass capacitors for the radio.

As illustrated in [Figure 3-13](#), there is an integrated internal RX switch and separate differential PA on the IC. The IC signals RFIN/OUT_P and RFIN/OUT_M are bidirectional and are connected for both for TX

and RX. When receiving, the RX switch is enabled to the internal LNA and the PA is disabled. When transmitting, the RX switch is disabled (isolating the LNA) and PA is enabled. The bias supply provides the reference voltage to the balun that converts a single-ended antenna to the differential interface required by the device.

The user need only provide a proper impedance match to the 50-Ω single-ended RF interface signal (RF_RX_TX). The antenna would commonly be a 50-Ω impedance element.

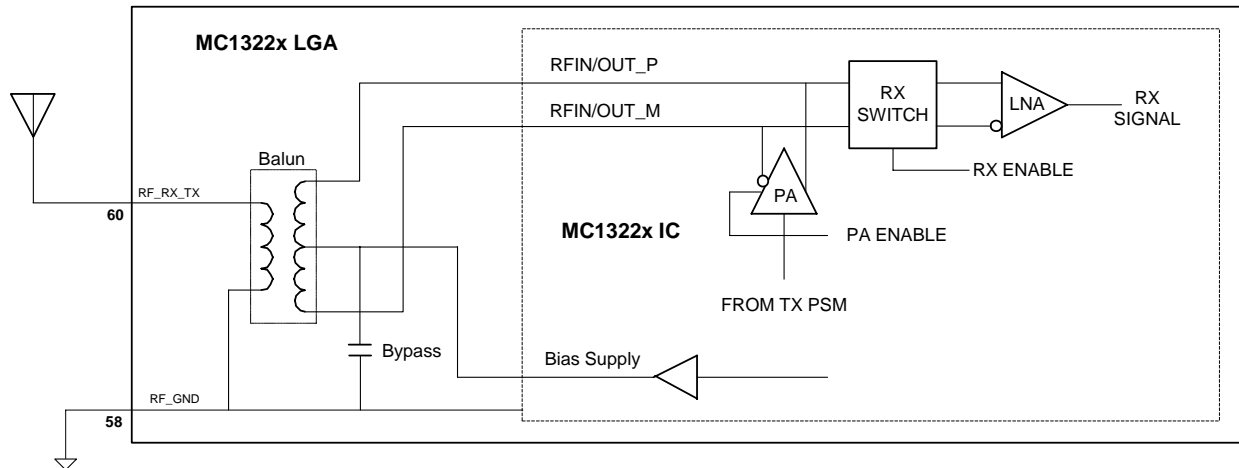


Figure 3-13. MC1322x Standard Single-Ended RF Connection

3.7.2 Dual-Port RF Operation

For those applications choosing to include an LNA for greater sensitivity, a PA for greater output gain or multiple antennas, The MC1322x provides hardware support.

3.7.2.1 RF Analog Signals

For dual port operation:

- A separate set of differential PA outputs are provided for a dual-port design. The differential PA outputs can be used with an external PA for extended gain.
- In a dual-port design, the device RF_RX_TX node is used as an input only. An external LNA is used in-line with the incoming RF signal.

3.7.2.2 RF Control Outputs

Four separate GPIO signals are available to control/enable external components and are designated as:

- ANT_1/GPIO42
- ANT_2/GPIO43
- TX_ON/GPIO44
- RX_ON/GPIO45

The MC1322x GPIO Function Select (see [Section 11.5.1, “GPIO Function Select”](#)) in the GPIO Module enables the ANT_1 and ANT_2 signals as one of two modes:

1. The designated GPIO (default) - general purpose, not RF control
2. Controlled by Transceiver (MACA) state machine (i.e., ANT_1 or ANT_2) - if enabled, the signal has a default rest state and toggles only for TX.

The MC1322x GPIO Function Select in the GPIO Module enables the TX_ON and RX_ON signals as one of three modes:

1. The designated GPIO (default) - general purpose, not RF control
2. Controlled by TX state machine - the TX state machine is pre-programmed to have a rest state for the IO (high or low) and toggles the signal while the state machine is active.
3. Controlled by RX state machine - the RX state machine is pre-programmed to have a rest state for the IO (high or low) and toggles the signal while the state machine is active.

[Table 3-5](#) summarizes control and operation of the RF control outputs.

Table 3-5. RF External Control Signal Summary

Signal Name	Pin Number	Function 0 [00] Default	Function 1 [01] TX Active ¹	Function 2 [10] RX Active ²	Comment
ANT_1	56	GPIO42	Rest = 0 Active = 1	NA	ANT_1 can be made active high for TX only
ANT_2	57	GPIO43	Rest = 1 Active = 0	NA	ANT_2 can be made active low for TX only
TX_ON	52	GPIO44	Rest = 0 Active = 1	Rest = 0 Active = 1	TX_ON can be made active high on for TX or RX mode; but only one configuration can be programmed at a time.
RX_ON	59	GPIO45	Rest = 1 Active = 0	Rest = 1 Active = 0	TX_ON can be made active low on for TX or RX mode; but only one configuration can be programmed at a time.

¹ Warm-up time is provided during the TX cycle for an external component before the PA ramps up to power.

² Warm-up time is provided during the RX cycle for an external component before the receiver is active.

NOTE

- The GPIO Driver easily facilitates control of the RF GPIO. See [Appendix B, “MC1322x Software Driver Utilities”](#)
- In low power modes, the RF GPIO do not retain their state and default to inputs which should be taken low in low power state.

3.8 Low Power Considerations

For battery operation, minimizing current draw for all conditions is of major concern. For the MC1322x, device usage is loosely divided into low power options and normal run options.

The MC1322x allows for a wide variation of low power options, however, the most basic modes include:

- Reset - The MC1322x can be held in a reset state via the hardware reset pin. This has the absolute lowest power, but the longest recovery time.
- Hibernate - Has the lowest power (excluding reset), but does not have the reference oscillator running. Exit from Hibernate is via low power oscillator (a default 2 KHz low accuracy oscillator or an optional 32.768 KHz crystal oscillator) timers or KBI asynchronous interrupts.
- Doze - Very similar to Hibernate, Doze keeps the reference oscillator running. It has the advantage of a low cost accurate time base while in sleep condition (although at the cost of additional current).

NOTE

- Entering Hibernate or Doze mode is controlled via the CRM module as described in [Chapter 5, “System Management \(Including CRM\)”](#).
- [Section 3.3.2, “Using GPIO in Low Power Modes”](#) describes GPIO in low power.

For run-time operation there are also options to help minimize current consumption:

- Clock control allows for varying CPU and bus clocks
- Peripherals can be enabled/disabled as required
- A bus-steal utility function allows for minimizing the MCU current while the radio is active

3.8.1 Controlling Low Power Current

Although reset is the lowest power option, this is typically only used when the MC1322x is a subsystem to some other controller. In standalone mode, where the device is the “system”, variations on sleep modes are used for lowest power. There are several programmable options that influence power (i.e., current draw) when low power operation is desired:

- Is the reference oscillator kept alive during sleep mode. This is the primary difference when designating sleep modes;
 - a) Hibernate mode does not keep the reference oscillator alive.
 - b) Doze mode does keep the reference oscillator alive. There is no appreciable gain in startup time by keeping the oscillator running
- Amount of RAM contents retained during sleep mode.
- Is the MCU kept alive during sleep mode
- What is the required accuracy of the sleep time
- How fast is the required wakeup time. There is a trade-off between lowest power and faster wakeup time (from sleep to active radio)

Table 3-6. MC1322x Typical Currents for Engineering Silicon

Characteristics	Typ Current	Unit
Hibernate current - RAM retained (8k, 32k, 64k, or 96k) 2KHz onboard oscillator or 32 kHz crystal oscillator CPU off (stop mode)/contents retained Radio off ADCs not available 8 Kbyte RAM retention 32 Kbyte RAM retention 64 Kbyte RAM retention 96 Kbyte RAM retention	 1.1 1.5 3 5	 μA μA μA μA
Doze current - RAM retained (8k, 32k, 64k, or 96k) Onboard 24 MHz oscillator on (high frequency accuracy) CPU off (stop mode)/contents retained Radio off ADCs available, but inactive 8 Kbyte RAM retention 32 Kbyte RAM retention 64 Kbyte RAM retention 96 Kbyte RAM retention	 60 61 63 65	 μA μA μA μA
Idle current - All RAM active Reference oscillator on (24 MHz) at 1.2 VDC CPU on at 1MHz Reference clock available to all peripherals Radio off ADCs available, but inactive	 750	 μA
Run current - All RAM active Reference oscillator on (24 MHz) at 1.2 VDC CPU on at reference frequency Radio off Reference clock available to all peripherals ADCs available, but inactive	 3.3	 mA
Receive current - All RAM active Reference oscillator on (24MHz) at 1.2 VDC Radio RX on (receiving data) Reference clock available to all peripherals ADC1 available, but inactive CPU on at 2 MHz	 21	 mA

Table 3-6. MC1322x Typical Currents for Engineering Silicon (continued)

Characteristics	Typ Current	Unit
Transmit current - All RAM active Reference oscillator on (24MHz) at 1.2 VDC Radio TX on (sending data @ 0 dBm) Reference clock available to all peripherals ADCs available, but inactive CPU clock at 2 MHz	20	mA
MCU Retention - incremental current if MCU retention is enabled	8	μA

3.8.1.1 Hibernate Mode versus Doze Mode

Keeping the reference oscillator running during sleep operation is the only real difference between Hibernate and Doze modes:

- Hibernate has a significant advantage with lower current draw vs. Doze mode
- There is no significant advantage in startup time for Doze mode (even though the reference oscillator is running)
- The advantage of Doze mode is that a very accurate timebase for the RTC and wakeup timers can be kept without use and cost of a 32.768 KHz crystal

3.8.1.2 Retaining RAM Contents

The MC1322x RAM is provided and controlled as a four blocks of 8Kbytes, 24Kbytes, 32Kbytes, and 32Kbytes, respectively (see [Section 4.2, “Features”](#)). There are factors to consider when retaining the RAM contents:

- Retaining an additional block of RAM has minimal impact on Hibernate or Doze current
- Retain only as much RAM as required for basic wakeup operation after low power mode if wakeup time is not an issue.
- RAM recovery time (loading from FLASH) has significant impact on wakeup recovery time. Approximately 2 ms are added to wakeup recovery time for every Kbyte of RAM that needs restored from FLASH

3.8.1.3 Retaining MCU Contents

The Sleep Control (SLEEP_CNTL) Register of the CRM (see [Section 5.9.3, “Sleep Control \(SLEEP_CNTL\)”](#)) contains a MCU_RET control bit (default = off) that allows state retention of all the digital logic during sleep modes, although at a reduced voltage.

- If the MCU_RET condition is not set - MCU recovery from low power is the same as a cold start or reset condition, other than some amount of RAM contents have been retained (as programmed by user). The code starts executing from the bottom of RAM and the CPU is exiting from a reset condition.
 - Key registers/parameters can be saved in retained RAM or NVM before entering sleep mode

- Applications code should be written to avoid problems with this condition
- If the MCU_RET condition is set - The digital logic (including the MPU) is not reset.
 - Program execution starts from the bottom of RAM, but all other CPU registers are intact
 - The cost of MCU_RET is additional current during the sleep mode condition.
 - Recovery time for applications software may be quicker because MCU initialization is minimized

3.8.1.4 Required Accuracy of Sleep Time and RTC

Use of the low frequency oscillator options for sleep mode saves power and perhaps cost. wakeup can be programmed via the wakeup timer or the RTC timer that are clocked by the default 2 KHz low accuracy oscillator or the optional 32.768 KHz crystal oscillator.

- The 2KHz oscillator is used for applications not requiring a high accuracy delay. Simple situations such as a sensor waking, taking a reading, and use wireless means to report results may be used with the 2KHz oscillator. The power-down timing delay is not critical and low cost is important.
- The optional 32.768KHz oscillator is useful for accurate timing delay or maintaining an accurate RTC on a continuous basis even through low power. Although a second low cost crystal is required for this oscillator, the RTC can run continuously as an accurate system clock and the wakeup timer now has a very accurate wakeup delay.

The alternative is to use the reference oscillator during low power (Doze mode) and use the reference clock/128 as the timer clock. The advantage is again a very accurate clock for RTC and wakeup delay without use of a second crystal. The penalty is a significant increase in low power current.

3.8.2 Wakeup or Recovery from Low Power Modes

wakeup from low power consists of two parts which include the wakeup mechanisms and the recovery time.

3.8.2.1 Wakeup Mechanisms

The MC1322x wakeup mechanisms include:

1. Reset - If the device is in a low power mode and a hardware reset occurs, the active signal would override the low power condition and put the device in full reset. Upon release of the hardware reset signal, the device will wakeup and boot from a cold start condition.
2. Timer-based - There are two separate timers available that can generate a wakeup interrupt
 - a) wakeup timer - This timer is a 32-bit counter that can be clocked from the 2kHz ring oscillator, the reference oscillator divided by 128, or the 32.768kHz oscillator. The source is selected as a sleep option. This counter is zeroed on entry into sleep and can be used for a very accurate and/or long sleep period. The time-out period is set via a register and the interrupt is generated when the counter and register value match.
 - b) Real Time Clock (RTC) - This is a second 32-bit timer that can be clocked from the 2kHz ring oscillator or the 32.768kHz oscillator. The RTC runs continuously (also during normal operation) and can generate a periodic interrupt based on a register value. Once a given

interrupt is generated, a new match value for the counter is generated internally, and the next interrupt will be generated based on the programmed delay count. This timer has the advantage that a wakeup can be based on a rolling, accurate timed-interval as opposed to a delay from entering sleep.

3. KBI input transition - Four KBI signals (inputs KBI_7:KBI_4) remain powered during sleep mode and a transition on the input can be enabled to generate an asynchronous interrupt to the device. The transition allows an external event (such as a button push) to wakeup the device.

Any combination of these events can be used as sleep mode wakeup. Details of implementing these options are discussed in [Chapter 5, “System Management \(Including CRM\)”](#).

3.8.2.2 Wakeup Recovery Time

The wakeup recovery time can be an important parameter for some applications where it is required to receive or transmit a frame quickly. The recovery time is variable and is impacted by how much RAM contents must be restored, circuit start-up, and how much of the device must be re-initiated. Generally, there is a trade-off between sleep power and recovery time; the lower the sleep current draw, the longer the recovery time.

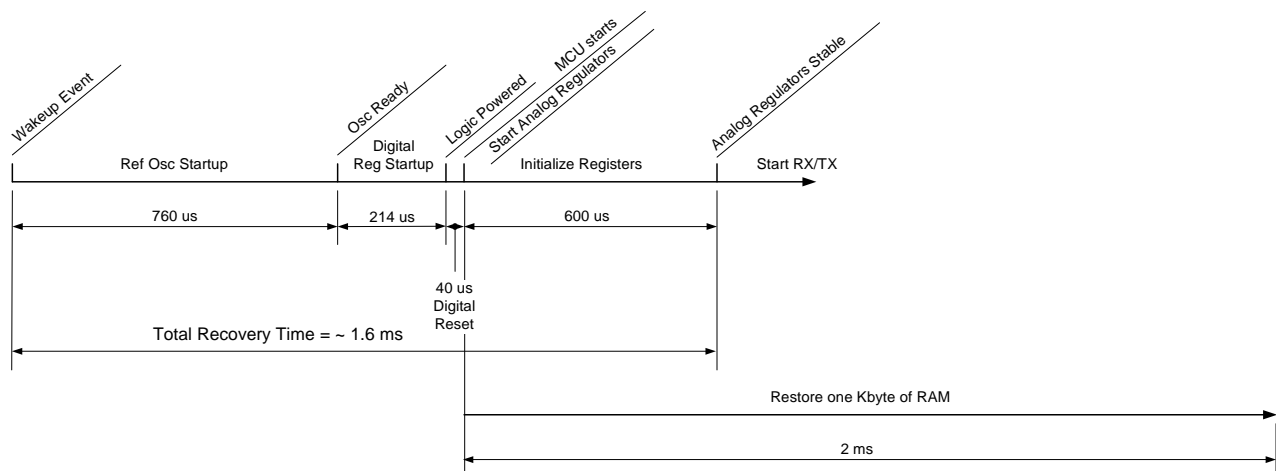


Figure 3-14. Hibernate wakeup Recovery Time

[Table 3-14](#) illustrates a typical timeline for the recovery process. The recovery timeline starts with a wakeup event and includes the following simplified steps:

1. CRM voltage reference / reference oscillator startup (typical 760-1000 us) - The voltage reference and reference oscillator (if needed) start basically in parallel. It is most common to use Hibernate for best power, and the reference oscillator is not running. There is a gain in start-up delay depending on the low power clock source:
 - Start-up with the 2 KHz oscillator typically is about 1000us
 - Start-up with either the reference clock/128 active (Doze mode) or the 32KHz oscillator active is typically 760us
2. Digital regulator startup (214 us) - The main regulator for the digital logic is enabled and must stabilize.
3. Digital logic reset (40 us) - All digital logic is reset.

NOTE

If MCU_RET is enabled, the digital logic reset pulse is not required and is disabled.

- CPU clock starts and core starts executing code from lowest RAM page.

NOTE

- wakeup steps are common to this point
 - Impact of sleep options and recovery code impact wakeup delay beyond this point.
 - For fastest recovery time to TX/RX, all required RAM has been retained.
 - Radio analog regulators must be initiated immediately for best case recovery time
- Start radio analog regulators (600 us) - Software can be running initializing radio and other logic.
 - Initiate an RX or TX - **Minimum time to active radio is about 1.6 ms**

NOTE

- This best case wakeup time assumes no reloading of required RAM contents.
- The base case wakeup time is assuming all required initialization is done within the 600us analog regulator stabilization time
- To recover RAM, the contents must be read from serial FLASH. Approximately 48 clocks/byte are required to update RAM using the default 24 MHz reference frequency. This translates to about 2 ms for 1024 bytes (1 Kbyte). **THIS TIME GETS ADDED TO RECOVERY TIME.**

3.8.3 Controlling Run-time Operating Current

When the MC1322x is not in sleep mode there are strategies that can be used to minimize run-time power:

- Control CPU and peripheral clock rate - The CPU and peripheral clocks run at the same rate and are controlled by the System Control (SYS_CNTL) Register in the CRM (see [Section 5.9.1, “System Control \(SYS_CNTL\)”](#)). The clock can sometimes be slowed down if the application does not require high throughput.

NOTE

If the radio is active, i.e., an RX or TX sequence is in progress, there is a minimum required clock rate depending on mode:

- For 802.15.4 mode, the CPU clock must be no less than 2MHz
 - For mixed mode, the higher 8MHz clock rate is required
- Enable bus-steal module for active radio sequences - The MC1322x has a unique bus-steal module (see [Section 5.9.4, “Bus Stealing Control \(BS_CNTL\)”](#)) to minimize MCU current drain during radio operation and save overall power. The MCU data bus has three bus masters that include the

MACA DMA function (for moving packet data during a RX or TX), the CPU core, and the bus-steal module. The DMA works on a cycle-steal basis and has highest priority (it is also the reason there is a minimum required clock rate during an active radio sequence). Under normal operation, the CPU dominates the data bus usage and releases it as required by the DMA. If enabled, the bus-steal module is only active during a radio operation and “steals” clock cycles from the CPU. This allows the clock to run at a higher rate but not burn as much power because the CPU is halted for the programmed number of cycles on a cyclical basis. The CPU can also be halted to wait for an interrupt while the bus-steal is active.

- Enable peripheral functions as required - All peripherals can be enabled individually. Enable peripherals only as required. One caution is that a given peripheral may require a higher peripheral clock rate that would set a minimum required rate during TX or RX.

3.9 Bootloader

The bootloader is stored in the MC1322x onboard ROM. After exiting a reset, the CPU starts executing from the onboard ROM to boot the device.

NOTE

The bootloader is generally described in this section. Detailed information on procedures and data format are given in [Appendix C, “Bootloader Reference”](#).

3.9.1 Overview

Stored in the ROM, a very simple bootstrap program is executed out of reset. The most common mode of operation is that a valid target image has been stored in the onboard FLASH, the FLASH contents get transferred into RAM, and the actual application starts executing from RAM. This bootstrap will only load a configurable number of consecutive bytes from the FLASH into the RAM area beginning at address 0x0040_0000.

If the exit from reset has resulted from wakeup from a sleep mode (versus a hardware reset), the bootstrap will vector to the start of RAM and begin executing from there. The lowest page of RAM is always saved in low power operation.

If a valid FLASH image is unavailable, the bootstrap can load a configurable number of consecutive bytes from a secondary boot source into the beginning of the RAM area (0x0040_0000). The secondary boot source options are:

- Load the target image from UART1 port
- Load the target image from the SPI port (as slave) attached to a master device
- Initiate the SPI port as a master and load target image from external slave (serial FLASH)

NOTE

During boot, the JTAG/Nexus debug port gets enabled for all operations other than a secure mode (protected) FLASH image. The debug mode can always take command of the system, as would be common during development/debug operations.

- Load the target image from I²C (serial EEPROM)

3.9.2 Exception Vectors

The ARM7TDMI-S processor expects that its exception vectors reside at memory addresses 0x0000_0000 to 0x0000_001F (see [Section 7.9.9, “Exception Vectors”](#)). However, the ROM is mapped into the low memory address space (0x0000_0000 through 0x0001_3FFF), and as a result, the exception vectors also reside in ROM and are hard coded.

To allow the user to place exception service routines at his desired RAM location, the ROM exception vector entries redirect or jump program execution to the bottom of RAM (0x0040_0000 to 0x0040_001F), see [Section 4.4, “Exception Vectors”](#). As an example, the normal undefined instruction vector (0x0000_0004) will jump to the RAM address 0x0040_0004, and the user should treat address 0x0040_0004 as his undefined instruction vector.

NOTE

The bootstrap will starting loading code into 0x0040_0000. The beginning of the code must be treated as exception vectors!

3.9.3 Bootstrap Flow

The [Figure 3-15](#) shows the flow chart for the bootstrap. The flow provides recovery from low power modes as well as reset.

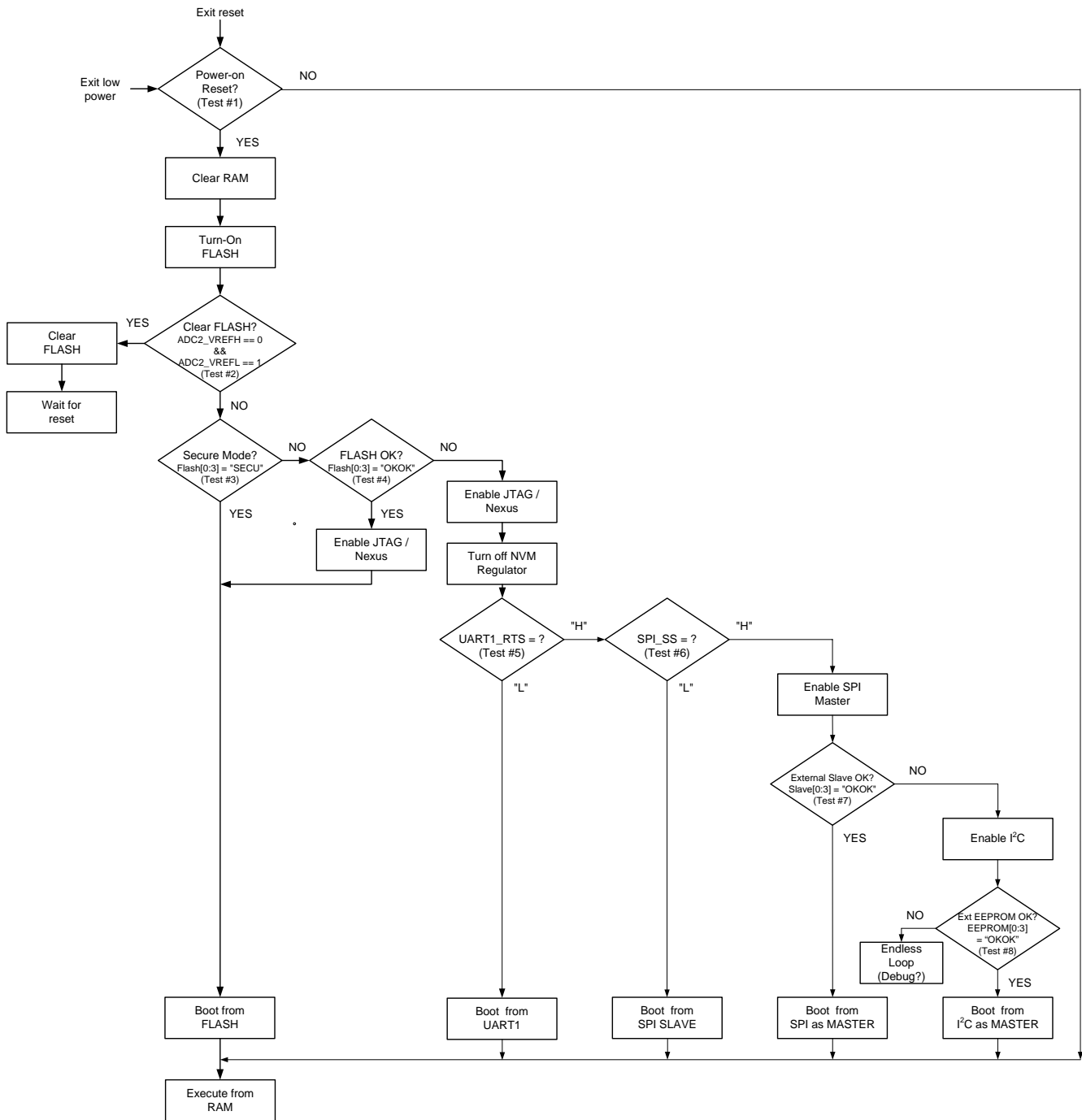


Figure 3-15. ROM Bootstrap Flow Chart

3.9.3.1 Flow Description

The bootstrap flow is entered as the result of exiting a low power mode or exiting reset:

1. TEST #1 - Is entry from a power-on reset (POR) condition?
 - NO: Proceed to execute from RAM - This action is the result of wakeup from Hibernate or Doze modes. The program jumps directly to the beginning of the RAM and starts executing from there. The lowest RAM sector PAGE0 (address 0x0400_0000) is always kept powered during Hibernate or Doze. It is the responsibility of the programmer to provide proper RAM and MCU recovery based on the power down options (such as how much RAM has been kept powered and how much has to be reloaded).
 - YES: Proceed with the boot flow. -
 - Clear RAM
 - Turn-on (power-up) the serial FLASH
 - Go to TEST #2
2. TEST #2 - Clear FLASH? (Two signals are tested for their input state to see if the condition is met, i.e., $ADC2_VREFH == 0 \ \&\& \ ADC2_VREFL == 1$. See [Section 3.9.5, “FLASH Erase or Recovery Mode”](#))
 - YES: Proceed to clear FLASH
 - Clear FLASH contents
 - Wait for reset
 - NO: Go to Test #3
3. Test #3 - FLASH in secure mode? (The contents of FLASH [0:3] are tested for the ASCII value “SECU”. See [Section 3.9.6, “Secure Mode”](#))
 - YES: Proceed to boot from FLASH and execute. Do not enable debug to allow examination of FLASH
 - Load RAM from FLASH (boot). See [Section 3.9.4, “Booting from FLASH”](#)
 - Jump to RAM and execute
 - NO: Go to Test #4
4. Test #4 - FLASH is OK? (The contents of FLASH [0:3] are tested for the ASCII value “OKOK”. The contents of the FLASH are valid for boot, but not secured.)
 - YES: Enable debug, proceed to boot from FLASH and execute.
 - Enable JTAG/Nexus
 - Load RAM from FLASH (boot). See [Section 3.9.4, “Booting from FLASH”](#)
 - Jump to RAM and execute
 - NO: Enable debug and continue
 - Enable JTAG/Nexus
 - Disable FLASH (turn off NVM regulator)
 - Go to Test #5
5. Test #5 - Boot from UART1? (The GPIO signal UART1_RTS is tested for its input state, i.e., $UART1_RTS = ?$)

- LOW (yes)
 - Proceed to load RAM from UART1
 - Jump to RAM and execute
 - HIGH (no): Go to Test #6
6. Test #6 - Boot from SPI Slave Mode? (The GPIO signal SPI_SS is tested for its input state, i.e., SPI_SS = ?)
- LOW: An external SPI Master is present
 - Place SPI port in Slave mode
 - Proceed to load RAM from onboard SPI Slave port
 - Jump to RAM and execute
 - HIGH: Attempt boot from external SPI Slave (typically external FLASH)
 - Place onboard SPI port in Master mode
 - Read four bytes
 - Go to Test #7
7. Test #7 - Is external SPI Slave present and are data correct? (The data are tested for the ASCII value “OKOK”)
- YES: Proceed to boot from external FLASH and execute
 - Proceed to load RAM through onboard SPI Master
 - Jump to RAM and execute
 - NO: Attempt boot from external I²C EEPROM
 - Enable I²C as Master
 - Read four bytes data from external slave
 - Go to Test #8
8. Test #8 - Is external EEPROM present and are data correct? (The data are tested for the ASCII value “OKOK”)
- YES: Proceed to boot from external EEPROM and execute
 - Proceed to load RAM through I²C
 - Jump to RAM and execute
 - NO: Enter hold loop
 - Enter endless loop
 - Either reset or debug port is required to restore control

NOTE

- Interrupts are disabled during boot.
- After code is loaded into RAM, execution jumps to address 0x0400_0000 and starts from there.
- After exiting boot, GPIO pins are not all in the default state, but retain any setup values from boot.

3.9.4 Booting from FLASH

After FLASH has been determined to be valid (through the “SECURE” or “OK” bytes) the boot flow loads the subsequent four bytes (32-bit, little-endian), and uses that as a length field parameter of how many bytes to load into RAM. The boot flows then loads the next consecutive “length” number of bytes from FLASH into RAM starting with the first RAM address (0x0040_0000). Once the code is loaded into RAM, the bootstrap jumps to the first RAM address(0x0040_0000) and continues execution from there.

3.9.5 FLASH Erase or Recovery Mode

If the FLASH contents need to be updated, or otherwise cleared, the FLASH can be erased through the boot process. This can be achieved by:

- ADC2_VREFH is enabled as an input with pull-up resistor, and ADC2_VREFL is enabled as an input with pull-down resistor.
- These pins are then tested for $ADC2_VREFH == 0 \ \&\& \ ADC2_VREFL == 1$. The user forces this condition typically through jumpers, and this condition being true forces the FLASH to be erased.
- When executing this option, wait a few seconds for the FLASH to be cleared, and then set $ADC2_VREFH = 1$ and $ADC2_VREFL = 0$ (non-recovery mode) before resetting the device.
- The FLASH is now cleared, and with a reset, the MC1322x will follow the boot flow to enable debug mode through JTAG or Nexus.

NOTE

When the FLASH is erased, the top 4 Kbyte sector is left untouched. This sector is reserved for production use.

3.9.6 Secure Mode

In order to prevent the FLASH contents from being unnecessarily dumped, or otherwise altered through use of the JTAG/Nexus interface(s), the JTAG/Nexus pinouts are disabled when the device is powered up. Only if the bootstrap determines that the FLASH is not secured (“SECU” not found), will it enable the JTAG/Nexus interfaces; otherwise they remain disabled.

Once the device is secured, and the JTAG/Nexus interfaces remain disabled, they can only be re-enabled by forcing the FLASH to be cleared (i.e., enter recovery mode).

3.9.7 GPIO Function Upon Boot Exit

Depending on the boot option, certain GPIO are left in an altered mode (from default) when exiting boot. The following summary lists modified GPIO based on boot option.

- Boot from internal flash.
 - GPIO_38 (ADC2_VRefH) - GPIO mode, input, read from pad, pullup
 - GPIO_39 (ADC2_VRefL) - GPIO mode, input, read from pad, pulldown
- Boot from UART1.
 - GPIO_38 (ADC2_VRefH) - GPIO mode, input, read from pad, pullup

- GPIO_39 (ADC2_VRefL) - GPIO mode, input, read from pad, pulldown
- GPIO_14 (UART1_TX) - Function1 mode
- GPIO_15 (UART1_RX) - Function1 mode, pullup
- GPIO_16 (UART1_CTS) - Function1 mode
- GPIO_17 (UART1_RTS) - Function1 mode, pullup
- Boot from SPI (master or slave).
 - GPIO_38 (ADC2_VRefH) - GPIO mode, input, read from pad, pullup
 - GPIO_39 (ADC2_VRefL) - GPIO mode, input, read from pad, pulldown
 - GPIO_17 (UART1_RTS) - GPIO mode, input, read from pad, pullup
 - GPIO_4 (SPI_SS) - Function1 mode, pullup
 - GPIO_5 (SPI_MISO) - Function 1 mode
 - GPIO_6 (SPI_MOSI) - Function 1 mode
 - GPIO_7 (SPI_SCK) - Function 1 mode
- Boot from I2CEEPROM
 - GPIO_38 (ADC2_VRefH) - GPIO mode, input, read from pad, pullup
 - GPIO_39 (ADC2_VRefL) - GPIO mode, input, read from pad, pulldown
 - GPIO_17 (UART1_RTS) - GPIO mode, input, read from pad, pullup
 - GPIO_4 (SPI_SS) - Function 1 mode, pullup
 - GPIO_5 (SPI_MISO) - Function 1 mode
 - GPIO_6 (SPI_MOSI) - Function 1 mode
 - GPIO_7 (SPI_SCK) - Function 1 mode
 - GPIO_12 (I2C_SCL) - Function 1 mode
 - GPIO_13 (I2C_SDA) - Function 1 mode

3.9.8 Bootstrap version.

The Bootstrap Software Version is located at address 0x0000_0020 in ROM and has four bytes.

Chapter 4

Memory

4.1 Introduction

The MC1322x ARM7 core has a register-based instruction set and the CPU registers are not located in the memory map. The on-board memory resources include:

- 128 Kbyte serial FLASH memory (that is mirrored into RAM)
- 96 Kbyte SRAM (98304_{dec} bytes actual)
- 80 Kbyte ROM
- MCU peripheral registers and buffers
- Transceiver control registers

4.2 Features

- 96 Kbyte SRAM in 4 blocks; addressed contiguously
 - RAM0: 8 Kbytes, 2 Kwords (2048 x 32 bits)
 - RAM1: 24 Kbytes, 6 Kwords (6144 x 32 bits)
 - RAM2: 32 Kbytes, 8 Kwords (8192 x 32 bits)
 - RAM3: 32 Kbytes, 8 Kwords (8192 x 32 bits)
- All read or write accesses require a minimum of two system clock cycles
- Stall signal generated for read after write cycles
- SRAM have been divided into blocks to allow for power savings.
 - Block RAM0 is always powered.
 - While sleeping, 1, 2, or 3 of the remaining RAM blocks can be turned off.
 - While sleeping, powered RAM blocks are placed in a low voltage mode for data retention.
 - As more RAM blocks are turned on, more current is required. The disadvantage is that boot time can be longer as required to reload RAM from FLASH.
- 80 Kbyte ROM
 - 20 Kwords (20480 x 32 bits)
 - Contains bootstrap code, 802.15.4 MAC (no security) and drivers. The MAC software builds on the lower level hardware capability of the transceiver and MACA. All code except the bootstrap is “patchable”.
- Serial FLASH (NVM)
 - 128 Kbyte

- Accessed via a dedicated SPI module designated as the SPIF. Freescale provides an access driver.
- The FLASH erase, program, and read access are programmed through the SPIF port.
- Program code gets mirrored into SRAM for execution. The FLASH is accessed at boot time to load/initialize RAM. All CPU program access is from RAM or ROM.
- Highest 4 Kbyte sector is reserved for factory use.
- Write accessible under program control.
 - Use provided driver.
 - Can also be used for non-volatile parameter, look-up table, and optional program code storage

4.3 Memory Map

The MC1322x summary memory map is shown in [Table 4-1](#). In this table only the base address of the various modules is shown. For a fully detailed memory map of peripheral registers, See [Appendix A, “MC1322x Register Address Map”](#). Also, in each module chapter the base address and register address table for the associated module is given.

Table 4-1. MC1322x Top Level Memory Map

Base Address	Module	Size (Bytes)	Type
0x0000_0000	ROM	80k	R
0x0040_0000	SRAM	96k	R/W
0x0040_0000	RAM0	8k	R/W
0x0040_2000	RAM1	24k	R/W
0x0040_8000	RAM2	32k	R/W
0x0041_0000	RAM3	32k	R/W
0x8000_0000	GPIO		R/W
0x8000_1000	SSI		R/W
0x8000_2000	SPI Port		R/W
0x8000_3000	Clock and Reset Module (CRM)		R/W
0x8000_4000	MAC Accelerator (MACA)		R/W
0x8000_5000	UART1		R/W
0x8000_6000	I ² C		R/W
0x8000_7000	TIMER		R/W
0x8000_8000	Advanced Security Module (ASM)		R/W
0x8000_9000	Modem Synthesizer Write Functions		R/W
0x8000_9200	Modem Transmit Sequence Manager		R/W
0x8000_9400	Modem Radio Receiver Functions		R/W

Table 4-1. MC1322x Top Level Memory Map (continued)

Base Address	Module	Size (Bytes)	Type
0x8000_9600	Modem Radio Transmitter Functions		R/W
0x8000_9800	Modem Radio Frequency Synthesizer		R/W
0x8000_9A00	Modem Tracking Oscillator Controller		R/W
0x8000_A000	Radio Analog Write Functions		W
0x8000_A000	Radio Analog Read Functions		R
0x8000_B000	UART2		R/W
0x8000_C000	Reserved		-
0x8000_D000	ADC Module		R/W
0x8002_0000	Interrupt Controller		R/W

4.4 Exception Vectors

The ARM architecture places the exception vector addresses at the bottom of the memory map, i.e., start address 0x0000_0000. With the MC1322x, the onboard ROM is located at address page 0x0000_0000 and does not allow the user to change vector contents. To overcome this issue, the ROM redirects the vector table to the base of the RAM at address 0x0040_0000. Table 4-2 shows the exception vector addresses with both the original address and the redirected user address in RAM.

Table 4-2. MC1322x ARM Exception vectors

ROM Address	Exception	RAM Address	Notes / Usage
0x0000_0000	Reset (boot start)	0x0040_0000	Start RAM address after boot.
0x0000_0004	Undefined instruction	0x0040_0004	Supported
0x0000_0008	Software interrupt	0x0040_0008	Supported
0x0000_000C	Abort (Prefetch)	0x0040_000C	Supported ¹
0x0000_0010	Abort (Data)	0x0040_0010	Supported ¹
0x0000_0014	Reserved	0x0040_0014	Reserved
0x0000_0018	IRQ	0x0040_0018	Supported
0x0000_001C	FIQ	0x0040_001C	Supported

¹ Supported for RAM access and UART modules; not supported by all other peripherals

NOTE

Address 0x0000_000 is still the reset vector in ROM, and execution starts here in ROM for the boot process. Address 0x0040_0000 becomes the start address of execution from RAM after boot (reset or restart from a low power condition).

4.5 ROM and RAM

The MC1322x executes all program out of the 80 kbyte ROM or the 96 kbyte SRAM. The ROM is factory programmed and contains boot code, drivers, utilities, and different communication protocol services. After recovery from a reset condition, execution always starts from address 0x0000_0000.

The RAM is broken into 4 pages for the purpose of allowing different options for power-down. Page RAM0 is 8192 bytes and is always powered under a sleep condition. RAM1, RAM2, and RAM3 pages are 24576, 32768, and 32768 kbytes respectively. These can be kept powered or un-powered in sleep mode.

The RAM is loaded from the serial FLASH at boot time and any powered-down page must be re-loaded after wake-up from sleep (Hibernate or Doze). Recovery time can vary greatly as a result. After wake-up, execution always starts at address 0x0040_0000. The recovery routine for reloading powered-down SRAM must reside in page RAM0.

Figure 4-1 shows the MC1322x ROM and SRAM memory map.

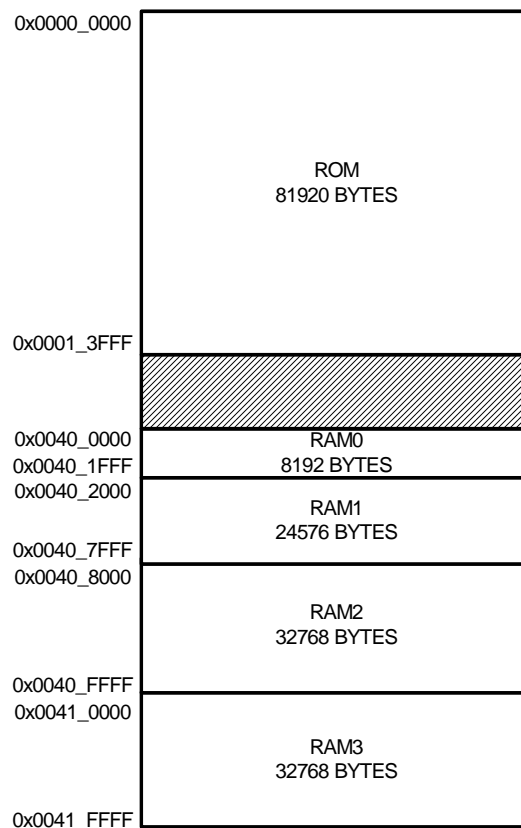


Figure 4-1. Memory map of the mem module (80 kbyte ROM and 96 kbyte RAM)

4.6 Serial FLASH (NVM)

The MC1322x non-volatile memory (NVM) is the a 128 Kbyte serial FLASH. The serial FLASH is accessed via a dedicated SPI module designated as the SPIF. The FLASH erase, program, and read capability are all programmed through the SPIF port. The valid FLASH contents are accessed at boot time to load/initialize RAM for program execution. The FLASH is also accessed to re-initialized/restore RAM that had been un-powered during sleep.

With 80 kbytes of RAM and 128 kbytes of FLASH, there is 48 Kbytes of excess capacity for other useful purposes than the main program code. The top 4 kbyte sector is reserved for factory use. There is sufficient capacity for non-volatile data store and optional application profiles if desired.

Some of the characteristics of the serial FLASH include:

- SPI serial interface with 13 MHz maximum data clock
- Accessed via a dedicated SPI module designated as the SPIF.
- 15 mA erase, program, or read current
- 5 μ A maximum standby current
- 75 ms sector or block erase duration
- 150 ms chip erase duration
- Accessible under program control

4.6.1 FLASH Access

A driver for FLASH access (erase, program, and read) is available for integration with the user's application. See [Appendix B, "MC1322x Software Driver Utilities"](#).

4.6.2 FLASH Security

The FLASH is accessed during the boot process. The FLASH must contain valid data and it also can be made secure for the boot process as discussed in [Section 3.9.6, "Secure Mode"](#).

The JTAG/Nexus interfaces are disabled when the MC1322x is powered up. Only if the bootstrap determines that the FLASH is not secured, will it enable the JTAG/Nexus interfaces; otherwise they remain disabled.

The first four bytes of the FLASH are reserved for a validation word that is used to verify that the FLASH contains valid data, as well, to verify if the FLASH is secure. If the validation word is "OKOK" the data are valid, but not protected. If the validation word is "SECU" the data are valid AND secure.

Once the device is secured and the JTAG/Nexus interfaces remain disabled, they can only be re-enabled by forcing the FLASH to be cleared (i.e., enter recovery mode).

4.6.3 FLASH Recovery (Erase)

As discussed in [Section 3.9.5, “FLASH Erase or Recovery Mode”](#), the FLASH can be erased via the boot process to be “recovered” to a known clear condition. Also, erasing parts or all of FLASH through program control is possible.

4.7 Peripheral Register Access

The MCU peripherals control and status registers vary in register width and allowed access type.

- 32-bit write access is always allowed.
- Some registers are 32-bit write access only. Reference the individual register descriptions for limitations as to 8-bit or 16-bit write access.
- 8-bit, 16-bit, and 32-bit read accesses are always allowed. Any access will always provide 32-bits of read data, and all bits/fields not present in the register will default to “0” on the data bus.

Chapter 5

System Management (Including CRM)

5.1 Management Overview

This chapter details the management of the PiP including reset, power management, wake-up from low power, clock control, and KBI interface. Management is controlled by the Clock/Reset/Power Module (CRM) which is a dedicated unit designed to handle all top level clock, reset and power functions for the MC1322x. The CRM controls the voltage regulators on MC1322x for power management. The CRM also contains a small block called the Sleep Module that runs when the rest of the device entirely powered down and it keeps time during sleep and wake modes.

5.1.1 Management Features

- Manages system reset and available software initiated system reset
- Controls clock gating for power savings
- Power management of internal regulated voltage sources including buck regulator
- Real Time Clock (RTC)
- Sleep mode management
 - 2 types of sleep: Hibernation or Doze
 - Controlled power down
 - Programmable degree of power-down
 - Critical programmed values are retained during sleep
- Wake-up mode management
 - Chip is gracefully powered up.
 - Clocks are automatically turned on.
 - wake-up available by programmable wake-up and RTC timers.
 - wake-up available by external interrupts from KBI pads.
- Watchdog (COP) surveillance timer
- Bus stealing mode to keep ARM quiet during idle periods.
- Management of ring oscillator and optional 32.768 KHz oscillators
- Management of reference oscillator
- Management of MCU core and peripheral clocks

5.1.2 System Reset

The MC1322x provides complete management of system reset and recovery/initialization from reset. There are several sources of reset:

- Hardware reset signal RESETB - The reset input is an active low, asynchronous signal. Holding RESETB low causes the lowest power mode for the device (typically less than 300 nA) where all circuitry is disabled except for the CRM Sleep Module. There is no onboard pull-up resistor attached to RESETB to further reduce current. Upon release of RESETB to high, the device recovers to a run condition.
- Power-on reset (POR) - In a wireless node where the MC1322x is the main controller, RESETB can be tied to VBATT so that the device powers-up as VBATT becomes valid. To guarantee a proper start-up procedure, there is a power-on reset circuit that holds the equivalent of an active reset signal until the VBATT voltage reaches a minimum threshold voltage. The POR will then be released to allow the device start-up sequence to proceed.

NOTE

The external RESETB is tied into the POR such that after RESETB is released, the POR circuit must also release to allow start-up to proceed.

- Software reset - The CRM has the Software Reset Register (address 0x8000_3050) where writing to the register can cause a system reset. The affect is the same as exiting a POR and starting a start-up sequence.
- Waking from Hibernate or Doze modes without MCU state retention - One option for Hibernate or Doze modes is to NOT retain all the MCU, Modem, and Analog Control states. This is controlled by the MCU_RET Bit 6 of the Sleep Control Register (address 0x80003008). If the MCU retention bit is not set, then the wake-up from low power mode(s) resets the MCU.

NOTE

- After exit from an MCU reset or any other reset initiated start-up sequence, program execution begins at the reset boot vector address 0x0000_0000.
- After exit from a wake-up sequence with an MCU reset, program execution begins at address 0x0040_0000, the beginning of RAM.

5.1.3 System Clocks

This section provides a brief overview of the system clocks.

5.1.3.1 Clock Sources

The MC1322x has 3 possible clock sources, i.e., the reference oscillator (13 - 26 MHz with 24 MHz default), an onboard self-contained 2 kHz ring oscillator, and an optional 32.768 khz crystal oscillator.

- Reference oscillator - is the primary clock source for the device
 - Default frequency is 24 MHz, although a frequency from 16 MHz to 26 MHz can be used.
 - Normally a crystal is used with the onboard amplifier although an external source can be used.

- Frequency accuracy of +/-40 ppm for IEEE 802.15.4 compatibility must be maintained over all conditions.
- [Section 3.5, “Reference Oscillator”](#) describes detailed use of the oscillator.
- The frequency of the crystal oscillator can be trimmed by changing onboard load capacitance. [Section 5.9.16, “Reference XTAL Control \(XTAL_CNTL\)”](#) describes the control register that adjust capacitive loading.
- Is divided by 128 and used as the wake-up timer source during Doze mode.
- 2 kHz ring oscillator - is the default oscillator for the Real Time Clock (RTC) and the Sleep Module when in or waking-up from Hibernate or Doze. The specified default frequency accuracy of the oscillator over temperature is from 1 kHz to 4 kHz. However, onboard resources are available to trim the oscillator accuracy.
- Optional 32.768 KHz crystal oscillator - is available with the addition of an external 32.768 KHz crystal (see [Section 3.6, “32.768 kHz Crystal Oscillator \(use optional\)”](#)). This oscillator typically replaces the use of the ring oscillator if a low power, high accuracy source is desired for wake-up and the RTC.

5.1.3.2 Main System Clock Distribution

All main system clocks are derived from the reference oscillator. [Figure 5-1](#) shows a simplified diagram of the MC1322x clock distribution. The diagram does not illustrate clock power-down options.

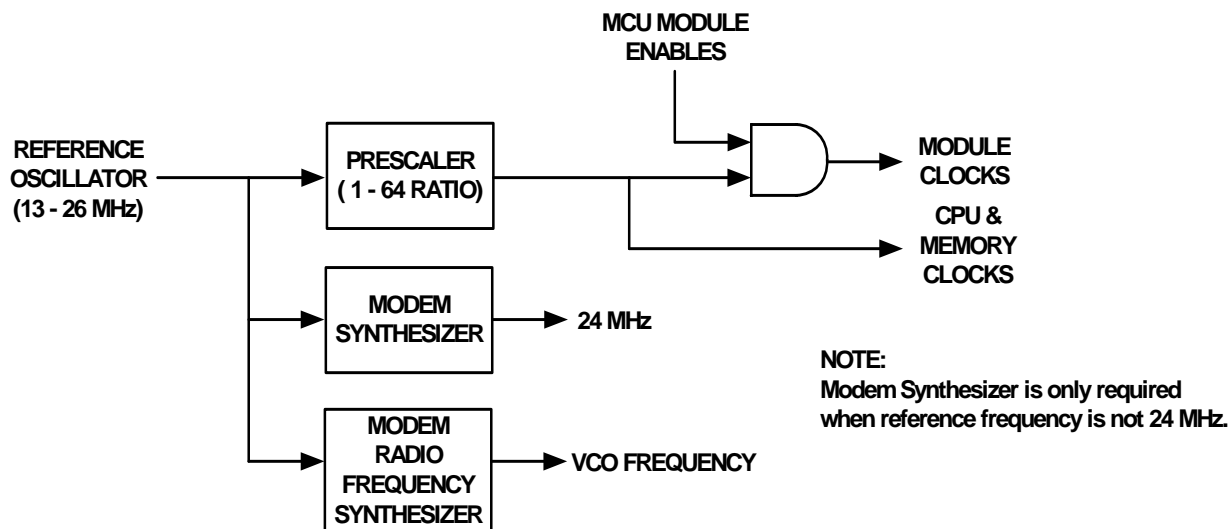


Figure 5-1. MC1322x Simplified Clock Distribution

The reference oscillator is normally 24 MHz, but can be from 13 - 26 MHz

- The CPU, memory, and MCU module clocks are all derived from a programmable divider (prescaler)
 - Prescaler divide ratio is set by the `xtal_clkdiv[5:0]` field of the System Control Register - divide ratios programmable from 1 to 64 (default = 1). Lower core clock rates reduce power, but also reduce performance.

- MCU module clocks are enabled individually by their respective module enables - the module clock enables are located within register sets for each module. The module source clocks are further modified by the module circuitry as to baud rates, count frequencies, etc.
- The Modem Radio Frequency Synthesizer generates the radio VCO frequency - it can use any reference frequency from 13 to 26 MHz. The radio channel frequencies are set by appropriately programming the synthesizer using the known reference frequency.
- The Modem Synthesizer
 - The modem synthesizer is only used for the receiver and only outputs 24MHz.
 - The receiver always requires a 24 MHz reference clock and the modem synthesizer is only used to generate this 24 MHz when the reference oscillator is NOT 24 MHz.
 - The external PLL filter is used by the modem synthesizer and is only used with a non-24MHz reference clock.

5.1.4 System Power Distribution

Figure 5-3 shows a diagram of power distribution. There are a number of different linear regulators that supply different functional blocks on the device. Power can be enabled/disabled to the different blocks based on the operational mode.

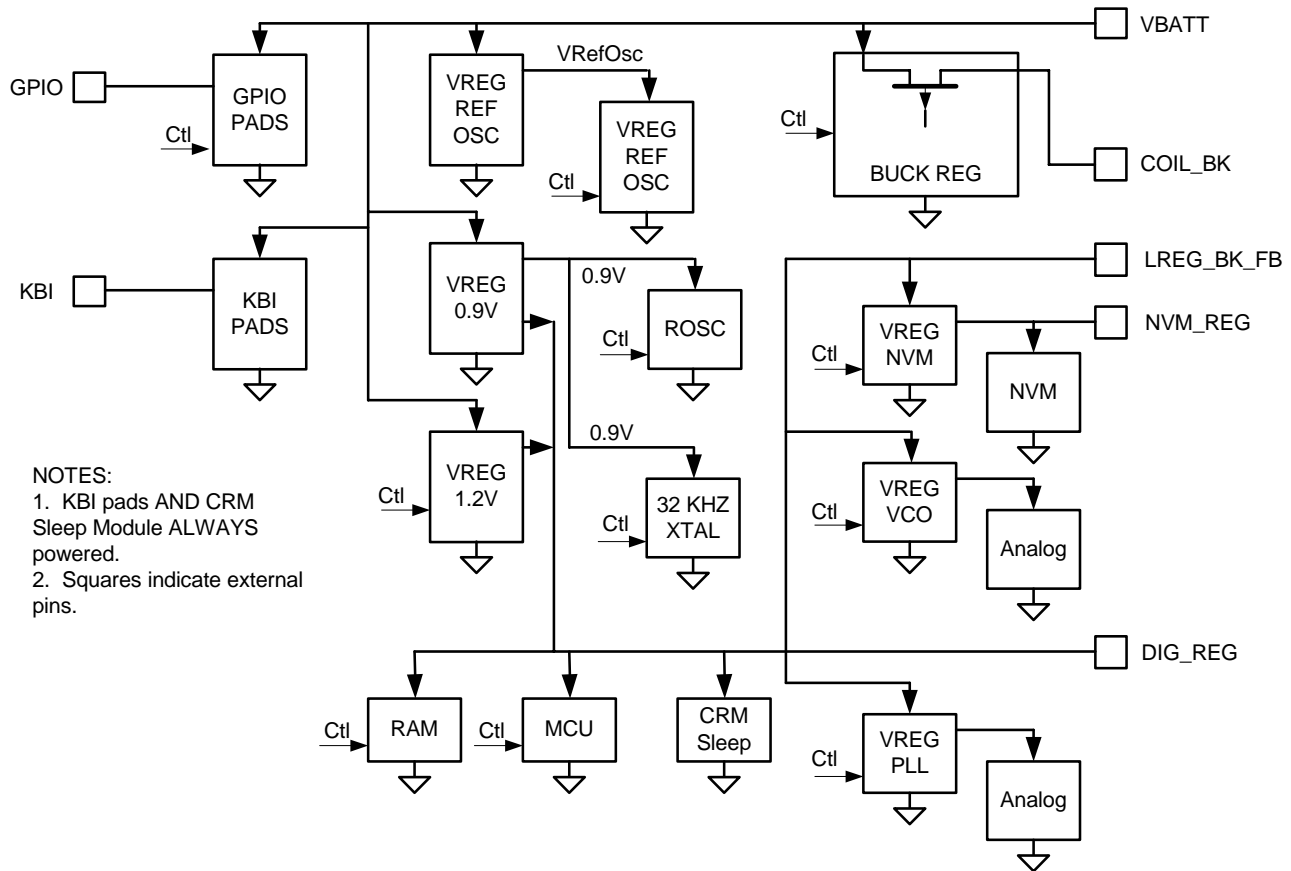


Figure 5-2. MC1322x Power Distribution

Consider the following:

- The KBI pads and CRM Sleep Module are always powered except when RESETB is active. The GPIO pads are not powered in reset and typically in low power modes. See [Section 3.3.1, “GPIO Pin State During Reset, Default, and Low Power Modes”](#) for a detailed discussion of IO in different modes.
- In sleep modes
 - The CRM Sleep Module maintains control
 - The different oscillators can be enabled (one at a time) depending on options
 - RAM0 page is always powered; RAM1, RAM2, and RAM3 can be optionally powered
- Power to the device can be connected in three separate ways, see [Section 3.2.1, “Power Pin Descriptions”](#) for details
- The BUCK REG is an optional switching regulator and is controlled via the CRM

5.1.5 KBI Interface Signals

The KBI signals are unique among the IO signals and service two common purposes:

- The KBI signals are meant to be used as IO during low power modes - The KBI signals along with low power control circuitry are kept powered during low power modes, and always revert to CRM control during low power operation. See [Section 5.9.2, “Wake-up Control \(WU_CNTL\)”](#).
 - KBI_3 : KBI_0 - These signals revert to outputs whenever low power mode is initiated. Default out of reset is all outputs in the high state. Once the device is initiated, the outputs can be programmed to revert to either output high or output low.
 - KBI_0 - In addition to being a powered output, KBI_0 is unique in that it can be programmed as a wake-up signal to external devices. In sleep mode KBI_0 is an output low, but upon wake-up it transitions to high, and an external device can monitor the signal for a wake-up event.
 - KBI_7 : KBI_4 - These signals revert to inputs that can be enabled to asynchronously wake-up the device. The input can be programmed as to level or transition sensitivity and can also be enabled to generate an interrupt request to the CPU.
- The KBI_7 : KBI_4 are always available in normal operation to asynchronously generate an input signal transition interrupt request to the CPU. The interrupt request can be enabled and programmed as to level or transition sensitivity in the CRM. However, when not in low power modes, the GPIO module controls these pins so that they must be enabled as inputs with an appropriate pulldown or pullup. Common use of this capability is a keypad or pushbutton interface

5.2 Clock/Reset/Power Module (CRM) Overview

The following sections details functions and operations of the CRM. The CRM contains the Sleep Module with wake-up timer and Real Time Clock, status and control registers, clock controls, bus steal module, and the COP timer.

5.2.1 CRM Block Diagram

Figure 5-3 shows the CRM block diagram.

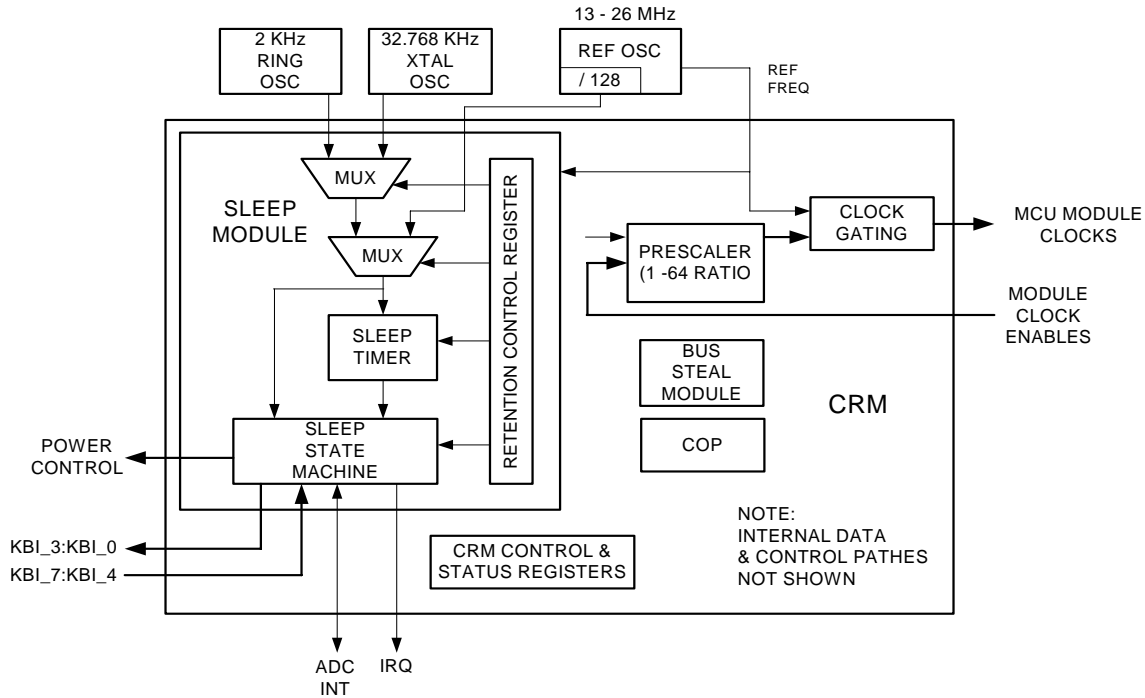


Figure 5-3. CRM Block Diagram

5.2.2 Signal Descriptions

Table 5-1 lists external signals associated with the CRM and gives their descriptions.

Table 5-1. CRM External Signals

Signal Names	Direction	Active	Comments
RESETB	Digital Input	Low	Asynchronous system reset
KBI_0_HST_WK	Digital Input/Output (Output only for reset & low power)	High	1) Can be used as a keyboard control output (can also be programmed as GPIO). 2) Can be used as a wake-up output when exiting sleep mode. 3) Retains power during reset and power-down.
KBI_3 - KBI_1	Digital Input/Output (Output only for reset & low power)	High	1) Can be used as a keyboard control outputs (can also be programmed as GPIO). 2) Retain power during reset and power-down.
KBI_7 - KBI_4	Digital Input/Output (Input only for reset & low power)	Low	1. When used as KBI inputs, the 4 signals can provide an asynchronous interrupt to exit sleep mode or as keyboard inputs (can also be programmed as GPIO). 2.) Retain power during reset and power-down.
XTAL_32_IN	Analog Input	-	Optional 32.768 kHz crystal oscillator input
XTAL_32_OUT	Analog Output	-	Optional 32.768 kHz crystal oscillator output

Table 5-1. CRM External Signals (continued)

Signal Names	Direction	Active	Comments
COIL_BK	Power Switch Output	-	Buck converter coil drive output
LREG_BK_FB	Power Input	-	1) Voltage input to onboard regulators 2) Buck regulator feedback voltage

5.2.3 Sleep Module

The sleep module of the CRM is always powered, integrated with the power-up sequence when exiting reset, and manages entering and exiting low power modes.

5.2.3.1 Modes of Operation

The Sleep Module assists in managing modes of operation.

- Reset active - this is the lowest power option for the MC1322x and only the Sleep Mode of the CRM is powered.
- Power Up from Reset

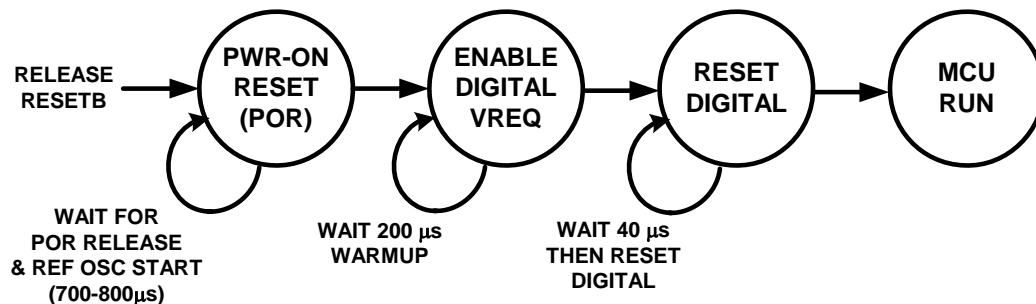


Figure 5-4. Power-up from Reset State Diagram

The Sleep module of the CRM performs power-up of the MCU after power on reset (POR). Figure 5-4 shows the sequence after release of RESETB. After POR, the reference oscillator is turned-on, and a wait occurs until the reference oscillator is ready (700usec). When the oscillator is ready, the full turn on of the Digital 1.2V supply can occur. A full digital reset is provided and then the reference clock (nominally 24MHz) is gated to the MCU core and the MCU core is fully functional.

- Normal Operation - In normal operation, the MCU core runs, and the MCU system clock runs at the frequency set by the prescaler (reference frequency divided by the prescaler ratio). Although not true modes, the device can run in various power configurations
 - Run / Idle - the CPU is running and the transceiver is idling and MC1322x can quickly enable transceiver operations. The MCU core and peripheral are running with full capability.
 - Transmit or Receive - similar to Run, this is not an actual CRM mode but is simply the MCU active and the transceiver active

- The MACA operation is independent of the CPU and uses DMA for moving data to/from RAM (stealing cycles away from the core). The MCUs effective run rate can be reduced lower via the CPU clock (as low as 2MHz for 802.15.4 mode)
- The Bus Steal Module (BSM) in the CRM can also be enabled to further reduce power
- Sleep Modes - There are two basic sleep modes, i.e., Hibernate and Doze

5.2.3.2 Sleep Operation

Because low power operation is extremely important to battery operation. The MC1322x has extensive resources to manage power and provide low power or “sleep” modes. There are 2 types of sleep mode that MC1322x can enter: Hibernate and Doze.

There is only difference between modes is that Doze mode keeps the reference oscillator alive to supply the sleep timing clock. The reference frequency is divided by 128 and supplied to the counters. Alternatively, in Hibernate the ring oscillator is used as a default and if the 32 kHz oscillator is available it can be used as the alternative. The 32 kHz option provides an accurate time base at lowest current because it is used in Hibernate. Using Doze allows an accurate time base with lower cost (no 32 kHz crystal is used) but at the expense of higher current while in low power mode.

For both modes:

- Most of the system is powered off, except for the selected oscillator, timer support logic, and the RAM0 page (additional RAM can be retained).
 - For Hibernate mode, the default oscillator is the ring oscillator. The 32 kHz crystal oscillator is an option if the crystal is present.
 - For Doze mode the reference oscillator divided-by-128 is the clock source.
- RAM retention is programmable
 - Memory is powered by 0.9V, but is not accessible (it just retains state).
 - Four combinations of RAM retention are possible. Each of the pages RAM1, RAM2, and RAM3 can be also enabled.
 - Low power current is increased as more RAM retains its state
- All the states of the MCU, Modem, and analog control can be retained.
 - If the MCU State Retention bit (MCU_RET) is set, the states of this logic are retained
 - Low power current is increased as the logic retains its state.
- The states of the GPIO can be retained
 - In low power mode, the GPIO normally are disconnected from the top voltage rail except for the KBI signals. Setting the DIG_PAD_EN bit causes the GPIO to stay powered in their programmed state. The KBI signals still revert to CRM KBI control.
 - The DIG_PAD_EN will be ignored if the MCU State retention bit is not set.
- Wake-up from sleep is possible from three sources
 - An external wake-up signal from KBI inputs KBI_7:KBI_4
 - The internal wake-up timer
 - The Real Time Clock time-out

Another option of auto ADC operation during sleep is available only in hibernate mode.

Table 5-2. Power Modes and Clocks

MCU Mode	MAX IDD Limit	CLOCKING					
		CPU	RAM	ROM	Periph	Digital Modem	CRM/ Sleep
			2kx32 6kx32 8kx32 8kx32	20kx32		24MHz from XTAL or PLL	2kHz/ 32kHz/ Ref Osc
Reset	0.3uA	OFF	OFF	OFF	OFF	OFF	OFF
Hibernate	1uA ¹	OFF	OFF	OFF	OFF	OFF	2kHz or 32kHz
Doze	23uA	OFF	OFF	OFF	OFF	OFF	Ref osc/ 128
Idle	0.9mA	Ref osc	Ref osc	Ref osc	Ref osc	OFF	Ref osc
Run	5mA	Ref osc	Ref osc	Ref osc	Ref osc	ON	Ref osc
Receive	20mA	Ref osc	Ref osc	Ref osc	Ref osc	ON	Ref osc
Transmit	20mA	Ref osc	Ref osc	Ref osc	Ref osc	ON	Ref osc

¹ Amount of RAM retention programmable - MAX IDD limit assumes ONLY the 2kx32 is ON

5.2.3.3 Sleep Module Clock Structure

The Sleep Module clock structure is shown in [Figure 5-5](#).

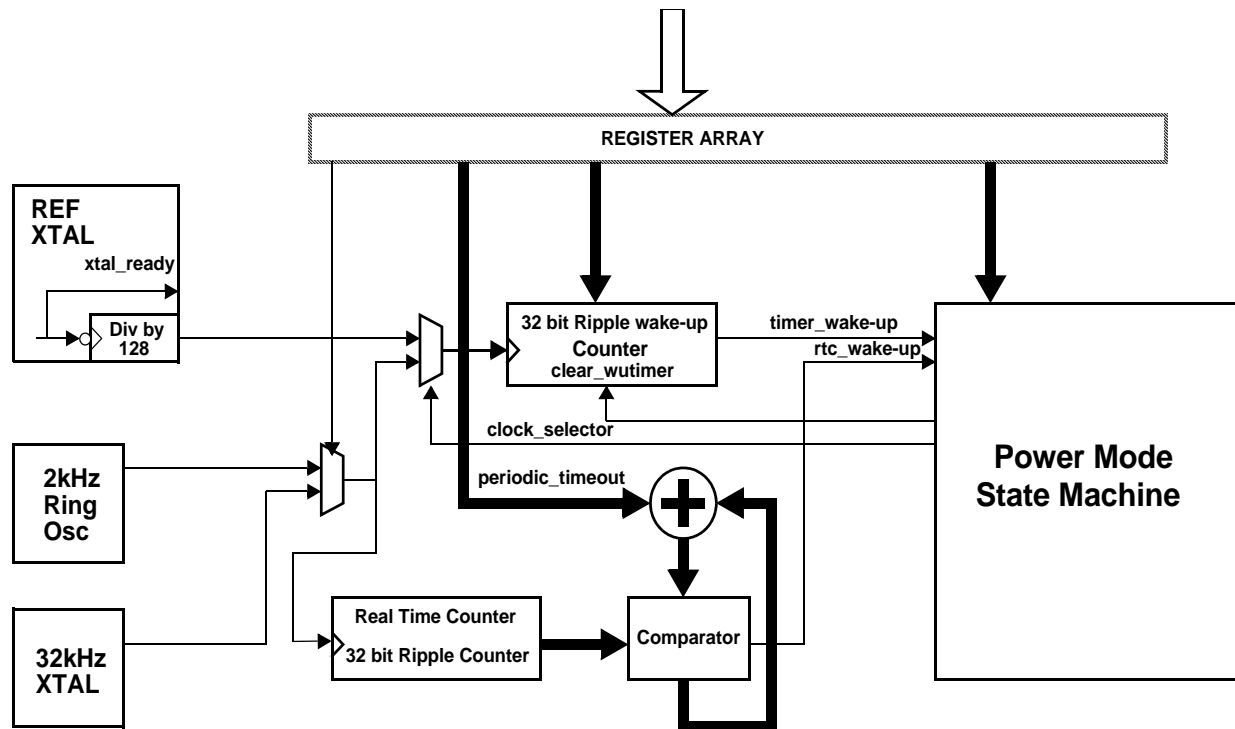


Figure 5-5. Sleep Module Clock Structure

The clock sources include a 2 kHz ring oscillator, an optional 32.768 kHz crystal oscillator, and the reference oscillator frequency divided-by 128. The RTC is only driven from the ring oscillator or the 32.768 kHz crystal oscillator (if present and enabled). The wake-up timer is driven from the available low power clock source, i.e., ring oscillator or crystal oscillator in Hibernate, and alternatively, the reference oscillator divided-by 128 in Doze.

5.2.3.3.1 2kHz Ring Oscillator

The 2kHz ring oscillator is the default clock source for the Sleep Module when coming out of reset or Hibernate for the Real Time Clock (RTC) and the wake-up counter. The specified default frequency accuracy of the oscillator over temperature is from 1 kHz to 4 kHz. However, onboard resources are available to trim the oscillator accuracy.

The accuracy of the ring oscillator for wake-up is typically unimportant. However, it may be desired to have a more accurate wake-up period from Hibernate. The CRM provides The Calibration Control Register (CAL_CNTL), Calibration XTAL Count Register (CAL_COUNT) and Ring Oscillator Control Register (RINGOSC_CNTL) for managing the ring oscillator.

The Ring Oscillator Control Register (see [Section 5.9.15, “Ring Oscillator Control \(RINGOSC_CNTL\)”](#)) provides:

- Ring Oscillator Enable bit - The default is the oscillator enabled exiting POR reset. If the 32.768 kHz oscillator is used the ring oscillator must be disabled via this bit.
- Frequency tuning (ROSC_CTUNE[3:0] and ROSC_FTUNE[4:0] fields) - The frequency of the oscillator can be adjusted by changing the loading capacitance. The ROSC_CTUNE[3:0] field (default 0x6) changes the loading in 1 pF steps and the ROSC_FTUNE[4:0] field (default 0x17) changes the loading in 160 fF steps.

The Calibration Control Register (see [Section 5.9.13, “Calibration Control Register \(CAL_CNTL\)”](#)) and Calibration XTAL Count Register (see [Section 5.9.14, “Calibration XTAL Count \(CAL_COUNT\)”](#)) provide the means for managing calibration.

- Calibration Control Register - provides the control for the calibration process. All fields typically would get written at one time
 - Calibration Enable bit (CAL_EN) - Default is zero or off. If the bit gets written to one, the calibration process starts. Writing a zero while the calibration process is in progress will abort the process.
 - Calibration Interrupt Enable bit (CAL_IEN) - Default is disabled. This will enable an interrupt from the CAL_DONE status when calibration cycle is complete.
 - Calibration Time-out Value field (CAL_TIMEOUT[15:0]) - The value in this field determines the number of ring oscillator counts in the calibration sequence. A maximum value equates to about 32.768 sec. assuming the oscillator is near 2 kHz.
- Calibration XTAL Count Register - provides a full 32-bit count value of the number of reference oscillator clocks during the calibration cycle.

The user can use the calibration cycle to tune the ring oscillator frequency. Enable the calibration cycle (at the same time setting the calibration time-out period). After the cycle is complete, read the XTAL count register and compare the result with the expected number of reference oscillator cycles. The oscillator loading can then be adjusted to correct the frequency. Repeat the process as needed.

5.2.3.3.2 32.768 kHz Crystal Oscillator

The 32.768 kHz option oscillator requires use of an external crystal (see [Section 3.6, “32.768 kHz Crystal Oscillator \(use optional\)”](#)). At the expense of the crystal, an accurate RTC timebase and wake-up timer can be provided. The CRM provides a control bit in the System Control Register (SYS_CNTL) and the 32kHz XTAL Control Register (XTAL32_CNTL) for managing the oscillator.

The System Control Register (see [Section 5.9.1, “System Control \(SYS_CNTL\)”](#)) includes the XTAL32_EXISTS control Bit 5.

- This bit is used to switch the internal clock circuitry AFTER the 32 kHz clock is available
- This bit should only be set to one after the 32 kHz oscillator is enabled and verified as running.

The 32kHz XTAL Control Register (see [Section 5.9.17, “32kHz XTAL Control \(XTAL32_CNTL\)”](#)) provides:

- Buffer Gain Control (XTAL32_GAIN[1:0] field) - This 2-bit field allows changing the buffer gain. Default is maximum gain and it is recommended that this field be left default.
- 32kHz Crystal Enable (XTAL32_EN bit) - Setting this bit to one enables the 32 kHz oscillator.

5.2.3.3.3 Switching from Default Ring Oscillator to 32 kHz Oscillator

Switching to the 32.768 kHz oscillator should consider the following:

Ring Oscillator Control Register (see [Section 5.9.15, “Ring Oscillator Control \(RINGOSC_CNTL\)”](#)) provides:

- The ring osc should be disabled first by clearing the Ring Oscillator Enable bit in the Ring Oscillator Control Register.
- The XTAL32_EN bit is then set in the 32kHz XTAL Control Register
- The 32 kHz may require up to 5 sec. to start. The CPU must wait to verify clock startup
- It is suggested to verify 32 kHz startup
 - Enable a timer for a 1 second time-out and interrupt
 - Wait for timer interrupt
 - Read RTC_COUNT value
 - Loop and repeat; exit loop after RTC value has been observed to change validating that the 32 kHz oscillator is clocking the RTC. (Engaging bus_steal can save power during the verification loop time)
- Set the XTAL32_EXISTS bit in the SYSTEM_CNTL register

NOTE

Only a hard or soft reset can turn the XTAL32 off.

5.2.3.3.4 Reference Oscillator Divided-by 128

The reference oscillator is kept alive in Doze low power mode. This has the disadvantage of higher current draw than Hibernate, but has the advantage that it allows for an accurate wake-up delay.

- During Doze the prescaler oscillator frequency (typically 187.5 kHz) drives the wake-up timer. As a result, the wake-up timer has a very accurate clock source for use in establishing the sleep period.
- The RTC continues to be clocked by the ring oscillator (or 32 kHz if enabled) during Doze
- The divide-by 128 clock cannot be used for an accurate RTC as the RTC can only be clocked from the ring oscillator or the 32 kHz oscillator.

NOTE

Typically, use of Doze mode has no advantage if the 32 kHz oscillator is available. The 32 kHz oscillator provides an accurate timebase for wake-up from Hibernate and does not require the higher current draw of Doze mode due to the reference oscillator running.

5.2.3.4 Wake-up Operation

Exiting reset, Hibernate or Doze modes, MC1322x will jump to Idle mode where the MCU will be fully functional and operates from the reference oscillator (the frequency is determined by the programmable prescaler; default is divide-by-1 or full speed). Communication with all mapped registers is possible including the digital modem and RF analog control. The transceiver is completely off. The Sleep module of the CRM gracefully controls this transition.

The Hibernate mode is initiated by setting the HIB bit in the Sleep Control Register. Software then must wait for the sleep time sync (SLEEP_SYNC) bit, do maintenance of timers like in MACA, and clear this status bit. Clearing this bit will power down the system almost immediately. Waiting for the SLEEP_SYNC can take up to 1 full cycle of the Hibernate clock. When awakened, read the CRM Status register to see if it was a wake-up from Hibernate.

The Doze mode is initiated by setting the DOZE bit in the Sleep Control Register. Software then must wait for the sleep time sync (SLEEP_SYNC) bit, do maintenance of timers like in MACA, and clear this status bit. Clearing this bit will power down the system almost immediately. Waiting for the SLEEP_SYNC can take up to 1 full cycle of the Doze clock. When awakened, read the CRM Status register to see if it was a wake-up from Doze.

5.2.3.5 Power Mode State Machine

The Power Mode State Machine properly guides MC1322x into and out of the sleep modes of Hibernation or Doze. This circuit controls the reference oscillator and power regulators.

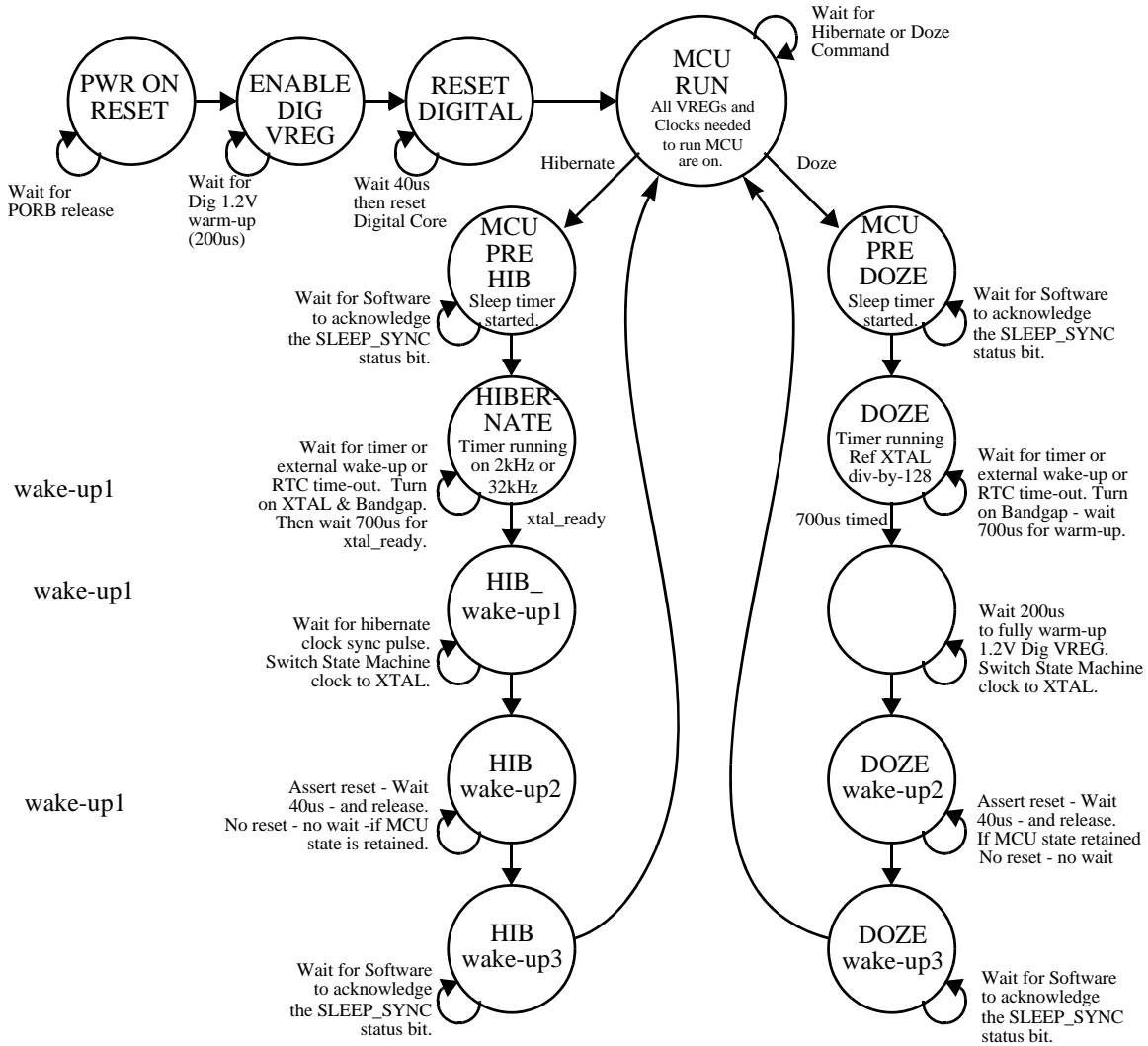


Figure 5-6. Power Mode State Diagram

Figure 5-4 shows the sequence for entering MCU RUN state from a “cold start” reset. There are two other main “loops” in the Power Mode State Machine, i.e., the Hibernate loop and the Doze Loop. These are entered from the MCU RUN state and are initiated via a hibernate or Doze command from the Sleep Control (SLEEP_CNTL). On assertion of the HIB or DOZE bits, a sleep mode sequence will be initiated. If HIB, then proceed to MCU_PRE_HIB; else if DOZE, then go to MCU_PRE_DOZE.

NOTE

In the following loop descriptions, the SLEEP_SYNC (Sleep Time Synchronizer) status bit is discussed. This status bit has the purpose of coordinating system timing and clocks during entering/exiting low power modes. Its use is discussed in Section 5.9.7, “Status (STATUS)” register, Bit 0. Software should use this bit for sleep and wake-up procedures.

Hibernate Loop

- **MCU_PRE_HIB** - This state is the transition state between "MCU RUN/Idle" mode and "Hibernate" mode. Upon entering this state, all clocks and power are fully on. MCU_PRE_HIB state stages the turn-off of clocks and power. Upon exit, all non-doze power supplies and clocks will be off.
 - Start the wake-up timer (from zero) and set the SLEEP_SYNC bit.
 - Wait for the SLEEP_SYNC bit to be cleared by software.
 - Freeze all MCU clocks and power down the system, except for the sleep timers and logic
- **HIBERNATE** - This state is the actual hibernate state. On entering this state, the Sleep Module is running solely on the Hibernate Clock.
 - Wait for wake-up - wake-up timer time-out, an external wake-up signal, or an RTC time-out.
 - On wake-up event, start wake-up - start main voltage reference, reference oscillator regulator and reference oscillator.
 - Wait for 700 μ sec. for reference oscillator ready.
- **HIB_wake-up1** - this is the first state for wake-up sequence after the reference oscillator is ready.
 - Warm-up and turn-on the digital 1.2V regulator
 - Synchronize and switch the state machine clock to the reference oscillator
 - The MCU and all the RAMs will be connected to the digital 1.2V regulator, but no clocks will be running to MCU.
 - Turn-on the pad ring (apply VBATT to GPIO buffers)
- **HIB_wake-up2** - this state provided to allow MCU reset if the MCU retention has NOT been enabled.
 - If MCU_RET bit is not set - apply a 40usec reset.
 - If the MCU_RET bit is set - no digital reset
 - Enable MCU clock.
- **HIB_wake-up3** - wait state before entering RUN condition
 - Set the SLEEP_SYNC status bit -which indicates the precise clock increment of the sleep timer
 - Wait for the SLEEP_SYNC bit to be cleared by software - on clearing the SLEEP_SYNC bit, the state machine turns on the MACA clock and returns to the MCU_RUN state.

Doze Loop

- **MCU_PRE_DOZE** - This state is the transition state between "MCU RUN/Idle" mode and "Doze" mode. Upon entering this state, all clocks and power are fully on. MCU_PRE_DOZE state stages the turn-off of clocks and power. Upon exit, all non-doze power supplies and clocks will be off with the exception of the reference oscillator
 - Start the wake-up timer and set the SLEEP_SYNC bit.
 - Wait for the SLEEP_SYNC bit to be cleared by software.
 - Freeze all MCU clocks and power down the system, except for the sleep timers (including reference oscillator) and logic

- **DOZE** - This state is the actual doze state. On entering this state, the Sleep Module is running solely on the Doze Clock (reference oscillator \div 128)
 - Wait for wake-up - wake-up timer time-out, an external wake-up signal, or an RTC time-out.
 - On wake-up event, start wake-up - start main voltage reference
 - Wait for 700 μ sec.
- **DOZE_wake-up1** - this is the first state for wake-up sequence after the reference oscillator is ready.
 - Warm-up and turn-on the digital 1.2V regulator
 - The MCU and all the RAMs will be connected to the digital 1.2V regulator, but no clocks will be running to MCU.
 - Turn-on the pad ring (apply VBATT to GPIO buffers)
- **DOZE_wake-up2** - this state provided to allow MCU reset if the MCU retention has NOT been enabled.
 - If MCU_RET bit is not set - apply a 40 μ sec reset.
 - If the MCU_RET bit is set - no digital reset
 - Enable MCU clock.
- **DOZE_wake-up3** - wait state before entering RUN condition
 - Set the SLEEP_SYNC status bit - which indicates the precise clock increment of the sleep timer
 - Wait for the SLEEP_SYNC bit to be cleared by software - on clearing the SLEEP_SYNC bit, the state machine turns on the MACA clock and returns to the MCU_RUN state.

5.2.3.6 Wake-up Timer

The wake-up timer is a 32-bit counter that is only used in Hibernate or Doze low power modes. The wake-up timer is clocked by the clock source available during the programmed sleep mode (see [Figure 5-5](#)):

- Hibernate enables the ring oscillator or optionally the 32.768 kHz crystal oscillator source
 - With the 2 kHz ring oscillator, maximum sleep time is ~24.85 days
 - With the 32.768 kHz oscillator, maximum sleep time is ~36.4 hours
- Doze enables the reference osc \div 128 (typically 187.5 kHz) - maximum sleep time is ~6.36 hours

The CRM provides the following interfaces to the wake-up counter:

- **TIMER_WU_EN** Bit (wake-up Control Register, Bit 0) - the wake-up timer enable bit activates wake-up timer option from Doze or Hibernate.
- **TIMER_WU_IEN** Bit (wake-up Control Register, Bit 16) - the wake-up timer interrupt enable bit activates the interrupt request from a wake-up timer event.
- **HIB_WU_EVT** Bit (Status Register, Bit 1) and **DOZE_WU_EVT** Bit (Status Register, Bit 2) - if either the wake-up timer or the RTC cause a wake-up event, the appropriate status bit will be set.
- **Wake-up Timeout Register (WU_TIMEOUT) Register** - sets the value for the timeout counter that triggers the time-out event.

- Wake-up Count Register (WU_COUNT) - reads the full 32-bit present value of the wake-up counter.

To use the wake-up timer to cause a wake-up event to exit low power mode:

1. Write wake-up Timeout Register with the desired value - the value must be based on the desired delay and the activated low power clock source.
2. Set the TIMER_WU_EN Bit to enable the wake-up timer.
3. Set the TIMER_WU_IEN Bit if an interrupt request is desired.
4. Enable the low power mode (after any other options are set) by setting either the HIB control bit (Bit 0) or the DOZE control bit (Bit 1) in the Sleep Control Register.

After wake-up from sleep, the HIB_WU_EVT status will be set if caused by the wake-up timer.

5.2.3.7 Real Time Counter (RTC)

The RTC is a 32-bit counter that can be used to maintain a continuous timebase (with time “tick”) and also provide a periodic wake-up from sleep. The RTC is continuously clocked by the selected low frequency clock source (see [Figure 5-5](#)), i.e., the ring oscillator or the 32.768 crystal oscillator (maximum time for the 32.768 oscillator is ~36.4 hours).

The CRM provides the following interfaces to the RTC:

- RTC_WU_EN Bit (wake-up Control Register, Bit 1) - the RTC timer enable bit activates the RTC time-out option. This can be used simply as a periodic interrupt (for a time “tick”) or as wake-up from Doze or Hibernate.
- RTC_WU_IEN Bit (wake-up Control Register, Bit 16) - the RTC time-out interrupt enable bit activates the interrupt request from a RTC timer event.
- RTC_WU_EVT Bit (Status Register, Bit 3) - if the RTC has a time-out event, this bit will be set.
- HIB_WU_EVT Bit (Status Register, Bit 1) and DOZE_WU_EVT Bit (Status Register, Bit 2) - if either the wake-up timer or the RTC cause a wake-up event, the appropriate status bit will be set.
- RTC Periodic Timeout Register (RTC_TIMEOUT) Register - sets the period between time-out events. As soon as the timeout period occurs, the next timeout point will be calculated from the present RTC count and saved.
- RTC Count Register (RTC_COUNT) - reads the full 32-bit present value of the wake-up counter.

5.2.3.8 External Wake-Up Event

A transition on any of the KBI_7 : KBI_4 signals can be enabled to asynchronously wake-up the device. Any or all 4 inputs can be enabled. The input transition can be programmed as to be level or edge sensitive to enable wake-up and generate an interrupt request to the CPU if desired. All the KBI signal (input and output) are unique in that they remain powered by default during sleep mode.

Use of the KBI signals as wake-up events is discussed in [Section 5.9.2, “Wake-up Control \(WU_CNTL\)”](#) and [Section 5.9.7, “Status \(STATUS\)”](#)

5.3 Managing Low Power Mode

The MC1322x has a number of options for low power operation (see [Section 5.2.3.2, “Sleep Operation”](#)). In addition to the two primary hibernate and doze modes, there are options for RAM retention, MCU state retention, GPIO pad state retention, auto ADC operation, and wake-up source. The options used and the interface to the CRM impact management of entering and recovery from low power operation.

5.3.1 Entering Low Power Mode

Management of entry into low power mode should consider:

- The user must consider impact of low power options (see [Section 3.8, “Low Power Considerations”](#)) on recovery from low power. Especially important is -
 - Amount of RAM retained - when returning from low power mode, the recovery time will be greatly impacted by how much RAM content needs reloaded.
 - MCU retention - when returning from low power with MCU retention enabled, the CPU will start from the condition or state at which it stopped and will continue. If retention is disabled (advantageous for lower power), the CPU will start executing from the bottom of RAM as in a start-up condition. Software must be able to recover properly, and good practice may be to save an image of system status (in retained RAM) before entering sleep.
- Options must be set prior to entry into the low power state (via setting either the HIB or DOZE bit in the Sleep Control Register)
- The Sleep Module state machine sets the SLEEP_SYNC bit (see [Section 5.2.3.5, “Power Mode State Machine”](#)) as the first step to enter low power mode. This function is to allow optional management of the MACA timer. **SOFTWARE MUST CLEAR THE SLEEP_SYNC BIT TO FULLY ENTER LOW POWER MODE.**

5.3.2 Recovery from Low Power Mode

Recovery from low power mode should consider:

- Similar to entering sleep, the Sleep Module state machine sets the SLEEP_SYNC bit as part of exiting sleep. This function is to allow optional management of the MACA timer. **SOFTWARE MUST CLEAR THE SLEEP_SYNC BIT TO FULLY EXIT LOW POWER MODE.**
- All enabled interrupts must be serviced as part of the recovery process.
- The options selected, the RAM retained, and the system state when entering sleep mode will impact how the recovery process is managed and the length of the recovery time.
- The Auto ADC mode requires special handling for recovery.

5.3.3 Auto ADC Mode

The MC1322x supports a version of Hibernate that allows the CRM Sleep Module to wake-up the system periodically and allow the ADC module to take samples independent of CPU activity. Conditions include:

- This mode is enabled by the AUTO_ADC control (Wake-up Control Register, Bit 3)
- Can be used in hibernate mode only (ring oscillator or 32 kHz oscillator is clock source).

- MCU state retention must be enabled (MCU_RET bit set). DIG_PAD_EN need not be set.

The suggested implementation flow is shown in [Figure 5-7](#).

- The AUTO_ADC and MCU_RET must be set before entering Hibernate.
- The RTC Timer is programmed for a periodic wake-up period that initiates the ADC activity.
- The ADC must be programmed with
 - Desired sample mode
 - Threshold(s) to which an input level will be compared
 - Interrupt enabled if the threshold is tripped.
- The wake-up timer is programmed with a value large enough to cover a desired number of periodic wakeups. In this manner, the wake-up timer serves as a watchdog timer in case the ADC activity never trips a programmed threshold.

In observing [Figure 5-7](#), after all proper initialization the hardware flow proceeds as follows:

1. **Start Hibernate** - Hibernate is entered.
2. **Sleep Time-out Event?** - The device will sleep until awakened
 - Wake-up timer - Go to #7 Normal MCU Wake-up (the wake-up timer is set for a much longer delay).
 - RTC periodic timeout - Go to #3 Wake-up MCU with CPU halted
3. **Wake-up MCU with CPU Halted** - The MCU_RET has kept the MCU powered. The CRM Sleep Module starts the reference oscillator. The bus steal module is used to hold the CPU in-active.
4. **Start ADC Module** - The Sleep module starts the ADC module which does one programmed sequence.
5. **ADC Interrupt?** - Threshold exceeded; interrupt set?
 - Yes - Go to #8 Unhalt CPU
 - No - Got to # 6 Re-enter Hibernate
6. **Re-enter Hibernate** - ADC signals Sleep Module that it has completed. Re-enter hibernate (stop reference osc); go to #2 Sleep Timeout.
7. **Normal MCU Wake-up** - Sleep module exits hibernate in a normal manner with wake-up timer status set.
8. **Unhalt CPU** - The ADC has asserted an interrupt request, the CPU is unhalted (bus steal module releases it), and the CPU will run.

NOTE

The CPU MUST clear the _AUTO_ADC, respond to the ADC interrupt, and then put the system back to sleep.

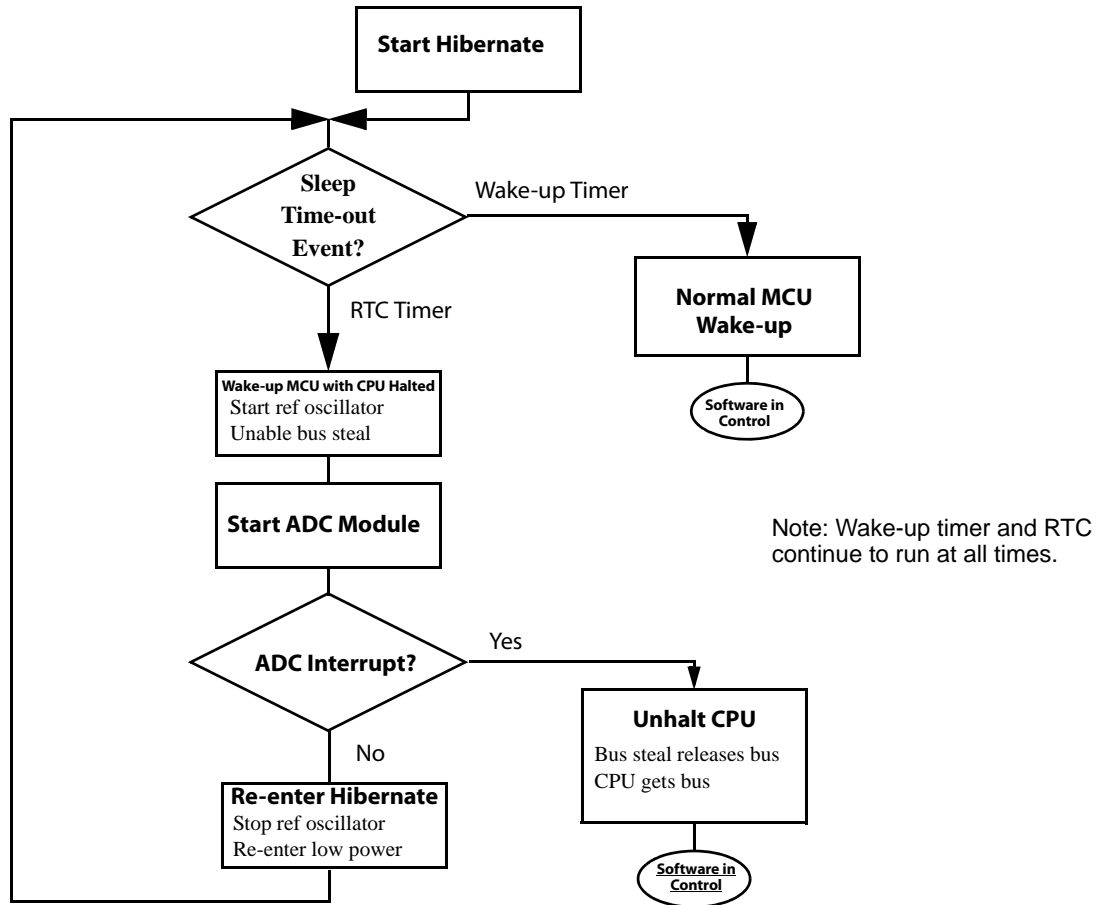


Figure 5-7. Auto ADC Flow Diagram

5.4 Managing Reference Oscillator

Use of the reference oscillator is discussed in [Section 3.5, “Reference Oscillator”](#). The CRM provides control of the reference oscillator via the Reference XTAL Control Register (see [Section 5.9.16, “Reference XTAL Control \(XTAL_CNTL\)”](#)). The reference oscillator uses a crystal of frequency from 13 - 26 MHz, although 24 MHz is typical and normally recommended.

To minimize system cost, all required crystal capacitive loading is provided onboard the MC1322x. The user, however, must trim the capacitance via the control register to meet the required frequency tolerance of ± 40 ppm over temperature.

NOTE

It is recommended that the software trim the crystal oscillator frequency to ± 10 ppm at room temperature to ensure the larger tolerance over temperature. Supply voltage variation to the device has little affect as the oscillator has its own onboard voltage regulator.

When exiting a POR reset condition, the default loading to the crystal is minimal (XTAL_CTUNE[4:0] = 0x00 and XTAL_FTUNE[4:0] = 0x00). Typically the crystal will require added capacitance to have the frequency within spec; the default frequency will be too high. Typical practice has found that once a hardware implementation has been characterized (with a given crystal), the settings for XTAL_CTUNE[4:0] = 0x00 and XTAL_FTUNE[4:0] can be pre-determined and used as initialization parameters.

NOTE

It is considered best practice that frequency accuracy be measured as part of a product's final test procedure during manufacture.

When waking from hibernate, the CRM has retained the programmed loading and starts the oscillator with the programmed loading. It need not be restored as part of the wake-up process.

5.5 Bus Steal Function

The bus steal function (see [Section 5.9.4, “Bus Stealing Control \(BS_CNTL\)”](#)) is primarily intended to lower average power (current) while allowing the CPU to run. The CPU bus has three possible masters, i.e., the CPU, the DMA for TX/RX, and the bus steal function. If enabled, the bus steal will “steal” clock cycles from the CPU causing the CPU to stay in a halted state to lower average device current. The bus steal function can be used in several modes:

- During a modem function only - if the BS_EN control bit is set, the bus steal will only steal cycles from the CPU during an active modem function. During this period, the DMA is functional and always gets the bus when needed as it has highest priority. The bus steal is inactive all other times. This mode helps limit current draw during peak radio/modem activity.
- Full time (manual enable) - if the BS_MAN_EN control bit is set, the bus steal will be functional continually until disabled. This allows a lower “idle” where little CPU activity is needed. If the DMA is active, it still has highest priority.
- Wait for Interrupt Request - if the BS_EN is active and the WAIT4IRQ control bit is active, the bus steal function is active until an interrupt request goes active; then the bus steal is released so the CPU runs at full bus speed
- During Auto ADC Mode - for AUTO_ADC mode, the bus steal function is used to halt the CPU while the ADC is active on a periodic basis. See [Section 5.3.3, “Auto ADC Mode”](#).

The bus steal does not totally shutdown the CPU, it steal cycles on a duty cycle determined by the ARM_OFF_TIME[5:0] control field. Also, the DMA has the highest priority to control the bus, so that the bus steal will release the bus on request from the DMA when data needs transferred.

[Figure 5-8](#) shows the bus steal state diagram.

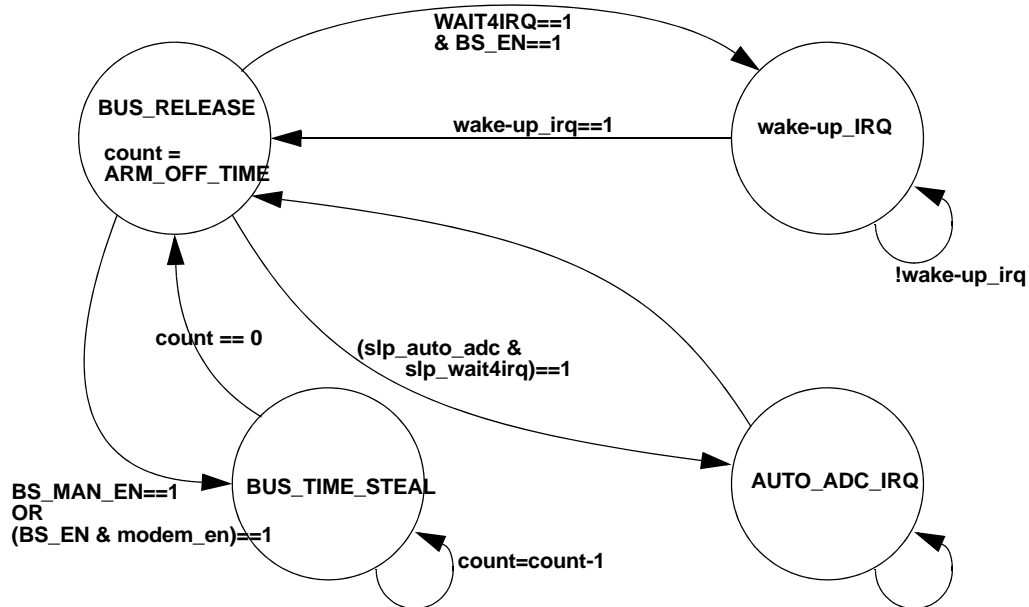


Figure 5-8. Bus Steal State Diagram

5.6 Computer Operating Properly (COP) or Watchdog Timer Module

The COP timer module can force a system reset or generate an interrupt if an application fails to maintain the timer through expected servicing. To prevent the COP time-out, the software must reset/service the COP timer periodically before the timer times out. If the application program gets lost and fails to reset the COP before it times out, a system reset is generated to force the system back to a known starting point, or alternatively, an interrupt can be generated to the CPU

Important features:

- The watchdog is by default disabled exiting device reset, and must be enabled for use.
- Programmable time out period (between 87ms and 11sec, in 128 steps).
- The watchdog is disabled for ARM debug mode
- The COP time-out can cause a complete reset POR power-up sequence or interrupt
- Programmable response to COP time-out (interrupt or reset, default is interrupt)
 - Cause a complete reset POR power-up sequence
 - Cause interrupt request
- COP does not run in sleep modes

The CRM provides the COP Control Register (COP_CNTL) and the COP Service Register (COP_SERVICE) for managing the oscillator.

The COP Control Register (see [Section 5.9.5, “COP Control \(COP_CNTL\)”](#)) includes:

- COP_TIMEOUT[6:0] field - sets the time-out value of the COP timer. The period can range from 0.087 to 11.187 seconds with a nominal 24 MHz reference clock

- CPO_COUNT[6:0] field - this is the present value of the COP timer. Every count represents 0.087 seconds (24 MHz clock)
- Three COP control bits -
 - COP_EN - COP enable bit
 - COP_OUT - selects desired COP time-out result, i.e., reset or interrupt request
 - COP_WP - locks present state of COP control. Can only be over-ridden with a system reset.

The COP Service Register (see [Section 5.9.6, “COP Service \(COP_SERVICE\)”](#)) is written to service/reset the COP timer.

- The value 0xc0DE5AFE must be written to reset the counter
- Writing the correct value also clears the interrupt request if that option is enabled (COP_OUT = 1)

Use of the COP reset option allows a higher level of system integrity for a deployed system, in that there is an extra level recovery is possible. Use of the interrupt option can be useful for helping debug software.

Concerning low power COP usage:

- If MCU retention is not used (MCU_RET = 0), the software does need to disable the COP before MC1322x enters sleep mode. This is because all registers will be reset (including the COP_CNTL) when exiting low power mode.
- If MCU retention is used (MCU_RET = 1), software must carefully maintain the COP because the COP counter will be frozen during sleep mode and will start counting again upon wake-up. The software must ensure that the COP does not time out before MC1322x has had a chance to wake up and reset the COP counter.

5.7 Interrupts

The CRM Module has a single interrupt request signal CRM_IRQ that is connected to the interrupt controller. The CRM_IRQ can be generated from a number of possible sources which are logically OR'ed such that any of them can generate an interrupt request if enabled. [Table 5-3](#) lists the CRM interrupt sources and their characteristics.

Table 5-3. CRM Interrupt Sources

Item	Status Bit	Mask Bit	Source Description	Interrupt Clear Mechanism
1	HIB_WU_EVT / DOZE_WU_EVT	TIMER_WU_IEN	Hibernate or Doze wake-up event due to wake-up timer	Writing a 1 to the proper HIB_WU_EVT or DOZE_WU_EVT status bit
2	RTC_WU_EVT	RTC_WU_IEN	RTC timeout event (wake-up or periodic)	Writing a 1 to the RTC_WU_EVT status bit
3	EXT_WU_EVT[3:0]	EXT_WU_EN[3:0]	<ul style="list-style-type: none"> • An external wake-up event has occurred on the associated KBI pin during low power mode • An external transition event has occurred while pin is programmed as GPIO input 	Writing a 1 to the associated EXT_WU_EVT status bit

Table 5-3. CRM Interrupt Sources

Item	Status Bit	Mask Bit	Source Description	Interrupt Clear Mechanism
4	CAL_DONE	CAL_IEN	Calibration period is complete for ring oscillator calibration	Writing a 1 to the CAL_DONE status bit
5	COP_EVT	COP_OUT	COP timer has timed out	Writing a 1 to the COP_EVT status bit or writing the correct value to the COP_SERVICE register

5.7.1 Wake-up Timer Time-out

The wake-up timer can be used along with the RTC and KBI inputs to exit either Hibernate or Doze mode. If the wake-up timer fires, the appropriate status bit, i.e., either HIB_WU_EVT or DOZE_WU_EVT will be set. An interrupt request will be generated for either if the TIMER_WU_IEN mask bit is set enabling the interrupt. The interrupt request is cleared by writing a one to the appropriate status bit.

5.7.2 Real Time Clock Time-out

The RTC timer can be used along with the wake-up timer and KBI inputs to exit either Hibernate or Doze mode. If the RTC timer fires, the RTC_WU_EVT status bit will be set. An interrupt request will be generated if the RTC_WU_IEN mask bit is set enabling the interrupt. The interrupt request is cleared by writing a one to RTC_WU_EVT status bit.

5.7.3 External wake-up Signals KBI_7:KBI_4

The GPIO designated as KBI_7, KBI_6, KBI_5, and KBI_4 can individually be programmed to generate an interrupt request on a level change or edge transition. If this occurs during low power mode, it can wake-up the MC1322x, and alternatively, the interrupt requests can be enabled during normal run mode. If an external signal transition occurs the appropriate bit in the EXT_WU_EVT[3:0] status field will be set. An interrupt request will be generated if the mask bit in the EXT_WU_EN[3:0] field is set enabling the interrupt. The interrupt request is cleared by writing a one to the appropriate bit in the EXT_WU_EVT[3:0] status field.

5.7.4 Ring Oscillator Calibration Done

Calibration of the 2 kHz oscillator can take a significant amount of time. The CAL_DONE status signifies the completion of the calibration cycle. This event can initiate an interrupt request if enabled with the CAL_IEN bit. The interrupt request is cleared by writing a one to the CAL_DONE status.

5.7.5 COP Time-out

The COP timer can be used either for code debug or recovery from a “run-away” code situation. When the COP_EVT status bit is set the COP timer has expired, and this status can generate an interrupt request

instead of a system reset if enabled by the COP_OUT control bit. The interrupt request is cleared by either serving the COP_SERVICE register or by writing a one to the COP_EVT status.

5.8 CRM Register Memory Map

The CRM Module is programmed via a set of memory-mapped registers

- The base address is **0x80003000**.
- The register memory map for the module is listed in [Table 5-4](#)
- [Table 5-4](#) also shows which registers are retained in sleep mode

Table 5-4. CRM Module Register Memory Map

Address	Name	Access Type	Retained in Sleep Mode	Width (Bits)	Access Width
Base + 0x00	System Control (SYS_CNTRL)	R/W	Yes	32	All
Base + 0x04	Wake-up Control (WU_CNTRL)	R/W	Yes	32	All
Base + 0x08	Sleep Control (SLEEP_CNTRL)	R/W	Yes	32	All
Base + 0x0C	Bus Stealing Control (BS_CNTRL)	R/W	No	32	All
Base + 0x10	COP Control (COP_CNTRL)	R/W	No	32	All
Base + 0x14	COP Service (COP_SERVICE)	R/W	No	32	32 only
Base + 0x18	Status (STATUS)	R/W	No	32	All
Base + 0x1C	Module Enable Status (MOD_STATUS)	R	No	32	All
Base + 0x20	Wake-up Count (WU_COUNT)	R	Yes	32	32 only
Base + 0x24	Wake-up Timer Compare (WU_TIMEOUT)	R/W	Yes	32	32 only
Base + 0x28	Real Time Count (RTC_COUNT)	R/W	Yes	32	32 only
Base + 0x2C	RTC Periodic Wake-up Time-out (RTC_TIMEOUT)	R/W	Yes	32	32 only
Base + 0x30	Reserved				
Base + 0x34	Calibration Timer Set (CAL_CNTRL)	R/W	No	32	16/32 only
Base + 0x38	Calibration XTAL Count (CAL_COUNT)	R	No	32	All
Base + 0x3C	2kHz Ring Oscillator Control (RINGOSC_CNTRL)	R/W	Yes	32	16/32 only
Base + 0x40	Reference XTAL Control (XTAL_CNTRL)	R/W	Yes	32	16/32 only
Base + 0x44	32kHz XTAL Control (XTAL32_CNTRL)	R/W	Yes	32	All
Base + 0x48	Voltage Regulator Control (VREG_CNTRL)	R/W	No	32	All
Base + 0x4C	Reserved				
Base + 0x50	Software Reset (SW_RST)	R/W	No	32	32 only
Base + 0x54	Reserved				
Base + 0x58	Reserved				
Base + 0x5C	Reserved				

5.9 CRM Registers and Control Bits

5.9.1 System Control (SYS_CNTL)

The SYS_CNTL register is used to control and configure the system for power supply, clock rate, and other device options. The register contents are retained in sleep mode.

	SYS_CNTL								Base+0x00								
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16	
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0	
			XTAL_CLKDIV[5:0]									XTAL32_EXISTS	JTAG_SECU_OFF	SPIF_1P8V_SEL	PADS_1P8V_SEL	PWR_SOURCE[1:0]	
TYPE	r	r	rw	rw	rw	rw	rw	rw	r	r	r	rw	rw	rw	rw	rw	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	

Table 5-5. SYS_CNTL Register Bit Descriptions

Bit Number	Description	Operation
14-31	Reserved	
8-13	<p>xtal_clkdiv[5:0] — Reference oscillator divider (prescaler). This value determines the ratio of the prescaler that divides the reference oscillator frequency to the entire MCU (both CPU core and peripheral source clocks).</p> <ul style="list-style-type: none"> It does not affect the rate of the Modem clock. Divide ratio = xtal_clkdiv[5:0] + 1 Maximum divisor is 64 All baud rates and peripheral clocks must be determined from this base frequency 	Default = 0x00; divide ratio = 1 (MCU clocks run at reference oscillator rate)
6-7	Reserved	
5	<p>XTAL32_EXISTS — 32kHz crystal exists. Setting this bit selects the 32kHz oscillator to the Sleep Module mux</p> <ul style="list-style-type: none"> This bit will be set to 1 after the 32kHz is operational after enabling the 32kHz XTAL (with XTAL32_EN during IDLE mode) A time of 1 to 5 second delay can occur before this oscillator is ready. On system reset, this 32kHz XTAL must be restarted. 	0 = Disable (default) 1 = 32 kHz enabled

Table 5-5. SYS_CNTL Register Bit Descriptions (continued)

Bit Number	Description	Operation
4	JTAG_SECU_OFF — JTAG Security Disable. This bit gets set by the ROM boot code unless the FLASH image is secured. See Section 3.9.6, “Secure Mode” .	1 = The JTAG and NEXUS modules enabled. 0 = The JTAG and NEXUS modules disabled (default).
3	Reserved	
2	PADS_1P8V_SEL - Output Drive Selector for all Digital Pads. This bit selects whether the digital pads are configured for high drive or standard drive	0 = Standard drive (default) 1 = High drive
0-1	PWR_SOURCE[1:0] — Selection of System Power Source. This value selects the type of supply that will be powering the system. This value will be retained until a hard or soft reset. On POR start-up, a VBATT battery supply is assumed unless over-programmed. On any subsequent wake-up from hibernate or doze, the system will immediately use the over-programmed supply configuration	0x0 = Default (VBATT) See Table 5-6

Table 5-6. Power Supply Selections

PWR_SOURCE	Power Source	Detail
2'b00	VBATT	Any enabling of the Buck regulator will be ignored. (default)
2'b01	Buck Regulation	Enables buck regulator
2'b10	External regulated 1.8V	Enabling the Buck or the 1.8V regulator will be ignored.
2'b11	Reserved	N/A

NOTE

If an external regulated 1.8V supply is selected instead of VBATT battery, then on any subsequent wake-up from hibernate or doze, the system will immediately use the correct voltage supply configuration.

- Using an external 1.8V supply will cause the enabling of the Buck Regulator to be ignored.
- If Bit 1 is set while the BUCK_EN is set, then the external 1.8V supply will not be enabled.

5.9.2 Wake-up Control (WU_CNTL)

The WU_CNTL register is used to enable/disable the wake-up events which can come from an external signal or from internal timers. For external wake-ups signals, the polarity and edge/level sense can be programmed for the wake-up signal, and for these or the timer wake-up events, an interrupt request can be generated if enabled.

The control fields EXT_WU_IEN[3:0], EXT_WU_EDGE[3:0], and EXT_WU_EN[3:0] are always active. This allows an interrupt request to be generated based on an input transition from a keyboard or pushbutton transition, even during run mode.

- These fields control the buffer pad logic during low power modes, but the GPIO Module controls the pad logic during run mode.
- During run mode, a pad must be programmed as a GPIO input and a pulldown or pullup enabled (as appropriate) via the GPIO Module to make use of the interrupt request capability.

The register contents are retained in sleep mode.

	WU_CTRL								Base+0x04							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	EXT_OUT_POL								EXT_WU_IEN[3:0]						RTC_WU_IEN	TIMER_WU_IEN
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	r	r	rw	rw
RESET	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	EXT_WU_POL[3:0]				EXT_WU_EDGE[3:0]				EXT_WU_EN[3:0]				AUTO_ADC	HOST_WAKE	RTC_WU_EN	TIMER_WU_EN
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-7. WU_CNTL Register Bit Descriptions

Bit Number	Description	Operation
28-31	EXT_OUT_POL[3:0] — External Wake-up Output Polarity (KBI 3, 2, 1, 0, respectively). These bits set the output state of the 4 external output drivers (i.e. KBI[3:0]) during sleep. The state of the output drivers equal the states of EXT_WU_POL[3:0]. Once awake, the GPIO controller takes control of these pins.	1 = External driver polarity high (Default) 0 = External driver polarity low.
24-27	Reserved	

Table 5-7. WU_CNTL Register Bit Descriptions (continued)

Bit Number	Description	Operation
20-23	<p>EXT_WU_IEN[3:0] — External Wake-up Interrupt Enable (KBI 7, 6, 5, 4, respectively). These bits enable the 4 external wake-up interrupts (KBI[7:4]) respectively.</p> <ul style="list-style-type: none"> • If disabled, no any interrupt request is generated • The status bit in CRM STATUS register will be asserted and must be cleared. • The status/interrupt request will be cleared immediately upon servicing 	<p>1 = External wake-up Interrupt request enabled 0 = External wake-up Interrupt request disabled (Default)</p>
18-19	Reserved	
17	<p>RTC_WU_IEN — Real Time Clock Wake-up Interrupt Enable. This bit enables the RTC timer compare interrupt request.</p> <ul style="list-style-type: none"> • If disabled, no any interrupt request is generated • The status bit in CRM STATUS register will be asserted and must be cleared. • The status/interrupt request will persist for 2 cycles of the low frequency hibernate clock after servicing. <p>Note: This may require that this interrupt be disabled in the RTI for these 2 cycles or until the status reads as clear.</p>	<p>1 = Enable RTC interrupt request 0 = Disabled. (Default)</p>
16	<p>TIMER_WU_IEN — Timer wake-up Interrupt Enable. This bit enables the sleep timer compare interrupt for hibernate or doze.</p> <ul style="list-style-type: none"> • If disabled, no interrupt, request is generated. • The HIB_WU_EVT bit or DOZE_WU_EVT bit in CRM STATUS register will be asserted and must be cleared • To service this interrupt request - a) service the status bit, b) disable the TIMER_WU_IEN until SLEEP_SYNC is cleared (the interrupt request will be active until the SLEEP_SYNC is cleared),. d) HIB_WU_EVT status bit can be cleared simultaneously with SLEEP_SYNC. 	<p>1 = Enable Wake-up timer interrupt request 0 = Disabled. (Default)</p>
12-15	<p>EXT_WU_POL[3:0] — External Wake-up Polarity (KBI 7, 6, 5, 4, respectively). These polarity bits control the detection state of the external wake-up inputs (i.e. KBI[7:4]).</p> <ul style="list-style-type: none"> • In low power mode, the KBI 7:4 always revert to inputs. • A pullup is enabled if a low level/negative edge is selected (bit = 0) • A pulldown is enabled if a high level/positive edge is selected (bit = 1), which is the default • During run mode these bits still function, but do not control the pad direction or pullup/pulldown. <p>Note: EXT_WU_POL[x] should only be changed when EXT_WU_EN[x] is disabled otherwise a false wake-up may be activated.</p>	<p>1 = Detect High level or positive edge. (Default) 0 = Detect Low level or negative edge.</p>

Table 5-7. WU_CNTL Register Bit Descriptions (continued)

Bit Number	Description	Operation
8-11	<p>EXT_WU_EDGE[3:0] — External Wake-up Edge or Level Sense (KBI 7, 6, 5, 4, respectively). These bits select whether external wake-up events will be edge or level detected.</p> <ul style="list-style-type: none"> If level sense is selected, wake-up will occur whenever EXT_WU_EN[x] is set to 1 and the state of the external wake-up signal matches the EXT_WU_POL[x] bit. To clear the interrupt request, the external pin must be returned to the inactive state and then the status bit, EXT_WU_STAT can be cleared. If edge sense is selected, then the wake-up will be asserted after an active edge is detected on the external wake-up pin. The status bit can be cleared at any time. <p>Note: The pulse width of the signal must be at least 2 clocks of whatever clock is running the edge detector. The clocks can be 2kHz or 32kHz in Hibernate or the high speed XTAL/128 (nominal 24MHz/128 = 187.5kHz) in Doze.</p>	<p>1 = Edge sensitive. 0 = Level sensitive. (Default)</p>
4-7	<p>EXT_WU_EN[3:0] — External wake-up Enables (KBI 7, 6, 5, 4, respectively). These bits control which of the 4 external signals are enabled to wake-up the MC1322x from hibernate or doze.</p> <p>Note: If none of the external wake-ups are enabled, a timer source must be selected as the wake-up source.</p>	<p>1 = External wake-up enabled 0 = External wake-up Disabled (Default)</p>
3	<p>AUTO_ADC — Enable the Autonomous ADC Mode.</p> <ul style="list-style-type: none"> This bit enables the CRM to wake-up only the ADC to take samples and then go back to sleep. This wake/sleep process will happen independently of the CPU (i.e. the ARM will be in a "halt" state) unless the ADC generates an interrupt request. If the ADC does issue an interrupt request, the CPU MUST clear AUTO_ADC and then tend to the ADC. The CPU also needs to put the system back to sleep. This mode can only be used in hibernate with MCU State Retention mode MCU_RET=1(DIG_PAD_EN need not be set). It is suggested that this mode be used with the RTC Timer with a periodic wakeup. It is also suggested that the Wake-up Timer be programmed with a value large enough to cover the number of desired periodic wakeups. Here the Wake-up Timer serves as a watchdog timer in case the ADC samples never trip a programmed threshold. 	<p>1 = Enable the Autonomous ADC Mode 0 = Disable the Autonomous ADC Mode (default)</p>
2	<p>HOST_WAKE — Enable Master Wake pin. This bit enables MC1322x to be a host wake-up device for a system.</p> <ul style="list-style-type: none"> This bit enables pin KBI0/GPIO22 as a wake-up output. While sleeping, KBI0 will be 0 and upon wake-up, KBI0 will transition to 1. While set to 1, HOST_WAKE will take priority over any programming on GPIO22. Clearing HOST_WAKE to zero will allow the GPIO22 to be used as general purpose output again. The initial enabling of HOST_WAKE will cause the Host-Wake pin to go from 0 to 1 assuming that the GPIO22 had been programmed to 0. 	<p>1 = Enable Host-Wake function/pin 0 = Disable Host-Wake Function/pin (default)</p>

Table 5-7. WU_CNTL Register Bit Descriptions (continued)

Bit Number	Description	Operation
1	RTC_WU_EN — Real Time Clock Wake-up Enable. This bit enables the Real Time Clock (RTC) timer compare for exiting low power mode.	1 = Enabled. 0 = Disabled. (Default)
0	TIMER_WU_EN — Timer wake-up Enable. This bit enables the wake-up timer compare for exiting low power mode.	1 = Enabled. 0 = Disabled. (Default)

5.9.3 Sleep Control (SLEEP_CNTL)

The SLEEP_CNTL register is used to control the various sleep options of the system including RAM retention, MCU retention, and GPIO pad retention. as well as, enable sleep mode (hibernate vs. doze). The register contents are retained in sleep mode.

	SLEEP_CNTL								Base+0x08							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									DIG_PAD_EN	MCU_RET	RAM_RET[1:0]				DOZE	HIB
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	r	r	wr0	wr0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-8. SLEEP_CNTL Register Bit Descriptions

Bit Number	Description	Operation
8-31	Reserved	
7	<p>DIG_PAD_EN — Enable Digital (GPIO) Pad Power. This bit enables retention of power to the digital I/O pad ring (all pads associated with GPIOs) in sleep mode.</p> <ul style="list-style-type: none"> The DIG_PAD_EN will only be acted on if MCU_RET is also set true. Typically, the pad ring is turned off for power savings in sleep. <p>Note: KBI_0 through KBI_7 are always powered.</p>	<p>1 = Enable pad power retention 0 = Disable pad power retention (default)</p>
6	<p>MCU_RET — MCU State Retention. This bit enables all the states of the MCU, Modem, and Analog Control to be saved during sleep mode.</p>	<p>1 = Enable MCU state retention 0 = Disable MCU state retention (default)</p>
4-5	<p>RAM_RET[1:0] — RAM page Retention. These bits select the amount of RAM that is retained during sleep mode. RAM is selected by Page 0 (8kbytes), Page 1 (24kbytes), Page 2 (32kbytes) and Page 3 (32kbytes)</p>	<p>2'b00: 8k bytes (Default - Page 0 only retained) 2'b01: 32k bytes total (Pages 0 & 1) 2'b10: 64k bytes total (Pages 0, 1, & 2) 2'b11: 96k bytes total (All pages)</p>
2-3	Reserved	

Table 5-8. SLEEP_CNTL Register Bit Descriptions (continued)

Bit Number	Description	Operation
1	<p>DOZE — Put system into Doze Mode. Writing a 1 to this bit will start the power down sequence for the entire chip except for the doze timer which will be running at the REF XTAL frequency divided by 128.</p> <ul style="list-style-type: none"> • A read of this bit always returns zero. • Once this bit is set, hardware synchronizes the Bus clock with the Doze clock - this could take up to 2 cycles of the Doze clock. • During this synchronization period, software must poll the SLEEP_SYNC bit in the STATUS register. Once the SLEEP_SYNC bit is set, software can read and store the MACA timer and do any other Doze tasks that need completed before fully entering doze. • Finally, the software must clear the SLEEP_SYNC bit. This allows final complete power-down of the MCU. 	<p>1 = Enable start of Doze sequence 0 = Doze not enabled (default) (Always reads as 0)</p>
0	<p>HIB — Put system into Hibernate Mode. Writing a 1 to this bit will start the power down sequence for the entire chip except for the hibernate timer which will be running at 2kHz from the internal ring oscillator or optionally 32kHz XTAL.</p> <ul style="list-style-type: none"> • A read of this bit always returns zero. • Once this bit is set, the hardware will synchronize the Bus clock with the Hibernate clock - this could take up to 2 cycles of the Hibernate clock. • During this synchronization period, software must poll the SLEEP_SYNC bit in the STATUS register. Once the SLEEP_SYNC bit is set, software can read and store the MACA timer and do any other Hibernate tasks that need completed before fully entering hibernate. • Finally, the software must clear the SLEEP_SYNC bit. This allows final complete power-down of the MCU. 	<p>1 = Enable start of Hibernate sequence 0 = Hibernate not enabled (default) (Always reads as 0)</p>

5.9.4 Bus Stealing Control (BS_CNTL)

The CPU bus has three possible masters, i.e., the CPU, the DMA for TX/RX, and the bus steal function. If enabled, the bus steal module will “steal” clock cycles from the CPU causing the CPU to stay in a halted state to lower average device current. The bus steal function can be used in several modes (see [Section 5.5, “Bus Steal Function”](#)):

- During a modem function only - if the BS_EN control bit is set, the bus steal will only steal cycles from the CPU during an active modem function.
- Full time (manual enable) - if the BS_MAN_EN control bit is set, the bus steal will be functional continually until disabled.
- Wait for Interrupt Request - if the BS_EN is active and the WAIT4IRQ control bit is active, the bus steal function is active until an interrupt request goes active; then the bus steal is released so the CPU runs at full bus speed.
- During Auto ADC Mode - for AUTO_ADC mode, the bus steal function is used to halt the CPU while the ADC is active on a periodic basis. See [Section 5.3.3, “Auto ADC Mode”](#).

The bus steal does not totally shutdown the CPU, it steal cycles on a duty cycle determined by the ARM_OFF_TIME[5:0] control field. Also, the DMA has the highest priority to control the bus, so that the bus steal will release the bus on request from the DMA when data needs transferred.

This register controls the bus steal function, and the register contents are not retained in sleep mode.

	BS_CNTL								Base+0x0C							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			ARM_OFF_TIME[5:0]											BS_MAN_EN	WAIT4IRQ	BS_EN
TYPE	r	r	rw	rw	rw	rw	rw	rw	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-9. BS_CNTL Register Bit Descriptions

Bit Number	Description	Operation
14-31	Reserved	
8-13	ARM_OFF_TIME[5:0] — Bus Steal Cycles. This value determines the number of clock cycles to steal away from the CPU. The maximum number of cycles to steal is 64 (value = 0). The bus is stolen for N cycles (value); then the CPU gets the bus for 2 cycles. The bus is then stolen again for another N cycles.	0x00 = Default (64 cycles) 0x02 = 2 cycles (50% duty cycle) 0x1F = 63 cycles (CPU active 4.6%)
3-4	Reserved	
2	BS_MAN_EN — Manual Bus Stealing Enable. The bit enables the bus stealing directly (without need for transceiver activity) through software. <ul style="list-style-type: none"> • This bit has higher priority than BS_EN. • Bus stealing is activated immediately when this bit is set (ARM_OFF_TIME should also be set). 	1 = Immediately enable the bus stealing. 0 = Disable the bus stealing (default) Note: Disabling can take longer (based on ARM_OFF_TIME value) because the ARM is being stalled from executing code.
1	WAIT4IRQ — Wait for Interrupt Request. Setting this bit to 1 halts the CPU and makes it wait for an interrupt to be released. <ul style="list-style-type: none"> • This bit control is only available when the BS_EN is true • Read back of this bit will always be zero - bit is self clearing. 	1 = Halt CPU and wait for interrupt. 0 = Wait for interrupt in-active (default)
0	BS_EN — Bus Stealer Enable. This bit allows bus steal to automatically start and steal CPU cycles whenever the transceiver is active. The bus steal automatically stops when the transceiver operation is complete. <ul style="list-style-type: none"> • ARM_OFF_TIME should also be set when enabling BS_EN • This bit also enables a “Wait for Interrupt” (WAIT4IRQ) where the ARM is halted and needs an interrupt to be freed. 	1 = Enable the automatic bus steal 0 = Disable bus steal (default) Note: Disabling can take longer (based on ARM_OFF_TIME value) because the CPU is being stalled from executing code.

5.9.5 COP Control (COP_CNTL)

The COP Control register configures the watchdog timer which allows recovery from a runaway code situation. The COP timer is clocked by the reference oscillator and the delay period is programmable from 87 ms to 11 seconds (with 24 MHz reference clock). The register contents are not retained in sleep mode.

	COP CONTROL								Base+0x10								
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16	
										COP_COUNT[6:0]							
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0	
		COP_TIMEOUT[6:0]												COP_WP	COP_OUT	COP_EN	
TYPE	r	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	rw	rw	rw	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 5-10. COP_CNTL Register Bit Descriptions

Bit Number	Description	Operation
23-31	Reserved	
16-22	COP_COUNT[6:0] — COP Counter Value. This read-only value is the running count in the COP timer. Every count of the COP timer is 87msec when running at 24MHz.	0x00 = Default
15	Reserved	
8-14	COP_TIMEOUT[6:0] — COP Time-out Period. The value in this register determines the time-out period of the COP counter. The COP counter counts reference oscillator clocks (nominally 24MHz).	See Table 5-11 . 0x00 = Default
3-7	Reserved	
2	COP_WP - COP Write Protect. Setting this bit disables a write to the COP CONTROL register. Once set, the register becomes read only, and this condition can only be cleared via a reset	1 = Protection set; register is read only 0 = Protection not set; register is read/write (default)
1	COP_OUT - COP Output. This bit selects the result of a COP time-out; 1) interrupt request, or 2) system reset	1 = COP time-out generates an interrupt request 0 = COP time-out generates a system reset (default)
0	COP_EN - COP Enable. The bit enables operation of the COP. THIS BIT CAN ONLY BE CHANGED WHEN COP_WP IS SET TO ZERO	1 = COP enabled 0 = COP disabled (default)

To determine the required COP_TIMEOUT[6:0] field setting for a desired time-out period, see [Table 5-11](#). With a 24 MHz reference frequency (typical usage) the time-out period can vary from 0.087 to 11.18 seconds.

Table 5-11. COP Time-Out Values

Value (hex)	Timeout Period Formula	Time out Period @ 24MHz (sec)
0x00	$\frac{(\text{Value} + 1) \times 2^{(21)}}{\text{RefXTALFreq}}$	0.087
0x01		0.175
0x02		0.262
0x03		0.350
		..
		..
0x7F		11.18

The following additionally describes use of the COP_CNTL register:

- To read the COP time, the COP_COUNT[6:0] field can be read.
- COP_TIMEOUT[6:0] should be written before the COP is enabled.

NOTE

Once the COP has been enabled, the recommended procedure for changing the COP_TIMEOUT[6:0] is:

- Disable the COP
- Write to COP_TIMEOUT
- Re-enable the COP.

The COP counter is not reset by a write to COP_TIMEOUT. Changing TIMEOUT while the COP is enabled will result in a time-out period that differs from the expected value.

- COP_CNTL bits cannot be changed after COP_WP is set to 1 until a reset occurs.
- In Hibernate or Doze modes, the COP is disabled along with the entire MCU. However, if the state of the MCU is retained, then the COP counter will also be retained, although frozen during sleep. When the core awakens, the COP will continue counting where it was stopped before entering the sleep mode.

5.9.6 COP Service (COP_SERVICE)

The COP Service register must be periodically written (serviced) before the COP timer times out. Use 32-bit access (read and write) only. The register contents are not retained in sleep mode.

- Writing the value 0xC0DE5AFE resets the COP timer.
- If the interrupt output option is selected (COP_OUT == 1) then the correct service write will clear the interrupt.
- Reading the register always returns 0xC0DE5AFE.

COP SERVICE									Base+0x14							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
COP_SERVICE[31:16]																
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	1	1	0	0	0	0	0	0	1	1	0	1	1	1	1	0
COP_SERVICE[15:0]																
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	1	0	1	1	0	1	0	1	1	1	1	1	1	1	0

Table 5-12. COP SERVICE Register Bit Descriptions

Bit Number	Description	Operation
0-31	COP_SERVICE[31:0] - COP Service Register. The register must be written with 0xC0DE5AFE to reset COP timer	Default read = 0xC0DE5AFE

5.9.7 Status (STATUS)

The CRM Status register reports CRM events and status. Reading this register does not alter the values of the status bits, however, writing a 1 to certain status bits clears them (in Table 5-13, these bits are signified by notation “rw1c”). The register contents are not retained in sleep mode.

	STATUS								Base+0x18							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
													VREG_1P5V_RDY	VREG_1P8V_RDY	VREG_BUCK_RDY	
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
						COP_EVT	CAL_DONE		EXT_WU_EVT[3:0]			RTC_WU_EVT	DOZE_WU_EVT	HIB_WU_EVT	SLEEP_SYNC	
TYPE	r	r	r	r	r	r	rw1c	r	rw1c	rw1c	rw1c	rw1c	rw1c	rw1c	rw1c	rw1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-13. STATUS Register Bit Descriptions

Bit Number	Description	Operation
20-31	Reserved	
19	<p>VREG_1P5V_RDY — 1.5V Analog Voltage Regulator (for transceiver) Ready Status. This status bit will be 1 when the 1.5V Regulator has been given enough time to warm up.</p> <ul style="list-style-type: none"> The warm up time is 600us at 26MHz if RX/TX and PLL have to be turned on or 200us if just the 1.5V Regulator for the RX/TX. There is no associated CPU interrupt. The 1.5V Regulator is enabled with the VREG_1P5V_EN bit in the VREG_CNTL register. 	<p>1 = 1.5V Regulator is ready to use 0 = 1.5V Regulator is not ready to use (default)</p>
18	<p>VREG_1P8V_RDY — 1.8V NVM Voltage Regulator Ready Status. This status bit will be 1 when the 1.8V Regulator has been given enough time to warm up.</p> <ul style="list-style-type: none"> The warm up time is 200us at 26MHz. There is no associated CPU interrupt. The 1.8V Regulator is enabled with the VREG_1P8V_EN bit in the VREG_CNTL register. 	<p>1 = 1.8V NVM Regulator is ready to use 0 = 1.8V NVM Regulator is not ready to use (default)</p>

Table 5-13. STATUS Register Bit Descriptions (continued)

Bit Number	Description	Operation
17	<p>VREG_BUCK_RDY — Buck Voltage Regulator Ready Status. This status bit will be 1 when the Buck Regulator has been given enough time to warm up.</p> <ul style="list-style-type: none"> The warm up time is 400us and 700us at 26MHz for regulation and bypass, respectively. There is no associated CPU interrupt. The Buck is enabled with the VREG_BUCK_EN bit in the VREG_CNTL register. This bit can also be used to determine when the buck has been properly bypassed in order to use the 1.8V NVM Regulator. Switching from Buck regulation to Buck bypass, while the NVM Regulator is on, is not allowed. If this condition occurs, then VREG_BUCK_RDY will be 0. The NVM Regulator (VREG_1P8V_EN) must be disabled first and then the Buck can be switched to bypass mode. 	<p>1 = Buck is ready to use 0 = Buck is not ready to use (default)</p>
11-16	Reserved	
10	<p>COP_EVT - COP Event Status. This read-only bit is set to indicate a COP timeout event status</p> <ul style="list-style-type: none"> It is usable is when the COP_OUT is enabled. It is cleared by writing the correct word to the COP_SERVICE register. If the COP_OUT is not set, a COP event will cause a complete reset of MCU and this status bit will be cleared. 	<p>1 = COP event has occurred 0 = No COP event has occurred (default)</p>
9	<p>CAL_DONE — Calibration Done. This bit indicates that a ring oscillator calibration event is complete.</p> <ul style="list-style-type: none"> Software should read the CAL_XTAL_CNT register to get the number of Reference oscillator cycles counted. The CAL_DONE bit is cleared by writing this bit position. 	<p>1 = Calibration done. 0 = Calibration not done (default)</p>
8	Reserved	
4-7	<p>EXT_WU_EVT[3:0] — External wake-up Event Status. These status bits indicate that an event has occurred on any one of 4 external KBI input pins to wake-up the processor.</p> <ul style="list-style-type: none"> These bits will only be set if enabled by their counterparts in the CRM WU_CNTL register. Writing a 1 to these bits will clear the status and clear the associated interrupt request, if also enabled in the WU_CNTL. Reading will not alter the contents. These 4 bits are ORed and sent to the ITC module. 	<p>1 = External wake-up Event has occurred. 0 = No External wake-up Event (default)</p>

Table 5-13. STATUS Register Bit Descriptions (continued)

Bit Number	Description	Operation
3	<p>RTC_WU_EVT — RTC Timer Time-out Event Status. This status bit reflects an RTC periodic timer event.</p> <ul style="list-style-type: none"> • The RTC can cause a wake-up from sleep mode or just provide periodic timeouts. • This bit will only be set if enabled by the RTC_WU_EN bit in the CRM WU_CNTL register. • Writing a 1 to this bit will clear the status and clear the associated interrupt request if enabled. • Reading will not alter the contents. • With the assertion of this status, status bit, the value in the RTC_COUNT register can be read. <p>Note: This bit requires some special handling because of the asynchronous nature between the bus clock and the RTC clock (2kHz or 32kHz). When software clears the RTC_WU_EVT, the bit will be cleared in 2 cycles of the slower RTC clock. Software cannot go back to sleep until this bit is cleared. If commanded to go to sleep while the RTC_WU_EVT bit is still set, the chip will wake up immediately.</p>	<p>1 = RTC Timer event has occurred. 0 = No RTC Timer event (default)</p>
2	<p>DOZE_WU_EVT — Doze Timer wake-up Event Status. This status bit reflects that the chip has awakened from doze.</p> <ul style="list-style-type: none"> • Wake-up from doze can be caused by the wake-up timer, the RTC or the external interrupt pads. • This bit will only be set if enabled by the TIMER_WU_EN bit in the CRM WU_CNTL register. • Writing a 1 to this bit will clear the status and clear the associated interrupt request if enabled. • Reading will not alter the contents. 	<p>1 = Timer wake-up Event has occurred. 0 = No Timer wake-up Event=</p>

Table 5-13. STATUS Register Bit Descriptions (continued)

Bit Number	Description	Operation
1	<p>HIB_WU_EVT — Hibernate Timer wake-up Event Status. This status bit reflects that the chip has awakened from hibernate.</p> <ul style="list-style-type: none"> • Wake-up from doze can be caused by the wake-up timer, the RTC or the external interrupt pads. • This bit will only be set if enabled by the TIMER_WU_EN bit in the CRM WU_CNTL register. • Writing a 1 to this bit will clear the status and clear the associated interrupt request if enabled. • Reading will not alter the contents. 	<p>1 = Hibernate wake-up Event has occurred. 0 = No Timer wake-up Event (default)</p>
0	<p>SLEEP_SYNC — Sleep Time Synchronizer. This status bit has 2 purposes:</p> <ul style="list-style-type: none"> • Either entering sleep mode and exiting wake-up. • When entering sleep mode, this bit is enabled the exact point at which the sleep timer has begun timing from 0. At this point, the software can read the MACA timer value which has been frozen by hardware and store it for later recovery. Once the MACA value is stored, the software needs to then clear this bit by writing a 1. Clearing this status bit will immediately freeze all MCU clocks and power down the system, except for the sleep timers and logic. Make sure all sleep mode maintenance is done prior to clearing this bit. Reading will not alter the contents. • Upon wake-up, software will need to immediately start polling this status bit. The bit is asserted at the exact point to which the sleep timer has counted; and the timer value can be fetched from the WU_COUNT register. This can be used to adjust the MACA timers with the precise amount of sleep time. The MACA can be loaded with the missing time and started. • This bit and wake-up procedure should be used for any type of wakeup. 	<p>1 = Sleep timer count has occurred. 0 = No counter change. (default)</p>

5.9.8 Module Enable Status (MOD_STATUS)

This read-only Module Enable Status register simply reports which modules are enabled or disabled.

	MOD_STATUS								Base+0x1C							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
															AIM_EN	
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NEX_EN	JTA_EN		ADC_EN	SPIF_EN	SSI_EN	I2C_EN	RIF_EN	TMR_EN	UART2_EN	UART1_EN	GPIO_EN	SPI_EN	ASM_EN	MACA_EN	ARM_EN
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 5-14. MOD_STATUS Register Bit Descriptions

Bit Number	Description	Operation
18-31	Reserved	
17	AIM_EN — Analog Interface Module enable status	1 = Enabled 0 = Not enabled
16	Reserved	
15	NEX_EN — Nexus Module enable status	1 = Enabled 0 = Not enabled
14	JTA_EN — JTAG Module enable status	1 = Enabled 0 = Not enabled
13	Reserved	
12	ADC_EN — Analog to Digital Converter Module enable status	1 = Enabled 0 = Not enabled
11	SPIF_EN — SPI FLASH Module enable status	1 = Enabled 0 = Not enabled
10	SSI_EN — Synchronous Serial Interface Module enable status	1 = Enabled 0 = Not enabled
9	I2C_EN — I2C Module enable status	1 = Enabled 0 = Not enabled
8	RIF_EN — Radio Interface Module enable status	1 = Enabled 0 = Not enabled

Table 5-14. MOD_STATUS Register Bit Descriptions (continued)

Bit Number	Description	Operation
7	TMR_EN — Timer Module enable status - If any clock to any counter is enabled, this bit is set.	1 = Enabled 0 = Not enabled
6	UART2_EN — Universal Asynchronous Receiver/Transmitter 2enable status	1 = Enabled 0 = Not enabled
5	UART1_EN — Universal Asynchronous Receiver/Transmitter 1 enable status	1 = Enabled 0 = Not enabled
4	GPIO_EN — General Purpose I/O Module enable status	1 = Enabled 0 = Not enabled
3	SPI_EN — Serial Peripheral Interface Module enable status	1 = Enabled 0 = Not enabled
2	ASM_EN — Advanced Security module enable status	1 = Enabled 0 = Not enabled
1	MACA_EN — MAC Accelerator enable status	1 = Enabled 0 = Not enabled
0	ARM_EN — CPU enable status	1 = Enabled 0 = Not enabled

5.9.9 Wake-up Count (WU_COUNT)

The WU_COUNT[31:0] register reflects the current count of the wake-up timer. Upon entering hibernate or doze mode, this counter will be reset, start at a count of zero and count up. Use 32-bit accesses only. The register contents are retained in sleep mode.

	WU_COUNT								Base+0x20							
Base+0x20	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	WU_COUNT[31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	WU_COUNT[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-15. WU_COUNT Register Bit Descriptions

Bit Number	Description	Operation
0-31	WU_COUNT[31:0] — Wake-up Timer Count. This is the running value of the CRM sleep timer that runs on the 2kHz, XTAL/128 or 32kHz (selectable by the XTAL32_EN bit in the XTAL32_CNTL register). <ul style="list-style-type: none"> • Entering sleep mode resets this timer to zero. • This value can used to adjust missing time in the MACA block. 	0x0000_0000 = Default from reset

There are three clocks that can run the sleep timer: Timer usage is shown in [Table 5-16](#).

- 2kHz ring oscillator
- 32kHz XTAL
- Reference oscillator divided-by 128 (nominally 24MHz/128).

Table 5-16. Timer Usage

	2kHz Ring Osc (Hibernate Mode)	32kHz XTAL (Hibernate Mode)	XTAL/128 (Doze Mode)
Max Sleep Time	24.8 days	36.4 hours	11.75 hours @ 13MHz 6.36 hours @ 24 MHz 5.875 hours @ 26MHz

5.9.10 Wake-up Time-out (WU_TIMEOUT)

The WU_TIMEOUT[31:0] register sets the time-out compare value for the wake-up timer. Use 32-bit accesses only. The register contents are retained in sleep mode.

		WU_TIMEOUT								Base+0x24							
		BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
		WU_TIMEOUT[31:16]															
TYPE		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		WU_TIMEOUT[15:0]															
		BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
TYPE		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-17. WU_TIMEOUT Register Bit Descriptions

Bit Number	Description	Operation
0-31	<p>WU_TIMEOUT[31:0] — Wake-up Time-out compare value. This value sets the sleep duration time for the wake-up timer for either hibernate or doze.</p> <ul style="list-style-type: none"> The true sleep duration will be longer than the value set because several sleep clock cycles are necessary to wake-up the MCU. Upon wake-up, the software should pole the appropriate status bit in the STATUS register and when the status bit goes true, it will then need to read the WU_COUNT register to find out exactly how long it has actually been asleep for. 	

5.9.11 RTC Count (RTC_COUNT)

The RTC_COUNT[31:0] register reflects the current count of the free-running real time clock timer. The 2kHz ring oscillator, or optionally, the 32kHz crystal oscillator must be enabled for the counter to be running. This is an up-counter, it starts at 0 and will rollover at 0xFFFF_FFFF. It cannot be reset by software. No special consideration has been given to the switch from 2kHz to 32kHz and visa-versa. 32-bit accesses only should be used for accuracy. The register contents are retained in sleep mode.

	RTC_COUNT								Base+0x28							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	RTC_COUNT[31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	RTC_COUNT[15:0]															
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-18. RTC_COUNT Register Bit Descriptions

Bit Number	Description	Operation
0-31	RTC_COUNT[31:0] — Real Time Clock (RTC) Count. This is the running value of the RTC timer.	

5.9.12 RTC Time-out (RTC_TIMEOUT)

This register sets the incremental time-out compare value for the RTC timer. Use 32-bit accesses only. The register contents are retained in sleep mode.

		RTC_TIMEOUT								Base+0x2C							
		BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
		RTC_TIMEOUT[31:16]															
TYPE		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		RTC_TIMEOUT[15:0]															
		BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
TYPE		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-19. RTC_TIMEOUT Register Bit Descriptions

Bit Number	Description	Operation
0-31	<p>RTC_TIMEOUT[31:0] — RTC Periodic Time-out. This value sets the incremental time at which an RTC interrupt will be periodically generated.</p> <ul style="list-style-type: none"> • An interrupt request based on an RTC time-out can be used while awake or can cause a wake-up event. • As soon as the time-out period occurs, the next time-out point (based on current RTC tcount) is calculated in hardware and saved. • A new time-out value can be written at any time, BUT the new time-out value will be calculated and effective only after the next RTC clock. 	

5.9.13 Calibration Control Register (CAL_CNTL)

The Calibration Control register allows the 2kHz Ring Oscillator to be calibrated against the Reference XTAL. Use 16-bit or 32-bit accesses only. The register contents are not retained in sleep mode.

	CAL_CNTL								Base+0x34							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
															CAL_IEN	CAL_EN
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	CAL_TIMEOUT[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-20. CAL_CNTL Register Bit Descriptions

Bit Number	Description	Operation
18-31	Reserved	
17	CAL_IEN - Calibration Interrupt Enable. This bit enables an interrupt request signifying that the calibration is complete. If disabled, there will be no interrupt request, however, the event status bit (CAL_DONE) in STATUS will be asserted and will need to be cleared.	1 = Interrupt request enabled 0 = Disabled. (Default)
16	CAL_EN — Calibration Enable. This bit enables the calibration sequence. This bit will be automatically cleared and the CAL_DONE bit in CRM STATUS will be set on completion of the calibration. Writing a zero to CAL_EN during the calibration will abort the process.	1 = Calibration enabled. 0 = Calibration disabled (Default) or cause calibration to abort.
0-15	CAL_TIMEOUT[15:0] — Calibration Time-out Value. This value determines how long the calibration is to last. The maximum amount of calibration time allowed (i.e. using all 16 bits) is 32.768sec at 2kHz. This value can be set at the same time as the CAL_EN.	

5.9.14 Calibration XTAL Count (CAL_COUNT)

This register contains the reference oscillator count value after a ring oscillator calibration sequence.

	CAL_COUNT								Base+0x38							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	CAL_COUNT[31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	CAL_COUNT[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-21. CAL_COUNT Register Bit Descriptions

Bit Number	Description	Operation
0-31	CAL_COUNT[15:0] and CAL_COUNT[31:16] — Calibration XTAL Count. This value contains the number of reference oscillator cycles that occurred during the calibration period. This value is valid after the CAL_DONE status or it's associated interrupt is asserted.	

5.9.15 Ring Oscillator Control (RINGOSC_CNTL)

This register controls the calibration sequence for the ring oscillator. Byte writes are not allowed for this register; use 16-bit or 32-bit access only. The register contents are retained in sleep mode.

	RINGOSC_CNTL								Base+0x3C							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
				ROSC_CTUNE[3:0]				ROSC_FTUNE[4:0]							ROSC_EN	
TYPE	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	1	1	0	1	0	1	1	1	0	0	0	1

Table 5-22. RINGOSC_CNTL Register Bit Descriptions

Bit Number	Description	Operation
13-31	Reserved	
9-12	ROSC_CTUNE[3:0] — Ring Oscillator Course Tune. This is the course tune adjustment for the ring oscillator. Each step changes loading by 1 pF	0x6 = Default
4-8	ROSC_FTUNE[4:0] — Ring Oscillator Fine Tune. This is the fine tune adjustment for the ring oscillator. Each step changes loading by 160 fF	0x17 = Default
1-3	Reserved	
0	ROSC_EN — Ring Oscillator Enable. The ring oscillator is enabled by default. This bit can disable the ring oscillator, but this should only be done when the 32kHz clock source is present (necessary for waking from hibernate mode).	1 = Ring Oscillator ON (default) 0 = Ring Oscillator OFF

5.9.16 Reference XTAL Control (XTAL_CNTL)

This register controls the loading to the reference oscillator. Byte writes are not allowed; use 16-bit or 32-bit access only. The register contents are retained in sleep mode.

	XTAL_CNTL								Base+0x40							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
							XTAL_CTUNE[4:0]				XTAL_FTUNE[4:0]					
TYPE	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					XTAL_IBIAS_SEL[3:0]											
TYPE	r	r	r	r	rw	rw	rw	rw	r	rw	rw	rw	r	rw	rw	rw
RESET	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	0

Table 5-23. XTAL_CNTL Register Bit Descriptions

Bit Number	Description	Operation
26-31	Reserved	
21-25	XTAL_CTUNE[4:0] — Reference oscillator coarse tuning for load capacitance: XTAL_CTUNE[4] - selects 4 pF load to crystal XTAL_CTUNE[3:0] - values 0-7 select load capacitance equal to programmed number, i.e., value 7 selects 7 pF	1) Default = 0x00 2) Recommended program value 0x15
16-20	XTAL_FTUNE[4:0] — Reference oscillator fine tuning for load capacitance. XTAL_FTUNE[4:0] selects from 0-5 pF in 32 steps of approximately 156 fF	1) Default = 0x00 2) Recommended program value 0x10
12-15	Reserved	
8-11	XTAL_IBIAS_SEL[3:0] — Reference oscillator bias select. These bits select different bias current for the oscillator amplifier. Note: Recommended that default be used as normal mode.	Default = 0x1F
7-0	Reserved	

5.9.17 32kHz XTAL Control (XTAL32_CNTL)

This register controls the use of the 32 kHz crystal oscillator. The register contents are retained in sleep mode.

	XTAL32_CNTL								Base+0x44							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
											XTAL32_EN					XTAL32_EN
TYPE	r	r	r	r	r	r	r	r	r	r	rw	rw	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Table 5-24. XTAL32_CNTL Register Bit Descriptions

Bit Number	Description	Operation
6-31	Reserved	
4-5	XTAL32_GAIN[1:0] - 32kHz XTAL Gain Selector. These bits select the gain for the 32kHz crystal oscillator. Maximum gain is 2'b11. Minimum gain is 2'b00. Note: RECOMMENDED TO LEAVE AS DEFAULT .	2b11 = Maximum gain (default)
1-3	Reserved	
0	XTAL32_EN — 32kHz XTAL Enable. This bit turns on the XTAL32 oscillator. <ul style="list-style-type: none"> 2kHz Ring Oscillator should be turned off first. Engaging the XTAL32 must be performed while awake. It is also recommended that the RTC register be monitored to determine that XTAL32 has started. One can consider going into Bus Stealer Mode and waiting for 32kHz to be engaged. The state is retained in and out of sleep modes. Only a hard or soft reset can turn the XTAL32 off. One might consider going into Bus Stealer Mode and waiting for 32kHz to be engaged. Sometime after setting XTAL32_EN to 1 (up to 5sec), the 32kHz crystal will be ready. 	1 = XTAL32 ON 0 = XTAL32 OFF (Cold start default)

5.9.18 Voltage Regulator Control (VREG_CNTL)

The VREG_CNTL register controls operation of a number of the power supply circuits. The register contents are not retained in sleep mode.

NOTE

All times stated here are based on a crystal of 26MHz (quickest warm-up). Lower frequency crystals will result in longer warm-up times. For example, a crystal of 13 MHz will nearly double the warm-up time for all regulators.

	VREG_CNTL								Base+0x48							
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
					BUCK_CLKDIV[3:0]				VREG_1P8V_EN	VREG_1P5V_SEL[1:0]		VREG_1P5V_EN[1:0]		BUCK_BYPASS_EN	BUCK_SYNC_REC_EN	BUCK_EN
TYPE	r	r	r	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0

Table 5-25. VREG_CNTL Register Bit Descriptions

Bit Number	Description	Operation
13-31	Reserved	
12	Reserved	
8-11	<p>BUCK_CLKDIV[3:0] — Divider Code Word for Buck Regulator Clock. These bits select the integer clock divider from decimal 2 to 16.</p> <ul style="list-style-type: none"> See Table 5-28 for usage. The code word needs to be properly selected so that the resultant Buck Clock will be as close as possible to 1.6MHz. The actual range of resultant Buck Regulator Clocks will be 1.50MHz to 1.70MHz. For example, with a 24MHz reference oscillator, the integer divider should be 15 and the BUCK_CLKDIV code word will be set to 4'b1111 (reset default). This will yield a Buck Clock of 1.6MHz. No clock will be generated if the Buck is disabled. 	0xF = Divisor is 15 (default)

Table 5-25. VREG_CNTL Register Bit Descriptions (continued)

Bit Number	Description	Operation
7	<p>VREG_1P8V_EN — NVM 1.8V Voltage Regulator Enable. This bit enables the 1.8V Voltage Regulator that supplies the Non-Volatile Memory (NVM).</p> <ul style="list-style-type: none"> This bit can be set only when MC1322x is awake, CPU running and not accessing the NVM. When first enabled: a) the voltage regulator first trickle charges the associated capacitor, b) after 200us, the full current drive capability of the 1.8V voltage regulator will be enabled. <p>Note: Do not start using the NVM before the 200us warm-up period - check the VREG_1P8V_RDY bit in the STATUS register for readiness. .</p> <p>Note: If the system is configured with an External 1.8V Supply, then VREG_1P8V_EN bit will be ignored. In this power configuration, the external 1.8V supply is externally connected to the 1.8 voltage regulator output pin, i.e., the 1.8V regulator is completely physically bypassed. Ensure that the PWR_SOURCE control bits are properly set in SYS_CNTL register.</p> <p>Note: If the Buck is enabled and then the 1.8V regulator is enabled, this will cause the 1.8V regulator to be bypassed since the Buck regulator output will already be approximately 1.8V. Thus, the VREG_1P8V_EN still needs to be set to 1 and 200us is still needed for warm up (check the VREG_1P8V_RDY status bit). In this mode, the regulator will not actually be turned, however, it will be bypassed and time is needed to charge the associated capacitor. Make sure that the PWR_SOURCE control bits are properly set in SYS_CNTL register.</p>	<p>1 = NVM Regulator enabled 0 = NVM Regulator disabled (default)</p>
5-6	<p>VREG_1P5V_SEL[1:0] - Analog 1.5V Voltage Regulator Current Selector These bits control the amount of current sourced out of the voltage regulator.</p> <ul style="list-style-type: none"> The selection is defined by the Table 5-27 . This regulator must be set to source 20mA or 40 mA before the transceiver is used 40mA is recommended for nominal or higher TX power settings. <p>Note: Common usage is to enable the 40mA option for normal operation so that power settings need not be changed between RX and TX operations</p>	<p>0x1 = 20mA source (default)</p>
3-4	<p>VREG_1P5V_EN[1:0] - Analog 1.5V Voltage Regulator Enable field. These bits turn on and off the 1.5V voltage regulators that supply the analog radio functions.</p> <ul style="list-style-type: none"> These bits can be set only when MC1322x is awake and running in the "IDLE" mode (transceiver not active). Table 5-26 shows the possible selections and the amount of time needed for warmup. Software can monitor the VREG_1P5V_RDY to determine when the regulator is fully on. These regulators put the transceiver in the "RUN" mode Typical usage is to program all regulators to on after exiting low power mode. <p>Note: If ONLY the RX/TX Regulator is on (VREG_1P5V_EN[1:0]=2'b01) and then all regulators are needed on, then the VREG_1P5V_EN[1:0] must first be programmed to 2'b00 (disabled) and then re-programmed to 2'b11.</p>	<p>0x0 = Off (default)</p>

Table 5-25. VREG_CNTL Register Bit Descriptions (continued)

Bit Number	Description	Operation
2	<p>BUCK_BYPASS_EN — Buck Regulator Bypass Enable. Setting this bit causes the system to bypass the Buck Voltage Regulator.</p> <ul style="list-style-type: none"> • Enable when the VBATT falls below 2.5V. • Bypassing the buck will require 700us in order to allow the battery to charge the buck capacitor. Once the 700us has elapsed, the 1.5V and 1.8V regulators can be enabled. • Application software can poll the BUCK_RDY bit in the CRM STATUS register to know when to continue with enabling the 1.5V or the 1.8V regulator. • The BUCK_EN takes priority over this BUCK_BYPASS_EN. However, if the buck is being bypassed, it does not make sense to enable the buck regulator. • If the system is configured with an External 1.8V Supply (i.e. SYSTEM_CNTL setting PWR_SOURCE = 2'b10), the BUCK_BYPASS_EN bit will be ignored. <p>Note: After POR, but before the use of the NVM, the system will not know if it is configured to use the Buck regulator. Therefore, the system will assume that a buck regulator will power the chip, and therefore, software MUST bypass the buck regulator whether there is a buck or not. Again, this is only needed after POR, but before enabling the 1.8V regulator and the NVM. The NVM Regulator must be disabled before switching the Buck Regulator from regulation to bypass mode. Check the VREG_BUCK_RDY set to 1 for a successful switch. This is handled by default in the ROM-based boot loader.</p>	<p>1 = Buck regulator bypass enabled 0 = Bypass disabled (default)</p>
1	<p>BUCK_SYNC_REC_EN — Buck Synchronous Rectifier Voltage Regulator Enable. This bit enables the synchronous rectifier of the Buck Regulator. It should always be set when the Buck regulator is enabled.</p>	<p>1 = Synch rect regulator enabled 0 = Synch rect regulator disabled (default)</p>
0	<p>BUCK_EN — Buck Voltage Regulator Enable. Setting this bit to 1 enables the buck regulator.</p> <ul style="list-style-type: none"> • The 1.5V and 1.8V regulators (for analog and NVM blocks) are powered by the buck regulator through the LREG_BK_FB pin. • The application software must allow 400us after the Buck is enabled to turn on either the 1.5V or the 1.8V Regulators. • Software can poll the BUCK_RDY bit in the CRM STATUS register to know when to turn-on the 1.5V or the 1.8V regulators. <p>Note: If the system is configured with an External 1.8V Supply (i.e. SYSTEM_CNTL setting PWR_SOURCE = 2'b10), the BUCK_EN bit will be ignored.</p> <p>Note: The Buck Regulator as a switcher is not efficient when VBATT is 2.5V or less. When VBATT <= 2.5V, the Buck Regulator must be bypassed, i.e., BUCK_BYPASS_EN bit must be set. This bypass will take 700us. Software can poll the VREG_BUCK_RDY bit in the STATUS register to know when to continue with turning on the 1.5V or the 1.8V regulator.</p> <ul style="list-style-type: none"> • If the Buck is enabled and then the 1.8V regulator is enabled, this will cause the 1.8V regulator to be bypassed since the Buck regulator output will already be approximately 1.8V. 	<p>1 = Buck regulator enabled 0 = Buck regulator disabled (default)</p>

Table 5-26. Analog 1.5 V Voltage Regulator Enable

VREG_1P5V_EN[1:0]	Regulator(s) Turned On	Time for Warm-up
2'b00	None	-
2'b01	RX/TX Regulator	200us
2'b11	RX/TX plus PLL Regulators	600us
2'b10	Not Legal	-

Table 5-27. 1.5V Voltage Regulator Current Selector

VREG_1P5V_SEL[1:0]	Current Sourced	Reason
2'b00	4mA	Pre-Transceiver State
2'b01 (default)	20mA	Normal Transceiver
2'b11	40mA	6dB Transceiver
2'b10	Not Legal	Not Legal

Table 5-28. Buck Clock Settings

BUCK_CLKDIV[3:0]	Actual Integer Divisor	Frequency Range of REF XTAL in which to use Divisor (MHz)	Resultant Buck Clock Frequency (MHz)
4'b0001	Constant Phase	Reserve for Test	N/A
4'b0010 to 4'b0111	2 to 7	N/A Reserve for Test	N/A
4'b1000	8	13.00 - 13.59	1.625 - 1.699
4'b1001	9	13.60 - 15.19	1.511 - 1.688
4'b1010	10	15.20- 16.79	1.520 - 1.679
4'b1011	11	16.80 - 18.39	1.527 - 1.672
4'b1100	12	18.40 - 19.99	1.533 - 1.666
4'b1101	13	20.00 - 21.59	1.538 - 1.661
4'b1110	14	21.60 - 23.19	1.543 - 1.656
4'b1111 (default)	15	23.20 - 24.79	1.547 - 1.653
4'b0000	16	24.80 - 26.00	1.550 - 1.625

5.9.19 Software Reset (SW_RST)

Writing to the Software Reset register causes a system reset, which provides a means to accomplish a system reset via software control. Only 32-bit accesses are used. The application must write the read return value to cause the reset.

		SW_RST								Base+0x50							
		BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
		SW_RST[31:16]															
TYPE		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET		1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1
		SW_RST[15:0]															
		BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
TYPE		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET		0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0

Table 5-29. SW_RST Register Bit Descriptions

Bit Number	Description	Operation
0-31	SW_RST[31:0] — Software Reset. The Software Reset register is used to initiate a complete reset of the system. <ul style="list-style-type: none"> To initiate the reset, write the value 0x87651234 to the SW_RST register. The value must be written as a 32-bit access - sequential byte or 16-bit writes will not cause a reset. Reading this register returns 0x87651234. 	0x87651234 = Default

Chapter 6

MC1322x 2.4 GHz ISM Band Transceiver

6.1 Introduction

The MC1322x transceiver consists of the 2.4 GHz radio, modem, and MAC Accelerator (MACA). This chapter provides the reference for the radio and the modem. The MACA is the primary control block and path for TX and RX data and is described in [Chapter 9, “MAC Accelerator \(MACA\)”](#).

6.2 Transceiver Overview

[Figure 6-1](#) shows a simplified block diagram of the MC1322x IEEE 802.15.4 transceiver. As the diagram shows, the transceiver is composed of the radio, modem and MACA blocks.

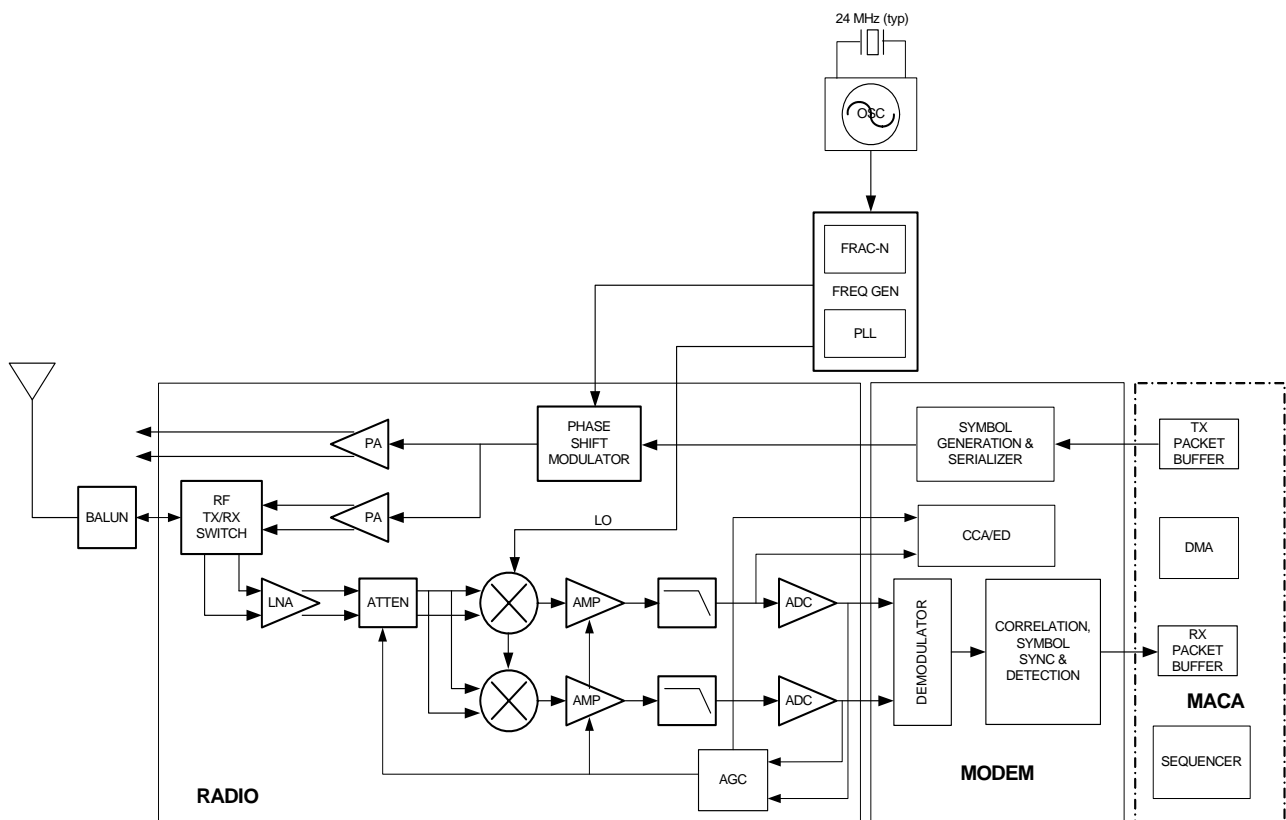


Figure 6-1. MC1322x IEEE 802.15.4 Transceiver Block Diagram

6.2.1 IEEE 802.15.4 Packet Structure

Figure 6-2 shows the IEEE 802.15.4 packet structure of the MC1322x. Payloads of up to 125 bytes are supported. The MC1322x adds a four-byte preamble, a one-byte Start of Frame Delimiter (SFD), and a one-byte Frame Length Indicator (FLI) before the data. A Frame Check Sequence (FCS) is calculated and appended to the end of the data.

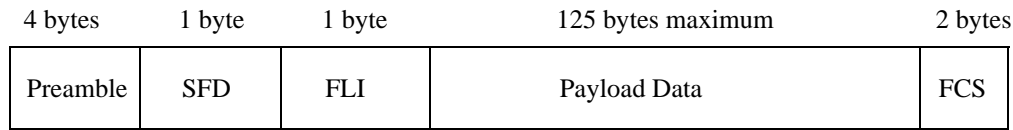


Figure 6-2. MC1322x Packet Structure

6.2.2 Transmit Path

The MACA provides the data path for both transmit and receive. For the transmit path, the MACA makes up the preamble, SRD, and FLI, and then fetches the payload data directly from memory with a simple DMA function. The DMA delivers the data on-a-demand basis to the MACA FIFO designated as the TX packet buffer.

The MACA transfers data (at the bit-level) to the Tx Modem which is responsible for the bit-to-symbol mapping, PN spreading, and PSM encoding. The standard 802.15.4 demodulator performs packet synchronization and data demodulation.

The modulated signal is delivered to the RF analog power amplifier output.

6.2.3 Receive Path

In the radio RF receive signal path, the RF input is converted to low IF In-phase and Quadrature (I & Q) signals through a down-conversion stage. The IF signals are amplified, filtered, and the digitally sampled to convert the receive data to the digital domain. The modem 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. A two-byte FCS value is also calculated on the received data and compared to the FCS value appended to the transmitted data, generating a Cyclical Redundancy Check (CRC) result.

In the MACA, the data is accumulated in the Rx data packet buffer. The payload data is passed directly to memory with a simple DMA function. The DMA delivers the data as available.

6.2.4 Independent Transceiver Operation

The MC1322x transceiver is capable of complete standalone operation. The MACA provides highest level control to the radio and modem and has the DMA for passing data to/from the RAM. The CPU is released to do other parallel tasks and also eliminates any timing dependency required to move data between the transceiver and memory. Also, the DMA works on a cycle steal basis from the CPU bus and as a result is very quick; data is transferred on a 4-byte basis.

Full interrupt capability is supported from completion of an operation or an exception condition occurring during an operation.

6.2.5 Modem Overview

The standard 802.15.4 demodulator performs packet synchronization and data demodulation.

The modem transmitter block consists of an interface to the MACA that transforms the data to be transmitted based on the 802.15.4 ZigBee standard into I/Q modulation signals for use by the analog phase shift modulator (PSM). The MACA transfers data (at the bit-level) already formatted with the correct 802.15.4 packet structure. The Tx Modem is responsible for the bit-to-symbol mapping, PN spreading, and PSM encoding.

The receiver block supports demodulation, correlation, symbol sync and detection. The detected symbols are passed to the MACA.

The receiver block also supports the CCA/ED level detection function for report CCA/ED level and LQI for a received packet.

6.3 Transceiver Register Memory Map Base Addresses

Each Transceiver module is programmed via a set of memory-mapped registers and their respective base addresses are listed in [Table 6-1](#).

Table 6-1. Base Addresses of Transceiver Modules

Base Address	Module Designation
0x8000_9000	Modem Synthesizer Write Functions
0x8000_91C0	Modem Synthesizer Read Functions
0x8000_9200	Modem Transceiver Sequence Manager
0x8000_9400	Modem Radio Receiver Functions
0x8000_9600	Modem Radio Transmitter Functions
0x8000_9800	Modem Radio (RF) Frequency Synthesizer
0x8000_9A00	Modem Tracking Oscillator (TOC)
0x8000_A000	Radio Analog Write Functions
0x8000_A1C0	Radio Analog Read Functions

NOTE

The transceiver radio and modem functions are highly configurable. The settings for these configurable parameters can be set via default values from reset or via initialization from direct programming. Many of these settings are provided by Freescale without benefit of explanation of the parameters. This is done to protect the intellectual property that the transceiver represents.

6.4 Modem Synthesizer

This section describes the synthesizer subsystem block.

6.4.1 Synthesizer Description

As described in [Section 5.1.3.2, “Main System Clock Distribution”](#) of [Chapter 5, “System Management \(Including CRM\)”](#) The modem synthesizer is part of the MC1322x clock distribution system. This module is only used if the reference oscillator frequency is not the default of 24 MHz. The modem requires a clock of 24 MHz and this block generates that frequency if the reference oscillator is not the 24 MHz standard. [Figure 6-3](#) highlights the synthesizer subsystem.

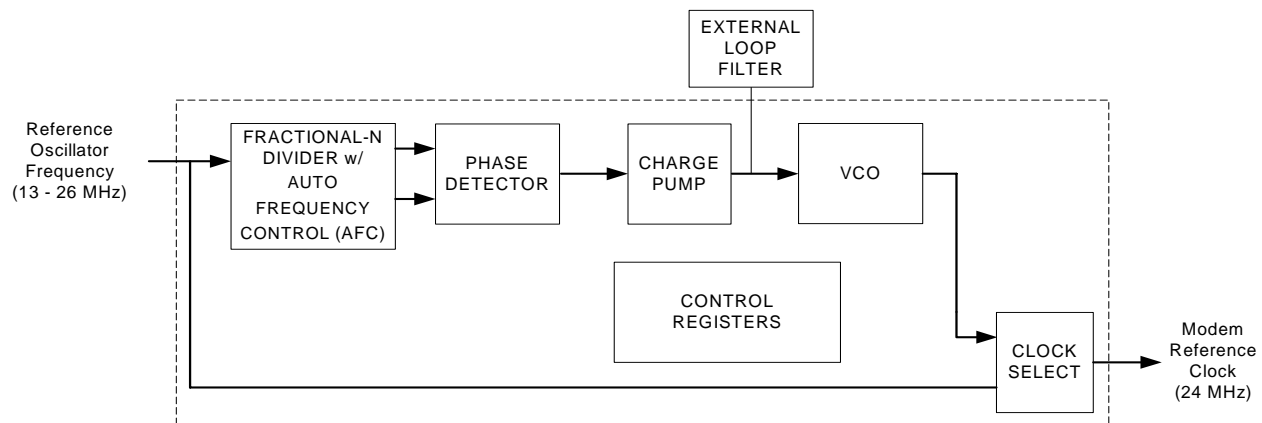


Figure 6-3. Modem Synthesizer Block Diagram

Features of the synthesizer include:

- Reference Clock — the main clock reference for the synthesizer is the external reference crystal oscillator.
- Modem clock synthesizer PLL (the modem reference clock must always be 24 MHz in functional mode)
 - A charge pump, voltage-controlled oscillator (VCO) and superfilter (or voltage regulator).
 - There is an external PLL loop filter which is a second order RC filter and is an integral part of the synthesizer loop. The charge pump and VCO are connected to the loop filter
 - A fractional-N division ($\Sigma\text{-}\Delta$ Loop Divider) architecture along with digital automatic frequency control (AFC), resulting in less than 1Hz frequency resolution at the VCO

- Synthesizer Control Registers — The register map consists of a 256-word register field, of which only a subset is utilized. All registers are readable.
- Startup State Machine — The startup state machine is controlled by the **syn_en** register bit. This logic enables/disables the clocks in the Synthesizer, thereby placing the MC1322x modem in battery-save. This logic optimally enables each analog component in the Synthesizer to reduce power consumption.

6.5 Modem Synthesizer Register Map

The registers in the synthesizer and their descriptions are shown below.

NOTE

- Only 32-bit accesses to synthesizer registers are supported.
- Modem Synthesizer uses two base addresses

Write base address is **0x8000_9000**

Read base address is **0x8000_91C0**

Table 6-2. Modem Synthesizer Write Register Memory Map

Offset		[31:24]	[23:16]	[15:8]	[7:0]	Access Type	Access Width
+ 0x00	SYN_ENABLE	Enable and Override Register				R/W	32-bit only
+ 0x04		Reserved					
+0x08	SYN_REFDIV	Reference Loop Divider Register				R/W	32-bit only
+ 0x0C	SYN_VCODIV	VCO Loop Divider Register				R/W	32-bit only
+ 0x10	SYN_VCOLOCK	VCO Lock Register (Reserved)				R/W	32-bit only
+ 0x14 to +0x28		Reserved					

6.5.1 Enable and Override Register

Enable and Override Register													Addr Base+0x00			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	pll_active	syn_en									dig_pll_en	vco_en	cp_en	sf_en_b	sf_float_cap_b	sf_vref_en_b
TYPE	rw	rw									rw	rw	rw	rw	rw	rw
RESET	1	0									0	0	0	1	0	1
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
							modem_clk_ctrl									
TYPE							rw	rw								
RESET							0	0								

Table 6-3. Enable and Override Register Bit Descriptions

Bit Number	Description	Operation
0-7	Reserved	
8-9	modem_clk_ctrl — Source and enable control for the modem clock.	00 = The modem clock is forced to logic low. 01 = The modem clock is the reference oscillator (xtal_clk). In MC1322x, this is used if a 24MHz reference crystal oscillator were employed. In this case, the post-modulo divider is bypassed.
10-15	Reserved	
16	Reserved	
17	Reserved	
18	Reserved	
19	Reserved	
20	Reserved	
21	Reserved	
22-29	Reserved	

Table 6-3. Enable and Override Register Bit Descriptions

Bit Number	Description	Operation
30	syn_en — Enable signal for entire clock synthesizer.	1 = A rising edge on this signal (software programs the bit to a 1 while it was previously 0), enables a synthesizer startup sequence, if pll_active is also asserted. Programming this bit to a 1 if it is already 1 will have no effect. This bit will also have no effect if pll_active is deasserted. 0 = A falling edge of this signal disables the clock synthesizer. Synthesizer shutdown is cold-turkey. Programming this bit to a 0 if it is already 0 will have no effect.
31	pll_active — Synthesizer controlled by syn_en or startup state machine	1 = Synthesizer is controlled by syn_en . When syn_en transitions low, the synthesizer is turned off. When syn_en transitions high, the synthesizer is enabled by an internal state machine. The sequence of events that is controlled by the assertion of syn_en is described in Section 6.4.1, “Synthesizer Description” . 0 = Synthesizer is controlled by the override bits. syn_en has no effect when pll_active is low.

6.5.2 Reference Loop Divider Register

The Reference Loop Divider Register must be programmed for the synthesizer to provide 24 MHz to the modem if a reference frequency other than 24 MHz is used. This register should only be changed while the synthesizer is disabled (**syn_en** = 0). See [Section 6.5.5, “Frequency Programming Example”](#) for details on programming a desired frequency.

		Reference Loop Divider											Addr Base+0x08			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE																
RESET																
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									pn							
TYPE									rw	rw	rw	rw	rw	rw	rw	rw
RESET									0	0	0	0	0	0	0	1

Table 6-4. Reference Loop Divider Register Bit Descriptions

Bit Number	Description	Operation
0-7	pn — Integer value of the reference loop divider. This value determines the reference frequency.	Valid values are 1 through 255, where a “1” represents a feed through of the input XTAL clock and a “255” represents a divide by 255. See Section 6.5.5, “Frequency Programming Example” for details on programming a desired frequency.

6.5.3 VCO Loop Divider Register

The VCO Loop Divider Register must be programmed for the synthesizer to provide 24 MHz to the modem if a reference frequency other than 24 MHz is used. This register should only be changed while the synthesizer is disabled (`syn_en = 0`). See [Section 6.5.5, “Frequency Programming Example”](#) for details on programming a desired frequency.

		VCO Loop Divider											Addr Base+0x0C			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	rn							rafc[24:16]								
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	rafc[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-5. VCO Loop Divider Register Bit Descriptions

Bit Number	Description	Operation
0-24	rafc — Fractional component of the divide value for the sigma delta VCO loop divider.	This value feeds into the three accumulator sigma delta. See Section 6.5.5, “Frequency Programming Example” for details on programming a desired frequency.
25-31	rn — Integer value of the VCO loop divider.	Valid values are 1 through 127, where a “1” represents a divide by 1 of the VCO input clock and a “127” represents a divide by 127. An offset of 3 is present in this divider when fractionalization is enabled. When fractionalization is disabled, the value programmed in <code>rn[6:0]</code> becomes the actual divide modulo.

6.5.4 VCO Lock Register

This register is reserved and should be left unmodified.

6.5.5 Frequency Programming Example

Multiple registers need to be programmed to synthesize the desired 24MHz modem frequency. These registers are shown below with a brief description.

Table 6-6. Registers Requiring Programming Before Being Synthesized

Register Name	Description
rn[6:0]	Integer value of the VCO loop divider for Synthesizer
pn[7:0]	Integer value of the reference XTAL loop divider for Synthesizer
rafc[24:0]	Fractional component of the divide value for the VCO loop divider

Suppose a XTAL frequency of 26MHz is used to synthesize the VCO frequency of 24MHz. The equation shown in [Figure 6-4](#) is used to calculate the values of **rn**, **pn**, and **rafc**.

$$F_{vco} = F_{ref} \times \left(RN + 3 + \frac{RAFC}{2^{25}} \right)$$

Figure 6-4. VCO Frequency Equation

where,

$$RN = \mathbf{rn}$$

$$RAFC = \mathbf{rafc}$$

F_{ref} = Reference Frequency (to the phase detector)

F_{vco} = VCO frequency to be synthesized (always 24 MHz)

A reference frequency (F_{ref}) of 3.25MHz is selected by the designer. This yields a value of **pn** equal to 8 based on the equation shown in [Figure 6-5](#).

$$F_{ref} = \frac{F_{xtal}}{pn} = \frac{26\text{MHz}}{8} = 3.25\text{MHz}$$

Figure 6-5. Reference Frequency Equation

A desired VCO frequency of 24MHz is to be synthesized. Solving the equations shown in [Figure 6-4](#) and [Figure 6-5](#) results in the values shown in [Figure 6-6](#).

$$24 = 3.25 \times \left(RN + 3 + \frac{RAFC}{2^{25}} \right)$$

$$\frac{24}{3.25} - 3 = \left(RN + \frac{RAFC}{2^{25}} \right)$$

$$4.384615385 = RN + \frac{RAFC}{2^{25}}$$

Figure 6-6. Equation Resolution End Result

This equation is then split into two equations to solve for the integer divide value (**rn**) and the fractionalization value (**rafc**) as shown in [Figure 6-7](#).

$$\begin{aligned} rn &= 4 \\ rafc &= \lfloor 0.384615385 \times 2^{25} \rfloor = 12905551 \end{aligned}$$

Figure 6-7. Integer Divide Value and Fractionalization Value Equation

RAFC is then converted to a hexadecimal base to yield a final value of 0xC4EC4E.

In summary, the final values to be programmed into **pn**, **rn**, and **rafc** are shown in [Table 6-7](#).

Table 6-7. Final Programming Values

Register	Final Value
pn	0x08
rn	0x04
rafc	0xC4EC4E.

6.6 Modem Transceiver Sequence Manager (TSM)

Within the modem, the Transceiver Sequence Manager subsystem controls operation of the radio during a transmission, reception, or CCA function. There are both a separate transmit state machine and a separate receive state machine that manages the receive/CCA sequences. The state machine manages warm-up of the RF circuitry, switching of the onboard RX/TX switch, timing of the sequence, and power-down of the circuitry. The sequence is initiated by the MACA where overall control of the radio is maintained.

[Figure 6-8](#) shows simple models of the TX, RX, and CCA/ED timing profiles. The radio RF VCO and analog circuitry is not powered in an idle condition to save battery life. As a result, for any radio sequence there is a 144 μ s warmup time to allow the radio circuitry to stabilize. For TX, this includes a 12 μ s PA power ramp-up so as not to “splash” noise into the adjacent frequency channels.

The total “on” time for TX or RX is dependent on the packet length.

- For TX, the on time is consistent for a given packet length, knowing the number of data bytes as well as the packet overhead bytes. Also, there is an additional short 12 μ s “warm down” time as the PA is powered down. Again, this is to prevent noise splatter as the PA turns off.
- For RX the time is more variable, because the receiver must be on in anticipation of the incoming packet, and once receiving the packet, the packet length will determine the additional on time.
- CCS or ED on time is always the same based 802.15.4 standard and related to the number of symbol times.

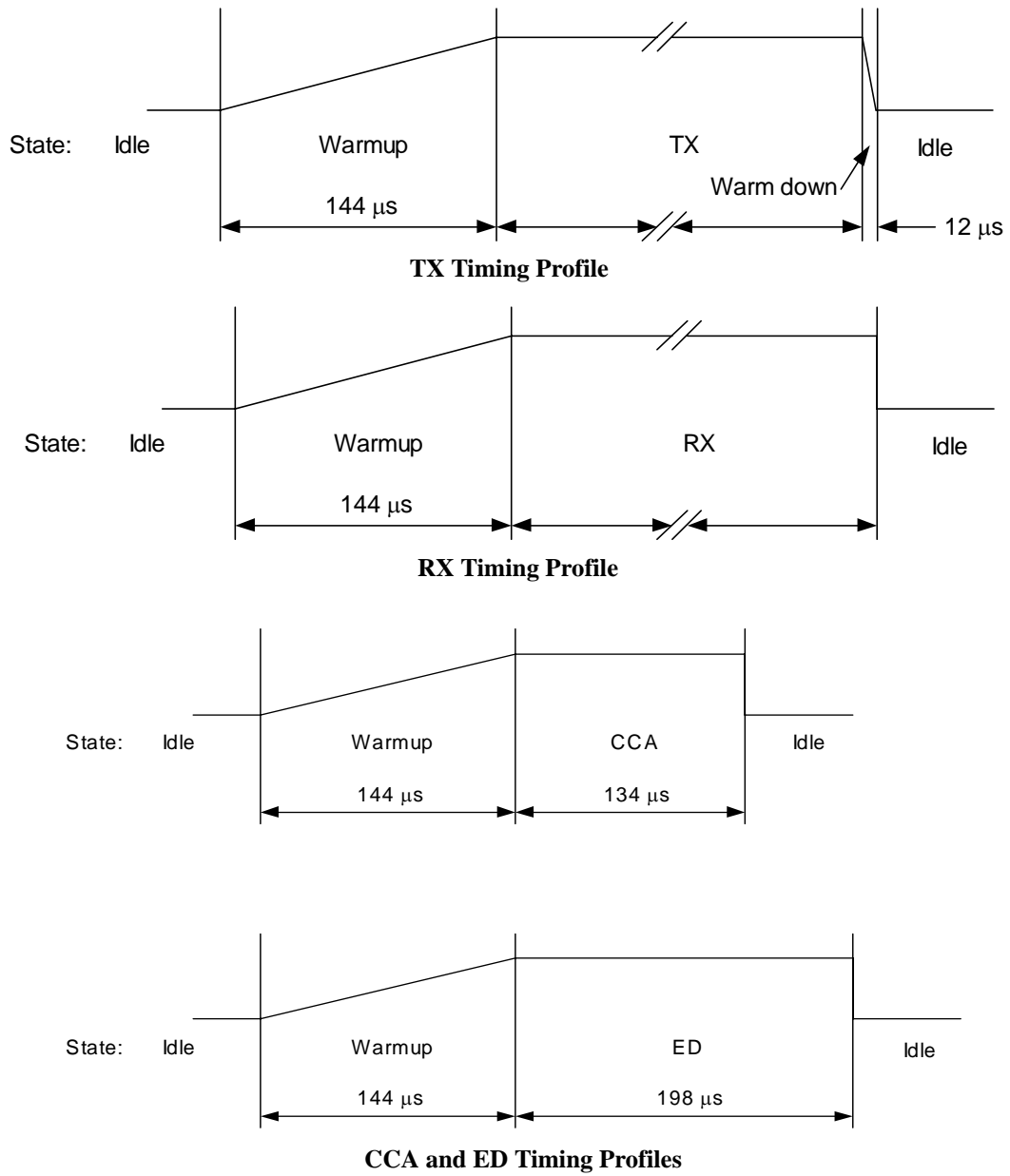


Figure 6-8. MC1322x TX, RX, and CCA/ED Timing Profiles

6.6.1 TSM Register Map

The Transceiver Sequence Manager is managed by a large register block. These registers use default and initialization values provided by Freescale. These registers should be considered reserved by the user.

NOTE

Modem Transceiver Sequence Manager base address is 0x8000_9200

Table 6-8. Modem Transceiver Sequence Manager Register Memory Map

Offset	[31:24]	[23:16]	[15:8]	[7:0]	Access Type	Access Width
+ 0x00 to +0x1FC	Reserved				-	-

6.7 Modem Radio Receiver Module

The radio receiver RF and IF analog functions are controlled by the Receiver Modem which has a large register block. These registers use default and initialization values provided by Freescale. These registers should be considered reserved by the user with the exception of the registers described in this section.

NOTE

Modem Receiver module base address is 0x8000_9400

Table 6-9. Modem Receiver Module Memory Map

Offset	[31:24]	[23:16]	[15:8]	[7:0]	Access Type	Access Width
+ 0x00 to +0x14	Reserved				-	-
+ 0x18	RX SFD Control Register (rx_sfd_ctrl_adr)				R/W	32-bit only
+ 0x1C to +0x5C	Reserved				-	-
+ 0x60	RX AGC CCA and ED Control Register (rx_agc_cca_ed_adr)				R/W	32-bit only
+ 0x64	RX AGC RSSI Parameters Register (rx_agc_rssi_adr)				R/W	32-bit only
+ 0x68 to + 0xFC	Reserved				-	-

6.7.1 SFD Control Register

Base+0x18	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
															sfd1_search	sfd2_search
TYPE															rw	rw
RESET															1	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
TYPE																
RESET	0	1	1	1	1	0	1	0	0	1	0	1	1	1	0	0

Table 6-10. SFD Control Register Bit Descriptions

Bit Number	Description	Operation
17-31	Reserved	
17	sfd1_search — SFD1 Search Enable. Control bit to enable/disable SFD1 search logic. SFD1 is the 802.15.4 standard SFD. If SFD1 search logic is disabled, the 802.15.4 demodulator will be disabled.	Reset value: 1 binary. 1 = Enable SFD1 search logic. 0 = Disable SFD1 search logic.
16	Reserved	
0-15	Reserved	

6.7.2 AGC CCA and ED Control Register

This register controls the mode of the CCA/ED function for the modem.

Base+0x60	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
										ed_en	ed_th					
TYPE										rw	rw	rw	rw	rw	rw	rw
RESET										0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	ed_th		cca_ed_measure_period										cca_mode	cca_en		
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0

Table 6-11. AGC Software Override Register Bit Descriptions

Bit Number	Description	Operation
23-31	Reserved	
22	ed_en - Energy Detect Enable	1 = Energy Detect is enabled. 0 = Energy Detect is disabled.
14-21	ed_th - Energy Detect Threshold. Threshold for determining a clear/busy channel.	
4-14	cca_ed_measure_period - CCA/ED Measure Period. Period to perform a CCA/ED measurement, in units of 8MHz clock cycles	
2-3	cca_mode - CCA Mode. CCA Mode, as defined in the 802.15.4 specification.	
1	cca_en - CCA Enable	1 = CCA is enabled. 0 = CCA is disabled.
0	Reserved???????????	

6.7.3 AGC RSSI Parameters Register

This register sets the sensitivity of the RSSI detection.

Base+0x6 4	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16	
																rss_i_adjust	
TYPE																rw	rw
RESET																0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0	
		rss_i_adjust					rss_i_sensitivity_plus10										
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-12. AGC RSSI Parameters Register Bit Descriptions

Bit Number	Description	Operation
18-31	Reserved	
10-17	rss_i_adjust - RSSI Adjust Factor. This value is subtracted from the raw power estimate from the AGC to obtain an absolute estimate of the signal at the antenna.	
0-9	rss_i_sensitivity_plus10 - RSSI sensitivity Plus 10dBm. This value represents the RSSI that the power estimate measures when the input to the AGC is at a power level of sensitivity plus 10dBm.	

6.8 Modem Radio Transmitter Module

The Modem Transmitter Module module consists of an interface to the MACA that transforms the data to be transmitted based on the 802.15.4 ZigBee standard into I/Q modulation signals for use by the analog phase shift modulator (PSM).

The MACA transfers data (at the bit-level) already formatted with the correct 802.15.4 packet structure. The Tx Modem is responsible for the bit-to-symbol mapping, PN spreading, and PSM encoding.

The radio Transmitter RF analog functions are controlled by the Receiver Modem which has a large register block. These registers use default and initialization values provided by Freescale. These registers should be considered reserved by the user.

NOTE

Modem Radio Transmitter Functions base address is 0x8000_9600

Table 6-13. Modem Transmitter Module Memory Map

Offset	[31:24]	[23:16]	[15:8]	[7:0]	Access Type	Access Width
+ 0x00 to +0x0C	Reserved				-	32-bit only

6.9 Modem Radio Frequency Synthesizer

The Modem Radio Synthesizer Module controls the activation and operation of the RF VCO. In addition, the synthesizer performs the fractional-N computations required to frequency-lock the VCO to a non-integer multiple of an external reference clock. Control of the RF synthesizer parameters is via the module’s control registers. The MACA is responsible for initiating sequences and the synthesizer is controlled the sequence managers.

Figure 6-9 shows a high-level block diagram of the RF PLL within the MC1322x, including the RF synthesizer subsystem.

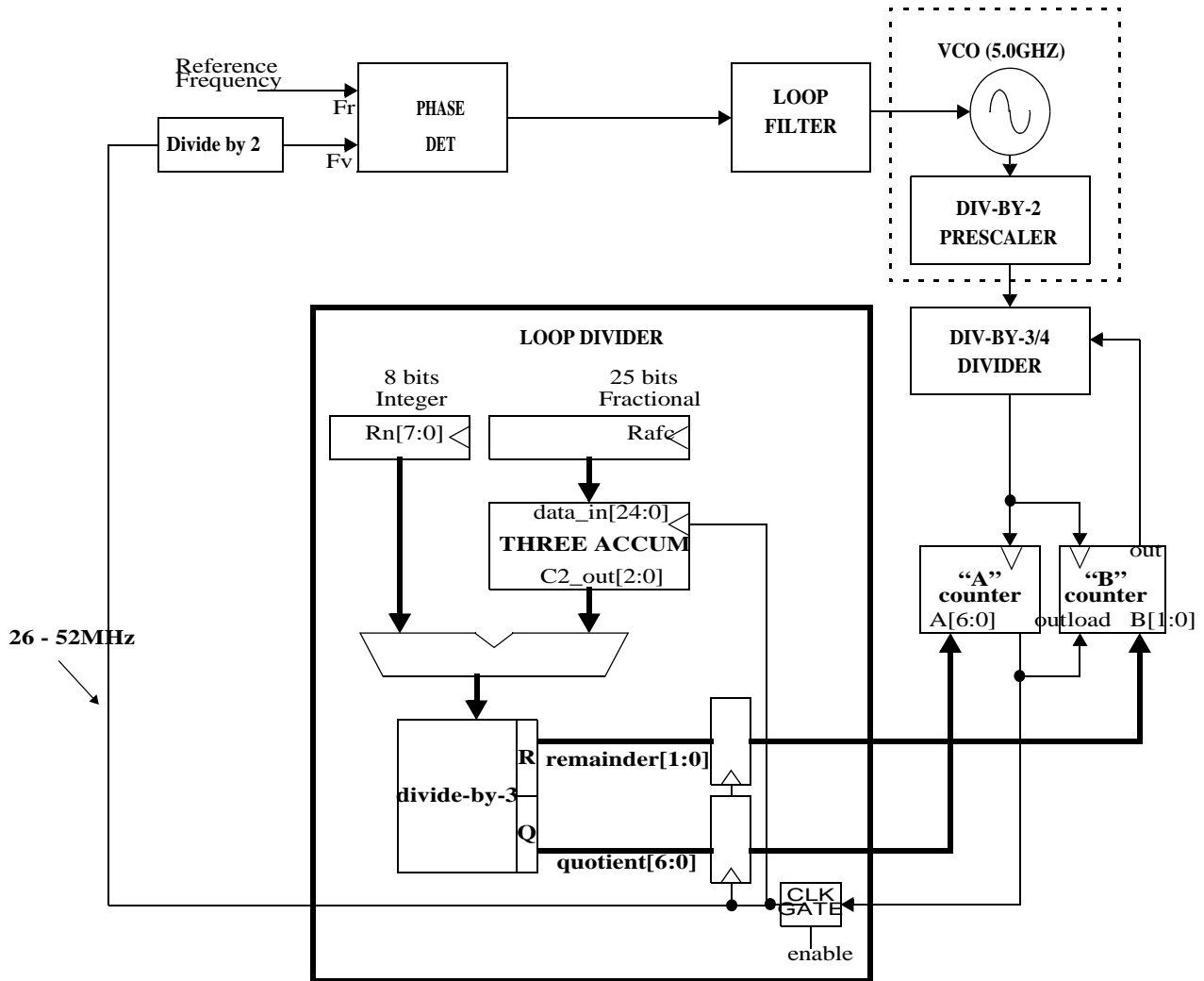


Figure 6-9. RF PLL Block Diagram

Features include:

- The main reference for the RF Synthesizer is an external reference crystal oscillator (Reference Frequency). The reference frequency can be 13MHz to 26MHz, but the default is 24MHz. The divided-down RF VCO (rfvco_div_clk) is also an input to the RF synthesizer subsystem; the VCO loop divider is designed to produce a rfvco_div_clk frequency equal to 2X (double) the xtal_clk frequency.
- The RF synthesizer contains a number of control registers. The register map consists of a 128-word (512B) register field, of which only a subset is utilized. All registers are readable. Some are also writable. Registers are 32-bits wide and only 32-bit-wide accesses should be performed.
- RF synthesizer is comprised of a phase detector, loop filter, voltage-controlled oscillator (VCO), a VCO prescaler, a VCO dual-modulus divider, and a fractional-N division architecture along with digital automatic frequency control (AFC).
- The default mode for the synthesizer start-up and power-down is through the MACA control and sequence managers.

6.9.1 RF Synthesizer Register Map

The MC1322x RF Synthesizer has the following register memory map

NOTE

- Only 32-bit accesses to RF Synthesizer registers are supported.
- Modem Radio Frequency Synthesizer base address is **0x8000_9800**

Table 6-14. RF Synthesizer Register Memory Map

Offset	[31:24]	[23:16]	[15:8]	[7:0]	Access Type	Access Width
+ 0x00 to 0x08	Enable and Override Register (Reserved)				R/W	32
+ 0x0C	VCO Loop Divider Integer (RFSYN_REFDIV)				R/W	32
+ 0x10	VCO Loop Divider Fractional (RFSYN_VCODIV_FRAC)				R/W	32
+ 0x14	VCO Lock (RFSYN_VCOLOCK)				R	32
+ 0x18 to 0x38	Reserved				-	32

6.9.2 VCO Loop Divider (Integer) Register

This register contains the field for the integer value of the VCO loop divider. This register should not be programmed when the VCO is active.

				VCO Loop Divider Integer								Addr Base+0x0C				
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE																
RESET																
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
									rn[7:0]							
TYPE									rw	rw	rw	rw	rw	rw	rw	rw
RESET									1	0	0	0	0	0	0	0

Table 6-15. VCO Loop Divider (Integer) Register Bit Descriptions

Bit Number	Description	Operation
8-31	Reserved	
0-7	rn[7:0] - Integer value of the VCO loop divider. This 8-bit value represents the integer part of the loop divisor.	Writes to this register do not take effect until the VCO Loop Divider (Fractional) Register is also written. The Integer and Fractional Registers should <i>always</i> be written as a pair, in sequence, regardless of whether fractionalization is enabled or not.

6.9.3 VCO Loop Divider (Fractional) Register

This register contains the field for the fractional value of the VCO loop divider. This register should not be programmed when the VCO is active.

								VCO Loop Divider Fractional						Addr Base+0x10				
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16		
								rafc[24:16]										
TYPE								rw	rw	rw	rw	rw	rw	rw	rw	rw		
RESET								0	0	0	0	0	0	0	0	0		
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0		
	rafc[15:0]																	
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 6-16. VCO Loop Divider (Fractional) Register Bit Descriptions

Bit Number	Description	Operation
25-31	Reserved	
0-24	rafc[24:0] - Fractional component of the divide value for the sigma delta VCO loop divider. This value feeds into the three accumulator sigma delta.	A write to this register should be preceded by a write to the VCO Loop Divider (Integer) Register. This is because updates to the Integer register do not take effect until the Fractional register is also written. The Integer and Fractional Registers should <i>always</i> be written as a pair, in sequence, regardless of whether fractionalization is enabled or not.

6.9.4 VCO Lock Register

				VCO Lock									Addr Base+0x14			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE																
RESET																
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
																rfvco_nlock
TYPE																r
RESET																-

This register is read-only.

Table 6-17. VCO Lock Register Bit Descriptions

Bit Number	Description	Operation
1-31	Reserved	
0	rfvco_nlock - RFVCO out-of-lock indicator, synchronized to the crystal reference clock.	1 = RFVCO out-of-lock 0 = RFVCO in lock

6.9.5 Frequency Programming Example

The register fields required for programming the RF synthesizer frequency are shown in Table 6-18.

Table 6-18. Frequency Programming

Register Name	Description
m[7:0]	Integer value of the VCO loop divider for RF Synthesizer
rafc[24:0]	Fractional component of the divide value for the VCO loop divider

The reference frequency of 24.000MHz is used to synthesize a VCO frequency of 2.405GHz (the lowest 802.15.4 channel). The following equation is used to calculate the values of **rn**, and **rafc**.

$$F_{vco} = 2 \times F_{ref} \times \left(RN + 3 + \frac{RAFC}{2^{25}} \right)$$

Figure 6-10. VCO Frequency Formula

where:

$RN = \mathbf{rn}$ (divider integer value)

$RAFC = \mathbf{rafc}$ (divider fractional component)

F_{ref} = Reference Frequency

F_{vco} = VCO frequency to be synthesized (channel center frequency)

NOTE

F_{vco} is the desired frequency out which is the actual frequency of the VCO oscillator divided-by-two.

A reference frequency (F_{ref}) of 24.0MHz is used, i.e., $F_{ref} = 24.000\text{MHz}$

A desired VCO frequency of 2.405GHz is to be synthesized. Solving the equation above results in values shown below.

$$2.405\text{GHz} = 2 \times 24\text{MHz} \times \left(RN + 3 + \frac{RAFC}{2^{25}} \right)$$

$$\frac{2.405\text{GHz}}{48\text{MHz}} - 3 = \left(RN + \frac{RAFC}{2^{25}} \right)$$

$$47.104166667 = RN + \frac{RAFC}{2^{25}}$$

Figure 6-11.

This equation is then split into two equations to solve for the integer divide value (**rn**) and the fractionalization value (**rafc**).

$$rn = 47$$

$$rafc = \lfloor 0.104166667 \times 2^{25} \rfloor = 3495253$$

Figure 6-12.

$RAFC$ is then converted to a hexadecimal base to yield a final value of 0x355555.

In summary, the final values to be programmed into **rn**, and **rafc** shown below.

Table 6-19. Final Programming Values

Register	Final Value
rn	47 = 0x2F
rafc	0x355555

6.10 Modem Tracking Oscillator Module (TOC)

NOTE

Modem Radio Transmitter Functions base address is **0x8000_9A00**

Table 6-20. Modem Tracking Oscillator Memory Map

Offset	[31:24]	[23:16]	[15:8]	[7:0]	Access Type	Access Width
+ 0x00 to +0x10	Reserved				-	32-bit only

6.11 Radio Analog Function Control

The radio analog function control determines operation of the radio analog elements.

NOTE

Modem Radio Analog Functions base address is **0x8000_A000**

Table 6-21. Modem Transmitter Module Memory Map

Offset	[31:24]	[23:16]	[15:8]	[7:0]	Access Type	Access Width
+ 0x00 to +0x04	Reserved				-	-
+ 0x20	Transmit Power Amp Control (PA_CTRL)				W	32
+ 0x24 to + 0x50	Reserved				-	-
+ 0x54	Transmit Power Adjust (PA_ADJ)				W	32
+ 0x58 to + 0x1BC	Reserved				-	-

6.11.1 Transmit Power Amp Control

This register controls the transmitter PA outputs and is write-only and have no read back visibility.

				Transmit Power Amp Control									Radio Base + 0x20			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE																
RESET																
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
			pa_switch													
TYPE																
RESET			1		0		1	0	0	0	0	0				

Table 6-22. Transmit Power Amp Control Register Bit Descriptions

Bit Number	Description	Operation
14-31	Reserved	
13	pa_switch - Controls which differential complementary PA outputs are enables	1 = PA1 enabled, connected to onboard balun (default) 0 = PA2 enabled, connected to PiP pads for use with external amplifier
0-12	Reserved	

6.11.2 Transmit Power Adjust

This register controls the transmitter PA output power level.

				Transmit Power Adjust								Radio Base + 0x54				
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
														pa_target_pwr_ovr[4]	pa_target_pwr_ovr[3]	pa_target_pwr_ovr[2]
TYPE																
RESET																
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	pa_target_pwr_ovr[1]	pa_target_pwr_ovr[0]														
TYPE																
RESET			1		0		1	0	0	0	0	0				

Table 6-23. Transmit Power Adjust Register Bit Descriptions

Bit Number	Description	Operation
19-31	Reserved	
14-18	pa_target_pwr_ovr[4:0] - Controls PA output level	See Table 6-24
0-13	Reserved	

Table 6-24. PA Power Out vs. Field Setting

pa_target_pwr_ovr[4:0]	Pout (dBm)
00	-30
01	-28
02	-26

Table 6-24. PA Power Out vs. Field Setting (continued)

pa_target_pwr_ovr[4:0]	Pout (dBm)
03	-24
04	-22
05	-20
06	-18
07	-16
08	-14
09	-12
0A	-10
0B	-10
0C	-8
0D	-6
0E	-4
0F	-2
10	0
11	2
12	4
13	6
14-1F	Not Allowed
15	

Chapter 7

Central Processing Unit (CPU)

7.1 ARM7TDMI-S Processor Overview

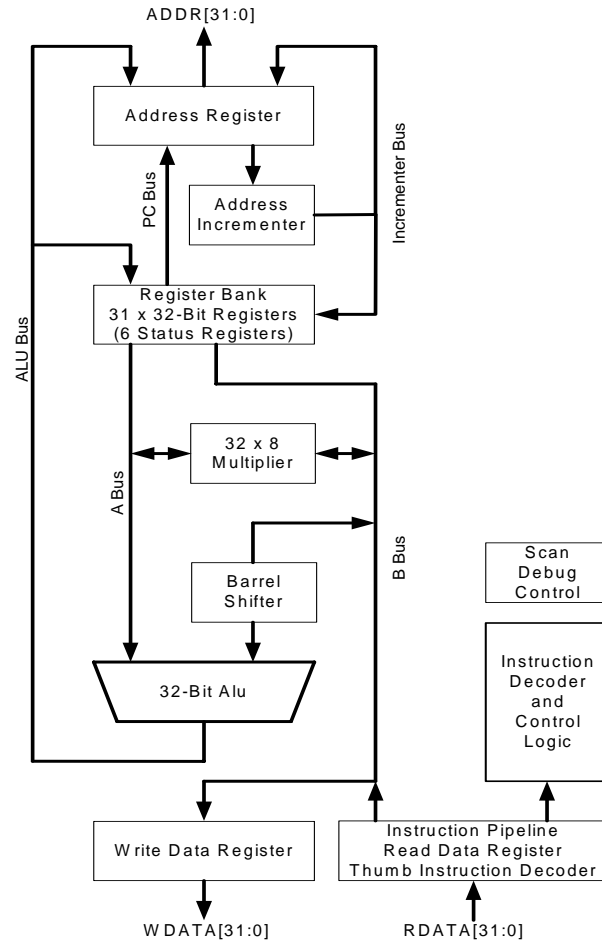


Figure 7-1. ARM7TDMI-S 32-Bit CPU Core

The MC1322x uses a ARM7TDMI-S processor (shown in [Figure 7-1](#)) which is a member of the ARM family of general-purpose 32-bit microprocessors that offers high performance for very low-power consumption and gate count. The ARM architecture is based on Reduced Instruction Set Computer (RISC) principles that give:

- High instruction throughput
- Excellent real-time interrupt response
- Low power

7.1.1 Memory Access

The ARM7TDMI-S CPU uses a single 32-bit data bus that carries both instructions and data. Only load, store, and swap instructions can access data from memory. The address bus is also 32 bits wide.

For data access on the data bus:

- The address is a byte address
- Data can be 8-bit bytes, 16-bit half words, or 32-bit words
- Words must be aligned to 4-byte boundaries
- Half words must be aligned to 2-byte boundaries

7.1.2 Memory Interface

The memory interface of the ARM7TDMI-S processor enables performance potential to be realized, while minimizing the use of memory. The MC1322x ARM7TDMI-S processor has three basic types of memory cycle:

- Internal cycle
- Nonsequential cycle
- Sequential cycle

7.1.3 Instruction Pipeline

The ARM7TDMI-S processor uses a pipeline to increase the speed of the flow of instructions to the processor. This enables several operations to take place simultaneously, and the processing, and memory systems to operate continuously.

A three-stage pipeline is used, so instructions are executed in three stages:

- Fetch
- Decode
- Execute.

The three-stage pipeline is shown in [Figure 7-2](#).

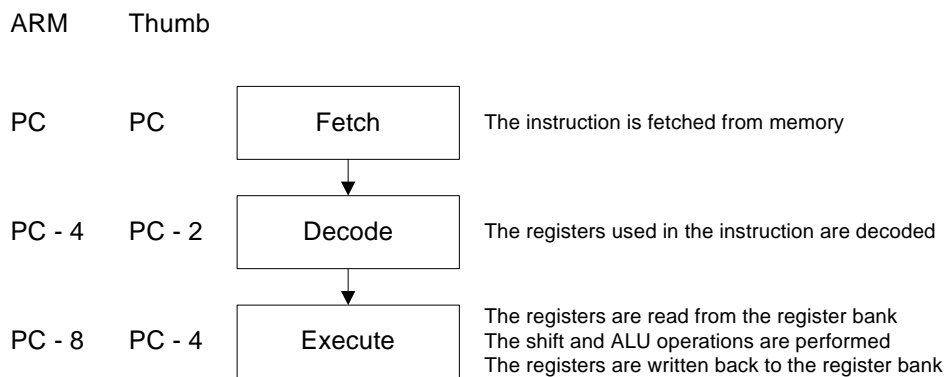


Figure 7-2. The Instruction Pipeline

The Program Counter (PC) points to the instruction being fetched rather than to the instruction being executed.

During normal operation, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

7.2 ARM7TDMI-S Architecture

The ARM7TDMI-S processor has two instruction sets:

- The 32-bit ARM instruction set
- The 16-bit Thumb instruction set

The ARM7TDMI-S processor is an implementation of the ARM architecture v4T. For full details of both the ARM and Thumb instruction sets, see the *ARM Architecture Reference Manual*.

7.2.1 Instruction Compression

Microprocessor architectures traditionally had the same width for instructions and data. Therefore, 32-bit architectures had higher performance manipulating 32-bit data and could address a large address space much more efficiently than 16-bit architectures.

16-bit architectures typically had higher code density than 32-bit architectures, and greater than half the performance.

Thumb implements a 16-bit instruction set on a 32-bit architecture to provide:

- higher performance than a 16-bit architecture
- higher code density than a 32-bit architecture.

7.2.2 The Thumb Instruction Set

The Thumb instruction set is a subset of the most commonly used 32-bit ARM instructions. Thumb instructions are each 16 bits long, and have a corresponding 32-bit ARM instruction that has the same effect on the processor model.

Thumb instructions operate with the standard ARM register configuration, allowing excellent interoperability between ARM and Thumb states.

On execution, 16-bit Thumb instructions are transparently decompressed to full 32-bit ARM instructions in real time, without performance loss.

Thumb has all the advantages of a 32-bit core:

- 32-bit address space
- 32-bit registers
- 32-bit shifter and Arithmetic Logic Unit (ALU)
- 32-bit memory transfer.

Thumb therefore offers a long branch range, powerful arithmetic operations, and a large address space.

Thumb code is typically 65% of the size of the ARM code and provides 160% of the performance of ARM code when running on a processor connected to a 16-bit memory system. Thumb, therefore, makes the ARM7TDMI-S processor ideally suited to embedded applications with restricted memory bandwidth, where code density is important.

The availability of both 16-bit Thumb and 32-bit ARM instruction sets gives designers the flexibility to emphasize performance, or code size on a subroutine level, according to the requirements of their applications. For example, critical loops for applications such as fast interrupts and DSP algorithms can be coded using the full ARM instruction set and linked with Thumb code.

7.3 About the Programmer's Model

The ARM7TDMI-S processor core implements ARM architecture v4T, which includes the 32-bit ARM instruction set and the 16-bit Thumb instruction set. The programmer's model is described fully in the *ARM Architecture Reference Manual* Processor operating states

The ARM7TDMI-S processor has two operating states:

ARM state	32-bit, word-aligned ARM instructions are executed in this state.
Thumb state	16-bit, half word-aligned Thumb instructions.

In Thumb state, the Program Counter (PC) uses bit 1 to select between alternate half words. Transition between ARM and Thumb states does not affect the processor mode or the register contents.

7.3.1 Switching State

You can switch the operating state of the ARM7TDMI-S core between ARM state and Thumb state using the BX instruction. This is described fully in the *ARM Architecture Reference Manual*.

All exception handling is performed in ARM state. If an exception occurs in Thumb state, the processor reverts to ARM state. The transition back to Thumb state occurs automatically on return.

7.3.2 Memory Formats

The ARM7TDMI-S processor views memory as a linear collection of bytes numbered in ascending order from zero:

- bytes 0 to 3 hold the first stored word
- bytes 4 to 7 hold the second stored word
- bytes 8 to 11 hold the third stored word

The ARM7TDMI-S processor can normally treat words in memory as being stored in big-endian format or little-endian format. However, the processor on the MC1322x has been modified to support only little-endian format.

In little-endian format, the lowest-numbered byte in a word is considered the least-significant byte of the word, and the highest-numbered byte is the most significant. So byte 0 of the memory system connects to data lines 7 to 0. This is shown in [Figure 7-3](#).

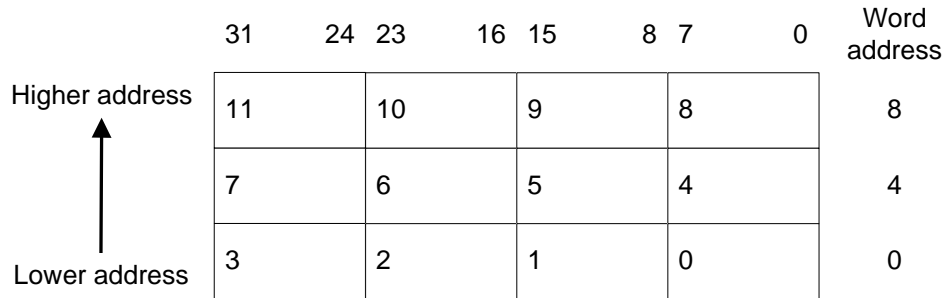


Figure 7-3. Little-endian addresses of bytes within words

7.4 Instruction Length

Instructions are either:

- 32 bits long (in ARM state)
- 16 bits long (in Thumb state)

7.5 Data Types

The ARM7TDMI-S processor supports the following data types:

- word (32-bit)
- half word (16-bit)
- byte (8-bit)

You must align these as follows:

- word quantities must be aligned to four-byte boundaries
- half word quantities must be aligned to two-byte boundaries
- byte quantities can be placed on any byte boundary

7.6 Operating Modes

The ARM7TDMI-S processor has seven operating modes:

- User mode is the usual ARM program execution state, and is used for executing most application programs.
- Fast interrupt (FIQ) mode supports a data transfer or channel process.
- Interrupt (IRQ) mode is used for general-purpose interrupt handling.
- Supervisor mode is a protected mode for the operating system.
- Abort mode is entered after a data or instruction prefetch abort.
- System mode is a privileged user mode for the operating system.
- Undefined mode is entered when an undefined instruction is executed.

Modes other than User mode are collectively known as privileged modes. Privileged modes are used to service interrupts, exceptions, or access protected resources.

7.7 Registers

The ARM7TDMI-S processor has a total of 37 registers:

- 31 general-purpose 32-bit registers
- 6 status registers.

These registers are not all accessible at the same time. The processor state and operating mode determine which registers are available to the programmer.

7.7.1 The ARM State Register Set

In ARM state, 16 general registers, and one or two status registers are accessible at any one time. In privileged modes, mode-specific banked registers become available. [Figure 7-4](#) shows which registers are available in each mode.

The ARM state register set contains 16 directly-accessible registers, r0 to r15. An additional register, the Current Program Status Register (CPSR), contains condition code flags, and the current mode bits. Registers r0 to r13 are general-purpose registers used to hold either data or address values. Registers r14 and r15 have the following special functions:

Link register	Register 14 is used as the subroutine Link Register (LR). r14 receives a copy of r15 when a Branch with Link (BL) instruction is executed. At all other times you can treat r14 as a general-purpose register. The corresponding banked registers r14_svc, r14_irq, r14_fiq, r14_abt, and r14_und are similarly used to hold the return values of r15 when interrupts and exceptions arise, or when BL instructions are executed within interrupt or exception routines.
Program counter	Register 15 holds the Program Counter (PC). In ARM state, bits [1:0] of r15 are zero. Bits [31:2] contain the PC. In Thumb state, bit [0] is zero. Bits [31:1] contain the PC.

In privileged modes, another register, the Saved Program Status Register (SPSR), is accessible. This contains the condition code flags, and the mode bits saved as a result of the exception that caused entry to the current mode.

See [Section 7.8, “The Program Status Registers”](#) for a description of the program status registers.

Banked registers have a mode identifier that shows to which User mode register they are mapped. These mode identifiers are shown in [Table 7-1](#).

Table 7-1. Register mode identifiers

Model	Mode identifier
User	usr
Fast interrupt	fiq
Interrupt	irq
Supervisor	svc

Table 7-1. Register mode identifiers (continued)

Model	Mode identifier
Abort	abt
System	sys
Undefined	und

FIQ mode has seven banked registers mapped to r8–r14 (r8_fiq–r14_fiq). In ARM state, most of the FIQ handlers do not have to save any registers. The User, IRQ, Supervisor, Abort, and undefined modes each have two banked registers mapped to r13 and r14, allowing a private stack pointer and LR for each mode. Figure 7-4 shows the ARM state registers.

ARM state general registers and program counter

System and User	FIQ	Supervisor	Abort	IRQ	Undefined
r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r2	r2
r3	r3	r3	r3	r3	r3
r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7
r8	r8_fiq	r8	r8	r8	r8
r9	r9_fiq	r9	r9	r9	r9
r10	r10_fiq	r10	r10	r10	r10
r11	r11_fiq	r11	r11	r11	r11
r12	r12_fiq	r12	r12	r12	r12
r13	r13_fiq	r13_svc	r13_abt	r13_irq	r13_und
r14	r14_fiq	r14_svc	r14_abt	r14_irq	r14_und
r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)

ARM state program status registers

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

 = banked register

Figure 7-4. Register organization in ARM state

7.7.2 The Thumb State Register Set

The Thumb state register set is a subset of the ARM state set. The programmer has direct access to:

- eight general registers, r0–r7
- the PC
- a Stack Pointer (SP)
- a Link Register (LR)
- the CPSR.

There are banked SPs, LRs, and SPSRs for each privileged mode. This register set is shown in [Figure 7-5](#).

Thumb state general registers and program counter

System and User	FIQ	Supervisor	Abort	IRQ	Undefined
r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r2	r2
r3	r3	r3	r3	r3	r3
r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7
SP	SP_fiq	SP_svc	SP_abt	SP_irq	SP_und
LR	LR_fiq	LR_svc	LR_abt	LR_irq	LR_und
PC	PC	PC	PC	PC	PC

Thumb state program status registers

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

 = banked register

Figure 7-5. Register organization in Thumb state

7.7.3 The Relationship Between ARM State and Thumb State Registers

The Thumb state registers relate to the ARM state registers in the following way:

- Thumb state r0–r7, and ARM state r0–r7 are identical
- Thumb state CPSR and SPSRs, and ARM state CPSR and SPSRs are identical
- Thumb state SP maps onto ARM state r13
- Thumb state LR maps onto ARM state r14

- The Thumb state PC maps onto the ARM state PC (r15).

These relationships are shown in [Figure 7-6](#).

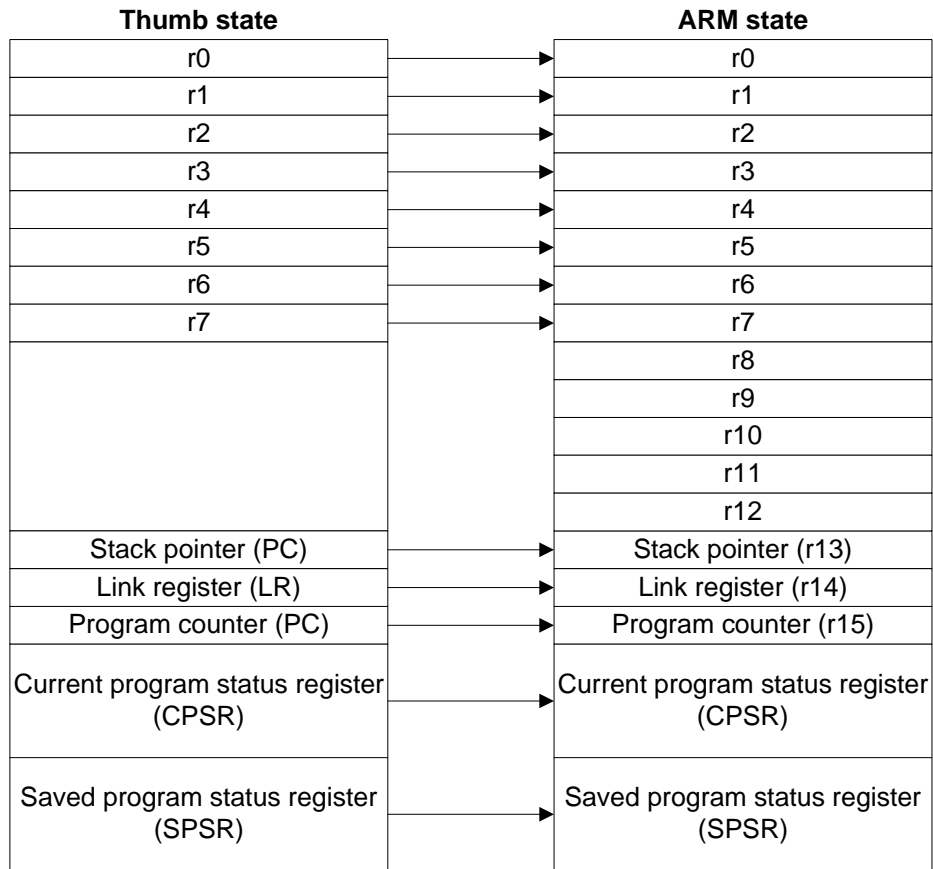


Figure 7-6. Mapping of Thumb state registers onto ARM state registers

Registers r0–r7 are known as the low registers. Registers r8–r15 are known as the high registers.

7.7.4 Accessing High Registers in Thumb State

In Thumb state, the high registers (r8–r15) are not part of the standard register set. The assembly language programmer has limited access to them, but can use them for fast temporary storage.

You can use special variants of the `MOV` instruction to transfer a value from a low register (in the range r0–r7) to a high register, and from a high register to a low register. The `CMP` instruction enables you to compare high register values with low register values. The `ADD` instruction enables you to add high register values to low register values. For more details, see the *ARM Architecture Reference Manual*.

7.8 The Program Status Registers

The ARM7TDMI-S core contains a CPSR and five SPSRs for exception handlers to use. The program status registers:

- hold the condition code flags
- control the enabling and disabling of interrupts
- set the processor operating mode

The arrangement of bits is shown in [Figure 7-7](#).

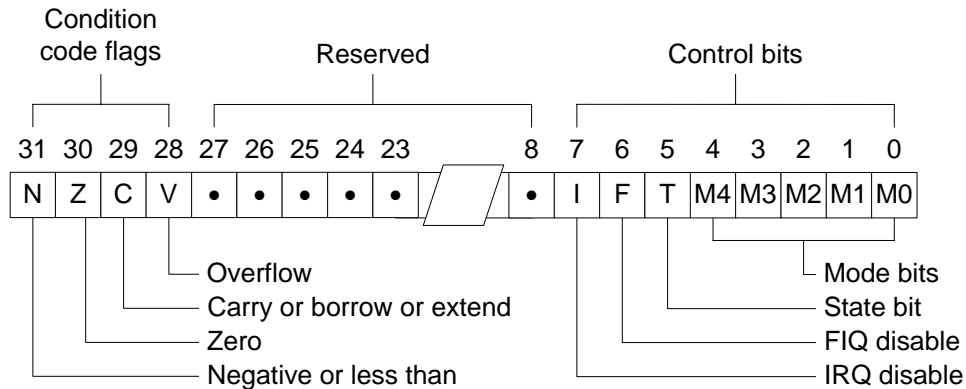


Figure 7-7. Program status register format

To maintain compatibility with future ARM processors, and as good practice, you are strongly advised to use a read-write-modify strategy when changing the CPSR.

7.8.1 The Condition Code Flags

The N, Z, C, and V bits are the condition code flags. You can set these bits by arithmetic and logical operations. The flags can also be set by MSR and LDM instructions. The ARM7TDMI-S processor tests these flags to determine whether to execute an instruction.

All instructions can execute conditionally in ARM state. In Thumb state, only the Branch instruction can be executed conditionally. For more information about conditional execution, see the *ARM Architecture Reference Manual*.

7.8.2 The Control Bits

The bottom eight bits of a PSR are known collectively as the control bits. They are the:

- [Section 7.8.3, “Interrupt Disable Bits”](#)
- [Section 7.8.4, “T Bit”](#)
- [Section 7.8.5, “Mode Bits”](#)

The control bits change when an exception occurs. When the processor is operating in a privileged mode, software can manipulate these bits.

7.8.3 Interrupt Disable Bits

The I and F bits are the interrupt disable bits:

- when the I bit is set, IRQ interrupts are disabled
- when the F bit is set, FIQ interrupts are disabled.

7.8.4 T Bit

The T bit reflects the operating state:

- when the T bit is set, the processor is executing in Thumb state
- when the T bit is clear, the processor executing in ARM state.

The operating state is reflected by the **CPTBIT** external signal.

Never use an MSR instruction to force a change to the state of the T bit in the CPSR. If you do this, the processor enters an unpredictable state.

7.8.5 Mode Bits

The M4, M3, M2, M1, and M0 bits (M[4:0]) are the mode bits. These bits determine the processor operating mode as shown in [Table 7-2](#). Not all combinations of the mode bits define a valid processor mode, so take care to use only the bit combinations shown.

Table 7-2. PSR mode bit values

M[4:0]	Mode	Visible Thumb state registers	Visible ARM state registers
10000	User	r0–r7, SP, LR, PC, CPSR	r0–r14, PC, CPSR
10001	FIQ	r0–r7, SP_fiq, LR_fiq, PC, CPSR, SPSR_fiq	r0–r7, r8_fiq–r14_fiq, PC, CPSR, SPSR_fiq
10010	IRQ	r0–r7, SP_irq, LR_irq, PC, CPSR, SPSR_irq	r0–r12, r13_irq, r14_irq, PC, CPSR, SPSR_irq
10011	Supervisor	r0–r7, SP_svc, LR_svc, PC, CPSR, SPSR_svc	r0–r12, r13_svc, r14_svc, PC, CPSR, SPSR_svc
10111	Abort	r0–r7, SP_abt, LR_abt, PC, CPSR, SPSR_abt	r0–r12, r13_abt, r14_abt, PC, CPSR, SPSR_abt
11011	Undefined	r0–r7, SP_und, LR_und, PC, CPSR, SPSR_und	r0–r12, r13_und, r14_und, PC, CPSR, SPSR_und
11111	System	r0–r7, SP, LR, PC, CPSR	r0–r14, PC, CPSR

If users program an illegal value into M[4:0], the processor enters an unrecoverable state.

7.8.6 Reserved Bits

The remaining bits in the PSRs are unused but are reserved. When changing a PSR flag or control bits make sure that these reserved bits are not altered. Also, make sure that your program does not rely on reserved bits containing specific values because future processors might have these bits set to one or zero.

7.9 Exceptions

Exceptions arise whenever the normal flow of a program has to be halted temporarily, for example to service an interrupt from a peripheral. Before attempting to handle an exception, the ARM7TDMI-S core preserves the current processor state so that the original program can resume when the handler routine has finished.

If two or more exceptions arise simultaneously, the exceptions are dealt with in the fixed order given in [Section 7.9.10, “Exception Priorities”](#).

This section provides details of the exception handling on the ARM7TDMI-S processor:

- [Section 7.9.1, “Exception Entry/Exit Summary”](#)
- [Section 7.9.2, “Entering an Exception”](#)
- [Section 7.9.3, “Leaving an Exception”](#)

7.9.1 Exception Entry/Exit Summary

[Table 7-3](#) shows the PC value preserved in the relevant r14 on exception entry and the recommended instruction for exiting the exception handler.

Table 7-3. Exception entry and exit

Exception or entry	Return instruction	Previous state		Notes
		ARM r14_x	Thumb r14_x	
BL	MOV PC, R14	PC + 4	PC + 2	Where the PC is the address of the BL, SWI, undefined instruction Fetch, or instruction that had the Prefetch Abort.
SWI	MOVS PC, R14_svc	PC + 4	PC + 2	
Undefined instruction	MOVS PC, R14_und	PC + 4	PC + 2	
Prefetch Abort	SUBS PC, R14_abt, #4	PC + 4	PC + 4	
FIQ	SUBS PC, R14_fiq, #4	PC + 4	PC + 4	Where the PC is the address of the instruction that was not executed because the FIQ or IRQ took priority.
IRQ	SUBS PC, R14_irq, #4	PC + 4	PC + 4	
Data Abort	SUBS PC, R14_abt, #8	PC + 8	PC + 8	Where the PC is the address of the Load or Store instruction that generated the Data Abort.
RESET	Not applicable	-	-	The value saved in r14_svc on reset is UNPREDICTABLE.

7.9.2 Entering an Exception

When handling an exception the ARM7TDMI-S core:

1. Preserves the address of the next instruction in the appropriate LR. When the exception entry is from:
 - ARM state, the ARM7TDMI-S copies the address of the next instruction into the LR (current PC + 4, or PC + 8 depending on the exception)

- Thumb state, the ARM7TDMI-S writes the value of the PC into the LR, offset by a value (current PC + 4, or PC + 8 depending on the exception)

The exception handler does not have to determine the state when entering an exception. For example, in the case of a SWI, `MOVS PC, r14_svc` always returns to the next instruction regardless of whether the SWI was executed in ARM or Thumb state.

2. Copies the CPSR into the appropriate SPSR.
3. Forces the CPSR mode bits to a value which depends on the exception.
4. Forces the PC to fetch the next instruction from the relevant exception vector.

The ARM7TDMI-S core also sets the interrupt disable flags on interrupt exceptions to prevent otherwise unmanageable nestings of exceptions.

Exceptions are always handled in ARM state. When the processor is in Thumb state and an exception occurs, the switch to ARM state takes place automatically when the exception vector address is loaded into the PC.

7.9.3 Leaving an Exception

When an exception is completed, the exception handler must:

1. Move the LR, minus an offset to the PC. The offset varies according to the type of exception, as shown in [Table 7-4](#).
2. Copy the SPSR back to the CPSR.
3. Clear the interrupt disable flags that were set on entry.

The action of restoring the CPSR from the SPSR automatically restores the T, F, and I bits to whatever value they held immediately prior to the exception.

7.9.4 Fast Interrupt Request (FIQ)

The Fast Interrupt Request (FIQ) exception supports data transfers or channel processes. In ARM state, FIQ mode has eight private registers to remove the need for register saving (this minimizes the overhead of context switching). An FIQ is externally generated by taking the nFIQ signal input LOW.

Irrespective of whether exception entry is from ARM state, or from Thumb state, an FIQ handler returns from the interrupt by executing:

```
SUBS PC,R14_fiq,#4
```

You can disable FIQ exceptions within a privileged mode by setting the CPSR F flag. When the F flag is clear, the ARM7TDMI-S checks for a LOW level on the output of the FIQ synchronizer at the end of each instruction.

7.9.5 Interrupt Request (IRQ)

The Interrupt Request (IRQ) exception is a normal interrupt caused by a LOW level on the nIRQ input. IRQ has a lower priority than FIQ, and is masked on entry to an FIQ sequence. You can disable IRQ at any time, by setting the I bit in the CPSR from a privileged mode.

Irrespective of whether exception entry is from ARM state, or Thumb state, an IRQ handler returns from the interrupt by executing:

```
SUBS PC,R14_irq,#4
```

7.9.6 Abort

An abort indicates that the current memory access cannot be completed. It is signalled by the external ABORT input. The ARM7TDMI-S checks for the abort exception at the end of memory access cycles.

There are two types of abort:

- a Prefetch Abort occurs during an instruction prefetch
- a Data Abort occurs during a data access.

NOTE

- Memory access aborts are supported only for memory with the exception of the UART modules. All other peripheral functions do not support abort.
- Abort on misaligned accesses is not supported.

7.9.6.1 Prefetch Abort

When a Prefetch Abort occurs, the ARM7TDMI-S core marks the prefetched instruction as invalid, but does not take the exception until the instruction reaches the execute stage of the pipeline. If the instruction is not executed because a branch occurs while it is in the pipeline, the abort does not take place.

After dealing with the reason for the abort, the handler executes the following instruction irrespective of the processor operating state:

```
SUBS PC,R14_abt,#4
```

This action restores both the PC and the CPSR and retries the aborted instruction.

7.9.6.2 Data Abort

When a Data Abort occurs, the action taken depends on the instruction type:

- Single data transfer instructions (LDR, STR) write back modified base registers. The abort handler must be aware of this
- The swap instruction (SWP) aborts as though it had not been executed. (The abort must occur on the read access of the SWP instruction.)

- Block data transfer instructions (LDM, STM) complete. When write-back is set, the base is updated. If the instruction would have overwritten the base with data (when it has the base register in the transfer list), the ARM7TDMI-S prevents the overwriting

The ARM7TDMI-S core prevents all register overwriting after an abort is indicated. This means that the ARM7TDMI-S core always preserves r15 (always the last register to be transferred) in an aborted LDM instruction

The abort mechanism enables the implementation of a demand-paged virtual memory system. In such a system, the processor is allowed to generate arbitrary addresses. When the data at an address is unavailable, the Memory Management Unit (MMU) signals an abort. The abort handler must then work out the cause of the abort, make the requested data available, and retry the aborted instruction. The application program does not have to know the amount of memory available to it, nor is its state in any way affected by the abort.

After fixing the reason for the abort, the handler must execute the following return instruction irrespective of the processor operating state at the point of entry:

```
SUBS PC,R14_abt,#8
```

This action restores both the PC, and the CPSR, and retries the aborted instruction.

7.9.7 Software Interrupt Instruction

The Software Interrupt (SWI) is used to enter Supervisor mode, usually to request a particular supervisor function. A SWI handler returns by executing the following irrespective of the processor operating state:

```
MOVS PC, R14_svc
```

This action restores the PC and CPSR, and returns to the instruction following the SWI. The SWI handler reads the opcode to extract the SWI function number.

7.9.8 Undefined Instruction

When the ARM7TDMI-S processor encounters an instruction that neither it nor any coprocessor in the system can handle, the ARM7TDMI-S core takes the undefined instruction trap. Software can use this mechanism to extend the ARM instruction set by emulating undefined coprocessor instructions.

The ARM7TDMI-S processor is fully compliant with the ARM architecture v4T, and traps all instruction bit patterns that are classified as undefined.

After emulating the failed instruction, the trap handler executes the following irrespective of the processor operating state:

```
MOVS PC,R14_und
```

This action restores the CPSR and returns to the next instruction after the undefined instruction.

For more information about undefined instructions, see the *ARM Architecture Reference Manual*.

7.9.9 Exception Vectors

Table 7-4 shows the exception vector addresses. In the table, columns I and F represent the previous value.

Table 7-4. Exception vectors

ARM Address (ROM)	Functional Address ¹ (RAM)	Exception	Mode on entry	I state on entry	F state on entry
0x0000_0000	0x0040_0000	Reset	Supervisor	Disabled	Disabled
0x0000_0004	0x0040_0004	Undefined instruction	Undefined	I	F
0x0000_0008	0x0040_0008	Software interrupt	Supervisor	Disabled	F
0x0000_000C	0x0040_000C	Abort (Prefetch)	Abort	I	F
0x0000_0010	0x0040_0010	Abort (Data)	Abort	I	F
0x0000_0014	0x0040_0014	Reserved	Reserved	-	-
0x0000_0018	0x0040_0018	IRQ	IRQ	Disabled	F
0x0000_001C	0x0040_001C	FIQ	FIQ	Disabled	Disabled

¹ The normal ARM exception vectors are located in ROM and are not programmable. For all exceptions except Reset, the ROM vector function is a simple jump to the appropriate address in RAM such that the actual vector software handler can begin execution at the RAM vector address.

7.9.10 Exception Priorities

When multiple exceptions arise at the same time, a fixed priority system determines the order in which they are handled:

1. Reset (highest priority).
2. Data Abort.
3. FIQ.
4. IRQ.
5. Prefetch Abort.
6. Undefined instruction.
7. SWI (lowest priority).

Some exceptions cannot occur together:

- The Undefined Instruction and SWI exceptions are mutually exclusive. Each corresponds to a particular (non-overlapping) decoding of the current instruction.
- When FIQs are enabled and a Data Abort occurs at the same time as an FIQ, the ARM7TDMI-S core enters the Data Abort handler and proceeds immediately to the FIQ vector.

A normal return from the FIQ causes the Data Abort handler to resume execution.

Data Aborts must have higher priority than FIQs to ensure that the transfer error does not escape detection. You must add the time for this exception entry to the worst-case FIQ latency calculations in a system that uses aborts.

7.10 Interrupt Latencies

Interrupt latencies are described in:

- [Section 7.10.1, “Maximum Interrupt Latencies”](#)
- [Section 7.10.2, “Minimum Interrupt Latencies”](#).

7.10.1 Maximum Interrupt Latencies

When FIQs are enabled, the worst-case latency for FIQ comprises a combination of:

- T_{syncmax} , the longest time the request can take to pass through the synchronizer. T_{syncmax} is two processor cycles.
- T_{ldm} , the time for the longest instruction to complete. (The longest instruction is an LDM that loads all the registers including the PC.) T_{ldm} is 20 cycles in a zero wait state system.
- T_{exc} , the time for the Data Abort entry. T_{exc} is three cycles.
- T_{fiq} , the time for FIQ entry. T_{fiq} is two cycles.

The total latency is therefore 27 processor cycles, slightly less than 0.7 microseconds in a system that uses a continuous 40MHz processor clock. At the end of this time, the ARM7TDMI-S executes the instruction at `0x1c`.

The maximum IRQ latency calculation is similar, but must allow for the fact that FIQ, having higher priority, might delay entry into the IRQ handling routine for an arbitrary length of time.

7.10.2 Minimum Interrupt Latencies

The minimum latency for FIQ or IRQ is the shortest time the request can take through the synchronizer, T_{syncmin} plus T_{fiq} (four processor cycles).

7.11 Reset

When the nRESET signal goes LOW, the ARM7TDMI-S processor abandons the executing instruction.

When nRESET goes HIGH again the ARM7TDMI-S processor:

- Forces M[4:0] to b10011 (Supervisor mode).
- Sets the I and F bits in the CPSR.
- Clears the CPSR T bit.
- Forces the PC to fetch the next instruction from address `0x00`.
- Reverts to ARM state and resumes execution.

After reset, all register values except the PC and CPSR are indeterminate.

Chapter 8

Interrupt Controller (ITC)

8.1 MC1322x MCU Interrupt Operation Overview

The MCU interrupt request service can be viewed as comprised of three basic functions as illustrated in Figure 8-1.

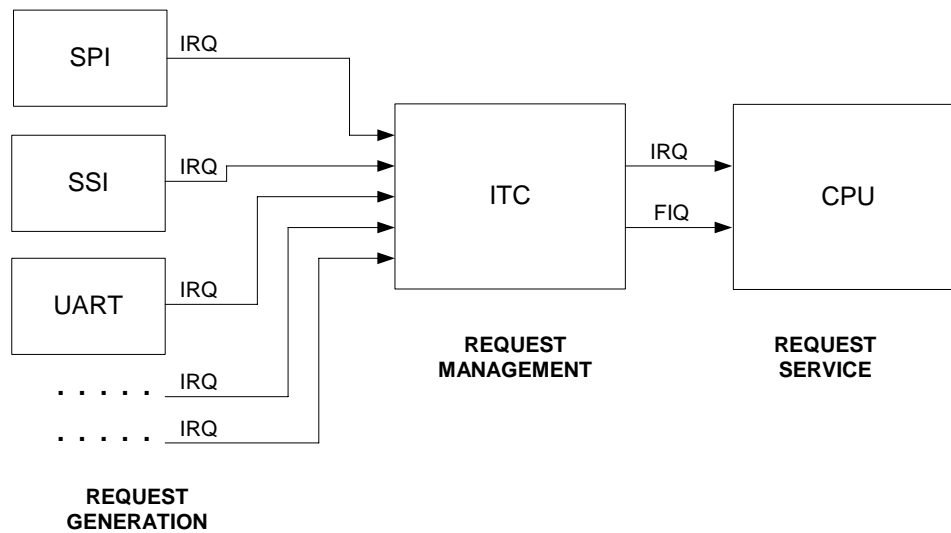


Figure 8-1. Interrupt Request Service Functions

- Request Generation - An interrupt request typically gets generated at a peripheral device. The peripheral control determines the source(s) of the interrupt request. There is a single request per peripheral.
- Request Management - The Interrupt Controller (ITC) accepts the interrupt request from each module source. These multiple requests are prioritized and mapped to one of two basic requests to the CPU, i.e., the fast interrupt request (FIQ) and the normal interrupt request (IRQ).
- Request Service - The CPU has the final responsibility of servicing the interrupt requests that have been mapped into the two simple signals of FIQ and IRQ.

The interrupt service is controlled as well as disabled or enabled across these three functions. Request generation is generated in the peripheral device.

NOTE

Freescale provides a software driver utility for the Interrupt Controller. This is briefly referenced in [Section Appendix B, “MC1322x Software Driver Utilities”](#) and is documented in the *MC1322x Software Driver Reference Manual*, (22XDRVRRM).

8.1.1 CPU Request Service

The CPU functionality is discussed in [Chapter 7, “Central Processing Unit \(CPU\)”](#) in greatest detail, but is summarized here as a review, and to show how the ITC relates to the request service.

The ARM7TDMI-S processor has seven operating modes and two of these are used to service interrupts:

- Fast interrupt (FIQ) mode supports a data transfer or channel process.
- Interrupt (IRQ) mode is used for general-purpose interrupt handling.

The interrupt modes are entered as exception processes. Exceptions arise whenever the normal flow of a program has to be halted temporarily, for example to service an interrupt from a peripheral. There are seven exception conditions and when multiple exceptions arise at the same time, a fixed priority system determines the order in which they are handled. The ARM7TDMI-S core imposes the following priority among the various exceptions:

- Reset (highest priority)
- Data abort
- Fast interrupt
- Normal interrupt
- Prefetch abort
- Undefined instruction and SWI (lowest priority)

The ARM7TDMI-S core has Program Status Registers (PSR) for exception handlers to use. The PSRs include a Current Program Status Register (CPSR) and five Stored Program Status Registers (SPSR). The program status registers:

- hold the condition code flags
- control the enabling and disabling of interrupts
- set the processor operating mode

The PSR arrangement of bits is shown in [Figure 8-2](#).

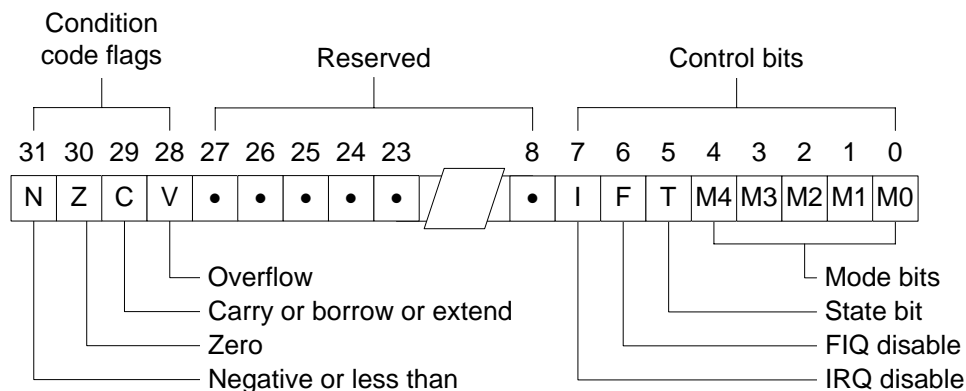


Figure 8-2. Program Status Register Format

The PSR bits important to interrupt service are the Interrupt Disable Bits:

- I Bit (IRQ Disable) - when the I bit is set, IRQ interrupts are disabled

- F Bit (FIQ Disable) - when the F bit is set, FIQ interrupts are disabled.

Before attempting to handle an exception, the ARM7TDMI-S core preserves the current processor state so that the original program can resume when the handler routine has finished. When handling an exception the ARM7TDMI-S core:

1. Preserves the address of the next instruction.
2. Copies the CPSR into the appropriate SPSR.
3. Forces the CPSR mode bits to a value which depends on the exception.
4. Forces the PC to fetch the next instruction from the relevant exception vector.

The ARM7TDMI-S core also sets the interrupt disable flags on interrupt exceptions to prevent otherwise unmanageable nestings of exceptions. Exceptions are always handled in ARM state.

When an exception service is completed, the exception handler must:

1. Restore the address of the next instruction.
2. Copy the SPSR back to the CPSR.
3. Clear the interrupt disable flags that were set on entry.

The action of restoring the CPSR from the SPSR automatically restores the T, F, and I bits to whatever value they held immediately prior to the exception.

8.1.1.1 Fast Interrupt Request (FIQ)

The Fast Interrupt Request (FIQ) exception supports data transfers or channel processes and is caused by a request on the FIQ internal signal. In ARM state, FIQ mode has eight private registers to remove the need for register saving (this minimizes the overhead of context switching).

FIQ exceptions can be disabled within a privileged mode by setting the CPSR F flag.

8.1.1.2 Interrupt Request (IRQ)

The Interrupt Request (IRQ) exception is a normal interrupt caused by a request on the IRQ internal signal. IRQ has a lower priority than FIQ, and is masked on entry to an FIQ sequence. IRQ can be disabled at any time, by setting the I bit in the CPSR from a privileged mode.

8.1.1.3 Software Interrupt Instruction

It should be noted the CPU also supports a Software Interrupt (SWI) that is used to enter Supervisor mode, usually to request a particular supervisor function. This has no relation to interrupt requests coming from IRQ and FIQ and has no connection to the ITC.

8.1.2 Interrupt Request Generation

Interrupt requests are generated within a peripheral function. Each function has all its internal interrupt request sources channelled to its single IRQ signal. The ITC then priorities these requests and maps them to either the IRQ or FIQ inputs to the CPU.

NOTE

The interrupt requests for a given module are described in the chapter on that module. The user is directed to the individual module descriptions in those chapters.

8.2 Interrupt Controller Overview

The ITC is a peripheral function that resides on the CPU data bus which collects interrupt requests from the peripheral sources and provides an interface to the CPU core. The ITC consists of a set of control registers and associated logic to perform interrupt masking and priority mapping of requests to normal and fast interrupts. The priority levels can be controlled by software by simply masking interrupts. Figure 8-3 shows a logical block diagram of the ITC.

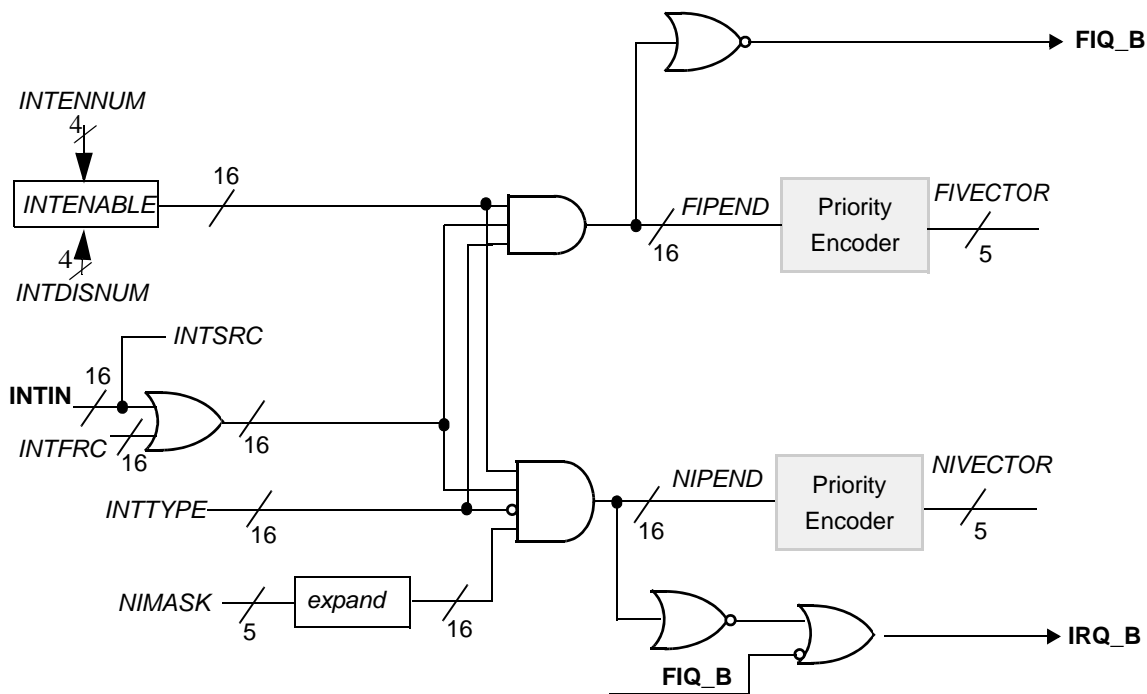


Figure 8-3. ITC Logic Block Diagram

8.3 ITC Features

The ITC provide the following features:

- Supports all internal interrupt sources
- Maps sources to normal and fast interrupt requests
- Indicates pending interrupt sources via a register for normal and fast interrupts
- Indicates highest priority interrupt number via register
- Independently enables/disables any interrupt source

- Provides a mechanism for software to force an interrupt

8.4 Interrupt Controller Operation

NOTE

In the following discussion, a number of registers/fields support 16 separate interrupt requests. However, due to the MC1322x peripherals implementation only 11 individual interrupt requests are used (see [Table 8-1](#)).

Referring to Figure 8-3, the Interrupt Source Register (INTSRC) is a 16-bit status register with a single interrupt source associated with each of the 16 bits. An interrupt request signal (INTIN are module IRQ inputs) is routed from each interrupt source to the INTSRC Register. This allows up to 16 distinct interrupt sources in an implementation. As an aid to software development and debug, interrupt requests can also be forced to be asserted by way of the Interrupt Force Register (INTFRC). Each bit in the INTFRC Register is logically ORED with the corresponding hardware request line and the result is routed into the INTSRC Register inputs.

For each interrupt request, there is a corresponding bit in the Interrupt Enable Register (INTENABLE) which allows masking (enable/disable) of the individual bits of the INTSRC register. The INTENABLE bits can be set or cleared by a direct write, or alternatively, the Interrupt Enable Number Register (INTENUM) can set one source at-a-time, or the Interrupt Disable Number Register (INTDISNUM) can clear (disable) one source at a time.

Once the interrupt request has been enabled (masked/unmasked), there is also an Interrupt Type Register (INTTYPE) which selects whether the interrupt source generates (is mapped to) a normal (IRQ) or fast (FIQ) interrupt to the ARM7TDMI-S core.

NOTE

Mapping and prioritization of interrupt requests is discussed in [Section 8.4.1, “ITC Prioritization of Interrupt Sources”](#).

For reading status and to generate the hardware IRQ and FIQ signals to the CPU core, there are two interrupt pending registers:

- Normal Interrupt Pending Register (NIPEND) - this register indicates all pending normal interrupt requests.
 - Each bit is the equivalent of the logical AND of the INTSRC bit, the INTENABLE bit, and the NOT of the INTTYPE bit.
 - The NIPEND Register bits are bit-wise ORed together to form the IRQ signal to the ARM7TDMI-S core.
 - The IRQ core input signal is maskable by the Normal Interrupt Disable Bit (I bit) in the Processor Status Register (CPSR). The Normal Interrupt Vector (NIVECTOR) indicates the vector index of highest priority pending normal interrupt.
- Fast Interrupt Pending Register (FIPEND) - this register indicates all pending fast interrupt requests.

- Each bit is the equivalent of the logical AND of the INTSRC bit, the INTENABLE bit, and the the INTTYPE bit.
- The FIPEND Register bits are bit-wise NORed together to form the FIQ signal routed to the ARM7TDMI-S core.
- The FIQ core input signal is maskable by the Fast Interrupt Disable Bit (F bit) in the CPSR. The fast interrupt vector (FIVECTOR) indicates the vector index of highest priority pending fast interrupt.

A second set of registers provides indication of the highest level pending interrupt request:

- Normal Interrupt Vector (NIVECTOR) - this register indicates the vector number of highest priority pending normal interrupt.
- Fast Interrupt Vector (FIVECTOR) - this register indicates the vector number of highest priority pending fast interrupt.

All interrupt controller registers are accessible in both supervisor and user modes. Writes attempted to read-only registers are ignored. These registers should be written with 32-bit stores only.

The Interrupt Force Register (INTFRC) is provided for software generation of interrupts. By enabling interrupts for these bit positions, software can force an interrupt request for debugging hardware interrupt service routines by providing an alternate method of interrupt assertion.

The interrupt requests are prioritized in the following sequence:

1. Fast interrupt requests, in order of highest number
2. Normal interrupt requests, in order of highest number

If two normal interrupts are asserted at the same time, the normal interrupt with the highest source number is selected.

8.4.1 ITC Prioritization of Interrupt Sources

The ITC module sets the ultimate priority of any interrupt request source. Prioritization is determined at two levels:

- The CPU has two interrupt request inputs, i.e., IRQ and FIQ. The fast interrupt request FIQ always has priority over the normal interrupt request IRQ.
- Once the CPU responds to an IRQ or FIQ, the pending interrupt requests are prioritized by source number where higher source numbers have higher priority.

To restate this, the interrupt requests are prioritized in the following sequence:

1. Fast interrupt requests, in order of highest source number.
2. Normal interrupt requests, in order of highest source number.

Table 8-1 shows the priority of the interrupt sources.

Table 8-1. Interrupt Sources

Interrupt Number	Source Name
15(highest)	not used
14	not used
13	not used
12	not used
11	not used
10	SPI
9	ADC
8	SSI
7	MACA
6	SPIF
5	TMR
4	I2C
3	CRM
2	UART2
1	UART1
0 (lowest)	ASM

8.4.2 Assigning and Enabling Interrupt Sources

As stated in the chapter overview, interrupt requests can be enabled/disabled in the three basic functions, i.e., the CPU core, the ITC, and the peripheral function. At the system level to fully enable an interrupt request:

- The first step is to program the peripheral source(s) to generate interrupt requests
- The second step is to enable and assign interrupt requests in the ITC
- The final step is to enable the IRQ and or FIQ interrupt inputs in the core by clearing the normal interrupt disable (I) and/or the fast interrupt disable (F) bits in the program status register (CPSR).

Within the ITC, it first prioritizes the sources in [Table 8-1](#) via their bit position in the INTENABLE Register. The assigned bit position maps to the prioritization level, as an example, INTENABLE[2] enables the UART2 interrupt request).

Assigning and enabling of interrupt requests at the ITC is done at several levels:

1. An interrupt request from a given source must first be enabled via its assigned bit in the INTENABLE Register
 - The bits may be enabled or disabled directly by writing the INTENABLE Register.
 - One source at-a-time can be disabled via Interrupt Disable Number Register (INTDISNUM)
 - One source at-a-time can be enabled via Interrupt Enable Number Register (INTENNUM)

2. The enabled source must then be mapped to the IRQ or FIQ space via its assigned bit in the INTTYPE Register
3. Exclusive to the IRQ, all interrupt sources below a given level can be disabled via the Normal Interrupt Mask Register (NIMASK).
4. Finally, either or both the IRQ and FIQ can be enabled or disabled via the Interrupt Control Register (INTCNTL)

8.5 Interrupt Controller Memory Map

Table 8-2 provides an overview of the ITC register memory map.

- The ITC base address is 0x8002_0000.
- Register writes to ITC offset addresses above the FIPEND Register address (0x8002_0040 - 0x8002_01FF) are ignored.
- Register reads from write-only registers and ITC addresses above the FIPEND Register offset address read as all 0's.

Table 8-2. ITC Register Memory Map

Offset	[31:24]	[23:16]	[15:8]	[7:0]	Access Type	Access Width
+ 0x00	Interrupt Control Register (INTCNTL)				R/W	32-bit only write/ any read
+ 0x04	Normal Interrupt Mask Register (NIMASK)				R/W	32-bit only write/ any read
+ 0x08	Interrupt Enable Number Register (INTENUM)				W	32-bit only write
+ 0x0C	Interrupt Disable Number Register (INTDISNUM)				W	32-bit only write
+ 0x10	Interrupt Enable Register (INTENABLE)				R/W	32-bit only write/ any read
+ 0x14	Interrupt Type Register (INTTYPE)				R/W	32-bit only write/ any read
+ 0x18 to +0x24	Reserved				-	-
+ 0x28	Normal Interrupt Vector (NIVECTOR)				R	Any read
+ 0x2C	Fast Interrupt Vector (FIVECTOR)				R	Any read
+ 0x30	Interrupt Source Register (INTSRC)				R	Any read
+ 0x34	Interrupt Force Register (INTFRC)				R/W	32-bit only write/ any read
+ 0x38	Normal Interrupt Pending Register (NIPEND)				R	Any read
+ 0x3C	Fast Interrupt Pending Register (FIPEND)				R	Any read

8.6 ITC Registers

The following sections provide descriptions of the ITC registers.

8.6.1 Interrupt Control Register (INTCNTL)

The Interrupt Control Register (INTCNTL) enables/disables the interrupt arbiters. This register should be updated with 32-bit stores only.

INTCNTL			Interrupt Control Register										Addr \$8002_0000			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
												NIAD	FIAD			
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-3. INTCNTL Bit Descriptions

Bit Number	Description	Operation
Bits 31-21	Reserved	
Bit 20	<p>NIAD — Normal Interrupt Arbiter Disable. This bit, when asserted, prevents (or masks) the assertion of bus request to the core when the normal interrupt signal (nIRQ) is asserted. If an alternate master has ownership of the bus when a normal interrupt occurs, the bus is given back to the processor core <u>after</u> the DMA device has completed its accesses. Therefore, the IRQ_DIS bit does not affect alternate master accesses that are in progress.</p> <p>In order to prevent an alternate master from accessing the bus during an interrupt service routine, the interrupt flag should not be cleared until the end of the service routine.</p>	<p>0 = Allow normal (IRQ) interrupt request to CPU 1 = Mask or disable assertion of IRQ to CPU</p>

Table 8-3. INTCNTL Bit Descriptions (continued)

Bit Number	Description	Operation
Bit 19	<p>FIAD — Fast Interrupt Arbiter Disable. This bit, when asserted, prevents the assertion of bus request to the core when the fast interrupt signal (nFIQ) is asserted. If an alternate master has ownership of the bus when a fast interrupt occurs, the bus is given back to the processor core <u>after</u> the DMA device has completed its accesses. Therefore, the IRQ_DIS bit does not affect alternate master accesses that are in progress.</p> <p>In order to prevent an alternate master from accessing the bus during an interrupt service routine, the interrupt flag should not be cleared until the end of the service routine.</p>	<p>0 = Allow fast (FIQ) interrupt request to CPU 1 = Mask or disable assertion of FIQ to CPU</p>
Bits 18-0	Reserved	

8.6.2 Normal Interrupt Mask Register (NIMASK)

The Normal Interrupt Mask Register (NIMASK) controls the normal interrupt mask level. All normal interrupts (IRQ) with a priority lower than or equal to the NIMASK are disabled. The priority of normal interrupts are determined by their assigned interrupt number (see [Section Table 8-1., “Interrupt Sources”](#)). The reset state of this register does not disable any normal interrupts. This register should be updated with 32-bit stores only.

NOTE

Writing a value of 0x10 or greater to NIMASK[4:0] does not disable any normal interrupt.

NIMASK			Normal Interrupt Mask Register										Addr \$8002_0004			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
												NIMASK[4:0]				
TYPE	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Table 8-4. NIMASK Bit Descriptions

Bit Number	Description	Operation
Bits 31-3	Reserved	
Bits 4-0	NIMASK[4:0] — Normal Interrupt Mask. Controls normal interrupt mask level. All normal interrupts of priority level lower than or equal to the NIMASK are disabled.	0 = Disable normal interrupt 0 1 = Disable normal interrupt 1 and lower 2 = Disable normal interrupt 2 and lower ... 11 - 31 = Do not disable any normal interrupts

8.6.3 Interrupt Enable Number Register (INTENNUM)

The Interrupt Enable Number Register (INTENNUM) provides a hardware accelerated means of enabling of interrupt requests. Any write to the ENNUM[3:0] field enables one interrupt source. If the field equals 4b0000, then interrupt source 0 is enabled, etc. Multiple writes to this register enable a corresponding number of interrupts. This hardware mechanism alleviates the need for an atomic read/modify/write sequence to enable an interrupt source.

This register should be updated with 32-bit stores only. This register always reads back all 0s.

INTENNUM				Interrupt Enable Number Register									Addr \$8002_0008			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													ENNUM[3:0]			
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	sfclr	sfclr	sfclr	sfclr
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-5. INTENNUM Bit Descriptions

Bit Number	Description	Operation
Bits 31-4	Reserved	
Bits 3-0	ENNUM[3:0] — Interrupt Enable Number. Writing to this field enables the interrupt source associated with this value.	0 = Enable interrupt source 0 1 = Enable interrupt source 1 ... 10 = Enable interrupt source 10 11 - 15 = Unused

8.6.4 Interrupt Disable Number Register (INTDISNUM)

The Interrupt Disable Number Register (INTDISNUM) provides a hardware accelerated means of disabling of interrupt requests. Any write to this register disables one interrupt source. If the field equals

4b0000, then interrupt source 0 is disabled, etc. Multiple writes to this register disable a corresponding number of interrupts. This hardware mechanism alleviates the need for an atomic read/modify/write sequence to disable an interrupt source.

This register should be updated with 32-bit stores only. This register always reads back all 0s.

INTDISNUM				Interrupt Disable Number Register									Addr \$8002_000C			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													DISNUM[3:0]			
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	slfclr	slfclr	slfclr	slfclr
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-6. INTDISNUM Bit Descriptions

Bit Number	Description	Operation
Bits 31-4	Reserved	
Bits 3-0	DISNUM[3:0] — Interrupt Disable Number. Writing to this field disables the interrupt source associated with this value.	0 = Disable interrupt source 0 1 = Disable interrupt source 1 ... 10 = Disable interrupt source 10 11 - 15 = Unused

8.6.5 Interrupt Enable Register (INTENABLE)

The Interrupt Enable Register (INTENABLE) is used to enable pending interrupt requests to the CPU core. Bits in this register correspond to an interrupt source available in the system (Sources 0 through 10 are used). The reset state of this register is all interrupts masked.

NOTE

- The interrupts enabled in this register can be modified by writing directly to the INTENABLE Register, or by writing the INTENNUM Register to set bits, or by writing the INTDISNUM Register to clear bits.
- See [Table 8-1](#) for mapping of interrupt enable bits to interrupt sources.

This register should be updated with 32-bit stores only.

INTENABLE				Interrupt Enable Register									Addr \$8002_0010			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	INTENABLE[15:0]															
TYPE	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-7. INTENABLE Bit Descriptions

Bit Number	Description	Operation
Bits 31-16	Reserved	
Bits 15-0	<p>INTENABLE[15:0] — Interrupt Enable. These bits enable the corresponding interrupt source to request either a normal interrupt or a fast interrupt.</p> <ul style="list-style-type: none"> • A reset operation clears these bits. • If an interrupt source is enabled, it is mapped to the IRQ or FIQ type request depending on the associated INTTYPE setting. • Only Bits 10 - 0 map to a valid interrupt source 	<p>0 = Interrupt disabled 1 = Interrupt enabled</p>

8.6.6 Interrupt Type Register (INTTYPE)

The Interrupt Type Register (INTTYPE) is used to select whether a pending interrupt source, when enabled by the INTENABLE register, creates a normal interrupt request (IRQ) or a fast interrupt request (FIQ) to the CPU core. Bits in this register correspond to an interrupt source available in the system (Sources 0 through 10 are used). The reset state of this register is that all interrupts generate a normal interrupt.

INTTYPE				Interrupt Type Register									Addr \$8002_0014			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	INTTYPE[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-8. INTTYPE Bit Descriptions

Bit Number	Description	Operation
Bits 31-16	Reserved	
Bits 15-0	<p>INTTYPE[15:0] — Interrupt Type. These bits control whether the corresponding interrupt source requests a normal interrupt or a fast interrupt.</p> <ul style="list-style-type: none"> • A reset operation clears these bits. • The corresponding INTENABLE bit must be asserted to enable the interrupt request • Only Bits 10 - 0 map to a valid interrupt source 	<p>0 = Interrupt source generates a normal interrupt (nIRQ)</p> <p>1 = Interrupt source generates a fast interrupt (nFIQ)</p>

8.6.7 Normal Interrupt Vector (NIVECTOR)

The Normal Interrupt Vector Register (NIVECTOR) provides the highest pending normal interrupt number. This number can be directly used as an index into a vector table to select the highest pending normal interrupt source. This register value is derived directly from the normal interrupt pending register (NIPEND).

NIVECTOR				Normal Interrupt Vector								Addr \$8002_0028				
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													NIVECTOR[3:0]			
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 8-9. NIVECTOR Bit Descriptions

Bit Number	Description	Operation
Bits 31-4	Reserved	
Bits 3-0	NIVECTOR[3:0] — Normal Interrupt Vector. Indicates number for the highest pending normal interrupt.	0 - 10 = Highest pending normal interrupt 11 - 15 = Unused (invalid)

8.6.8 Fast Interrupt Vector (FIVECTOR)

The Fast Interrupt Vector Register (FIVECTOR) provides the highest pending fast interrupt vector number. This number can be directly used as an index into a vector table to select the highest pending fast interrupt source. This register value is derived directly from the fast interrupt pending register (FIPEND).

FIVECTOR				Fast Interrupt Vector									Addr \$8002_002C			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													FIVECTOR[3:0]			
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 8-10. FIVECTOR Bit Descriptions

Bit Number	Description	Operation
Bits 31-4	Reserved	
FIVECTOR Bits 31-0	FIVECTOR[3:0] — Fast Interrupt Vector. Indicates number for the highest pending fast interrupt.	0 - 10 = Highest pending fast interrupt 11 - 15 = Unused (invalid)

8.6.9 Interrupt Source Register (INTSRC)

The Interrupt Source Register (INTSRC) reflects the status of all interrupt request inputs into the interrupt controller. Unused bit positions always read zero (no request pending). The state of this register out of reset is determined by the peripheral circuits generating the requests; normally, the requests would be inactive.

INTSRC				Interrupt Source Register									Addr \$8002_0030			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	INTSRC[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-11. INTSRC Bit Descriptions

Bit Number	Description	Operation
Bits 31-16	Reserved	
Bits 15-0	INTSRC[15:0] — Interrupt Source. Indicates the state of the corresponding hardware interrupt source. <ul style="list-style-type: none"> • Only Bits 10 - 0 map to a valid interrupt source • Bits 15 - 11 are not used 	0 = Interrupt source unasserted 1 = Interrupt source asserted

8.6.10 Interrupt Force Register (INTFRC)

The Interrupt Force Register (INTFRC) allows for software generation of interrupts for each of the possible interrupt sources for functional or debug purposes.

This register should be updated with 32-bit stores only.

INTFRC				Interrupt Force Register									Addr \$8002_0034			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
INTFRC[15:0]																
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-12. INTFRC Bit Descriptions

Bit Number	Description	Operation
Bits 31-16	Reserved	
Bits 15-0	INTFR[15:0] — Interrupt Source Force Request. Used to force a request for the corresponding interrupt source. <ul style="list-style-type: none"> • Only Bits 10 - 0 map to a valid interrupt source • Bits 15 - 11 are not used 	0 = Standard interrupt operation 1 = Interrupt force asserted

8.6.11 Normal Interrupt Pending Register (NIPEND)

The Normal Interrupt Pending Register (NIPEND) indicates all pending normal (IRQ) interrupt requests. This register status is determined by the Interrupt Enable Register (INTENABLE), Interrupt Type Register (INTTYPE), and the Interrupt Source Register (INTSRC).

NIPEND				Normal Interrupt Pending Register									Addr \$8002_0038			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPEND[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-13. NIPEND Bit Descriptions

Bit Number	Description	Operation
Bits 31-16	Reserved	
Bits 15-0	<p>NIPEND[15:0] — Normal Interrupt Pending Field. Indicates all pending normal interrupt requests to the CPU core. For an active pending normal request:</p> <ul style="list-style-type: none"> • A corresponding normal interrupt enable bit must be set • A corresponding IRQ must be active • Only Bits 10 - 0 map to a valid interrupt source • Bits 15 - 11 are not used 	<p>0 = No normal interrupt request 1 = Normal interrupt request pending</p>

8.6.12 Fast Interrupt Pending Register (FIPEND)

The Fast Interrupt Pending Register (FIPEND) indicates all pending fast (FIQ) interrupt requests. This register status is determined by the Interrupt Enable Register (INTENABLE), Interrupt Type Register (INTTYPE), and the Interrupt Source Register (INTSRC).

FIPEND				Fast Interrupt Pending Register									Addr \$8002_003C			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	FIPEND[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-14. FIPEND Bit Descriptions

Bit Number	Description	Operation
Bits 31-16	Reserved	
FIPEND Bits 15-0	FIPEND[15:0] — Fast Interrupt Pending Field. Indicates all pending fast interrupt requests to the CPU core. For an active pending fast request: <ul style="list-style-type: none"> • A corresponding fast interrupt enable must be set • A corresponding IRQ must be active • Only Bits 10 - 0 map to a valid interrupt source • Bits 15 - 11 are not used 	0 = No fast interrupt request 1 = Fast interrupt request pending

Chapter 9

MAC Accelerator (MACA)

9.1 Overview

This chapter describes the low-level MAC and PHY link controller, which together with software running on the ARM core, implements the baseband protocols and other low-level link routines for media access and link control.

The MACA provides the following:

- The interface between the network layer and the RF modem
- A reduction in MCU load
- Contains embedded features that control parts of the IEEE 802.15.4 PHY and MAC layers
- Provides the protocol and control mechanisms required for channel access
- Builds the transmit packets
- Parses the received packets
- Handles acknowledgements and TxPoll sequences independent of the ARM processor

9.2 Features

The following is a partial list of 802.15.4 Standard network features:

- Over-the-air data rates of 250 KB/s
- Star or peer-to-peer topology
- Allocated 16 bit short or 64 bit extended addresses
- Allocation of Guaranteed Time Slots (GTSs)
- Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) channel access (slotted and non-slotted modes)
- Fully acknowledged protocol for transfer reliability
- Low power consumption
- Energy detection (ED)
- Link quality indication (LQI)
- 16 channels in the 2.450 GHz band

The following is a list of MACA core features:

- Sequence manager for auto sequences:
 - Automatic acknowledgment frame reception on transmitted packets
 - Automatic acknowledgment frame transmission on received packets

- Auto-Rx for continuous reception as coordinator
- Auto sequence for transmitted MAC data.request
- Assist for efficient response to MAC data.requests
- Embedded channel assessment in sequence
- Support for sequences with slotted CSMA-CA mode access
- Timer triggered or immediately executed actions
 - Actions can be triggered on either native or relative clock
 - Relative clock is based on native (free-running) clock with an added offset
- Support for extended Rx for reception in random backoff and battery life extension
- Support for promiscuous mode
- Shared memory between ARM core and MACA for Tx/Rx data using a dedicated DMA
- Packet manager
 - Handles preamble data
 - Handles frame check sequence (CRC)
 - Embedded header filter for received packets
- Beacon Support Mode
- Controlled by CPU accessible registers

9.3 Primary Functionality

As already stated, to help reduce MCU loading, the MACA constructs the transmit packets, parses the received packets, and handles acknowledgements and TxPoll sequences independent of the ARM processor.

For packet transmit, the MACA constructs the entire packet which includes the preamble, Start of Frame Delimiter (SFD) and the Frame Check Sequence (FCS). During receive, the modem recognizes the preamble and SFD, provides a pulse to mark the first bit of frame length, starts receiving the packet with the first bit of frame length and checks the FCS. [Figure 9-1](#) shows a diagram of an 802.15.4 packet.

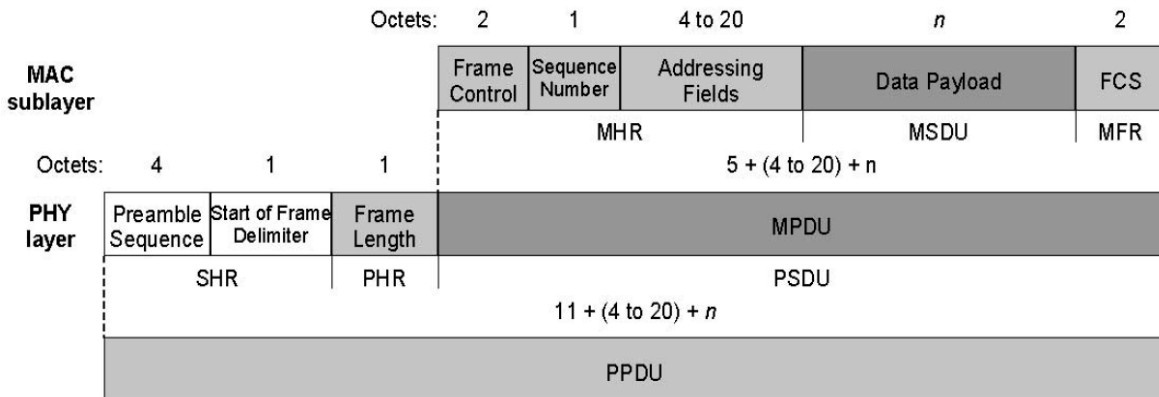


Figure 9-1. 802.15.4 Packet Diagram

The MACA supports slotted and non-slotted CSMA-CA modes. Slotted and non-slotted CSMA-CA use a basic time unit called a Backoff Period (BP). A BP unit is equal to 20 Symbols.

Slotted CSMA-CA For use in beacon-enabled networks, slotted CSMA-CA channel access divides time into slots. The slots are aligned with the start of beacon transmission. Each time a device needs to transmit data frames during the contention access period (CAP), it locates the boundary of the next slot and then waits for a random number of slots. If the channel is busy (following the BP) the device waits for another random number of slots before trying to access the channel again. If the channel is idle, the device begins transmission on the next available slot boundary.

Non-slotted CSMA-CA For use in non-beaconed networks, non-slotted CSMA-CA is random. Every time a device needs to transmit data frames or MAC commands, it waits for a random period of time. If the channel is idle (following the BP) it transmits the data. If the channel is busy (following the BP) the device waits for another random time period before accessing the channel again. Acknowledgment frames are sent without using any CSMA-CA method.

For more information on slotted and non-slotted CSMA-CA modes, refer to the 802.15.4 Standard.

9.4 Block Diagram

Figure 9-2 shows a functional overview of the MACA and its interfaces.

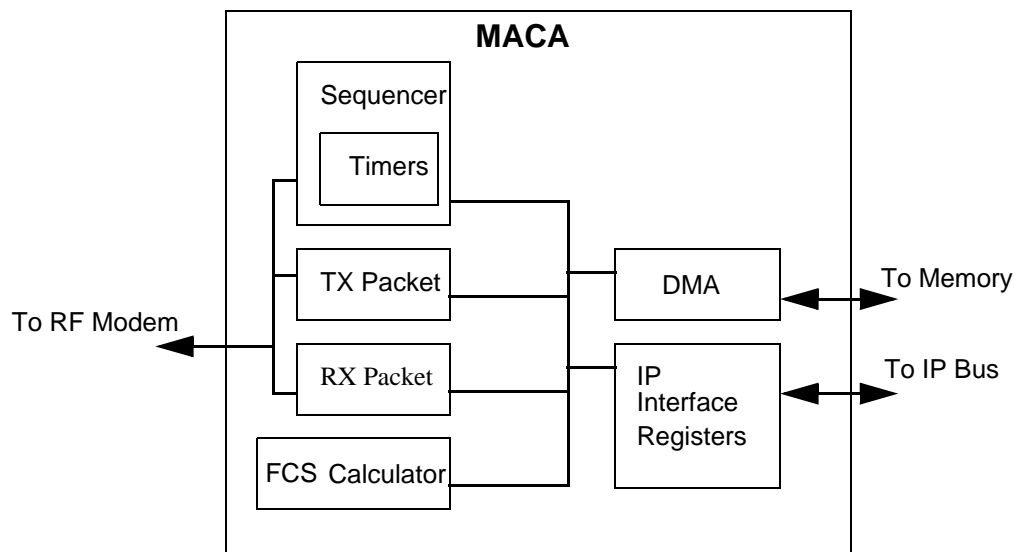


Figure 9-2. MACA

The following list provides a brief description of each MACA module and its functionality:

- **Sequencer** — The internal state machines are responsible for controlling user requested auto-sequences. See [Section 9.5.1, “Sequencer”](#) for more information about sequences. See subsections for information about the following sequencer details:
 - Timer triggered start of actions ([Section 9.5.1.1, “Timer Triggered Start Time”](#))

- Transmission sequence, [Section 9.5.1.2, “Transmission Sequence”](#), as well as the following items:
 - Automatic acknowledgment frame reception
 - Poll sequence handling
- Receive sequence, [Section 9.5.1.3, “Receiving Sequence”](#), as well as the following items:
 - Receive time-out
 - Extended Rx
 - Transmission while receiver active
 - Handling received MAC data request (POLL)
 - Generation of acknowledgement frame
- Clear Channel Assessment (CCA)
- Energy Detection (ED)
- Clock generation
- Interrupt generation
- Packet generation
- Packet receiver
- FCS handler
- The Dedicated Direct Memory Access (DDMA) transfers data between memory and the MACA buffers ([Section 9.5.2, “Dedicated Direct Memory Access \(DDMA\)”](#))
- The Random generator is a 32-bit random number generator which is also usable for random number generation in security applications ([Section 9.5.3, “Random Generator”](#))
- The Radio/modem control is an interface to the modem ([Section 9.5.4, “Radio Modem Control”](#))
- Beacon mode support ([Section 9.5.5, “Beacon Mode Support”](#))

9.5 Module descriptions

The following sections provide more details about the MACA core features and their functionality. All modules are controlled through memory mapped registers. See [Section 9.6, “MACA Register Memory Map”](#) for detailed register descriptions.

9.5.1 Sequencer

The sequencer handles auto-sequences which are a set of actions executed with respect to timing requirements defined by the ARM specification. Individual independent actions are:

- Clear Channel Assessment (CCA)
- Transmit
- Receive
- TxAck
- RxAck
- TxPoll

By combining these actions into longer autonomous auto-sequences, the MCU receives fewer interrupts, and the sequence internal timing requirements are managed by the hardware. Sequences are controlled by the MACA_CONTROL register.

NOTE

Writing to the control register should only be done when a sequence is fully completed because a new action terminates any currently running sequence. Aborting a sequence generates an ActionComplete_irq with an “aborted” status.

When a sequence completes (finished, aborted, or failed) it generates an ActionComplete_irq. Additional information about why the sequence was completed is located in the MACA_STATUS register COMPLETE_CODE field which contains the most recent status code.

The sequencer provides the following auto-sequences and actions:

- Energy Detect (ED) — See [Section 9.5.1.5, “Energy Detection \(ED\) Block”](#)
- TxSequence — [CCA]-[CCA]-Tx-[RxAck]. See [Section 9.5.1.2, “Transmission Sequence”](#)
- TxPollSequence — [CCA]-[CCA]-Tx(MAC-Data.request)-RxAck-[RxData]¹-[TxAck]. See [Section 9.5.1.2.2, “Poll Sequence”](#)
- RxSequence — Rx-[TxAck]-[Rx]. See [Section 9.5.1.3, “Receiving Sequence”](#)
- RxSequence(MAC-Data.request) — Rx-TxAck-TxData-[Rx]. This is an alternative sequence if the MAC data request is received, see [Section 9.5.1.3.5, “Handling Received MAC Data Request”](#)
- Wait — This is for no operation, but it generates an interrupt on time-out. See [Section 9.5.1.6, “Wait \(Timer Controlled NOP\)”](#).

9.5.1.1 Timer Triggered Start Time

All sequences can be started immediately or they can be timer triggered. If an action is timer triggered, the start clock must be written before the control register is written. Timer triggered actions are a one-shot action only. That is, once the action is started (or has been aborted), the timer comparator circuit is disabled. When the action starts, an ActionStarted_irq is generated.

9.5.1.1.1 Late Start

In the event that a starting time occurs in the past, this late start must be detected by calculating the time to action start. Because the timers are 32 bits in length, they exceed the 802.15.4 Standard requirements of at least 24-bit timers running at symbol clock (62.5 kHz). This allows detection for actions occurring in the past. If time to action is more than 31 bits (negative), then it is assumed to have occurred in the past, and the action is terminated immediately. In this case, an ActionComplete_irq is generated, and a “Late Start” reason code is placed in the status register. See [Section 9.7.5, “MACA_STATUS”](#) for further details about complete codes, and [Section 9.5.1.9, “Interrupt Generation”](#) for more information about interrupts.

Detection is performed immediately upon writing to the start clock register (MACA_STARTCLK). This detection mechanism is also applied to MACA_CPLTIM and MACA_SFTTIM registers.

¹. Receiver is only started if the received ACK indicates data pending.

A special instance of “Late Start” is when a timer-triggered action is started by writing to MACA_CONTROL register but no active timer is running. In this case, the action is aborted immediately with the generation of a ActionComplete_irq, with the “Late Start” status.

9.5.1.2 Transmission Sequence

When a transmission sequence starts, it can be initiated with no CCA or slotted/non-slotted CSMA-CA CCA. These are controlled by the MACA_CONTROL register. If CCA is required, the sequence terminates if the channel is detected as busy. A termination is sent to the software by generating an “ActionComplete_irq” interrupt, and it sets the result code to “Channel Busy” in the MACA_STATUS register.

The timing between CCA sequences and transmission start is fixed for both slotted CSMA-CA and non-slotted CSMA-CA CCA mode.

For transmissions in slotted CSMA-CA mode, the CCA must begin on an offset boundary. An offset boundary is defined as the beginning of a beacon, and then every 20 symbols. Transmit actions are started using a timer trigger, and the software is responsible for correctly aligning the sequence.

If the transmission sequence succeeds, it completes by generating an “ActionComplete_irq” interrupt, and the “Success” status. If a requested acknowledgment frame is not returned, the sequence completes with the “No Ack” status. Retransmission and backoff time calculation are the responsibility of the software.

Data is transferred from memory to the packet generator using DDMA, see Section 9.5.2, “Dedicated Direct Memory Access (DDMA)”.

The internal timing for aligning CCA back-to-back is controlled by the CCADELAY bit, and the delay between CCA and transmit (in both beacon and non-beacon mode) is controlled by the TXDELAY bit. Both bit settings are located in the MACA_CCADELAY register. The delay between the transmitted packet and the opening of the receiver for acknowledgement is controlled by the MACA_RXACKDELAY register. This register also contains the receiver window length.

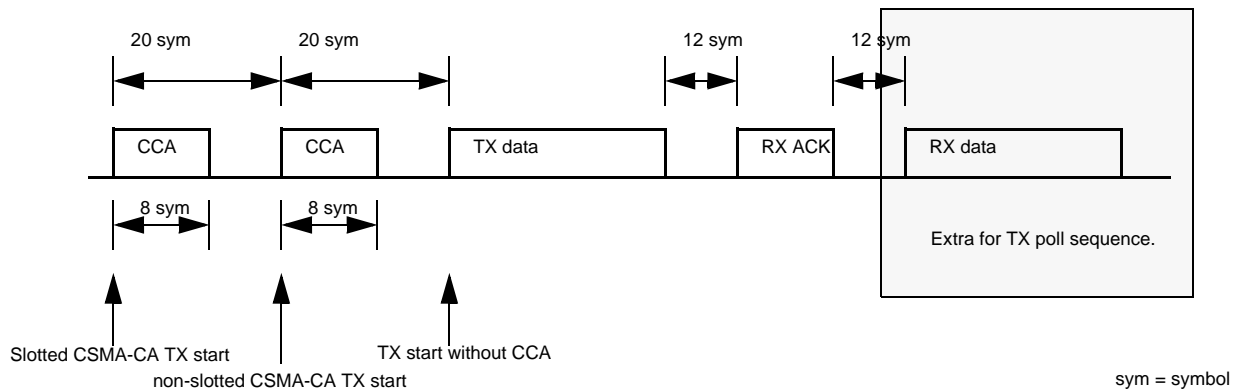


Figure 9-3. Transmit Transmission Timeline

9.5.1.2.1 Automatic Acknowledgement Reception

Upon completion of data transmission, the sequencer may be instructed to receive an acknowledgement frame. An acknowledgment frame can be expected to arrive 11 symbols from the last bit transmitted.

NOTE

The turnaround time is approximately 12 symbols so an accuracy of +/- one symbol is implied.

The acknowledgment frame includes a sequence number which is matched against the sequence number in the transmitted packet. If the sequence number does not match, then the acknowledgment frame is discarded. The packet receiver is responsible for extracting the sequence number. The transmitted sequence number is mirrored in the TXSEQNR register.

The data received in the acknowledgement frame should not be transferred to memory using DDMA but only used internally in the sequencer.

NOTE

The sequence number and acknowledgement request is embedded in the TxData, but instead of having hardware parse this information, it is supplied through the MACA_TXLEN and MACA_TXSEQNR registers.

9.5.1.2.2 Poll Sequence

An extension of the transmission sequence supports autonomous handling of the MAC Data.request, which is used for the device to retrieve data from the coordinator as for example, when polling the coordinator. This sequence is selected by setting the “POLL” bit in the MACA_CONTROL register.

NOTE

This sequence must be set to transmit.

In the acknowledgement frame that is received as the response to the Data.request, the “Frame pending” bit inside the frame control field indicates that the coordinator has data available and that the device should open its receiver. If the “frame pending” bit is set, the sequencer prepares for data reception within 11 symbols from receiving the acknowledgment frame. The sequencer opens the receiver for data reception. In this case, a receive time-out must be specified and it is the responsibility of the software to calculate the time-out. If data is received, a DataIndication_irq interrupt is issued, the complete clock is cancelled, and the poll sequence is complete by generating an ActionComplete_irq and setting the status to “success”.

If no data is received before the time-out, the sequence completes by generating an ActionComplete_irq and setting the status to “time-out”.

9.5.1.2.3 Random Backoff Before Transmission

Prior to transmissions using CCA, a random backoff is performed. It is the responsibility of the software to calculate and implement this period by setting up a timer triggered action.

9.5.1.3 Receiving Sequence

Receive operations may be initiated using timer triggered start or they can be started immediately after writing to the control register. Once the receiver is started, the sequencer waits for sync-detection from the RF modem and then starts to receive data into memory using the DDMA. The first byte transferred starts with the frame control. The last byte of data transferred is the last byte of the payload. The frame length can be read from the GET_RX_LVL register.

The receiver only transfers data to memory once for every time a pointer is written to the DDMA pointer register. Writing a new pointer implies buffer availability. When a packet is received, the buffer is used and a new buffer must be provided (through a new write to the pointer register). If a sync is detected, but no buffer is available, the remainder of the packet is handled as a failed packet. See [Section 9.7.20, “MACA_DMARX”](#) for more information about the MACA_DMARX pointer register.

Once data is fully received and the FCS is valid, the sequencer evaluates the frame control field. If the acknowledgement request bit is set, the auto-sequence continues by transmitting an acknowledgement frame.

Whenever data is fully received (and acknowledged if required) a DataIndication_irq interrupt is issued. If the sequencer finishes at this point, an ActionComplete_irq is generated with a “success” status. A data indication interrupt is guaranteed to occur before (or at the same time as) a complete interrupt.

Upon completion of the receive sequence, the receiver is restarted immediately if the “AUTO” bit is set in the control word. Otherwise, the sequence completes with a “Success” status (the packet was successfully received), and any time outs are aborted. This means that if the filter or checksum fails, the receive sequence is always restarted regardless of the state of “AUTO”.

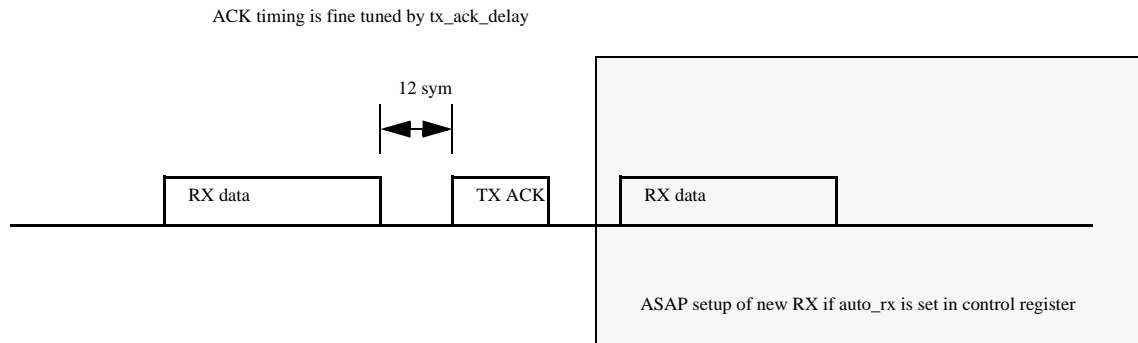


Figure 9-4. Receive Operation Timeline with Optional Receiver Restart

9.5.1.3.1 Receive Time-out

When the receive time-out occurs, the receive sequence is terminated immediately, even though a sync was found and a reception was in progress. In this case, the sequence is completed, and an interrupt ActionComplete_irq is generated. The complete status is set to “time-out”. Time-out is controlled by the MACA_CPLTIM register.

9.5.1.3.2 Soft Time-out

An alternative to the hard time-out provided by the MACA_CPLTIM register, is a less harsh receive time-out referred to as a soft time-out. The soft time-out does not terminate a receive sequence if a packet is being received (for example, when an SFD is found and end of packet not reached). Instead, the time-out is postponed and generated at the end of the packet after filter rejects, checksum fails, or packet successfully received. The radio is also returned to Idle mode and an ActionComplete_irq is generated, with the “SftPndTimeout” status. If the time-out triggers and is not being postponed, the action completes with ActionComplete_irq and the “SftTimeout” status.

The soft-time-out is programmed using the MACA_SFTTIM register.

9.5.1.3.3 Battery Life Extension

Battery life extension is implemented using a soft-time-out, and with the “AUTO” bit disabled in MACA_CONTROL register. This allows the receiver to be open for a given time bounded by MACA_SFTTIM and MACA_CPLTIM register settings. This setting can provide the following scenarios:

- No data received — ActionComplete_irq generated at the soft-time-out, with “SftTimeout” status. Any complete timer is aborted.
- Successful data before soft time-out — (DataIndication_irq), ActionComplete_irq with “success” status, because the receiver is not restarted. Any complete timer is aborted.
- Successful data while soft time-out — (DataIndication_irq if data Ok), ActionComplete_irq with “SftPndTimeout” status. Any complete timer is aborted.
- If the “hard” time-out (given by the MACA_CPLTIM register) is reached in any scenario, the receive is aborted with the ActionComplete_irq with “time-out” status. Any complete timer is aborted.

9.5.1.3.4 Transmission While Receiver Active

As a coordinator, the receiver is usually enabled all the time to allow devices to transmit at any time. But if the coordinator wants to initiate a direct data transfer, a random backoff must be calculated, and the receiver temporarily stopped. In order to gently stop the receiver at a calculated time, use the soft time-out process. During the random backoff period, the receiver is intentionally allowed to run until the backoff period ends. However, do not force the receiver off if a reception is in progress.

For this functionality to work, users can program and use the MACA_SFTTIM register while the receiver is active. This sets a “soft” time-out which behaves almost like battery life extension. However, instead of completing the action on the very first successfully received packet, the receiver is restarted. (If “AUTO” is set in the MACA_CONTROL register).

This means that this sequence does not complete until either a (pending) soft time-out or a hard time-out occurs. The following scenarios describe this behavior.

1. No data received — An ActionComplete_Irq is generated on the “soft” time-out (“SftTimeout” status). Any complete timer is aborted.

2. Successful data before soft time-out — (DataIndication_irq interrupts occur every time a successful packet is received). An ActionComplete_irq is generated on the “soft” time-out (“SftTimeout” status).
Any complete timer is aborted.
3. Successful data while soft time-out — An ActionComplete_irq is generated on end of packet (“SftPndTimeout” status) regardless of packet success (filter or checksum passed).
Any complete timer is aborted.
4. If the “hard” time-out (provides by the MACA_CPLTIM register) is reached in any scenario, the receiver is aborted, and an ActionComplete_irq is generated with the “time-out” status.
Any complete timer is aborted.

9.5.1.3.5 Handling Received MAC Data Request

A receive sequence special case is when a MAC data request is received. In this special case, the remote side expects a data response. There are two ways of responding with data:

- Responding immediately
- With random backoff and CCA before transmission

As shown in Figure 9-5, there are two methods of transmitting a data response:

Upper	Data may be transmitted immediately (12 symbols) after acknowledgement, and no CCA is needed.
Lower	Data may be transmitted later, but a CCA is required. In slotted CSMA-CA mode, the transmissions must be aligned to backoff boundaries.

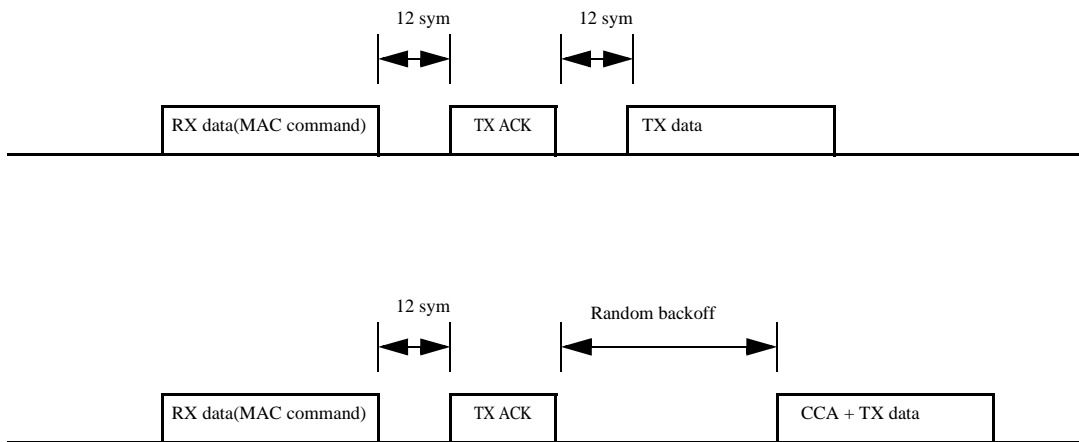
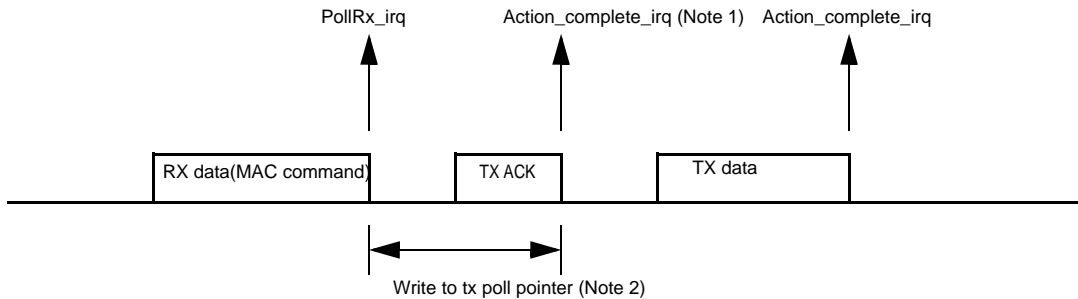


Figure 9-5. Data Response Methods

A faster response results in lower power consumption because the receiver (usually a battery powered device) would otherwise have to open its receiver for longer time periods).

In unslotted mode, a fast response must start transmission 12 symbols after the acknowledgment frame. This is the slotted CSMA-CA mode required transmission time to be aligned with the backoff boundary. See Section 9.5.5.1, “Slotted CSMA-CA Mode Timing”. This leaves very little time left for the software to determine what possible data to transfer and to setup the transmission. To assist accomplishing a fast response, a “PollRx_irq” interrupt is generated on the end of the MAC request (CRC valid, and

acknowledgment frame about to be transmitted), see [Figure 9-5](#). When the software sees this interrupt, the data response can be determined (assuming that there is data in indirect queue to the polling device). By writing the data pointer to the MACA_DMAPOLL register, the DDMA is armed for transmission data, and writing to this register also informs the sequence to continue with fast data response.



Note 1: Action_complte_irq will occur if maca_frmPend==0, or maca_frmPend updated after usage, or tx_poll pointer not updated before tx done.

Note 2: If tx poll pointer is written, Tx sequence will be started. Else, complete with "success" status.

Figure 9-6. Generating a Fast Response

If the MACA_DMAPOLL register is not written before the completion of the acknowledgment frame which indicates data pending, the sequencer is assumed to be a slow response, and does not start-up the transmitter. The auto-sequencer will then complete or restart the receiver (if AUTO enabled). In any case, a DataIndication_irq is generated. If the sequence completes, an ActionComplete_irq is generated.

NOTE

Ensure that the ActionComplete_irq does not occur before the DataIndication_irq.

If the MACA_DMAPOLL register is written after completion of the acknowledgment frame, the sequencer must immediately abort its sequence and issue a complete interrupt with the "Late start" status. For example, writing to the MACA_DMAPOLL register should only be done in the window between PollRx_irq generation and the completion of the following acknowledgement.

In slotted CSMA-CA mode, the acknowledgment frame must be transmitted on a backoff boundary (see [Section 9.5.5.1, "Slotted CSMA-CA Mode Timing"](#)). This extends the time available for setting up a fast response, but the procedure is the same as far as software is concerned.

It is possible to update the MACA_FRMPND register up until the time when the status is used to generate the acknowledgement response information. If the MACA_FRMPND register is zero (regardless if the MACA_DMAPOLL register is set up), the sequence terminates after the acknowledgement is transmitted.

If the MACA_FRMPND register is updated after the information is used in the acknowledgement, but before the completion of the transmitted acknowledgement, the MACA_FRMPND register is not updated, and the sequence completes after acknowledgement transmission with the ActionComplete_irq, and the "Late Start" status. Software can then read the MACA_FRMPND register to know the pending-frame contents of the transmitted acknowledgement.

If the packet is protected by a security integrity check, the software is responsible for validation.

9.5.1.3.6 Generating the Acknowledgement Frame

Acknowledgement frames are generated internally and contain 3 bytes of data (excluding FCS). The data is a frame control field and a sequence number. The sequence number is copied from the just received data packet. When building the frame control field, the “frame pending” bit must correspond to the state of the MACA_FRMPND register.

In unslotted mode, acknowledgement frames are transmitted 12 symbols after reception of data frames that require acknowledgement, but in slotted CSMA-CA mode, the frame must be further delayed until a backoff boundary. See [Section 9.5.5.1, “Slotted CSMA-CA Mode Timing”](#) for further information about slotted CSMA-CA alignment.

The handling of minimum turnaround may be fine-tuned by using of the MACA_TXACKDELAY register.

9.5.1.4 Clear Channel Assessment (CCA)

CCA is handled by the RF modem. The action is setup by the MACA sequencer, and when the assessment completes, output from the RF modem is used to determine how the MACA auto sequence is continued.

A CCA has an outcome of either busy or clear. If the channel is busy, the “BUSY” bit in MACA_STATUS register is set. The follow-up section is controlled by the transmission sequence.

A CCA is always be used in conjunction with a transmission sequence.

9.5.1.5 Energy Detection (ED) Block

ED is handled by the RF modem. The action is setup by the MACA sequencer, and when the detection completes, the value is placed into the MACA_EDVALUE register. The ED function does not return a bus “busy” bit as it is only a measure of how much energy is present. See [Section 9.7.8, “MACA_EDVALUE”](#).

9.5.1.6 Wait (Timer Controlled NOP)

By programming the time-out register and afterwards writing a “NOP” action command to the control register, the MACA module will be idle until the time-out triggers. At this point, a CompleteAction_irq is generated with the “time-out” status.

9.5.1.7 Header Filtering Block

Received data must be filtered according to the 802.15.4 Standard. The filter may be disabled by enabling promiscuous mode (The PRM bit in the MACA_CONTROL register). If the filter is disabled, auto-acknowledgement is not enabled.

A packet is received if it meets the following criteria:

- The frame type subfield of the frame control field can not contain an illegal frame type
- If the frame type indicates that the frame is a beacon frame, the source PAN identifier must match *macPANId* unless *macPANId* is equal to 0 x ffff. In this case, the beacon frame must be accepted, regardless of the source PAN identifier

- If a destination PAN identifier is included in the frame, it must match the *macPANId* or it must be the broadcast PAN identifier (0 x ffff)
- If a short destination address is included in the frame, it must match either the *macShortAddress* or the broadcast address (0 x ffff). Otherwise, if an extended destination address is included in the frame, it must match the *aExtendedAddress*
- If only source addressing fields are included in the data or MAC command frame, the frame must be accepted only if the device is a PAN coordinator and the source PAN identifier matches the *macPANId*

If the filter determines that the currently received data is not destined for this device, the reception is stopped, and the receiver is restarted, otherwise the packet is received until its end.

The *macPANId* is extracted from MACA_MACPANID register and the *macShortAddress* is found in the MACA_MAC16ADDR register. The extended address is found by combining the MACA_MAC64HI and MACA_MAC64LO registers.

To determine the current mode of operation, the control register bit (ROLE) is set when the device is a PAN coordinator.

The usage of the short or extended address is based on the received frame control field. The packet address information is extracted in the packet receiver. See [Section 9.5.1.11, “Packet Receiver”](#).

An additional check is applied to the packet length as a sanity check:

- Length < 2 — Reject
- Length < 5 — Reject if not promiscuous mode

If the receiver filter fails for any reason, or the CRC fails, then the receiver is restarted.

9.5.1.7.1 Buffer Starvation Prevention

An extension to the header filter prevents buffer starvation which occurs on routers where incoming data is received (and acknowledged) using the last available buffer. This means that any further incoming packets are discarded. In the worst case, the system can not empty the outgoing queue if that data needs to be POLLED out.

To avoid this scenario, the filter must reject all MAC command packets that are not Data.requests. (See [Section 9.5.1.3.5, “Handling Received MAC Data Request”](#)) for information about whether the POLL bit is set in the MACA_FLTREJ register.

9.5.1.7.2 Explicit Packet Type Rejection

As an option, an extension to the header filter rejects specific packet types. Packet types are found in the the first 3 bits of the frame control field. Frame types identified by CMD, ACK, DATA, and BCN in the MACA_FLTREJ register must be rejected.

9.5.1.7.3 Auto-Reception of Acknowledgement

If the sequencer is set up for automatic reception of the acknowledgment frame, then only these types of packets are allowed. Any other packet type is rejected. This is controlled internally in the MACA.

9.5.1.7.4 Ensure Frame Control Field Reserved Bits are Zero

To ensure that reserved bits in the frame control field are zero, the FC_MASK field in MACA_FLTRES register must be AND'ed with the received frame control field. If the outcome is non-zero, it indicates that some reserved bit(s) are set, and the packet is rejected. This approach allows explicit rejection/acceptance of 802.15.4b packets.

9.5.1.7.5 Detection of Data Request

A special extension of the filter is the detection of received data request packets. This requires the receive filter to check the payload data of MAC command frames and check for a match against the command frame identifier equal to 0x04. Integrity (security) bits in the packet are skipped and validated in software.

9.5.1.8 Clock Generation

All timers are 32-bit wide, and run at 250kHz. Triggers are armed by writing to the appropriate register (MACA_CPLCLK). Triggers are one-shot only and need to be re-armed once expired. Also, if timers are aborted, a re-arm is also needed.

In order to detect timer triggers occurring “in the past”, the MACA checks the difference between the current clock and the time until a trigger should occur. If the difference is larger than 31 bits (a negative result that corresponds to more than 2 hours into the future) it is assumed to be an error (or as already occurred) and an ActionComplete_irq is generated with the “Late start” status. After this, the sequencer is in the idle state.

If the complete clock is written to while armed and the sequencer is not idle, a new clock is set. For the remainder of the new time-out, the receiver operates in extended mode until completion of the sequence. A timer can be aborted by writing to the MACA_TMRDIS register.

In low power mode, the CRM module provides a low-power system timer and wakes up the MACA.

9.5.1.9 Interrupt Generation

The MACA uses interrupts as event creation to the software. All interrupt sources, except Reset_irq, are maskable. Interrupt sources are read from the MACA_IRQ register, and are handshaked by setting bits in MACA_CLRIRQ register. Using this handshake method only affects specified interrupt sources.

The software may also generate an interrupt by manually setting the interrupt source through writes to the MACA_SETIRQ register.

There are multiple interrupt sources in the MACA but they are all OR'ed into one interrupt request source from the module. The combined source is passed to the Interrupt Controller (ITC)

9.5.1.9.1 MACA Accelerator Interrupts

The following lists shows the individual interrupt sources inside the MACA:

- ActionComplete_irq
- PollRx_irq
- DataIndication_irq

- For debugging, the following interrupt sources are also present:
 - Timeout_irq
 - SyncFound_irq
 - ActionStarted_irq
 - CrcFail_irq
 - FilterFail_irq
 - RxLevel_irq

Figure 9-7 shows some example scenarios that use interrupts throughout a packet transmission and reception.

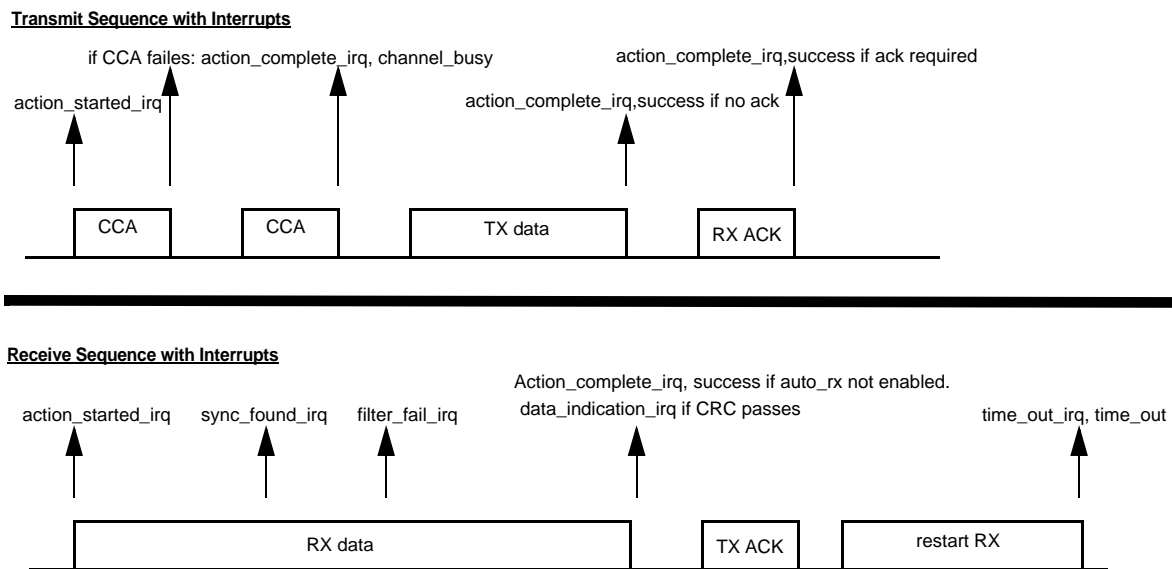


Figure 9-7. Transmit and Receive Sequences

9.5.1.9.2 ActionComplete_irq

The ActionComplete_irq is generated when a sequence action is completed. Sequence actions are started by writing an action to the control register. The sequencer then performs the required actions and moves to idle mode after completion. When generating this interrupt, the MACA_STATUS register must also be updated to reflect the complete reason code.

If the start timer trigger is armed and a trigger occurs before an action is written to the control register, an ActionComplete_irq is generated instead of an ActionStarted_irq. Every time this interrupt is generated, the sequencer moves to idle mode.

To ensure that the RF modem is idle before a new action is issued, this interrupt may be delayed by writing the time in symbols to MACA_EOFDELAY register. Any time outs are aborted and the complete clock MACA_CPLTIM register is updated.

9.5.1.9.3 PollRx_irq

The PollRx_irq is generated during an auto-sequence which has received a MAC data indication, and the FCS passes. See [Section 9.5.1.3.5, “Handling Received MAC Data Request”](#) for further description. The PollRx_irq is generated instead of a the DataIndication_irq.

9.5.1.9.4 DataIndication_irq

When data, MAC, or beacons are received during receive mode, and FCS and header filtering passes, the DataIndication_irq interrupt is generated when the packet is received.

If a receive action finished at this point, it is required that the DataIndication_irq is generated before (or at the same time as) an ActionComplete_irq.

9.5.1.9.5 Timeout_irq

The timeout_irq interrupt is set when a complete clock triggers.

9.5.1.9.6 SyncFound_irq

This interrupt is generated during receive mode, when a packet is detected and frame delimiter and length are received. The interrupt is triggered on the end boundary of the length field. At this point, the clock is also latched into the MACA_TIMESTAMP register.

9.5.1.9.7 ActionStarted_irq

This interrupt is generated when the start clock triggers and a pending action was written to the control register. If the clock triggers and no action is written to the control register (since setting up the start timer trigger), an ActionComplete_irq is generated instead.

9.5.1.9.8 CrcFail_irq

This interrupt is generated when the checksum fails.

9.5.1.9.9 FilterFail_irq

This interrupt is generated if the packet is rejected due to header filtering.

9.5.1.9.10 RxLevel_irq

This interrupts is generated when the buffer FIFO level matches or exceeds the value programmed by the MACA_SETRXLVL register.

9.5.1.10 Packet Generation

The packet generation module generates the packet data which is passed to the RF modem. This includes internal generation of the synchronization header and length field, and control of the DDMA for reading transmit data from memory. It also inserts the frame control sequence (FCS) in the end of the packet. For details about calculating the FCS, see [Section 9.5.1.12, “Frame Control Sequence \(FCS\) Handler”](#).

9.5.1.11 Packet Receiver

During receive mode, the packet receiver ensures that data is transferred into memory using the DDMA, and validates the frame control sequence. Also, the packet receiver interfaces to the header filtering block for validating packet destination, and extracts the header information from the packet, as well as the sequence number for acknowledgement reply/test.

9.5.1.12 Frame Control Sequence (FCS) Handler

The FCS Handler is a 16-bit CRC. Input is provided by either the packet generator or the packet receiver. The intermediate result is kept inside this module, and is read by the packet generator or receiver to apply or check the FCS.

If the checksum fails, the receiver is restarted.

9.5.1.13 Action finish clock

Whenever a complete interrupt (see [Section 9.5.1.9.2](#), “`ActionComplete_irq`”) is generated, the clock will be latched at this point too. This shall be right at the end of a transmitted or received packet. The value is placed in the `MACA_CPLTIM` register.

9.5.2 Dedicated Direct Memory Access (DDMA)

This module transfers data from memory to the packet generator or from the packet receiver to memory. Only the memory source/destination pointers are configurable.

DDMA transfers are controlled by the MACA packet receiver module or the MACA packet generator module, and will stall any access to the bus by other peripherals or the ARM core.

9.5.3 Random Generator

The 32 bit random number generator is used for backoff time calculation but can also be used for security purposes. The read returns a 32-bit random number while a write seeds the LFSR with an initial value. The LFSR is free running from the bus clock (13Mhz to 26Mhz). There are three ways the initial value can be set.

1. A hard system reset. (sets start value to 32'h1)
2. A soft reset from the register map. (sets start value to 32'h1)
3. A write to the register to “seed” the start value with the value written to the register.

The random generator has the following features.

- The LFSR used is a 32 bit primitive polynomial. It will repeat in 2^{32} clocks
- If somehow a group of “nodes” could be powered up and released from reset at precisely the same time, set with the same seed value and then, assuming that the frequency of the clock crystal of each node are offset by only 1ppm, the random number generator will be un-correlated in 0.078 seconds of operation

- The LFSR is running from the same clock as the ARM7. Even back-to-back reads return different values

9.5.4 Radio Modem Control

Interface to the RF modem is a high-level generation of signal starting actions as follows:

- Receive
 - Including fast restart of the receiver without a warm up or warm down period
- Transmit
- CCA
- ED
- Force Idle

Internal warm up timing in the sequences are not maintained by the accelerator module. Also, the interface must allow the software to change the frequency of operation. It is possible to place the RF modem into test-mode for evaluation. This requires the following modes:

- Continuous receive
- Continuous transmit with modulation
- Continuous transmit without modulation

The system must be able to enter these states from software. The RF modem returns a signal to indicate the current state (Rx, Tx, CCA, ED, Idle) to the MACA. Communication between RF modem and the MACA may be parallel or serial, depending on which implementation is most efficient.

9.5.4.1 Radio Test Modes

It is possible to enter continuous Rx, and Continuous Tx (with and without modulation). The control of these test modes is outside the scope of the MAC accelerator discussion.

9.5.5 Beacon Mode Support

Beacon handling is performed entirely in software and thus is completely responsible for timing in the system.

9.5.5.1 Slotted CSMA-CA Mode Timing

In slotted CSMA-CA mode, all transmissions are transmitted on a beacon offset boundary. The beacon defines the first boundary, and consecutive boundaries are placed at 20 symbol intervals.

Because a transmission is delayed by a warm-up period, it must be started at some point prior to the actual transmission of data. This is accomplished with the MACA_SLOTOFFSET register, which contains an initial counter value. This value is loaded when a beacon is received or transmitted. See [Section 9.7.18, “MACA_SLOTOFFSET”](#) for details about how to do this. This will align the counter to include any internal delays. The offset is used when the RSTO bit is set in the control register.

9.5.6 FIFO Buffer Interrupts

For mitigation purposes, it is possible to handle filtering in software. To accomplish this, a “stream” mode is supported.

It is also possible to set a receive level interrupt (after N bytes received) which generates a RxLevel_irq. The trigger is level triggered and is compared against the value provided by the MACA_SETRXLVL register. This interrupt allows the CPU to fetch the packet data from RAM before the packet has been completely received. In this way, software filtering can be accomplished in software as the packet is still being received.

The level may be set multiple times for an optimum software filtering of the header. In addition, the software can read the current number of received bytes using the MACA_GETRXLVL register.

9.6 MACA Register Memory Map

The MACA module is programmed through memory mapped registers. [Table 9-1](#) lists all registers in the MACA module and their relative address in memory.

NOTE

- Only 32-bit access is supported.
- The MACA base address is **0x8000_4000**

Table 9-1. MACA Memory Map

Address	Register name	Access
Base + 0x00	MACA Version Number (MACA_VERSION)	R
Base + 0x04	Reset (MACA_RESET)	R/W
Base + 0x08	Random number (MACA_RANDOM)	R/W
Base + 0x0C	Control (MACA_CONTROL)	W
Base + 0x10	Status (MACA_STATUS)	R
Base + 0x14	Frame Pending (MACA_FRMPND)	R/W
Base + 0x18	Frequency channel (MACA_FREQ)	R/W
Base + 0x1C	Energy Detect Result (MACA_EDVALUE)	R
Base + 0x40	Enable timers (MACA_TMREN)	R/W
Base + 0x44	Disable timers (MACA_TMRDIS)	R/W
Base + 0x48	Clock (MACA_CLK)	R/W
Base + 0x4C	Start clock (MACA_STARTCLK)	R/W
Base + 0x50	Complete clock (MACA_CPLCLK)	R/W
Base + 0x54	Soft timeoutClock (MACA_SFTCLK)	R/W
Base + 0x58	Clock offset (MACA_CLKOFFSET)	R/W
Base + 0x5C	Relative clock (MACA_RELCLK)	R/W
Base + 0x60	Action complete timestamp (MACA_CPLTIM)	R

Table 9-1. MACA Memory Map

Address	Register name	Access
Base + 0x64	Tx slot offset adjustment (MACA_SLOTOFFSET)	R/W
Base + 0x68	Receive time stamp (MACA_TIMESTAMP)	R
Base + 0x80	DMA Rx Data pointer (MACA_DMARX)	R/W
Base + 0x84	DMA Tx Data pointer (MACA_DMATX)	R/W
Base + 0x88	DMA Tx poll response pointer (MACA_DMAPOLL)	R/W
Base + 0x8C	Tx length (MACA_TXLEN)	R/W
Base + 0x90	Tx sequence number (MACA_TXSEQNR)	R/W
Base + 0x94	Set Rx Level Interrupt (MACA_SETRXLVL)	R/W
Base + 0x98	Read number of received bytes (MACA_GETRXLVL)	R
Base + 0xC0	Interrupt register (MACA_IRQ)	R
Base + 0xC4	Interrupt clear register (MACA_CLRIRQ)	W
Base + 0xC8	Interrupt set register (MACA_SETIRQ)	W
Base + 0xCC	Interrupt mask register (MACA_MASKIRQ)	R/W
Base + 0x100	MAC PAN ID (MACA_MACPANID)	R/W
Base + 0x104	MAC short address (MACA_MAC16ADDR)	R/W
Base + 0x108	High MAC extended address (MACA_MAC64HI)	R/W
Base + 0x10C	Low MAC extended address (MACA_MAC64LO)	R/W
Base + 0x110	Filter Rejection Mask (MACA_FLTREJ)	R/W
Base + 0x114	Clock divider(MACA_CLKDIV)	W
Base + 0x118	warmup(MACA_WARMUP)	R/W
Base + 0x11C	preamble(MACA_PREAMBLE)	R/W
Base + 0x120	lfsr whitening seed (MACA_WHITESEED)	R/W
Base + 0x124	Frame sync word 1(MACA_FRAMESYNC0)	R/W
Base + 0x128	Frame sync word 2(MACA_FRAMESYNC1)	R/W
Base + 0x140	Tx acknowledgement delay (MACA_TXACKDELAY)	R/W
Base + 0x144	Rx acknowledgement delay (MACA_RXACKDELAY)	R/W
Base + 0x148	End of frame delay (MACA_EOFDELAY)	R/W
Base + 0x14C	CCA delay (MACA_CCADELAY)	R/W
Base + 0x150	(MACA_RXEND)	R/W
Base + 0x154	TX CCA delay (MACA_TXCCADELAY)	R/W
Base + 0x158	MACA_KEY3	R
Base + 0x15c	MACA_KEY2	R
Base + 0x160	MACA_KEY1	R

Table 9-1. MACA Memory Map

Address	Register name	Access
Base + 0x164	MACA_KEY0	R
Base + 0x180	MACA_OPTIONS	W

9.7 MACA Register Description

The following sections provide detailed register descriptions.

9.7.1 MACA_VERSION

The MACA_VERSION register determines the version of the MACA. The register has a major and minor version number field. The minor version field indicates directly compatible versions. The major version field requires code changes in the driver layer among other things. (Major = 1, Minor = 0 corresponds to version 1.00).

MACA_VERSION									Base + 0x00							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved								MAJOR							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Reserved									MINOR							
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved								MINOR							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-2. MACA_Version Bits

Name	Description	Settings
Bit 31-24	Reserved bits — Read as zero and written with zero for future compatibility.	N/A
MAJOR Bit 23-16	Major version code — Reflects the major version of the MACA module.	
Bit 15-8	Reserved bits — Read as zero and written with zero for future compatibility.	N/A
MINOR Bit 7-0	Minor version code — Reflects the minor version of the MACA module.	

9.7.2 MACA_RESET

Writing to this register initiates a software reset of the MACA. After this reset, all internal registers are set to their reset values. This register also controls the signal that the MACA sends back to the clock controller module. If the MACA is not used for an extended period of time, the MACA clock can be disabled by writing a 0 to the CLK_ON bit. This saves additional power.

	MACA_RESET								Base + 0x04							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved														CLK_ON	RST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-3. MACA_RESET Register Bits

Name	Description	Settings
Bit 31-2	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
RST Bit 0	MACA Reset—Setting this bit initiates a module reset, completed by reset interrupt. This is NOT a self clearing bit. To release the module from reset, this bit must be cleared. This allows the software to hold the module in reset for any length of time.	1 = Reset 0 = Do not reset
CLK_ON Bit 1	MACA CLK_ON—Setting this bit causes the clock control module to activate the clock to the MACA. The default setting is “0”. By default the MACA is on. To save additional power when the MACA is not in use, the software can clear this bit to turn the MACA clocks off.	1 = MACA clock on 0 = MACA clock off

9.7.3 MACA_RANDOM

The MACA_RANDOM register generates random numbers. Writing to this register initializes the engine with a seed. Reading from this register returns a new 32-bit random value every time the register is read.

	MACA_RANDOM								Base + 0x08							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	(read)DATA / (write)SEED															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	(read)DATA / (write)SEED(cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-4. MACA_RANDOM Bits

Name	Description	Settings
DATA Bit 31-0 (read)	Read Random Data—Every read access returns a new random value.	
SEED Bit 31-0 (write)	Generator Seed—Initializes the random generator with new seed.	

9.7.4 MACA_CONTROL

The MACA_CONTROL register controls the MACA. This register starts auto sequences (receive, transmit), and associated behavior for these sequences. If timer triggers are used for the sequence, these must be setup before writing to the control register. Writing to the control register clears the status register and sets the complete code to “not completed”.

NOTE

All bits must be properly set for the entire sequence when writing to the MACA_CONTROL register. Any further writes are taken as start-up of a new auto sequence.

MACA_CONTROL									Base + 0x0C							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved											ISM	PRE_COUNT			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	RSTO	RSV	ROLE	NOFC	PRM	REL	ASAP	BCN	AUTO	LFSR	TM	MODE		SEQUENCE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-5. MACA_CONTROL Bits

Name	Description	Settings
Bit 31-21	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
ISM Bit 20	Reserved	
PRE_COUNT Bit 19-16	Preamble repeat counter. This bit field can be set to a value of 0 to 15. To repeat the preamble pattern 8 times, set this field to 7.	
RSTO Bit 15	Reset Slot Offset—Setting this bit resets the internal 20-counter for slotted CSMA-CA access. For TX, the counter is reset after transmission of length field. For RX, the counter is reset after reception of length field.	1 = Reset counter 0 = Do not reset counter
RSV Bit 14	Reserved	
ROLE Bit 13	Current Role—This bit sets the operating role needed for proper header filtering.	1 = PAN Coordinator 0 = Not PAN Coordinator
NOFC Bit 12	No Frame Check—Setting this bit disables check for correct checksum on received packets.	1 = Disable FCS 0 = Enable FCS
PRM Bit 11	Promiscuous Mode—Setting this bit enables promiscuous mode. No data filtering is performed on received packets.	1 = Promiscuous mode 0 = Normal mode
rel Bit 10	Clock Selector—This bit selects either use of absolute or relative clock for timing actions in the sequence.	1 = Relative clock 0 = Absolute clock

Table 9-5. MACA_CONTROL Bits

Name	Description	Settings
Bit 31-21	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
ASAP Bit 9	Start Action Sequence ASAP—Setting this bit starts the sequence immediately. Otherwise, the start-up is postponed until the start clock condition is satisfied.	1 = Start ASAP 0 = Start timer triggered
BCN Bit 8	Filter Beacon Only—In receive mode, setting this bit filters out packets which are not beacon packets.	1 = Only receive beacon packets 0 = Receive all packets
AUTO Bit 7	Restart Rx Automatically—In receive mode, this bit instructs the sequencer to restart receiver after receiving data. Otherwise the sequence completes when the first valid data has been received.	1 = Restart Rx 0 = Complete after first packet
LFSR Bit 6	Reserved	
TM Bit 5	Test mode-- Setting this bit either forces the MACA to continuously send the preamble pattern, or forces the MACA to continuously receive data. Write 32'h00000223 to the control register for continuos TX. Write 32'h00000224 to the control register for continuos RX.	1=test mode 0=normal mode
MODE Bit 4-3	Transmission Access Mode—These control bits set up transmission with CCA, and mode (slotted CSMA-CA has 2 CCA, non-slotted CSMA-CA has 1 CCA prior to transmission).	0 = No CCA 1 = Non slotted CSMA-CA 2 = Slotted CSMA-CA 3 = Reserved
SEQUENCE Bit 2-0	Sequence—These bits set the new mode of operation of the MACA. Abort forces the sequencer into idle. Wait is a sequence which is idle, but generates a complete on time-out. Poll is an extended Tx sequence for transmitting MAC data requests. CCA and Energy Detect (ED) are single actions.	0 = Nop 1 = Abort 2 = Wait 3 = Tx 4 = Rx 5 = TxPoll 6 = CCA 7 = ED

9.7.5 MACA_STATUS

The MACA_STATUS register contains the completion codes, which provide additional information from the MACA when a completion action interrupt is generated. This register is cleared when writing to the MACA_CONTROL register which sets the COMPLETE_CODE to “Not completed”.

	MACA_STATUS								Base + 0x10							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	TO	CRC	BUSY	OVR	zigbee	reserved							COMPLETE_CODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-6. MACA_Status Bits

Name	Description	Settings
Bit 31-16	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
TO Bit 15	Time-out—This bit is set when a time-out occurred in the last completed sequence.	1 = Time-out 0 = No time-out
CRC Bit 14	Checksum Failed—If this bit is set, the last sequence detected a bad CRC.	1 = CRC failure detected 0 = No CRC failure detected
BUSY Bit 13	Channel Busy Detection—The CCA detected a busy channel.	1 = Busy channel detected 0 = No busy channel detected
OVR Bit 12	Rx Buffer Overrun—No Rx DMA point available when data received. Packet discarded.	1 = Rx overrun detected 0 = No overrun detected
Bit 11	Reserved	

Table 9-6. MACA_Status Bits

Name	Description	Settings
Bit 10-4	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
COMPLETE_CODE Bit 3-0	Sequence Complete Codes—These bits reflect the status of the last completed sequence.	0 = Success 1 = Time-out 2 = Channel Busy 3 = CRC failed 4 = Aborted 5 = No Ack 6 = No data 7 = Late start 8 = ExtTimeout 9 = ExtPndTimeout 10 = not used 11 = not used 12 = pll unlock 13 = External abort 14 = Not completed 15 = DMA bus error

9.7.6 MACA_FRMPND

The MACA_FRMPND register sets the proper frame pending information in acknowledgement response to a MAC data request packet. This reflects the status of the indirect transmission queue (empty or not).

	MACA_FRMPND								Base + 0x14							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved															PND
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-7. MACA_FRMPND Bits

Name	Description	Settings
Bit 31-1	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
PND Bit 0	Acknowledgement Frame Pending Status—This bit is copied into acknowledgement responses to MAC data request packets and instructs the remote side to open its receiver for data.	1 = Data available 0 = No data pending

9.7.7 MACA_FREQ

The MACA_FREQ register sets up the RF modem front end for selected frequency channel.

See [Section 9.7.7, “MACA_FREQ”](#) for setting frequency.

	MACA_FREQ								Base + 0x18							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved												CHANNEL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-8. MACA_FREQ Bits

Name	Description	Settings
Bit 31-4	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
CHANNEL Bit 3-0	Frequency Channel—The currently active frequency channel (channel 0 equals 2405 MHz, and channel 15 equals 2480 MHz).	0 = 2405 MHz 1 = 2410 MHz 15 = 2480 MHz

9.7.8 MACA_EDVALUE

Not used.

9.7.9 MACA_TMREN

The MACA_TMREN register enables the timers. Only timers selected by the respective bits are affected. Reading this register returns the status of the individual timers.

	MACA_TMREN								Base + 0x40							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved												SFT	CPL	STRT	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-9. MACA_TMREN Bits

Name	Description	Settings
Bit 31-16	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
Bit 14-3	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
SFT Bit 2	Enable Soft Complete Clock —This bit enables the soft complete clock.	1 = Enable clock 0 = Clock not affected
CPL Bit 1	Enable Complete Clock —This bit enables the complete clock circuit.	1 = Enable complete clock 0 = Clock not affected
STRT Bit 0	Enable Start Clock —This bit enables the start clock circuit.	1 = Enable start clock 0 = Clock not affected

9.7.10 MACA_TMRDIS

The MACA_TMRDIS register disables the timers. Only the timers selected by the respective bits are affected. Reading this register returns the status of the individual timers.

	MACA_TMRDIS								Base + 0x44							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved										SFT_OFF	CPL_OFF	STRT_OFF	SFT	CPL	STRT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-10. MACA_TMRDIS Bits

Name	Description	Settings
Bit 31-4	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
SFT_OFF Bit 5	Abort Soft Complete Clock—This bit aborts a running soft complete clock, and terminates it immediately. If it is armed, a time-out is generated.	1 = Abort soft clock 0 = Clock not affected
CPL_OFF Bit 4	Abort Complete Clock—This bit aborts a running complete clock, and terminates it immediately. If it is armed, a time-out is generated.	1 = Abort complete clock 0 = Clock not affected
STRT_OFF Bit 3	Abort Start Clock—This bit aborts a running start clock, and terminates it immediately. If it is armed, a time-out is generated.	1 = Abort start clock 0 = Clock not affected
SFT Bit 2	Disable Soft Complete Clock—This bit disables the c'soft' complete clock circuit.	1 = Disable soft clock 0 = Clock not affected
CPL Bit 1	Disable Complete Clock—This bit disables the complete clock circuit.	1 = Disable complete clock 0 = Clock not affected
STRT Bit 0	Disable Start Clock—This bit disables the start clock circuit.	1 = Disable start clock 0 = Clock not affected

9.7.11 MACA_CLK

The MACA_CLK register is a 32-bit clock running at 250kHz. Writing to this register sets a new absolute clock. Ensure that the timers are not active because they will not be modified accordingly.

	MACA_CLK								Base + 0x48							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	ZIGBEE_CLOCK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	ZIGBEE_CLOCK (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-11. MACA_CLK Bits

Name	Description	Settings
ZIGBEE_CLOCK Bit 31-0	ZigBee (802.15.4 Standard) Absolute Clock—Reflects the current absolute clock.	250 kHz clock

9.7.12 MACA_STARTCLK

The MACA_STARTCLK register is a one-shot trigger used for starting actions. The clock is matched against either the absolute or relative clock, depending on the “SEL” bit in the MACA_TMREN register. If enabled, and once a match occurs, an ActionStarted_irq interrupt is generated and the timer is automatically disabled. In addition, the sequencer state machine uses this as an internal signal to proceed with its current auto-sequence.

	MACA_STARTCLK								Base + 0x4C							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	START_CLOCK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	STAR_CLOCK (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-12. MACA_STARTCLK Bits

Name	Description	Settings
START_CLOCK Bit 31-0	802.15.4 Standard Trigger Start Clock—The desired trigger time for an action to start.	250 kHz clock

9.7.13 MACA_CPLCLK

The MACA_CPLCLK register is a one-shot trigger used for terminating actions (primarily receive actions). The clock is matched against either the absolute or relative clock, depending of the “SEL” bit in the MACA_TMREN register. If enabled, and once a match occurs, an internal signal is generated for use in the sequencer. Normally, it terminates a receive sequence.

	MACA_CPLCLK								Base + 0x50							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	COMPLETE_CLOCK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	COMPLETE_CLOCK (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-13. MACA_CPLCLK Bits

Name	Description	Settings
COMPLETE_CLOCK Bit 31-0	802.15.4 Standard Trigger Complete Clock—The desired trigger time for an action to finish.	

9.7.14 MACA_SFTCLK

The MACA_SFTCLK register is a one-shot trigger that completes receive actions with the “extended” time-out. The clock is matched against either the absolute or relative clock, depending of the “SEL” bit in the MACA_TMREN register. If enabled, and once a match occurs, an internal signal is generated for use in the sequencer. See [Section 9.5.1.3.2, “Soft Time-out”](#).

	MACA_SFTCLK								Base + 0x54							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	SOFT_CLOCK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	SOFT_CLOCK (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-14. MACA_SFTCLK Bits

Name	Description	Settings
SOFT_CLOCKEXT Bit 31-0	Soft Complete Clock—The desired trigger time for an action to finish if no packet is detected. Trigger is postponed until packet completion (or rejection) if receiver has detected SFD.	

9.7.15 MACA_CLKOFFSET

The MACA_CLKOFFSET register is the offset between the absolute and relative clock. Updating the relative clock also updates this register. Likewise, updating the offset register updates the relative clock. The relative clock is defined as absolute clock + offset.

MACA_CLKOFFSET									Base + 0x58							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	CLOCK_OFFSET															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	CLOCK_OFFSET (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-15. MACA_CLKOFFSET Bits

Name	Description	Settings
CLOCK_OFFSET Bit 31-0	Clock offset—The offset between the absolute clock and the relative clock.	250 kHz clock

9.7.16 MACA_RELCLK

The MACA_RELCLK register is a 32-bit clock running at 250kHz and is based on the absolute clock + offset. Writing to this register sets a new relative clock and updates the offset. Ensure that the timers are not active because they will not be modified accordingly.

MACA_RELCLK									Base + 0x5C							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	RELATIVE_CLOCK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	RELATIVE_CLOCK (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-16. MACA_RELCLK Bits

Name	Description	Settings
RELATIVE_CLOCK Bit 31-0	Relative Clock—Reflects the current relative clock, and sets a new relative clock.	250 kHz clock

9.7.17 MACA_CPLTIM

The MACA_CPLTIM register is a timestamp latched upon the completion of an auto-sequence. Depending on the “SEL” bit in the MACA_TMREN register, either the absolute or relative clock is latched.

	MACA_CPLTIM								Base + 0x60							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	COMPLETE_TIME															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	COMPLETE_TIME (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-17. MACA_CPLTIM Bits

Name	Description	Settings
COMPLETE_TIME Bit 31-0	Complete Time—Timestamp of the time a sequence completed.	250 kHz clock

9.7.18 MACA_SLOTOFFSET

The MACA_SLOTOFFSET register properly aligns access to the channel in slotted CSMA-CA mode. An internal backoff counter counts from 0 to 79. Access is then only allowed every 20 symbols (clock runs at 4x symbol speed). To account for radio setup time and radio delay chains, the MACA_SLOTOFFSET sets up an initial offset to the received/transmitted beacon.

If the “RSTO” bit is set in the control register, and a beacon is received, the internal backoff counter is loaded with the contents of MACA_SLOTOFFSET (the offset is loaded after reception of the length field). If the “RSTO” is set, and the sequence is a transmission, the offset is loaded after transmission of the length field.

	MACA_SLOTOFFSET								Base + 0x64							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved				RX_SLOT_OFFSET											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved				TX_SLOT_OFFSET											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-18. MACA_SLOTOFFSET Bits

Name	Description	Settings
Bit 31-28	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
RX_SLOT_OFFSET Bit 27-16	Receive Slot Offset—The value for “resetting” the internal backoff counter for slotted CSMA-CA access. Each bit corresponds to an offset of 0.25 symbols. This values is used when synchronizing to a received beacon.	
Bit 15-12	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
TX_SLOT_OFFSET Bit 11-0	Transmit Slot Offset—The value used for “resetting” the internal backoff counter for slotted CSMA-CA access. Each bit corresponds to an offset of 0.25 symbols. This value is used when transmitting a beacon.	

9.7.19 MACA_TIMESTAMP

The MACA_TIMESTAMP register contains the latched clock for the last received packet. The clock is latched immediately after receiving the length field. Depending on the “SEL” bit in the MACA_TMREN register, either the relative or absolute clock is latched.

	MACA_TIMESTAMP								Base + 0x68							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	TIMESTAMP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	TIMESTAMP (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-19. MACA_TIMESTAMP

Name	Description	Settings
TIMESTAMP Bit 31-0	Timestamp—Latched clock time stamping the last received packet.	250 kHz clock

9.7.20 MACA_DMARX

The MACA_DMARX register sets up a receive pointer used for DMA transfer. The DMA transfers data from the MACA receive queue into the memory location referred to by the MACA_DMARX register. The register is not updated during reception of a packet because this would allow the sequencer to reload the pointer if, for example, the CRC fails on a received packet.

Writing to this register indicates buffer availability to the sequencer. When a DataIndication_irq is generated, the buffer is used. If no write to the MACA_DMARX is performed before a new packet sync is found, the sequencer handles the received packet as having a “bad” header. In addition, the “OVR” bit in the status register is set.

NOTE

The first transferred data will be the LQI and packet length. The receive buffer must be large enough to hold these bytes as well as the data.

	MACA_DMARX								Base + 0x80							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	DEST_PTR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	DEST_PTR (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-20. MACA_DMARX Bits

Name	Description	Settings
DEST_PTR Bit 31-0	DMA Destination Pointer—Pointer to memory where to transfer data.	

9.7.21 MACA_DMATX

The MACA_DMATX register sets up a transmission pointer used for DMA transfer. The DMA transfers data from memory to the MACA transmit queue. The register is not updated during transmission of a packet.

	MACA_DMATX								Base + 0x84							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	SRC_PTR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	SRC_PTR (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-21. MACA_DMATX Bits

Name	Description	Settings
SRC_PTR Bit 31-0	DMA Source Pointer—Pointer in memory where to transfer data from.	

9.7.22 MACA_DMAPOLL

The MACA_DMAPOLL register sets up a transmission pointer used for DMA transfer during a fast poll response sequence. Writing to this register arms an immediate reply to a remote data request (See [Section 9.5.1.3.5, “Handling Received MAC Data Request”](#)). The DMA transfers data from memory to the MACA transmit queue. The register is not updated during transmission of a packet.

	MACA_DMAPOLL								Base + 0x88							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	SRC_PTR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	SRC_PTR (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-22. MACA_DMAPOLL Bits

Name	Description	Settings
SRC_PTR Bit 31-0	DMA Source Pointer—Pointer in memory where to transfer data from.	

9.7.23 MACA_TXLEN

The MACA_TXLEN register specifies how many bytes of data is to be transmitted during a Tx auto-sequence (including checksum, but excluding PHY header). This register should be written prior to MACA_DMATX or MACA_DMAPOLL.

MACA_TXLEN									Base + 0x8C							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	RX_LEN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	res	TX_LEN														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-23. MACA_TXLEN Bits

Name	Description	Settings
RX_LEN Bit 30--16	Reserved bits —Sets the maximum length of a received packet. If the incoming packet is longer than this value, then it will be rejected.	
TX_LEN Bit 14-0	Tx payload length —Length of transmitted data including checksum. In 802.15.4 Standard mode only the lower 7 bits are used. {choose_ifsr,tx_len}	

9.7.24 MACA_TXSEQNR

The MACA_TXSEQNR register is a copy of the sequence number embedded in the data to transmit. It validates acknowledgement. If acknowledgement is not required, this register need not be updated for transmissions. This register must be written prior to MACA_DMATX or MACA_DMAPOLL.

	MACA_TXSEQNR								Base + 0x90							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved								TXSEQN							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-24. MACA_TXSEQNR Bits

Name	Description	Settings
Bit 31-8	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
TXSEQN Bit 6-0	Tx Sequence Number—Sequence number of transmitted packet for validating acknowledgement.	

9.7.25 MACA_SETRXLVL

The MACA_SETRXLVL register specifies the FIFO trigger level for incoming packets. Once the level is reached or exceeded (and data is transferred to the destination using DMA), an RxLevel_irq is generated.

It is possible to update the level register during packet reception which allows for multiple interrupts for packet processing.

	MACA_SETRXLVL								Base + 0x94							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	FIFO_LVL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-25. MACA_SETRXLVL Bits

Name	Description	Settings
Bit 31-8	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
FIFO_LVL Bit 15-0	FIFO Level—Indicates the number of bytes to receive before generating an RxLevel_irq.	

9.7.26 MACA_GETRXLVL

Reading the MACA_GETRXLVL register returns the number of bytes already received and transferred to memory using the DMA.

	MACA_GETRXLVL								Base + 0x98							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	RecvBytes															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-26. MACA_GETRXLVL Bits

Name	Description	Settings
Bit 31-7	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
RecvBytes Bit 15-0	Received Bytes!—Indicates the number of available received bytes.	

9.7.27 MACA_IRQ

The MACA_IRQ register is a read-only register that contains interrupt sources. Interrupts are handshaked by writing to the interrupt clear register (MACA_CLRIRQ).

	MACA_IRQ								Base + 0xC0							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	STRT	SYNC	CM	CRC	FLT	SFT	LVL	Reserved				RST	WU	DI	POLL	ACPL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-27. MACA_IRQ

Name	Description	Settings
Bit 31-16	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
STRT Bit 15	Action Started Interrupt—An auto-sequence is started, either immediately or by timer trigger.	1 = Clear interrupt source 0 = Leave source untouched
SYNC Bit 14	Sync Detected Interrupt—The modem has detected the beginning of a new packet	1 = Clear interrupt source 0 = Leave source untouched
CM Bit 13	Complete Clock Interrupt—The complete clock has generated a trigger.	1 = Clear interrupt source 0 = Leave source untouched
CRC Bit 12	Checksum Failed Interrupt—The checksum failed for the received packet.	1 = Clear interrupt source 0 = Leave source untouched
FLT Bit 11	Filter Failed Interrupt—The receive header filter failed.	1 = Clear interrupt source 0 = Leave source untouched
SFT Bit 10	Soft Complete Clock Interrupt—The soft complete clock has generated a trigger.	1 = Clear interrupt source 0 = Leave source untouched
LVL Bit 9	FIFO Level interrupt—The receive FIFO level is reached or exceeded.	1 = Clear interrupt source 0 = Leave source untouched
Bit 8-5	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
RST Bit 4	Reset Interrupt—A non maskable reset interrupt detected	1 = Clear interrupt source 0 = Leave source untouched
WU Bit 3	wake-up Interrupt—Low power mode has been exited.	1 = Clear interrupt source 0 = Leave source untouched
DI Bit 2	Data Indication Interrupt—During receive, a packet was successfully received.	1 = Clear interrupt source 0 = Leave source untouched

Table 9-27. MACA_IRQ

Name	Description	Settings
POLL Bit 1	Poll Indication Interrupt—Issued when data request received (and before ACK transmitted). MCU may then set MACA_FRMPND and prepare fast response.	1 = Clear interrupt source 0 = Leave source untouched
ACPL Bit 0	Action Complete Interrupt—Marks the completion of a complete auto-sequence.	1 = Clear interrupt source 0 = Leave source untouched

9.7.28 MACA_CLRIRQ

The MACA_CLRIRQ register clears interrupt sources. Writing a bit-field clears interrupts selected by set bits. Internally, the MACA module inverts the bit-field and “AND’s” it on the interrupt sources. For example, writing 0x0003 clears interrupt sources 0 and 1 (POLL and ACPL), but other sources are untouched.

	MACA_CLRIRQ								Base + 0xC4							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	STRT	SYNC	CM	CRC	FLT	SFT	LVL	Reserved				RST	WU	DI	POLL	ACPL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-28. MACA_CLRIRQ Bits

Name	Description	Settings
Bit 31-16	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
STRT Bit 15	Action Started Interrupt—An auto-sequence is started, either immediately or by timer trigger.	1 = Clear interrupt source 0 = Leave source untouched
SYNC Bit 14	Sync Detected Interrupt—The modem has detected the beginning of a new packet	1 = Clear interrupt source 0 = Leave source untouched
CM Bit 13	Complete Clock Interrupt—The complete clock has generated a trigger.	1 = Clear interrupt source 0 = Leave source untouched
CRC Bit 12	Checksum Failed Interrupt—The checksum failed for the received packet.	1 = Clear interrupt source 0 = Leave source untouched
FLT Bit 11	Filter Failed Interrupt—The receive header filter failed.	1 = Clear interrupt source 0 = Leave source untouched
SFT Bit 10	Soft Complete Clock Interrupt—The soft complete clock has generated a trigger.	1 = Clear interrupt source 0 = Leave source untouched
LVL Bit 9	FIFO Level interrupt—The receive FIFO level is reached or exceeded.	1 = Clear interrupt source 0 = Leave source untouched

Table 9-28. MACA_CLRIRQ Bits

Name	Description	Settings
Bit 8-5	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
RST Bit 4	Reset Interrupt—A non maskable reset interrupt detected (<u>TBD!!!</u>)	1 = Clear interrupt source 0 = Leave source untouched
WU Bit 3	wake-up Interrupt—Low power mode has been exited (<u>TBD in connection with CCM module</u>).	1 = Clear interrupt source 0 = Leave source untouched
DI Bit 2	Data Indication Interrupt—During receive, a packet has been successfully received.	1 = Clear interrupt source 0 = Leave source untouched
POLL Bit 1	Poll Indication Interrupt—Issued when data request received (and before ACK transmitted). MCU may then set MACA_FRMPND and prepare fast response. <u>TBD: Shall this be skipped if MACA FRMPND is clear?</u>	1 = Clear interrupt source 0 = Leave source untouched
ACPL Bit 0	Action Complete Interrupt—Marks the completion of a complete auto-sequence.	1 = Clear interrupt source 0 = Leave source untouched

9.7.29 MACA_SETIRQ

The MACA_SETIRQ register forces interrupt sources generated by software. Writing a bit-field sets interrupts selected by set bits. Internally, the MACA module “OR’s” the bit-field on the interrupt sources. For example, writing 0x0003, sets interrupt sources 0 and 1 (POLL and ACPL), but other sources are untouched.

	MACA_SETIRQ								Base + 0xC8							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	STRT	SYNC	CM	CRC	FLT	SFT	LVL	Reserved				RST	WU	DI	POLL	ACPL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-29. MACA_SETIRQ Bits

Name	Description	Settings
Bit 31-16	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
STRT Bit 15	Action Started Interrupt—An auto-sequence is started, either immediately or by timer trigger.	1 = Set interrupt source 0 = Leave source untouched
SYNC Bit 14	Sync Detected Interrupt—The modem has detected the beginning of a new packet	1 = Set interrupt source 0 = Leave source untouched

Table 9-29. MACA_SETIRQ Bits

Name	Description	Settings
CM Bit 13	Complete Clock Interrupt—The complete clock has generated a trigger.	1 = Set interrupt source 0 = Leave source untouched
CRC Bit 12	Checksum Failed Interrupt—The checksum failed for the received packet.	1 = Set interrupt source 0 = Leave source untouched
FLT Bit 11	Filter Failed Interrupt—The receive header filter failed.	1 = Set interrupt source 0 = Leave source untouched
SFT Bit 10	Soft Complete Clock Interrupt—The soft complete clock has generated a trigger.	1 = Set interrupt source 0 = Leave source untouched
LVL Bit 9	FIFO Level interrupt—The receive FIFO level is reached or exceeded.	1 = Set interrupt source 0 = Leave source untouched
Bit 8-5	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
RST Bit 4	Reset Interrupt—A non maskable reset interrupt detected (TBD!!!)	1 = Set interrupt source 0 = Leave source untouched
WU Bit 3	wake-up Interrupt—Low power mode has been exited (TBD in connection with CCM module).	1 = Set interrupt source 0 = Leave source untouched
DI Bit 2	Data Indication Interrupt—During receive, a packet has been successfully received.	1 = Set interrupt source 0 = Leave source untouched
POLL Bit 1	Poll Indication Interrupt—Issued when data request received (and before ACK transmitted). MCU may then set MACA_FRMPND and prepare fast response. TBD: Shall this be skipped if MACA_FRMPND is clear?	1 = Set interrupt source 0 = Leave source untouched
ACPL Bit 0	Action Complete Interrupt—Marks the completion of a complete auto-sequence.	1 = Set interrupt source 0 = Leave source untouched

9.7.30 MACA_MASKIRQ

The MACA_MASKIRQ register enables interrupts. Only masked interrupts sources generate a MACA interrupt the MCU.

	MACA_MASKIRQ								Base + 0xCC							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	STRT	SYNC	CM	CRC	FLT	SFT	LVL	Reserved						DI	POLL	ACPL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-30. MACA_MACKRIQ Bits

Name	Description	Settings
Bit 31-16	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
STRT Bit 15	Action Started Interrupt—An auto-sequence is started, either immediately or by timer trigger.	1 = Clear interrupt source 0 = Leave source untouched
SYNC Bit 14	Sync Detected Interrupt—The modem has detected the beginning of a new packet	1 = Clear interrupt source 0 = Leave source untouched
CM Bit 13	Complete Clock Interrupt—The complete clock has generated a trigger.	1 = Clear interrupt source 0 = Leave source untouched
CRC Bit 12	Checksum Failed Interrupt—The checksum failed for the received packet.	1 = Clear interrupt source 0 = Leave source untouched
FLT Bit 11	Filter Failed Interrupt—The receive header filter failed.	1 = Clear interrupt source 0 = Leave source untouched
LVL Bit 9	FIFO Level interrupt—The receive FIFO level is reached or exceeded.	1 = Clear interrupt source 0 = Leave source untouched
Bit 8-5	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
bit 4	Not used	
bit 3	Not used	
DI Bit 2	Data Indication Interrupt—During receive, a packet has been successfully received.	1 = Clear interrupt source 0 = Leave source untouched
POLL Bit 1	Poll Indication Interrupt—Issued when data request received (and before ACK transmitted). MCU may then set MACA_FRMPND and prepare fast response.	1 = Clear interrupt source 0 = Leave source untouched
ACPL Bit 0	Action Complete Interrupt—Marks the completion of a complete auto-sequence.	1 = Clear interrupt source 0 = Leave source untouched

9.7.31 MACA_MACPANID

Header filtering needs the macPANID value which is placed in the MACA_MACPANID register.

MACA_MACPANID									Base + 0x100							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PANID																
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	PANID															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 9-31. MACA_MACPANID Bits

Name	Description	Settings
Bit 31-16	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
PANID Bit 15-0	MAC PAN ID—Reflects the current MAC PAN Id for 802.15.4 Standard network.	

9.7.32 MACA_MAC16ADDR

The MACA_MAC16ADDR register is used in the header filtering and contains the assigned 16-bit short address of the device.

MACA_MAC16ADDR									Base + 0x104							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADDR																
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	ADDR															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 9-32. MACA_MAC16ADDR Bits

Name	Description	Settings
Bit 31-16	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
ADDR Bit 15-0	ADDR—Reflects the current short address for the device.	

9.7.33 MACA_MAC64HI

The MACA_MAC64HI register is the upper part of the assigned 64-bit IEEE address. Together with MACA_MAC64LO, it is used for header filtering on extended (64-bit) addresses.

	MACA_MAC64HI								Base + 0x108							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	EXT_ADDR_HI															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	EXT_ADDR_HI (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Name	Description	Settings
EXT_ADDR_HI Bit 31-0	High part of Extended Address—Reflects the assigned IEEE address.	

9.7.34 MACA_MAC64LO

The MACA_MAC64LO register is the lower part of the assigned 64-bit IEEE address.

	MACA_MAC64LO								Base + 0x10C							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	EXT_ADDR_LOI															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	EXT_ADDR_LO (cont)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-33. MACA_MAC64LO Bits

Name	Description	Settings
EXT_ADDR_LO Bit 31-0	Lower part of Extended Address—Reflects the assigned IEEE address.	

9.7.35 MACA_FLTREJ

The MACA_FLTREJ rejects incoming packets during header filtering.

	MACA_FLTREJ								Base + 0x110							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	FC_MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved							POLL	Reserved				CMD	ACK	DATA	BCN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-34. MACA_FLTREJ Bits

Name	Description	Settings
FC_MASK Bit 31-16	Frame Control Mask—The mask identifies which bits in frame control must be zero for packet accepting.	1 = bit must be zero 0 = bit is don't care
Bit 15-9	Reserved	
POLL Bit 8	Accept POLL packets—If this bit is set, only MAC Commands, which are actual “Data.request” (POLL) are accepted.	1 = Accept only POLL 0 = No restriction
Bit 7-4	Reserved	
CMD Bit 3	Reject Mac CMD packets—If this bit is set, Mac command types are rejected.	1 = Reject 0 = No rejection
ACK Bit 2	Reject ACK packets—If this bit is set, ACK packet types are rejected.	1 = Reject 0 = No rejection
DATA Bit 1	Reject Data packets—If this bit is set, Data packet types are rejected.	1 = Reject 0 = No rejection
BCN Bit 0	Reject Beacon packets—If this bit is set, beacon packet types are rejected.	1 = Reject 0 = No rejection

9.7.36 MACA_CLKDIV

The MACA_CLKDIV register sets the divider to generate the transmit clock.

MACA_CLKDIV									Base + 0x114							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	divider															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-35. MACA_CLKDIV Bits

Name	Description	Settings
divider Bit 15-0	integer—Example: Divide ratio is register value + 1. For example, if the bus clock is 24Mhz, set divider to 95 (0x5f) to set the transmit bit clock to 250khz.	

9.7.37 MACA_WARMUP

The MACA_WARMUP sets the values for Tx and Rx warmup.

MACA_WARMUP									Base + 0x118							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field					TX_WARMUP											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field					RX_WARMUP											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-36. MACA_WARMUP Bits

Name	Description	Settings
TX_WARMUP Bit 27-16	tx_warmup—This 12bit value sets the warmup time for the transmitter. For example, if this value is set to 10, then the sequencer tells the modem to start its warmup sequence 10 bits before the first bit of preamble is sent.	
RX_WARMUP Bit 11-0	rx_warmup—This 12bit value sets the warmup time for the receiver. For example, if the value is set to 10, then the sequencer tells the modem to start its warmup sequence 10 bits before the first bit of preamble is expected to be received.	

9.7.38 MACA_PREAMBLE

The MACA_PREAMBLE register sets the preamble pattern.

MACA_PREAMBLE									Base + 0x11C							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	PREAMBLE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	PREAMBLE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-37. MACA_PREAMBLE Bits

Name	Description	Settings
PREAMBLE Bit 31-0	preamble—This register sets the preamble pattern. In the control register there is a 4 bit field that controls the number of times this pattern is repeated.	

9.7.39 MACA_FRAME_SYNC0

The MACA_FRAME_SYNC0 register sets the first half of the framesync pattern.

MACA_FRAME_SYNC0									Base + 0x124							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	FRAMESYNC0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	FRAMESYNC0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-38. MACA_FRAME_SYNC0 Bits

Name	Description	Settings
FRAMESYNC0 Bit 31-0	framesync0—This register sets the lower half of the framesync pattern. The lower half is transmitted first.	

9.7.40 MACA_FRAME_SYNC1

The MACA_FRAME_SYNC0 register sets the second half of the framesync pattern.

	MACA_FRAME_SYNC1								Base + 0x128							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	FRAMESYNC1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	FRAMESYNC1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-39. MACA_FRAME_SYNC1 Bits

Name	Description	Settings
FRAMESYNC1 Bit 31-0	framesync1—This register sets the upper half of the framesync pattern. The upper half is transmitted after the lower half. In 802.15.4 Standard mode, only the lower 8 bits are used.	

9.7.41 MACA_TXACKDELAY

The MACA_TXACKDELAY register fine tunes acknowledgement transmissions to occur exactly 12 symbols after the last received bit. This register can adjust for radio warm-up and delay in Rx/Tx chain.

In slotted CSMA-CA mode, the delay is a “minimum” delay only. Transmission occurs on the first backoff slot after the delay. The delay is represented with a resolution of 0.25 symbols and a delay of 0 indicates that acknowledgement frames will be transmitted immediately after packet reception (For example, Tx warm-up starts immediately).

	MACA_TXACKDELAY								Base + 0x140							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved				TXPOLLDELAY											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved				TXACKDELAY											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-40. MACA_TXACKDELAY Bits

Name	Description	Settings
TXPOLLDELAY Bit 27-16	Tx PollDelay—Delay in 0.25 symbols from end of Poll command ack to beginning of transmitted polled data. This must greater than the transmitter warmup time.	
TXACKDELAY Bit 11-0	Tx Acknowledgement Delay—Delay in 0.25 symbols from packet reception to beginning of acknowledgement Tx warm-up.	

9.7.42 MACA_RXACKDELAY

This register can adjust for radio warm-up and delay in the Rx/Tx chain. The delay is represented with a resolution of 0.25 symbols and a delay of 0 indicates that acknowledgement must be transmitted immediately after packet reception (For example, Rx warm-up starts immediately).

MACA_RXACKDELAY									Base + 0x144							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved				RXAUTODELAY											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	RXACKDELAY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-41. MACA_RXACKDELAY Bits

Name	Description	Settings
Bit 31-28	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
RXAUTODELAY Bit 27-16	RX_AUTO_DELAY. Length of time (in bits) to disable receiver before restarting receiver.	
Bit 15-12	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
RXACKDELAY Bit 11-0	Rx Acknowledgement Delay—Determines the beginning of the acknowledgement frame search window. Measured in 0.25 symbols after Tx completed.	

9.7.43 MACA_EOFDELAY

The MACA_EOFDELAY register is a programmable delay for generation of the ActionComplete_irq interrupt. This ensures proper power down of radio chain before issuing a new action. This allows concurrent software processing while the radio is powering down. The delay is measured in increments of 0.25 symbols each.

	MACA_EOFDELAY								Base + 0x148							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved				EOF_DELAY											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-42. MACA_EOFDELAY Bits

Name	Description	Settings
Bit 31-7	Reserved bits—Read as zero and written with zero for future compatibility.	N/A
EOF_DELAY Bit 11-0	End of Frame Delay—Delay in 0.25 symbols from auto-sequence completes to generation of ActionComplete_irq interrupt.	

9.7.44 MACA_CCDELAY

The MACA_CCDELAY register fine tunes exactly when the CCA timing occurs. This register can adjust radio warm-up and delay in the Rx/Tx chain. The delay is represented with a resolution of 0.25 symbols. A delay of 0 indicates that the acknowledgement frame must be transmitted immediately after packet reception (For example, CCA warm-up starts immediately).

	MACA_CCDELAY								Base + 0x14C							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field					CCALENGTH											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field					CCADELAY											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-43. MACA_CCDELAY Bits

Name	Description	Settings
CCALENGTH Bit 27-16	CCALENGTH—Length of time to perform CCA.	
CCADELAY Bit 11-0	CCA Delay—Delay (in bits) from first CCA to second CCA in slotted CSMA-CA mode.	

9.7.45 MACA_RXEND

This register fine tunes reception of acknowledgement by specifying the end of the search window for slotted CSMA-CA and non-slotted CSMA-CA mode. The delay is represented with a resolution of 0.25 symbols. A delay of 0 indicates that the acknowledgement must be transmitted immediately after packet reception (For example, Tx warm-up starts immediately).

	MACA_RXEND								Base + 0x150							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved				RXSLOTTED_END											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	reserved				RXACK_END											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-44. MACA_RXEND

Name	Description	Settings
RXSLOTTED_END Bit 23-16	Rx Acknowledgement Window End in slotted CSMA-CA mode—Delay in 0.25 symbols from Tx completed to end of Rx ACK window in slotted CSMA-CA mode.	
RXACK_END Bit 11-0	Rx Acknowledgement Window End in Normal Mode—Delay in 0.25 symbols from Tx completed to end of Rx ACK window in normal mode	

9.7.46 MACA_TXCCADELAY

The MACA_TXCCADELAY register fine tunes exactly when the CCA timing occurs. This register can adjust for radio warm-up and delay in the Rx/Tx chain. The delay is represented with a resolution of 0.25 symbols. A delay of 0 indicates that the acknowledgement frame must be transmitted immediately after packet reception. (For example, CCA warm-up starts immediately).

MACA_TXCCADELAY									Base + 0x154							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved				TXCCADELAY											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-45. MACA_TXCCADELAY Bits

Name	Description	Settings
TXCCADELAY Bit 11-0	TX cca Delay—Delay (in bits) from end of CCA to TX start.	

9.7.47 MACA_KEY3

Reading this register returns a 32 bit random number.

MACA_KEY3									Base + 0x158							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-46. MACA_KEY3 Bits

Name	Description	Settings
KEY Bit 31,0	KEY—32 bit random value	

9.7.48 MACA_KEY2

Reading this register returns a 32 bit random number.

	MACA_KEY2								Base + 0x15C							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-47. MACA_KEY2 Bits

Name	Description	Settings
KEY Bit 31,0	KEY—32 bit random value	

9.7.49 MACA_KEY1

Reading this register returns a 32 bit random number.

	MACA_KEY1								Base + 0x160							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-48. MACA_KEY1 Bits

Name	Description	Settings
KEY Bit 31,0	KEY3—32 bit random value	

9.7.50 MACA_KEY0

Reading this register returns a 32 bit random number.

	MACA_KEY0								Base + 0x164							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-49. MACA_KEY0 Bits

Name	Description	Settings
KEY Bit 31,0	KEY—32 bit random value	

9.7.51 MACA_OPTIONS

The MACA_OPTIONS register can modify the behavior of the MACA.

	MACA_OPTIONS								Base + 0x							
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved												seed_key	pll_ignore	pll_tx	poll
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-50. MACA_OPTIONS Bits

Name	Description	Settings
Bit 31-4	Reserved bits —Read as zero and written with zero for future compatibility.	N/A
seed_key Bit 3	If this bit is a 1 then the key generation can be seeded with a write to the maca_random register. If this bit is a 0 then the register can not be seeded and will be in a random power up state. The flip-flops used in the key generation do not have a async reset.	
pll_ignore Bit 2	If this bit is a 1 then the MACA ignores the PLL unlock signal. If it is a 0 then the MACA responds.	
pll_tx Bit 1	If this bit is a 1 then the MACA only responds to the PLL unlock when the MACA is performing a transmit. If this bit is a 0 then the MACA responds as soon as PLL unlock is asserted.	
poll Bit 0	If this bit is a 1 then the DMA Tx poll response pointer (MACA_DMAPOLL) can be written to at any time. This allows writing to the pointer during the window between PollRx_irq generation and the completion of the following acknowledgement.	

Chapter 10

Advanced Security Module (ASM)

10.1 Overview

The ASM block is a hardware engine that encrypts/decrypts using the Advanced Encryption Standard (AES). It can perform "Counter" (CTR) and Cipher Block Chaining (CBC) encryption. The combination of these two modes of encryption are known as Counter with CBC-MAC (CCM) mode encryption. CCM is a generic authenticate and encrypt block cipher mode. CCM is only defined for use with 128-bit block ciphers, such as AES. The definition of CCM mode encryption is documented in the NIST publication SP800-38C. [Section 10.3.3, "CCM mode"](#) of this chapter provides a brief summary of how CCM mode works.

10.1.1 Features

The ASM has the following features:

- 32-bit access
- CTR encryption in 13 clock cycles.
- CBC encryption in 13 clock cycles.
- Encrypts 128 bits as a unit.
- The 128-bit data values (16 bytes) are each accessed as four 32-bit registers which are aligned to the 32-bit word boundaries.

10.2 ASM Module Block Diagram

The structure of the ASM module is shown in the block above. This structure combines both the CTR mode and CBC-MAC mode. This is also known as CCM mode encryption.

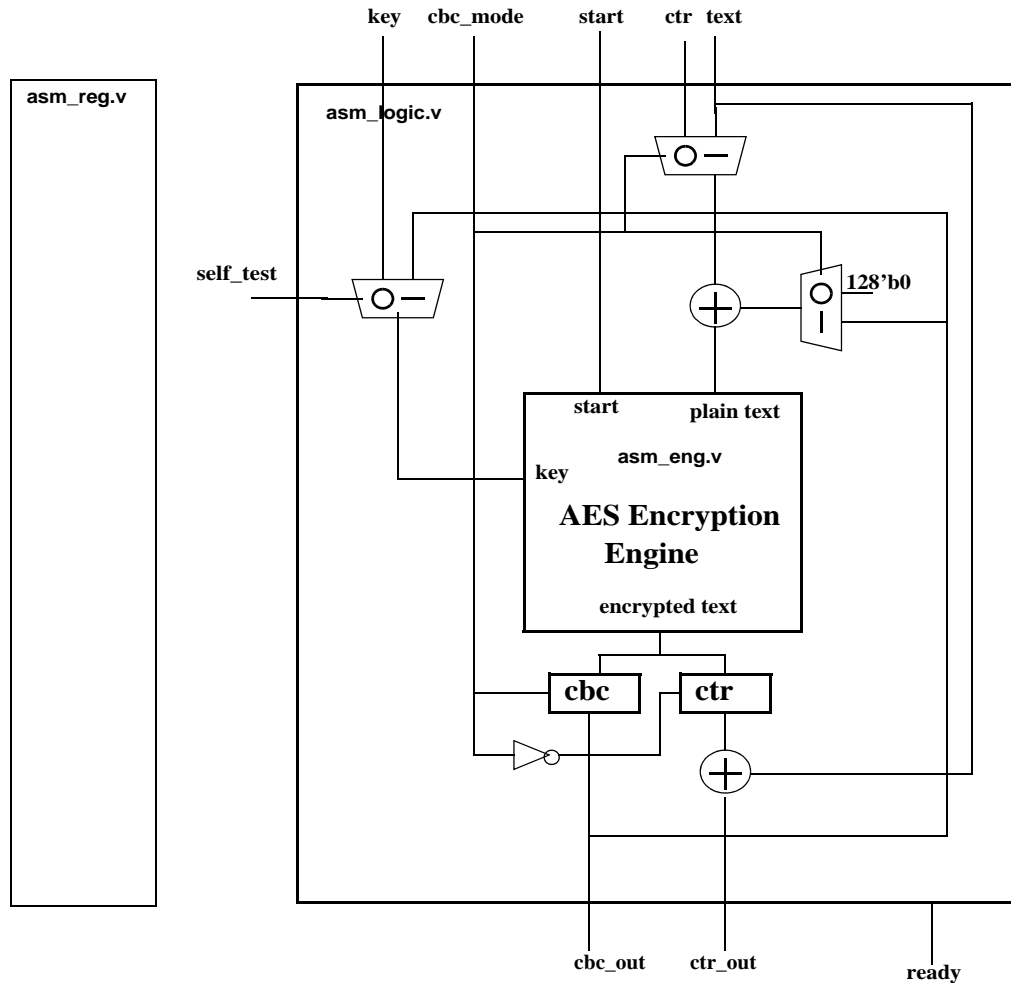


Figure 10-1. ASM Module Block Diagram

10.3 Mode of Operation

The ASM is designed to be loaded with data and then started with a self-clearing "start" bit. Four 32-bit registers of a key plus four 32-bit registers of a counter plus four 32-bit registers of text are necessary for "Counter" mode encryption. Cipher Block Chaining (CBC) mode needs only a key field and a text field programmed. Typically, only the text fields and counter fields need to be continuously written since the key field won't change.

The module has a built-in self test which requires 3330 clocks to complete. This test must be initiated by the software to make the module usable. Until this test is run and is passed, the ASM module is disabled. Typical usage would require that this test be run once when the module is powered up. The software can re-run this test at any time it becomes necessary to verify the operation of the encryption engine.

To use self-test:

1. Self-test mode is set by writing a "1" to the "self_test" bit.
2. Self-test is initiated by writing a "1" to the "start" bit.

3. After self_test is run (3330 clocks) -
 - If self test passes, the “test_pass” bit in Register 0x58 will be high.
 - If self-test fails, the “test_pass” bit will be cleared to a low, and the ASM module will be disabled and all outputs will be held to constant 0.
4. After using self-test, be sure and write a "0" to the "self_test" bit to clear this mode.

10.3.1 CTR mode Block Diagram

Counter mode encryption (CTR) is done by using the AES engine as shown in the block diagram above. The 128 bit key is written to register "key". The 128 bit counter value is written to register "ctr". The 128 bit data to encrypt is written to register "data". The start bit is set and the encrypted data can be read from the "ctr_result" register 13 clocks after the start bit is set. The status register has a "done" bit that the software can read to determine when the result is ready to be read.

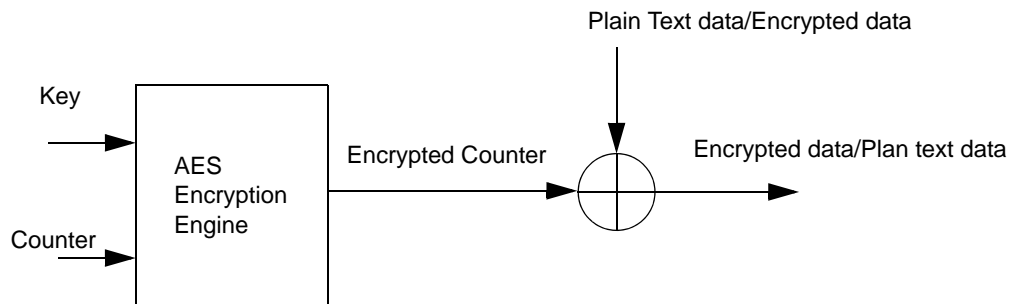


Figure 10-2. CTR Block

Decryption still uses the same AES encryption engine. The data to be decrypted is written to the "data" register. The decrypted result can be read from the "ctr_result" register.

10.3.2 CBC mode block diagram

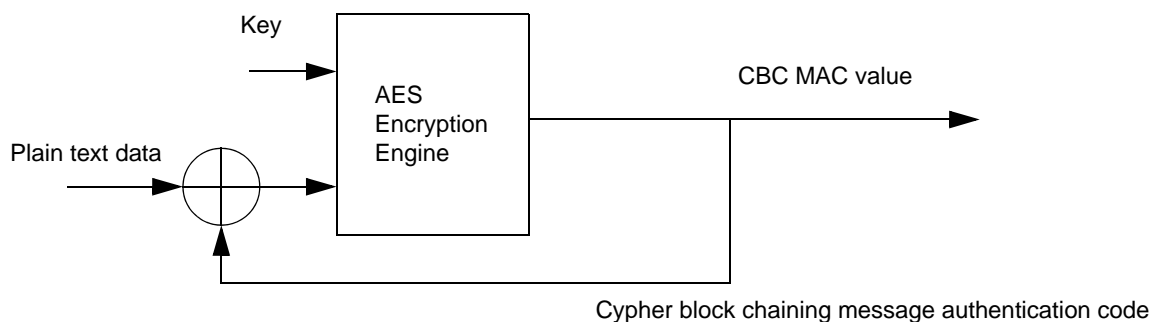


Figure 10-3. CBC Block

The CBC-MAC generation is done by using the AES engine as shown in the block diagram above. The 128 bit key is written to register "key". The 128 bit data value is written to register "data". The 128 bit data to encrypt is written to register "data". The start bit is set and the MAC value can be read from the cbc_result register 13 clocks after the start bit is set. The "done" bit can be read from the status register to determine when the result is complete.

10.3.3 CCM mode

CCM mode is the combination of using CTR mode for protecting the privacy of data, and using CBC mode to generate a MAC to protect the data from unauthorized modifications.

10.3.4 Boot mode

Boot mode is the default mode that the ASM module is in when it is released from reset. In boot mode the key value is set by an internal secret key. The boot ROM can then use the ASM module to decrypt or encrypt external ROM. When the boot ROM has completed this task it can then force the ASM module to normal mode by writing to the normal bit in the control register. It is not possible for the software to change the mode back to boot mode. While the module is in boot mode, the CCM modes (CTR and CBC) can be used for protecting the privacy of the external ROM.

10.3.5 Bypass mode

The ASM module can be put in bypass mode by setting the bypass bit in the control register. In bypass mode the data is passed through the module un-encrypted.

10.4 ASM module Block Diagram

The structure of the ASM module is shown in the block above. This structure combines both the CTR mode and CBC-MAC mode. This is also known as CCM mode encryption.

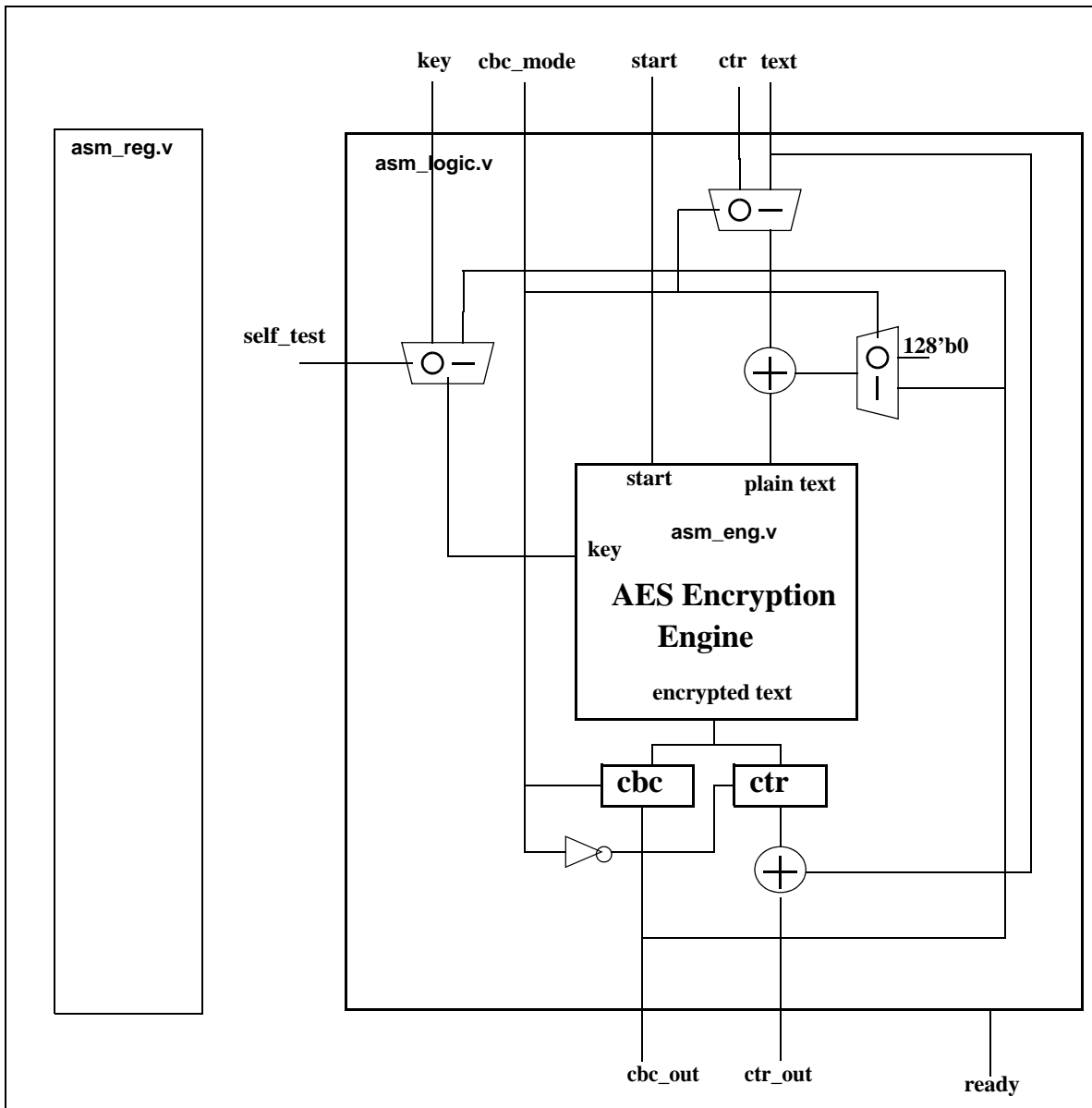


Figure 10-4. ASM Block

10.4.1 AES Encryption Engine Algorithm

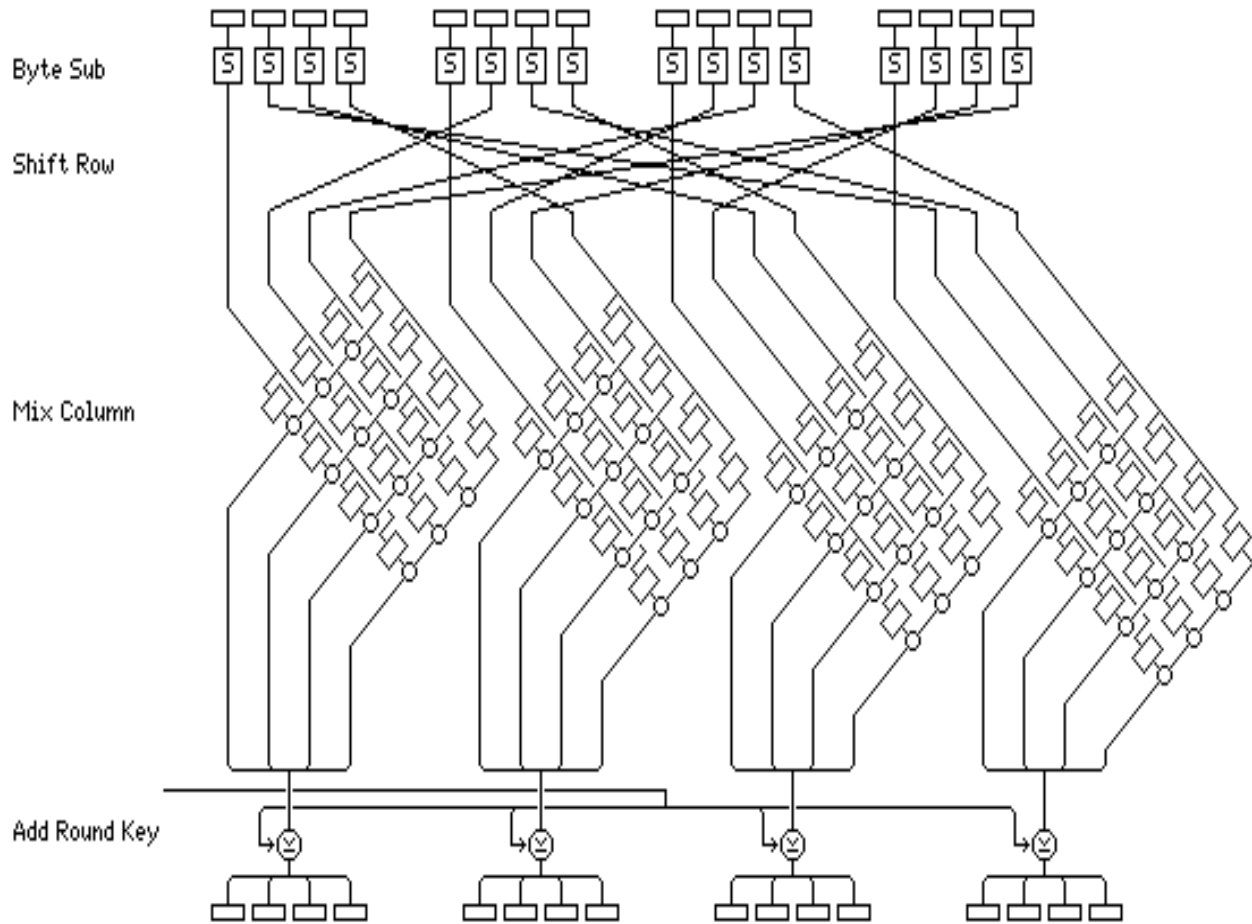


Figure 10-5. AES Encryption

The above diagram documents the encryption process for a single round. Each 128 bit data block is iterated through this structure 10 times. Each time though the round key is updated with a new value.

10.4.1.1 Byte substitution

This first part performs a non-linear byte substitution. Details of how this table is constructed can be found in section 5.1.1 of FIPS-197 available at the NIST web site. This operation is performed by module `asm_sub.v`

10.4.1.2 Shift row transformation

This section performs cyclic shifts at the byte level. Details of this operation can be found in section 5.1.2 of the FIPS-197 specification available at the NIST web site. This operation is performed by module `asm_shift.v`

10.4.1.3 Mix column(asm_mix.v)

This section performs column mixing at the byte level. Details of this operation can be found in section 5.1.3 of the FIPS-197 specification available at the NIST web site. This operation is performed by module asm_mix.v

10.4.1.4 Add round key(asm_addkey.v)

This section adds the round key. This is performed by an EXOR with the current round key with the data coming from the previous step (mix column). Details of this operation can be found in section 5.1.4 of the FIPS-197 specification available at the NIST web site. This operation is performed by module asm_addkey.v

10.4.1.5 Key expansion(asm_key.v)

The key value programmed by the software is used to generate a set of 10 keys. Each one is 128 bits. The details of this operation can be found in section 5.2 of the FIPS-197 specification. This operation is performed by module asm_key.v

10.5 Counter mode encryption

NOTE

13 clocks to complete

1. Program the key value in registers 00,04,08,0c
2. Set the bit "CTR" in register 54. The "CTR" bit is static.
3. Program the text value in registers 10,14,18,1c
4. Program the counter value in registers 20,24,28,2c
5. Set the "start" bit register 50. The "start" bit is a self clearing bit.
6. Poll for the done bit to be set to a "1", or wait for the ASM interrupt signal.
7. Read the encrypted text from registers 30,34,38,3c
8. Repeat steps 3-7 for each 128 bits block.

10.6 Message Authentication Code generation (MAC)

NOTE

13 clocks to complete

1. Program the key value in registers 00,04,08,0c
2. Set the bit "CBC" in register 54
3. Program the text value in registers 10,14,18,1c
4. Set the "start" bit register 50. The "start" bit is a self clearing bit.
5. Poll for the done bit to be set to a "1", or wait for the ASM interrupt signal.
6. Repeat steps 3-6 for each 128 bits block.

7. When all blocks have been processed, read the final MAC (authentication code) from registers 0x40, 0x44, 0x48, and 0x4c
8. The MAC accumulator should be cleared to prepare for the next payload. Writing a 1 to the clear bit will set the MAC accumulator back to all 0. The clear bit is self clearing. If for some reason the software needs to continue a MAC calculation from a previously saved value, the MAC accumulator can be pre-loaded with a initial value. The value you wish to load into the accumulator is written to registers 0x60, 0x64, 0x68, and 0x6c and then a 1 is written to the load_mac bit. The load_mac bit is self clearing.

10.7 Counter mode and Authentication mode encryption combined

NOTE

26 clocks to complete

1. Program the key value in registers 00,04,08,0c
2. Set the bit "CBC" and the "CTR" in register 54
3. Program the text value in registers 10,14,18,1c
4. Program the counter value in registers 20,24,28,2c
5. Set the "start" bit register 50. The "start" bit is a self clearing bit.
6. Poll for the done bit to be set to a "1", or wait for the ASM interrupt signal.
7. Read the encrypted text from registers 30,34,38,3c.
8. Repeat steps 3-7 for each 128 bits block.
9. When all blocks have been processed, read the final MAC (authentication code) from registers 40,44,48,4c

10.8 Signal Description

Table 10-1. Signal Properties

Signal	Input/ Output	Function	Reset State
clk	input	Clock	
rst_b	input	Asynchronous Reset	
byte_7_0	input	IP Byte Enable 7-0	
byte_15_8	input	IP Byte Enable 15-8	
byte_23_16	input	IP Byte Enable 23-16	
byte_31_24	input	IP Byte Enable 31-24	
scan_mode	input	Scan Mode	
write_b	input	IP Read (inverted)	
enable	input	IP Bus Module Enable	
address[15:0]	input	IP Address Bus	0x0000
write_data[31:0]	input	IP Write Data Bus	
read_data[31:0]	output	IP Read Data Bus	
asm_error	output	IP Transfer Error	

Table 10-1. Signal Properties (continued)

Signal	Input/Output	Function	Reset State
asm_wait	output	IP Transfer Wait	
asm_en	output	clock control	0
asm_irq	output	Interrupt	0

10.9 ASM Registers

The ASM registers, here depicted as 32-bit registers.

Table 10-2. ASM Registers (32 Bit)

Address	Register Name	Access
Base + 0x00	key0	R/W
Base + 0x04	key1	R/W
Base + 0x08	key2	R/W
Base + 0x0C	key3	R/W
Base + 0x10	data0	R/W
Base + 0x14	data1	R/W
Base + 0x18	data2	R/W
Base + 0x1C	data3	R/W
Base + 0x20	ctr0	R/W
Base + 0x24	ctr1	R/W
Base + 0x28	ctr2	R/W
Base + 0x2C	ctr3	R/W
Base + 0x30	ctr_result0	R
Base + 0x34	ctr_result1	R
Base + 0x38	ctr_result2	R
Base + 0x3C	ctr_result3	R
Base + 0x40	cbc_result0	R
Base + 0x44	cbc_result1	R
Base + 0x48	cbc_result2	R
Base + 0x4C	cbc_result3	R
Base + 0x50	control0	W (Self Clearing)
Base + 0x54	control1	R/W
Base + 0x58	status	R
Base + 0x60	mac0	R/W
Base + 0x64	mac1	R/W
Base + 0x68	mac2	R/W
Base + 0x6C	mac3	R/W

10.9.1 key0, key1, key2, key3 Registers

These four registers contain the 128 bit key used for encryption.

key0, key1, key2, key3				Key									Addr Base+0x00 Base+0x04 Base+0x08 Base+0x0C			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Data0[31:16]															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Data0[15:0]															
TYPE	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.9.2 data0,data1,data2,data3 Registers

These four registers contain the 128 data value that will be either encrypted or decrypted.

data0,data1,data2,data3				Data									Addr Base+0x10 Base+0x14 Base+0x08 Base+0x0C			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Data0[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Data0[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.9.3 ctr0, ctr1, ctr2, ctr3 Registers

These four registers contain the counter value used to encrypt the data in counter mode.

ctr0, ctr1, ctr2, ctr3				Counter									Addr Base+0x20 Base+0x24 Base+0x28 Base+0x2C			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Data0[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Data0[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.9.4 ctr_result0, ctr_result1, ctr_result2, ctr_result3 Registers

When the encryption is done, the counter mode encryption result can be read from this register.

ctr_result0, ctr_result1, ctr_result2, ctr_result3				Counter Result									Addr Base+0x30 Base+0x34 Base+0x38 Base+0x3C			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Data0[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Data0[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.9.5 cbc_result0, cbc_result1, cbc_result2, cbc_result3 Registers

When CBC-MAC generation is done, the result can be read from this register.

cbc_result0, cbc_result1, cbc_result2, cbc_result3				CBC Result									Addr Base+0x40 Base+0x44 Base+0x48 Base+0x4C			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Data0[31:16]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Data0[15:0]															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.9.6 Control 0 Register

The bits in this register are self clearing.

control0				Control 0 Register									Addr Base+0x50			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	clear_irq					LOAD_MAC	CLEAR	START								
TYPE	w	rw	rw	rw	rw	w	w	w	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

load_mac

Write "1" to pre-load MAC with a value. Writing a "0" does nothing.

clear_irq

Write "1" to clear the IRQ. Writing a "0" does nothing.

clear

Write "1" to clear all memory in ASM. Writing a "0" does nothing.

start

Write "1" to start the CBC or CTR mode encryption OR both. If self_test is set to "1" then "start" will start self test.

10.9.7 Control 1 Register

The bit in this register are not self clearing.

control1			Control 1 Register											Addr Base+0x54		
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	mask_irq					SELF TEST	CTR	CBC								
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
														bypass	mode	on
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

mask_irq

This bit control the masking of the IRQ.

1 = irq disabled.

0 = irq enabled.

self_test

Write "1" to start set mode to self test (not self clearing).

1 = Module in self test mode.

0 = Module not in self test mode.

ctr

Write 1 to put ASM in "Counter" encryption mode.

1 = Counter mode encryption will be performed.

0 = Counter mode encryption will not be performed.

cbc

Write 1 to put ASM in "CBC-MAC" generation mode.

1 = CBC-MAC generation will be performed.

0 = CBC-MAC generation will not be performed

bypass

Write 1 to put ASM in bypass mode.

- 1 = Module is in bypass mode. Data is passed through unmodified.
- 0 = Data will be encrypted.

normal_mode

Write 1 to put ASM in normal mode. By default the ASM module is in "boot" mode. In "boot" mode an internal secret key is used for all operations. This is to allow the software to use the ASM module to encrypt/decrypt the external ROM contents. Writing a "1" to the normal mode bit will change the mode from "boot" to "normal" mode. The key value can then be set by writing to the proper register. It is not possible for software to enter "boot" mode. The only way to return to "boot" is with a system reset. The boot ROM will then decide when to change mode from "boot" to "normal".

- 1 = Changes ASM module mode from "boot" to "normal".
- 0 = Does nothing.

on

Write 1 to turn the ASM module on.

- 1 = Module on.
- 0 = Module off.

To perform CTR or CBC takes 13 clocks. To perform both (both CTR and CBC set) takes 26 clocks.

10.9.8 Status Register

STATUS				Status Register									Addr Base+0x58				
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16	
	unused						TEST PASS	DONE	unused								
TYPE	rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0	
	unused																
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw	rw	rw	rw	rw	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

TEST_PASS

If this bit is "1" then the hardware passed the self test. If self test fails then the module is not usable and the outputs will be set to all "0". The default state for this bit is "0". The software must initiate a self test before the module can be used.

- 1 = self test has passed.
- 0 = self test has failed.

Done

If this bit is a "1" then the result registers can be read.

1 = The result in the CTR or CBC register is ready to be read.

0 = The result in the CTR or CBC register is not ready to be read.

10.9.9 mac0, mac1, mac2, mac3 Registers

mac0, mac1, mac2, mac3				MAC start value										Addr Base+0x60 Base+0x64 Base+0x68 Base+0x6C			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16	
	Data0[31:16]																
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0	
	Data0[15:0]																
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

This register can be used to set the starting value for CBC-MAC generation.

Chapter 11

General Purpose I/O Module

11.1 Introduction

The MC1322x has a possible 64 GPIOs. Many of these pins are shared with on-chip peripherals such as the timer, external interrupts, or communication channels. When these other modules or functions are not using these pins, they can be configured as general-purpose I/O control. Each I/O port can be configured as an input or output, has programmable pull-up or pull-down resistors, and has complete read and write capability.

11.2 Reset and Low Power GPIO Operation

The MC1322x has been designed to provide extremely low power operation (see [Section 3.3, “System Reset and Low Power GPIO Signal Connections”](#)). The device can be put into lowest power through two means:

- Hardware reset - When input RESETB is asserted low, the device is held in reset and the “off” condition (there is no internal pull-up resistor on the reset input). The only circuitry that remains powered is the POR control. The GPIO buffers have the supply removed from them to eliminate leakage. As a result, no GPIO signal should be tied or driven high while the RESETB is active. This prevents a GPIO from forward biasing the ESD diode on the pad and causing excessive leakage through the IO structure.

NOTE

If a GPIO is held or driven high during reset, no damage will be done. However, if reset is being used as a very low power state, this defeats that purpose.

- Programmed Hibernate or Doze modes - The MC1322x can also be programmed to enter Hibernate or Doze as a very low power state. Similar to reset, all the GPIO with the exception of the eight KBI signals have the supply removed from them to eliminate leakage. The KBI signal IO remain powered and in their programmed mode. Four of the KBI signals can also be used to generate a wake-up interrupt. Four of the KBI signals can be used as a control to disable peripheral circuitry that may pull a normal GPIO high during low power mode.
- The shared GPIO30-41 which are combined with the ADC_CHAN0- ADC_CHAN7, ADC_VREFH1, ADC_VREFH2, ADC_VREFL2, ADC_VREFL2 must have the input buffer enabled and the input pull-up keeper enabled to have the pads function as an input. This is unique to these shared pads.

11.3 Features

The MC1322x GPIO features include:

- 64 General-purpose IO pins
- Default to GPIO controlled as inputs with pull-down or pull-up resistors enabled (except for JTAG/NEXUS and some analog pins)
- Programmable input hysteresis
- Software controlled pull-ups or pull-downs on all pins
- Software controlled state keepers on all pins
- Logic levels specified at +/-1 mA drive current drive and 15pF loading for ac performance
- Logic levels specified at +/-5 mA drive current drive for higher low frequency loads such as LEDs
- Shared functionality with MCU peripherals or special functions
- The GPIO module/control generates no interrupts
- Eight “stay-alive” KBI signal for use during low power modes.

11.4 Pin Descriptions

The MC1322x has a total of 64 signals that can be used as parallel I/O pins. Consult [Chapter 2, “Pins and Connections”](#) for detailed information on signal names and pin assignments. All of these pins are available for general-purpose I/O when they are not used by another on-chip peripheral or function.

11.4.1 Pin Functionality

[Table 11-1](#) lists the shared functionality of the pins and where the functions are described.

Table 11-1. GPIO Shared Functionality

GPIO Pins	Alternate Function	Reference ¹
GPIO3-GPIO0	SSI	Chapter 16, “Synchronous Serial Interface Module (SSI)”
GPIO7-GPIO4	SPI	Chapter 15, “Serial Peripheral Interface Module (SPI)”
GPIO11-GPIO8	TIMER	Chapter 12, “Timer Module (TMR)”
GPIO13-GPIO12	I ² C	Chapter 14, “I2C Module”
GPIO21-GPIO14	UART	Chapter 13, “Universal Asynchronous Receiver/Transmitter Module (UART)”
GPIO29-GPIO22	KBI	Chapter 5, “System Management (Including CRM)”
GPIO41-GPIO30	ADC	Chapter 17, “Analog to Digital Converter Module (ADC)”
GPIO45-GPIO42	EXT RF CTL	Chapter 6, “MC1322x 2.4 GHz ISM Band Transceiver”
GPIO63-GPIO46	JTAG & Nexus	Chapter 18, “Development and Debug Support”

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

11.4.2 Pin Register Mappings

The 64 GPIO pins have alternative functions as described below:

- 4 JTAG I/Os reconfigurable as GPIO
- 22 GPIOs reconfigurable as interfaces to the peripherals
- 8 dedicated KBI GPIOs, 4 of which are usable as keyboard interrupts (KBIs) or external wake-up signals to the CRM
- 8 GPIOs reconfigurable as ADC analog inputs
- 4 control pins for external RF RX/TX control
- 14 Nexus pins (JTAG also)

Table 11-2 contains a listing of all the GPIOs by port number, corresponding MC1322x pin name, pin number, and functional modes.

Table 11-2. MC1322x GPIO Register Bit Mapping vs. Peripheral Function

Register X			Register X+4		
Bit	Function	Pin	Bit	Function	Pin
0	GPIO0 / SSI_TX	34	0	GPIO32 / ADC2 ³	3
1	GPIO1 / SSI_RX	33	1	GPIO33 / ADC3 ³	4
2	GPIO2 / SSI_FSYN	32	2	GPIO34 / ADC4 ³	5
3	GPIO3 / SSI_BITCK	31	3	GPIO35 / ADC5 ³	6
4	GPIO4 / SPI_SS	30	4	GPIO36 / ADC6 ³	7
5	GPIO5 / SPI_MISO	29	5	GPIO37 / ADC7_RTCK ³	8
6	GPIO6 / SPI_MOSI	28	6	GPIO38 / ADC2_VREFH ³	64
7	GPIO7 / SPI_SCK	27	7	GPIO39 / ADC2_VREFL ³	61
8	GPIO8 / TMR0	26	8	GPIO40 / ADC1_VREFH ³	63
9	GPIO9 / TMR1	25	9	GPIO41 / ADC1_VREFL ³	62
10	GPIO10 / TMR2	24	10	GPIO42 / ANT_1	56
11	GPIO11 / TMR3	23	11	GPIO43 / ANT_2	57
12	GPIO12 / I2C_SCL ¹	22	12	GPIO44 / TX_ON	52
13	GPIO13 / I2C_SDA ¹	21	13	GPIO45 / RX_ON	59
14	GPIO14 / UART1_TX	20	14	GPIO46 / TMS	12
15	GPIO15 / UART1_RX	19	15	GPIO47 / TCK	11
16	GPIO16 / UART1_CTS	18	16	GPIO48 / TDI	10
17	GPIO17 / UART1_RTS	17	17	GPIO49 / TDO	9
18	GPIO18 / UART2_TX	16	18	GPIO50 / MCKO	131
19	GPIO19 / UART2_RX	15	19	GPIO51 / MDO00	103
20	GPIO20 / UART2_CTS	14	20	GPIO52 / MDO01	102

Table 11-2. MC1322x GPIO Register Bit Mapping vs. Peripheral Function (continued)

Register X			Register X+4		
Bit	Function	Pin	Bit	Function	Pin
21	GPIO21 / UART2_RTS	13	21	GPIO53 / MDO02	112
22	GPIO22 / KBI_0_HST_WK ²	42	22	GPIO54 / MDO03	111
23	GPIO23 / KBI_1 ²	41	23	GPIO55 / MDO04	121
24	GPIO24 / KBI_2 ²	40	24	GPIO56 / MDO05	120
25	GPIO25 / KBI_3 ²	39	25	GPIO57 / MDO06	130
26	GPIO26 / KBI_4 ²	38	26	GPIO58 / MDO07	129
27	GPIO27 / KBI_5 ²	37	27	GPIO59 / MSEO0_B	114
28	GPIO28 / KBI_6 ²	36	28	GPIO60 / MSEO1_B	113
29	GPIO29 / KBI_7 ²	35	29	GPIO61 / RDY_B	122
30	GPIO30 / ADC0 ³	1	30	GPIO62 / EVTO_B	123
31	GPIO31 / ADC1 ³	2	31	GPIO63 / EVTI_B	132

¹ I2C pins are true bidirectional with open drain.

² The 8 KBI pads are always active, even during low-power modes. When they are being used as keyboard interrupts, they are controlled by the CRM module well as the GPIO

³ Analog-digital GPIO ports have their digital pads tri-stated when in analog functional modes. The digital pad is completely off.

11.5 Functional Description

Each input/output port is a single bit and highly configurable. It consists of all the control and data logic in the GPIO module port and any peripheral control associated with that bit. Figure 11-1 shows the typical block diagram of a single bit configuration.

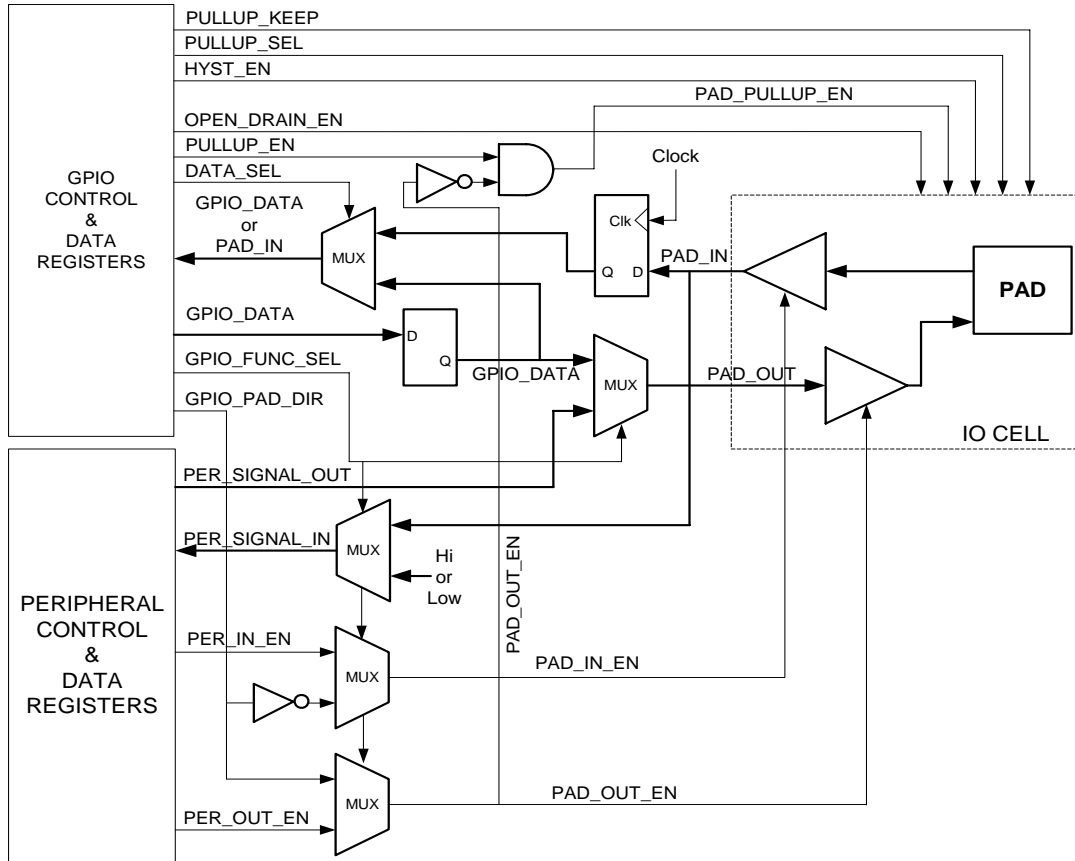


Figure 11-1. GPIO Typical Single-Bit Block Diagram.

Each single-bit port has a set of data and control register bits that determine the operation of that port. The GPIO data and control bits are organized such that all similar control or data are in a single register type. For 64 ports, two 32-bit wide registers of each type are typically used. An exception is the Function Select control which uses four registers total because the each select field is 2-bits wide. The GPIO register types are shown in Table 11-3.

Table 11-3. GPIO Register Types

Register Type	Function	Access Type
GPIO FUNCTION SELECT	Function select (1 of 4 possible)	R/W
GPIO PAD DIRECTION	Set pad direction only for GPIO modes	R/W
GPIO PAD DIRECTION SET	Used with present value of GPIO PAD DIRECTION register to set new bit values in GPIO PAD DIRECTION	W-only

Table 11-3. GPIO Register Types (continued)

Register Type	Function	Access Type
GPIO PAD DIRECTION RESET	Used with present value of GPIO PAD DIRECTION register to reset new bit values in GPIO PAD DIRECTION	W-only
GPIO DATA	Data register for GPIO modes only	R/W
GPIO DATA SET	Used with present value of GPIO DATA register to set new bit values in DATA REGISTER	W-only
GPIO DATA RESET	Used with present value of GPIO DATA register to reset new bit values in DATA REGISTER	W-only
PAD PULLUP ENABLE	Pad pull-up (pull-down) enable	R/W
PAD PULLUP SELECT	Select between a pull-up and a pull-down when pull-up is enabled	R/W
READ DATA CONTROL	Selects between GPIO DATA register and input pad for read data	R/W
PAD HYSTERESIS ENABLE	Enable input pad hysteresis	R/W
PAD KEEPER	Pad keeper enable	R/W

The GPIO block has overall control of the pads. However, the mode control will determine whether the pad is used with GPIO or a peripheral function. When the peripheral function is selected, that peripheral controls pad enables and data direction.

11.5.1 GPIO Function Select

The GPIO module has four registers dedicated to providing a 2-bit function select field for each GPIO (see [Table 11-10](#)). These control the four modes of operation: Normal Mode (default), Alternate1 Mode, Alternate2 Mode and Alternate3 Mode. [Table 11-4](#) lists the GPIO functionality versus function select for the various categories of alternative or peripheral functions.

NOTE

The default modes shown in [Table 11-4](#) occur upon exit of POR. These can be altered during the boot process, see [Section 3.9.7, “GPIO Function Upon Boot Exit”](#).

Table 11-4. General GPIO Function vs. Function Select

Category	Alternative Function	Function 0 [00] (Default)	Function 1 [01] Alternate 1	Function 2 [10] Alternate 2	Function 3 [11] Alternate 3
1	SSI	GPIO	SSI	GPIO	GPIO
2	SPI	GPIO	SPI	GPIO	GPIO
3	TIMER	GPIO	TIMER	GPIO	GPIO
4	I ² C	GPIO	I ² C	GPIO	GPIO
5	UART	GPIO	UART	GPIO	GPIO

Table 11-4. General GPIO Function vs. Function Select (continued)

Category	Alternative Function	Function 0 [00] (Default)	Function 1 [01] Alternate 1	Function 2 [10] Alternate 2	Function 3 [11] Alternate 3
6	KBI	GPIO/KBI_interrupt	GPIO/KBI_interrupt	GPIO/KBI_interrupt	GPIO/KBI_interrupt
7	ADC	GPIO except for ADC1 VREF pins and JTAG RTCK	ADC (analog) except ADC1 VREF H/L; GPIO for JTAG RTCK	GPIO	GPIO
8	EXT RF CTL	GPIO	TX control	RX control	GPIO
9	JTAG & Nexus	JTAG & Nexus	GPIO	GPIO	GPIO

The GPIO module always controls the selection and choice of pull-up or pull-down resistors and use of pad state keepers.

11.5.1.1 Normal (Function 0) Mode

Function 0 mode is the POR default mode which puts all IO to GPIO mode except the JTAG/NEXUS pins and the ADC1_VREF pins. For default:

- On exiting reset, all the GPIO except JTAG/NEXUS pins and the ADC1_VREF pins are set to inputs with input pull-down resistors enabled
- For the JTAG/NEXUS and ADC1_VREF pins, Function 0 mode is the normal non-GPIO functionality of the pins. For the JTAG/NEXUS input pins, the default is to have weak pull-ups enabled instead of pull-downs. Commonly, the JTAG or Nexus functionality is not used in the target application and the initialization of the MC1322x should reconfigure these pins.
- In GPIO Mode, the GPIO module
 - Controls the pad direction (input or output)
 - Provides the output data when driving the pad
 - Selects the source of data (pad or register) when reading the port

11.5.1.2 Alternate Modes

The function select provides three alternate modes, however they are highly redundant.

- For all GPIOs except the JTAG/NEXUS and ADC1_VREF pins, Function 1 mode can be thought of as “Peripheral Controlled Mode.” The peripheral function will control operation of the pad IF THE PERIPHERAL IS ENABLED.
- When enabled the ADC module puts its pads into analog mode and the digital pad function is totally disabled
- The external RF control (EXT RF CTL) has different functionality in Function 1 versus Function 2
- The peripheral module supplies the output enable to the I/O pad to control its direction, and the pad output data if its output enable is asserted. The peripheral can read the value of the data on the I/O pad if it is using the pin as an input.

11.5.2 Pull-up/Pull-down Resistors

Each GPIO pad has a selectable pull-up/pull-down resistor.

- The use of a pull-up/pull-down is selectable via the PAD PULLUP ENABLE registers
- The use of a pull-up versus a pull-down is selectable via the PAD PULLUP SELECT registers
- The pull-up/pull-down is only enabled when the pad is an input. It is disabled when the pad is programmed as an output.
- For a peripheral function, the pull-up/pull-down is selected via the GPIO module. If selected the pull-up/pull-down is active for a GPIO or peripheral input
- The POR GPIO default is as pull-down except for as noted in [Table 11-13](#).
- Typical values are 50 Kohms for pull-ups and pull-downs

[Table 11-5](#) shows the truth table for the pull-up/pull-down enable function.

Table 11-5. GPIO Pull-up/Pull-down Configuration

Pull-up En	Pad Output Enable	Pull-ups
0	0	disabled
1	0	enabled
X	1	disabled

11.5.3 Input Hysteresis

Each GPIO pad has a selectable input hysteresis function

- The use of hysteresis is selectable via the PAD HYSTERESIS ENABLE registers
- For a peripheral function, the input hysteresis is also selected via the GPIO module. If selected the hysteresis is active for a GPIO or peripheral input
- The POR GPIO default is with hysteresis disabled.

11.5.4 Pad Keeper Function

Each GPIO pad has a selectable pad keeper function

- The pad keeper will retain the last state held on the pad (when used as an input).
- The use of the pad keeper is selectable via the PAD KEEPER registers
- For a peripheral function, the pad keeper is selected via the GPIO module. If selected the keeper is active for the GPIO or peripheral input
- The POR GPIO default is with pad keeper disabled.

11.5.5 Port I/O Control

The GPIO module has primary control of the IO port. As stated in the previous sections the pull-up/pull-down, hysteresis, and pad keeper functions are controlled from the GPIO module independent of the function select state (operating mode). When GPIO mode is enabled the port I/O control uses the following functionality

11.5.5.1 GPIO Pad Direction Control

Each GPIO pad has a pad direction control

- The pad direction is determined by the GPIO PAD DIRECTION registers
- The direction defaults to inputs enabled
- The PAD DIRECTION registers can be accessed directly for read and write.
- The PAD DIRECTION registers can also be modified to set or reset specific IO
 - For each PAD DIRECTION register, there is a write-only GPIO PAD DIRECTION SET register. Writing a “1” to any bit in that register will set the corresponding bit in the PAD DIRECTION register and retain all other 1’s previously set in the PAD DIRECTION register.
 - For each PAD DIRECTION register, there is a write-only GPIO PAD DIRECTION RESET register. Writing a “1” to any bit in that register will reset (set to “0”) the corresponding bit in the PAD DIRECTION register and retain all other 0’s previously set in the PAD DIRECTION register.
- If the function select for a given IO does not enable GPIO, pad direction reverts to the assigned peripheral function

11.5.5.2 GPIO Data Control

Each GPIO pad has pad data control

- If the pad direction is selected as an output the data state of the pad is driven by the corresponding bit of the GPIO DATA register
 - The GPIO DATA registers default to all zeroes.
 - The GPIO DATA registers can be accessed directly for write.
 - The GPIO DATA registers can also be modified to set or reset specific IO
 - For each GPIO DATA register, there is a write-only GPIO DATA SET register. Writing a “1” to any bit in that register will set the corresponding bit in the GPIO DATA register and retain all other 1’s previously set in the GPIO DATA register.
 - For each GPIO DATA register, there is a write-only GPIO DATA RESET register. Writing a “1” to any bit in that register will reset (set to “0”) the corresponding bit in the GPIO DATA register and retain all other 0’s previously set in the PAD DIRECTION register.
- If the pad direction is selected as an input, data can be read from either the pad or the GPIO DATA register.
 - The corresponding READ DATA CONTROL register will determine for each IO whether an access of the GPIO DATA register will fetch a read of the stored internal DATA REGISTER bit or the pad input register.

- The default condition on the READ DATA CONTROL register is to read the pads.

11.5.5.3 Peripheral Port Control

When not in GPIO mode, the pad direction control comes from the peripheral, and input and output data are interfaced with the peripheral port.

- When in GPIO Mode, the data inputs into the peripheral modules are driven high (except for SSI, whose peripheral inputs need to be 0 for non-peripheral and non-master modes) and the output data enables from the peripherals are ignored.
- As stated previously pull-up/pull-down, hysteresis, and keepers are determined by the GPIO module while the port is in peripheral mode.
- The I²C port is a special case
 - Both the input pad and output pad are enabled for I²C operation
 - I²C is the only case when the output pad is enabled for open-drain operation
 - Pull-ups are always disabled for I²C pins in I²C operation

11.5.6 Special Signals and Conditions

There are special conditions that apply based on power management and pin usage

- If the hardware reset is active, the MC1322x is in the lowest power, reset condition. The GPIO pad power is disconnected; GPIO signals should not be driven high.
- On release from reset all GPIO revert to default conditions (JTAG/Nexus signals active). All GPIO are forced to inputs with pull-downs activated (except for JTAG/NEXUS pins)
- The 8 signals designated as KBI7-KBI0 are a special case

NOTE

See [Section 5.9.2, “Wake-up Control \(WU_CNTL\)”](#) for detailed information on control and use of the KBI signals.

- When the MC1322x is “awake”, the GPIO module is always in control. However, the KBI7-KBI4 interrupt request capability is always available and is controlled by the CRM and can be used as external interrupt requests. The GPIO module must configure the GPIO for KBI7-KBI4 as an input(s) with a pull-up/pull-down for the interrupt request to be used.
- When Hibernate or Doze low power mode is activated, all the KBI signals come under control of the CRM module
 - KBI3-KBI0 are configured as outputs. Out-of-default (POR), they are forced to the “1” state. They can be programmed for state in the WU_CNTL Register.
 - KBI0 can be enabled as an “out-of-wake-up” indicator.
 - KBI7-KBI4 are configured as inputs with pull-downs/pullups enabled. Out-of-default, they are forced to pull-downs enabled. A pull-up or pull-down is always enabled depending on the programmed interrupt request active high or active low.
 - KBI7-KBI4 can be enabled as asynchronous, “wake-up” interrupt requests.

11.6 GPIO Module Register Memory Map

The GPIO module is programmed via a set of memory-mapped registers

- The base address is 0x8000_0000
- The register memory map for the module is listed in [Table 11-6](#)
- The registers should only be accessed as 32 bits

Table 11-6. GPIO Module Register Memory Map

Address	Name	Access Type	Access Width
Base + 0x00	GPIO Pad Direction for GPIO 00-31 (GPIO_PAD_DIR0)	R/W	32 only
Base + 0x04	GPIO Pad Direction for GPIO 32-63 (GPIO_PAD_DIR1)	R/W	32 only
Base + 0x08	GPIO Data for GPIO 00-31 (GPIO_DATA0)	R/W	32 only
Base + 0x0C	GPIO Data for GPIO 32-63 (GPIO_DATA1)	R/W	32 only
Base + 0x10	GPIO Pad Pull-up Enable for GPIO 00-31 (GPIO_PAD_PU_EN0)	R/W	32 only
Base + 0x14	GPIO Pad Pull-up Enable for GPIO 32-63 (GPIO_PAD_PU_EN1)	R/W	32 only
Base + 0x18	GPIO Function Select for GPIO 00-15 (GPIO_FUNC_SEL0)	R/W	32 only
Base + 0x1C	GPIO Function Select for GPIO 16-31 (GPIO_FUNC_SEL1)	R/W	32 only
Base + 0x20	GPIO Function Select for GPIO 32-47 (GPIO_FUNC_SEL2)	R/W	32 only
Base + 0x24	GPIO Function Select for GPIO 48-63 (GPIO_FUNC_SEL3)	R/W	32 only
Base + 0x28	GPIO Data Select for GPIO 00-31 (GPIO_DATA_SEL0)	R/W	32 only
Base + 0x2C	GPIO Data Select for GPIO 32-63 (GPIO_DATA_SEL1)	R/W	32 only
Base + 0x30	GPIO Pad Pull-up Select for GPIO 00-31 (GPIO_PAD_PU_SEL0)	R/W	32 only
Base + 0x34	GPIO Pad Pull-up Select for GPIO 32-63 (GPIO_PAD_PU_SEL1)	R/W	32 only
Base + 0x38	GPIO Pad Hysteresis Enable for GPIO 00-31 (GPIO_PAD_HYST_EN0)	R/W	32 only
Base + 0x3C	GPIO Pad Hysteresis Enable for GPIO 32-63 (GPIO_PAD_HYST_EN1)	R/W	32 only
Base + 0x40	GPIO Pad Keeper Enable for GPIO 00-31 (GPIO_PAD_KEEP0)	R/W	32 only
Base + 0x44	GPIO Pad Keeper Enable for GPIO 32-63 (GPIO_PAD_KEEP1)	R/W	32 only
Base + 0x48	GPIO Data Set for GPIO 00-31 (GPIO_DATA_SET0)	W	32 only
Base + 0x4C	GPIO Data Set for GPIO 32-63 (GPIO_DATA_SET1)	W	32 only
Base + 0x50	GPIO Data Reset for GPIO 00-31 (GPIO_DATA_RESET0)	W	32 only
Base + 0x54	GPIO Data Reset for GPIO 32-63 (GPIO_DATA_RESET1)	W	32 only
Base + 0x58	GPIO Pad Direction Set for GPIO 00-31 (GPIO_PAD_DIR_SET0)	W	32 only
Base + 0x5C	GPIO Pad Direction Set for GPIO 32-63 (GPIO_PAD_DIR_SET1)	W	32 only
Base + 0x60	GPIO Pad Direction Reset for GPIO 00-31 (GPIO_PAD_DIR_RESET0)	W	32 only
Base + 0x64	GPIO Pad Direction Reset for GPIO 32-63 (GPIO_PAD_DIR_RESET1)	W	32 only

11.7 GPIO Module Registers and Control Bits

The following sections describe the functional registers in the GPIO module.

11.7.1 GPIO Pad Direction Registers (GPIO_PAD_DIR0 and GPIO_PAD_DIR1)

The GPIO Pad Direction Registers control the direction of the IO when the GPIO mode is selected. Two 32-bit registers (GPIO_PAD_DIR0 and GPIO_PAD_DIR1) are required to control the 64 possible GPIO.

GPIO_PAD_DIR0				FOR GPIO MODES ONLY - GPIOs GPIO_00 - GPIO_31									Addr Base+0x00			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x00	GPIO_31	GPIO_30	GPIO_29	GPIO_28	GPIO_27	GPIO_26	GPIO_25	GPIO_24	GPIO_23	GPIO_22	GPIO_21	GPIO_20	GPIO_19	GPIO_18	GPIO_17	GPIO_16
RESET	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x00	GPIO_15	GPIO_14	GPIO_13	GPIO_12	GPIO_11	GPIO_10	GPIO_09	GPIO_08	GPIO_07	GPIO_06	GPIO_05	GPIO_04	GPIO_03	GPIO_02	GPIO_01	GPIO_00
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GPIO_PAD_DIR1				FOR GPIO MODES ONLY - GPIOs GPIO_32 - GPIO_63									Addr Base+0x00-0x04			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x04	GPIO_63	GPIO_62	GPIO_61	GPIO_60	GPIO_59	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	GPIO_53	GPIO_52	GPIO_51	GPIO_50	GPIO_49	GPIO_48
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x04	GPIO_47	GPIO_46	GPIO_45	GPIO_44	GPIO_43	GPIO_42	GPIO_41	GPIO_40	GPIO_39	GPIO_38	GPIO_37	GPIO_36	GPIO_35	GPIO_34	GPIO_33	GPIO_32
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-7. GPIO Direction Register Bit Descriptions

Bit Number	Description	Operation
0-31	GPIO_PAD_DIRx[31:0] - These bits control the directions of the pins when in GPIO Mode only. In Peripheral Mode, these bits have no effect on the output enables or pull-up enables (for GPIOs which have no Peripheral mode, the bits always take effect).	0 (default) = Pin is an input. Pull-ups are dependent on the value of the GPIO_PAD_PU_EN registers. Exceptions: GPIOs 22-25 (KBI0-3). 1 = Pin is an output; pull-ups are disabled.

11.7.2 GPIO Data Registers (GPIO_DATA1 and GPIO_DATA0)

The GPIO Data Registers supply the data when the output is enabled and GPIO mode is selected. Two 32-bit registers (GPIO_DATA0 and GPIO_DATA1) are required to control the 64 possible GPIO.

GPIO_DATA0				FOR GPIO MODES ONLY - GPIOs GPIO_00 - GPIO_31									Addr Base+0x08			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x08	GPIO_31	GPIO_30	GPIO_29	GPIO_28	GPIO_27	GPIO_26	GPIO_25	GPIO_24	GPIO_23	GPIO_22	GPIO_21	GPIO_20	GPIO_19	GPIO_18	GPIO_17	GPIO_16
RESET	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x08	GPIO_15	GPIO_14	GPIO_13	GPIO_12	GPIO_11	GPIO_10	GPIO_09	GPIO_08	GPIO_07	GPIO_06	GPIO_05	GPIO_04	GPIO_03	GPIO_02	GPIO_01	GPIO_00
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GPIO_DATA1				FOR GPIO MODES ONLY - GPIOs GPIO_32 - GPIO_63									Addr Base+0x0C			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x0C	GPIO_63	GPIO_62	GPIO_61	GPIO_60	GPIO_59	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	GPIO_53	GPIO_52	GPIO_51	GPIO_50	GPIO_49	GPIO_48
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x0C	GPIO_47	GPIO_46	GPIO_45	GPIO_44	GPIO_43	GPIO_42	GPIO_41	GPIO_40	GPIO_39	GPIO_38	GPIO_37	GPIO_36	GPIO_35	GPIO_34	GPIO_33	GPIO_32
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-8. GPIO Data Register Bit Descriptions

Bit Number	Description	Operation
0-31	GPIO_DATA1x[31:0] - These bits contain the output data for the GPIOs when in GPIO Mode.	Default = 0 except GPIOs 22-25 (KBI0-3)

11.7.3 GPIO Pull-up Enable Registers (GPIO_PAD_PU_EN1 and GPIO_PAD_PU_EN0)

The GPIO pull-up enable registers select if a pull-up or pull-down is enabled if the pad is an input (for all modes). A pull-up or pull-down can be selected via the PAD_PULLUP_SELECT registers. Two 32-bit registers (GPIO_PAD_PU_EN0 and GPIO_PAD_PU_EN1) are required to control the 64 possible GPIO.

GPIO_PAD_PU_EN0				GPIOs GPIO_00 - GPIO_31									Addr Base+0x10			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x10	GPIO_31	GPIO_30	GPIO_29	GPIO_28	GPIO_27	GPIO_26	GPIO_25	GPIO_24	GPIO_23	GPIO_22	GPIO_21	GPIO_20	GPIO_19	GPIO_18	GPIO_17	GPIO_16
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x10	GPIO_15	GPIO_14	GPIO_13	GPIO_12	GPIO_11	GPIO_10	GPIO_09	GPIO_08	GPIO_07	GPIO_06	GPIO_05	GPIO_04	GPIO_03	GPIO_02	GPIO_01	GPIO_00
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
GPIO_PAD_PU_EN1				GPIOs GPIO_32 - GPIO_63									Addr Base+0x14			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x14	GPIO_63	GPIO_62	GPIO_61	GPIO_60	GPIO_59	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	GPIO_53	GPIO_52	GPIO_51	GPIO_50	GPIO_49	GPIO_48
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x14	GPIO_47	GPIO_46	GPIO_45	GPIO_44	GPIO_43	GPIO_42	GPIO_41	GPIO_40	GPIO_39	GPIO_38	GPIO_37	GPIO_36	GPIO_35	GPIO_34	GPIO_33	GPIO_32
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 11-9. GPIO Pull-up Enable Register Bit Descriptions

Bit Number	Description	Operation
0-31	GPIO_PAD_PU_ENx[31:0] - These bits control whether pull-ups/pull-downs are enabled for inputs in the various functional modes. Pull-ups/pull-downs are automatically disabled for outputs in these modes.	0 = Pull-ups/pull-downs disabled for inputs 1 = (Default) Pull-ups/pull-downs enabled for inputs

11.7.4 GPIO Function Select Registers (GPIO_FUNC_SEL3, GPIO_FUNC_SEL2, GPIO_FUNC_SEL1, and GPIO_FUNC_SEL0)

The GPIO Function Select Registers provide a 2-bit function select field for each GPIO. The 2-bit field selects one of 4 functions for each GPIO. Four 32-bit registers (GPIO_FUNC_SEL0, GPIO_FUNC_SEL1, GPIO_FUNC_SEL2, and GPIO_FUNC_SEL3) are required to control the 64 possible GPIO.

GPIO_FUNC_SEL0				GPIOs GPIO_15 - GPIO_0												Addr Base+0x18			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16			
Base+0x18	GPIO_15			GPIO_14		GPIO_13		GPIO_12		GPIO_11		GPIO_10		GPIO_9		GPIO_8			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0			
Base+0x18	GPIO_7			GPIO_6		GPIO_5		GPIO_4		GPIO_3		GPIO_2		GPIO_1		GPIO_0			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
GPIO_FUNC_SEL1				GPIOs GPIO_31 - GPIO_16												Addr Base+0x1C			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16			
Base+0x1C	GPIO_31			GPIO_30		GPIO_28		GPIO_28		GPIO_27		GPIO_26		GPIO_25		GPIO_24			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0			
Base+0x1C	GPIO_23			GPIO_22		GPIO_21		GPIO_20		GPIO_19		GPIO_18		GPIO_17		GPIO_16			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

GPIO_FUNC_SEL2				GPIOs GPIO_47 - GPIO_32									Addr Base+0x20			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x20	GPIO_47		GPIO_46		GPIO_45		GPIO_44		GPIO_43		GPIO_42		GPIO_41		GPIO_40	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x20	GPIO_39		GPIO_38		GPIO_37		GPIO_36		GPIO_35		GPIO_34		GPIO_33		GPIO_32	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GPIO_FUNC_SEL3				GPIOs GPIO_63 - GPIO_48									Addr Base+0x24			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x24	GPIO_63		GPIO_62		GPIO_61		GPIO_60		GPIO_59		GPIO_58		GPIO_57		GPIO_56	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x24	GPIO_55		GPIO_54		GPIO_53		GPIO_52		GPIO_51		GPIO_50		GPIO_49		GPIO_48	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-10. GPIO Functional Mode Select Register Bit Descriptions

Bit Number	Description	Operation
For each GPIO	GPIO_FUNC_SEL[1:0] - There is a 2-bit field for each GPIO that selects function	0 = Functional Mode 0 (default) 1 = Alternate Mode 1 (Function 1) 2 = Alternate Mode 2 (Function 2) 3 = Alternate Mode 3 (Function 3) The Functional Modes are different for each grouping of GPIOs. See Table 11-11 for a complete listing of GPIO port assignments and functional modes.

Table 11-11. Pin Function vs. Function Select State

GPIO PORT	Chip Pin Name	Functional 0 (Default)		Functional 1		Functional 2		Functional 3	
		FUNC_SEL = 0	I/O	FUNC_SEL = 1	I/O	FUNC_SEL = 2	I/O	FUNC_SEL = 3	I/O
PERIPHERAL CONNECTIONS									
GPIO_00	SSI_TX	GPIO_00	I/O	SSI_TX	Tri-O	GPIO_00	I/O	GPIO_00	I/O
GPIO_01	SSI_RX	GPIO_01	I/O	SSI_RX	I	GPIO_01	I/O	GPIO_01	I/O
GPIO_02	SSI_FSYNC	GPIO_02	I/O	SSI_MST_FSYNC SSI_SLV_FSYNC	Tri-O I	GPIO_02	I/O	GPIO_02	I/O
GPIO_03	SSI_BITCLK	GPIO_03	I/O	SSI_MST_BITCLK SSI_SLV_BITCLK	Tri-O I	GPIO_03	I/O	GPIO_03	I/O
GPIO_04	SPI_SS	GPIO_04	I/O	SPI_SS	I Tri-O	GPIO_04	I/O	GPIO_04	I/O
GPIO_05	SPI_MISO	GPIO_05	I/O	SPI_MISO	Tri-O I	GPIO_05	I/O	GPIO_05	I/O
GPIO_06	SPI_MOSI	GPIO_06	I/O	SPI_MOSI	I Tri-O	GPIO_06	I/O	GPIO_06	I/O
GPIO_07	SPI_SCK	GPIO_07	I/O	SPI_SCK	I Tri-O	GPIO_07	I/O	GPIO_07	I/O
GPIO_08	TMR0	GPIO_08	I/O	TMR0_IN TMR0_OUT	I O	GPIO_08	I/O	GPIO_08	I/O
GPIO_09	TMR1	GPIO_09	I/O	TMR1_IN TMR1_OUT	I O	GPIO_09	I/O	GPIO_09	I/O
GPIO_10	TMR2	GPIO_10	I/O	TMR2_IN TMR2_OUT	I O	GPIO_10	I/O	GPIO_10	I/O
GPIO_11	TMR3	GPIO_11	I/O	TMR3_IN TMR3_OUT	I O	GPIO_11	I/O	GPIO_11	I/O
GPIO_12	I2C_SCL	GPIO_12	I/O	I2C_SCL	I/O*	GPIO_12	I/O	GPIO_12	I/O
GPIO_13	I2C_SDA	GPIO_13	I/O	I2C_SDA	I/O*	GPIO_13	I/O	GPIO_13	I/O
GPIO_14	UART1_TX	GPIO_14	I/O	UART1_TX	Tri-O	GPIO_14	I/O	GPIO_14	I/O
GPIO_15	UART1_RX	GPIO_15	I/O	UART1_RX	I	GPIO_15	I/O	GPIO_15	I/O
GPIO_16	UART1_CTS	GPIO_16	I/O	UART1_CTS	O	GPIO_16	I/O	GPIO_16	I/O
GPIO_17	UART1_RTS	GPIO_17	I/O	UART1_RTS	I	GPIO_17	I/O	GPIO_17	I/O
GPIO_18	UART2_TX	GPIO_18	I/O	UART2_TX	Tri-O	GPIO_18	I/O	GPIO_18	I/O
GPIO_19	UART2_RX	GPIO_19	I/O	UART2_RX	I	GPIO_19	I/O	GPIO_19	I/O
GPIO_20	UART2_CTS	GPIO_20	I/O	UART2_CTS	O	GPIO_20	I/O	GPIO_20	I/O
GPIO_21	UART2_RTS	GPIO_21	I/O	UART2_RTS	I	GPIO_21	I/O	GPIO_21	I/O
DEDICATED GPIO CONNECTIONS									
GPIO_22	KBI_0**	GPIO_22	I/O	GPIO_22	I/O	GPIO_22	I/O	GPIO_22	I/O
GPIO_23	KBI_1**	GPIO_23	I/O	GPIO_23	I/O	GPIO_23	I/O	GPIO_23	I/O
GPIO_24	KBI_2**	GPIO_24	I/O	GPIO_24	I/O	GPIO_24	I/O	GPIO_24	I/O
GPIO_25	KBI_3**	GPIO_25	I/O	GPIO_25	I/O	GPIO_25	I/O	GPIO_25	I/O
GPIO_26	KBI_4**	GPIO_26/ KBI_4_Wake_Int	I/O	GPIO_26/ KBI_4_Wake_Int	I	GPIO_26/ KBI_4_Wake_Int	I/O	GPIO_26/ KBI_4_Wake_Int	I/O
GPIO_27	KBI_5**	GPIO_27/ KBI_5_Wake_Int	I/O	GPIO_27/ KBI_5_Wake_Int	I	GPIO_27/ KBI_5_Wake_Int	I/O	GPIO_27/ KBI_5_Wake_Int	I/O
GPIO_28	KBI_6**	GPIO_28/ KBI_6_Wake_Int	I/O	GPIO_28/ KBI_6_Wake_Int	I	GPIO_28/ KBI_6_Wake_Int	I/O	GPIO_28/ KBI_6_Wake_Int	I/O
GPIO_29	KBI_7**	GPIO_29/ KBI_7_Wake_Int	I/O	GPIO_29/ KBI_7_Wake_Int	I	GPIO_29/ KBI_7_Wake_Int	I/O	GPIO_29/ KBI_7_Wake_Int	I/O

Table 11-11. Pin Function vs. Function Select State (continued)

GPIO PORT	Chip Pin Name	Functional 0 (Default)		Functional 1		Functional 2		Functional 3	
ADC CONNECTIONS									
GPIO_30	ADC_Chan0***	GPIO_30	I/O	ADC Analog IP (Ch. 0)	Tri	GPIO_30	I/O	GPIO_30	I/O
GPIO_31	ADC_Chan1***	GPIO_31	I/O	ADC Analog IP (Ch. 1)	Tri	GPIO_31	I/O	GPIO_31	I/O
GPIO_32	ADC_Chan2***	GPIO_32	I/O	ADC Analog IP (Ch. 2)	Tri	GPIO_32	I/O	GPIO_32	I/O
GPIO_33	ADC_Chan3***	GPIO_33	I/O	ADC Analog IP (Ch. 3)	Tri	GPIO_33	I/O	GPIO_33	I/O
GPIO_34	ADC_Chan4***	GPIO_34	I/O	ADC Analog IP (Ch. 4)	Tri	GPIO_34	I/O	GPIO_34	I/O
GPIO_35	ADC_Chan5***	GPIO_35	I/O	ADC Analog IP (Ch. 5)	Tri	GPIO_35	I/O	GPIO_35	I/O
GPIO_36	ADC_Chan6***	GPIO_36	I/O	ADC Analog IP (Ch. 6)	Tri	GPIO_36	I/O	GPIO_36	I/O
GPIO_37	ADC_Chan7***	JTAG RTCK	O	ADC Analog IP (Ch. 7)	Tri	GPIO_37	I/O	GPIO_37	I/O
GPIO_38	ADC2_VrefH***	GPIO_38	I/O	ADC2_VrefH	Tri	GPIO_38	I/O	GPIO_38	I/O
GPIO_39	ADC2_VrefL***	GPIO_39	I/O	ADC2_VrefL	Tri	GPIO_39	I/O	GPIO_39	I/O
GPIO_40	ADC1_VrefH***	ADC1_VrefH	Tri	GPIO_40	I/O	GPIO_40	I/O	GPIO_40	I/O
GPIO_41	ADC1_VrefL***	ADC1_VrefL	Tri	GPIO_4	I/O	GPIO_41	I/O	GPIO_41	I/O
CONTROL PINS									
GPIO_42	TxRx_ANT_1	GPIO_42	I/O	TRX Seq Mgr GPO1 (antenna sw1)	O	GPIO_42	-	GPIO_42	I/O
GPIO_43	TxRx_ANT_2	GPIO_43	I/O	TRX Seq Mgr GPO2 (antenna sw2)	O	GPIO_43	I/O	GPIO_43	I/O
GPIO_44	TX_ON	GPIO_44	I/O	TX Seq Mgr GPO1 (PA)	O	RX Seq Mgr GPO1 (LNA)	O	GPIO_44	I/O
GPIO_45	RX_ON	GPIO_45	I/O	TX Seq Mgr GPO2	O	RX Seq Mgr GPO2	O	GPIO_45	I/O
JTAG CONNECTIONS (4)									
GPIO_46	TMS	JTA_TMS	I	GPIO_46	I/O	GPIO_46	I/O	GPIO_46	I/O
GPIO_47	TCK	JTA_TCK	I	GPIO_47	I/O	GPIO_47	I/O	GPIO_47	I/O
GPIO_48	TDI	JTA_TDI	I	GPIO_48	I/O	GPIO_48	I/O	GPIO_48	I/O
GPIO_49	TDO	JTA_TDO	O	GPIO_49	I/O	GPIO_49	I/O	GPIO_49	I/O
NEXUS PINS									
GPIO_50	MCKO	MCKO Msg Clk	O	GPIO_50	I/O	GPIO_50	I/O	GPIO_50	I/O
GPIO_51	MDO0	Msg Data Out	O	GPIO_51	I/O	GPIO_51	I/O	GPIO_51	I/O
GPIO_52	MDO1	Msg Data Out	O	GPIO_52	I/O	GPIO_52	I/O	GPIO_52	I/O
GPIO_53	MDO2	Msg Data Out	O	GPIO_53	I/O	GPIO_53	I/O	GPIO_53	I/O
GPIO_54	MDO3	Msg Data Out	O	GPIO_54	I/O	GPIO_54	I/O	GPIO_54	I/O
GPIO_55	MDO4	Msg Data Out	O	GPIO_55	I/O	GPIO_55	I/O	GPIO_55	I/O
GPIO_56	MDO5	Msg Data Out	O	GPIO_56	I/O	GPIO_56	I/O	GPIO_56	I/O
GPIO_57	MDO6	Msg Data Out	O	GPIO_57	I/O	GPIO_57	I/O	GPIO_57	I/O
GPIO_58	MDO7	Msg Data Out	O	GPIO_58	I/O	GPIO_58	I/O	GPIO_58	I/O

Table 11-11. Pin Function vs. Function Select State (continued)

GPIO PORT	Chip Pin Name	Functional 0 (Default)		Functional 1		Functional 2		Functional 3	
GPIO_59	MSEO0_B	Msg Start/End	O	GPIO_59	I/O	GPIO_59	I/O	GPIO_59	I/O
GPIO_60	MSEO1_B	Msg Start/End	O	GPIO_60	I/O	GPIO_60	I/O	GPIO_60	I/O
GPIO_61	RDY_B	Nexus Ready	O	GPIO_61	I/O	GPIO_61	I/O	GPIO_61	I/O
GPIO_62	EVTO_B	Event Out	O	GPIO_62	I/O	GPIO_62	I/O	GPIO_62	I/O
GPIO_63	EVTI_B	Event In	I	GPIO_63	I/O	GPIO_63	I/O	GPIO_63	I/O

11.7.5 GPIO Data Select Registers (GPIO_DATA_SEL1 and GPIO_DATA_SEL0)

The GPIO Data Select Registers determine the read data source for each GPIO when a read of the GPIO Data Register (GPIO_DATA1 and GPIO_DATA0) occurs. The data can be read from the pad or the GPIO Data Register. Two 32-bit registers (GPIO_DATA_SEL1 and GPIO_DATA_SEL0) are required to control the 64 possible GPIO.

GPIO_DATA_SEL0				GPIOs GPIO_00 - GPIO_31												Addr Base+0x28			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16			
Base+0x28	GPIO_31	GPIO_30	GPIO_29	GPIO_28	GPIO_27	GPIO_26	GPIO_25	GPIO_24	GPIO_23	GPIO_22	GPIO_21	GPIO_20	GPIO_19	GPIO_18	GPIO_17	GPIO_16			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0			
Base+0x28	GPIO_15	GPIO_14	GPIO_13	GPIO_12	GPIO_11	GPIO_10	GPIO_09	GPIO_08	GPIO_07	GPIO_06	GPIO_05	GPIO_04	GPIO_03	GPIO_02	GPIO_01	GPIO_00			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
GPIO_DATA_SEL1				GPIOs GPIO_32 - GPIO_63												Addr Base+0x2C			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16			
Base+0x2C	GPIO_63	GPIO_62	GPIO_61	GPIO_60	GPIO_59	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	GPIO_53	GPIO_52	GPIO_51	GPIO_50	GPIO_49	GPIO_48			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0			
Base+0x2C	GPIO_47	GPIO_46	GPIO_45	GPIO_44	GPIO_43	GPIO_42	GPIO_41	GPIO_40	GPIO_39	GPIO_38	GPIO_37	GPIO_36	GPIO_35	GPIO_34	GPIO_33	GPIO_32			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 11-12. GPIO Read Control Register Bit Descriptions

Bit Number	Description	Operation
0-31	GPIO_DATA_SELx[31:0] - Each bit controls whether data is read directly from the GPIO pad, or from the GPIO Data Register	0 = (Default) Data is read from the pad. 1 = Data is read from the GPIO Data Register

11.7.6 GPIO Pad Pull-up Select (GPIO_PAD_PU_SEL1 and GPIO_PAD_PU_SEL0)

The GPIO Pad Pull-up Select Registers determine whether a pull-up versus a pull-down resistor is used if the pad is an input and if the pull-up/pull-down is enabled via the GPIO pull-up enable registers. Two 32-bit registers (GPIO_PAD_PU_SEL1 and GPIO_PAD_PU_SEL0) are required to control the 64 possible GPIO.

GPIO_PAD_PU_SEL0				GPIOs GPIO_00 - GPIO_31									Addr Base+0x30			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x30	GPIO_31	GPIO_30	GPIO_29	GPIO_28	GPIO_27	GPIO_26	GPIO_25	GPIO_24	GPIO_23	GPIO_22	GPIO_21	GPIO_20	GPIO_19	GPIO_18	GPIO_17	GPIO_16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x30	GPIO_15	GPIO_14	GPIO_13	GPIO_12	GPIO_11	GPIO_10	GPIO_09	GPIO_08	GPIO_07	GPIO_06	GPIO_05	GPIO_04	GPIO_03	GPIO_02	GPIO_01	GPIO_00
RESET	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
GPIO_PAD_PU_SEL1				GPIOs GPIO_32 - GPIO_63									Addr Base+-0x34			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x34	GPIO_63	GPIO_62	GPIO_61	GPIO_60	GPIO_59	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	GPIO_53	GPIO_52	GPIO_51	GPIO_50	GPIO_49	GPIO_48
RESET	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x34	GPIO_47	GPIO_46	GPIO_45	GPIO_44	GPIO_43	GPIO_42	GPIO_41	GPIO_40	GPIO_39	GPIO_38	GPIO_37	GPIO_36	GPIO_35	GPIO_34	GPIO_33	GPIO_32
RESET	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-13. GPIO PAD Pull-up Select Register Bit Descriptions

Bit Number	Description	Operation
0-31	GPIO_PAD_PU_SELx[31:0] - Each bit controls the selection of pull-up versus pull-down for an output pad. The corresponding GPIO_PAD_PU_EN bit must be asserted to activate either weak pull device.	<p>0 = Weak pull-down (default for all except GPIOs 12-13 (I2C), 63 (EVTI_B), 46 (TMS), 47 (TCK) and 48 (TDI))</p> <p>1 = Weak pull-up (default for GPIOs 12-13 (I2C), 63 (EVTI_B), 46 (TMS), 47 (TCK) and 48 (TDI))</p>

11.7.7 GPIO Pad Hysteresis Enable Registers (GPIO_PAD_HYST_EN1 and GPIO_PAD_HYST_EN0)

The GPIO Pad Hysteresis Enable Registers determine whether hysteresis is used if the pad is an input. Two 32-bit registers (GPIO_PAD_HYST_EN1 and GPIO_PAD_HYST_EN0) are required to control the 64 possible GPIO.

GPIO_PAD_HYST_EN0				GPIOs GPIO_00 - GPIO_31								Addr Base+0x38				
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x38	GPIO_31	GPIO_30	GPIO_29	GPIO_28	GPIO_27	GPIO_26	GPIO_25	GPIO_24	GPIO_23	GPIO_22	GPIO_21	GPIO_20	GPIO_19	GPIO_18	GPIO_17	GPIO_16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x38	GPIO_15	GPIO_14	GPIO_13	GPIO_12	GPIO_11	GPIO_10	GPIO_09	GPIO_08	GPIO_07	GPIO_06	GPIO_05	GPIO_04	GPIO_03	GPIO_02	GPIO_01	GPIO_00
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GPIO_PAD_HYST_EN1				GPIOs GPIO_32 - GPIO_63								Addr Base+0x3C				
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x3C	GPIO_63	GPIO_62	GPIO_61	GPIO_60	GPIO_59	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	GPIO_53	GPIO_52	GPIO_51	GPIO_50	GPIO_49	GPIO_48
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x3C	GPIO_47	GPIO_46	GPIO_45	GPIO_44	GPIO_43	GPIO_42	GPIO_41	GPIO_40	GPIO_39	GPIO_38	GPIO_37	GPIO_36	GPIO_35	GPIO_34	GPIO_33	GPIO_32
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-14. GPIO Pad Hysteresis Enable Register Bit Descriptions

Bit Number	Description	Operation
Bit31-Bit0	GPIO_PAD_HYST_ENx[31:0] - Each bit can enable hysteresis for a GPIO pad when used as an input.	0 = (Default) Hysteresis disabled 1 = Hysteresis enabled

11.7.8 GPIO Pad Keeper Enable Registers (GPIO_PAD_KEEP1 and GPIO_PAD_KEEP0)

The GPIO Pad Keeper Enable Registers determine whether a pad “keeper” is enabled if the pad is an input. The keeper will retain the last state on the pin (high or low) if the pin ceases to be driven. Two 32-bit registers (GPIO_PAD_KEEP1 and GPIO_PAD_KEEP0) are required to control the 64 possible GPIO.

GPIO_PAD_KEEP0				GPIOs GPIO_31 - GPIO_0									Addr Base+0x40			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x40	GPIO_31	GPIO_30	GPIO_29	GPIO_28	GPIO_27	GPIO_26	GPIO_25	GPIO_24	GPIO_23	GPIO_22	GPIO_21	GPIO_20	GPIO_19	GPIO_18	GPIO_17	GPIO_16
RESET	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x40	GPIO_15	GPIO_14	GPIO_13	GPIO_12	GPIO_11	GPIO_10	GPIO_09	GPIO_08	GPIO_07	GPIO_06	GPIO_05	GPIO_04	GPIO_03	GPIO_02	GPIO_01	GPIO_00
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GPIO_PAD_KEEP1				GPIOs GPIO_32 - GPIO_63									Addr Base+0x44			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x44	GPIO_63	GPIO_62	GPIO_61	GPIO_60	GPIO_59	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	GPIO_53	GPIO_52	GPIO_51	GPIO_50	GPIO_49	GPIO_48
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x44	GPIO_47	GPIO_46	GPIO_45	GPIO_44	GPIO_43	GPIO_42	GPIO_41	GPIO_40	GPIO_39	GPIO_38	GPIO_37	GPIO_36	GPIO_35	GPIO_34	GPIO_33	GPIO_32
RESET	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1

Table 11-15. GPIO Pad Pull-up Keeper Register Bit Descriptions

Bit Number	Description	Operation
0-63	GPIO_PAD_KEEPx[31:0] - These bits enable pull-up keepers on the pads.	0 = (Default) Pull-up keep disabled for all but GPIOs 30-36, 38-39 1 = Pull-up keep enabled

NOTE

When the dual-type analog/digital GPIOs (GPIOs 30-41) are configured as inputs (e.g., GPIO inputs), the corresponding GPIO Pad Keeper Enable Register bits must be 1.

11.7.9 GPIO Data Set Registers (GPIO_DATA_SET1 and GPIO_DATA_SET0)

The GPIO Data Set Registers are used to set specific bits to “1” in the GPIO Data Registers. Two registers (GPIO_DATA_SET1 and GPIO_DATA_SET0) are required to control the 64 possible GPIO.

- The data set registers are write-only.
- Any bits that are “1” in the present value of GPIO_DATA0 or GPIO_DATA1 will still be “1” after a write to its corresponding data set register.
- Any bit written to a “1” in the data set register will cause that bit to be set to “1” in its data register.
- The final contents of the data register are the logic “OR” of the previous data register contents and the data set register write data.

NOTE

The GPIO_DATA0 and GPIO_DATA1 registers can be written directly to program all bits at one time.

GPIO_DATA_SET0				FOR GPIO MODES ONLY - GPIOs GPIO_00 - GPIO_31									Addr Base+0x48			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x48	GPIO_31	GPIO_30	GPIO_29	GPIO_28	GPIO_27	GPIO_26	GPIO_25	GPIO_24	GPIO_23	GPIO_22	GPIO_21	GPIO_20	GPIO_19	GPIO_18	GPIO_17	GPIO_16
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x48	GPIO_15	GPIO_14	GPIO_13	GPIO_12	GPIO_11	GPIO_10	GPIO_09	GPIO_08	GPIO_07	GPIO_06	GPIO_05	GPIO_04	GPIO_03	GPIO_02	GPIO_01	GPIO_00
GPIO_DATA_SET1				FOR GPIO MODES ONLY - GPIOs GPIO_32 - GPIO_63									Addr Base+0x4C			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x4C	GPIO_63	GPIO_62	GPIO_61	GPIO_60	GPIO_59	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	GPIO_53	GPIO_52	GPIO_51	GPIO_50	GPIO_49	GPIO_48
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x4C	GPIO_47	GPIO_46	GPIO_45	GPIO_44	GPIO_43	GPIO_42	GPIO_41	GPIO_40	GPIO_39	GPIO_38	GPIO_37	GPIO_36	GPIO_35	GPIO_34	GPIO_33	GPIO_32

Table 11-16. GPIO Data Set Register Bit Descriptions

Bit Number	Description	Operation
0-63	GPIO_DATA_SETx[31:0] - These bits set corresponding bits in the GPIO_DATAx Registers.	Write-only. Writing a 1 sets the corresponding bit in the GPIO_DATAx Register. Any existing bits set to 1 in GPIO_DATAx remain at 1.

11.7.10 GPIO Data Reset Registers (GPIO_DATA_RESET1 and GPIO_DATA_RESET0)

The GPIO Data Reset Registers are used to set specific bits to “0” in the GPIO Data Registers. Two registers (GPIO_DATA_RESET1 and GPIO_DATA_RESET0) are required to control the 64 possible GPIO.

- The data reset registers are write-only.
- Any bits that are “0” in the present value of GPIO_DATA0 or GPIO_DATA1 will still be “0” after a write to its corresponding data reset register.
- Any bit written to a “1” in the data reset register will cause that bit to be reset to “0” in its data register.

- The final contents of the data register are the logic “AND” of the previous data register contents and the bitwise inverse of the data reset register write data.

NOTE

The GPIO_DATA0 and GPIO_DATA1 registers can be written directly to program all bits at one time.

GPIO_DATA_RESET0				FOR GPIO MODES ONLY - GPIOs GPIO_00 - GPIO_31									Addr Base+0x50			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x50	GPIO_31	GPIO_30	GPIO_29	GPIO_28	GPIO_27	GPIO_26	GPIO_25	GPIO_24	GPIO_23	GPIO_22	GPIO_21	GPIO_20	GPIO_19	GPIO_18	GPIO_17	GPIO_16
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x50	GPIO_15	GPIO_14	GPIO_13	GPIO_12	GPIO_11	GPIO_10	GPIO_09	GPIO_08	GPIO_07	GPIO_06	GPIO_05	GPIO_04	GPIO_03	GPIO_02	GPIO_01	GPIO_00
GPIO_DATA_RESET1				FOR GPIO MODES ONLY - GPIOs GPIO_32 - GPIO_63									Addr Base+0x54			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x54	GPIO_63	GPIO_62	GPIO_61	GPIO_60	GPIO_59	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	GPIO_53	GPIO_52	GPIO_51	GPIO_50	GPIO_49	GPIO_48
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x54	GPIO_47	GPIO_46	GPIO_45	GPIO_44	GPIO_43	GPIO_42	GPIO_41	GPIO_40	GPIO_39	GPIO_38	GPIO_37	GPIO_36	GPIO_35	GPIO_34	GPIO_33	GPIO_32

Table 11-17. GPIO Data Reset Register Bit Descriptions

Bit Number	Description	Operation
0-63	GPIO_DATA_RESETx[31:0] - These bits reset corresponding bits in the GPIO_DATAx Registers.	Write-only. Writing a 1 resets the corresponding bit in the GPIO_DATAx Register. Any existing bits set to 0 in GPIO_DATAx remain at 0.

11.7.11 GPIO Pad Direction Set Registers (GPIO_PAD_DIR_SET1 and GPIO_PAD_DIR_SET0)

The GPIO Pad Direction Set Registers are used to set specific bits to “1” in the GPIO Pad Direction Registers (set GPIO to output). Two registers (GPIO_PAD_DIR_SET1 and GPIO_PAD_DIR_SET0) are required to control the 64 possible GPIO.

- The pad direction set registers are write-only.
- Any bits that are “1” in the present value of GPIO_PAD_DIR0 or GPIO_PAD_DIR1 will still be “1” after a write to its corresponding pad direction set register.
- Any bit written to a “1” in the pad direction set register will cause that bit to be set to “1” in its pad direction register.
- The final contents of the pad direction register are the logic “OR” of the previous pad direction register contents and the pad direction set register write data.

NOTE

The GPIO_PAD_DIR0 and GPIO_PAD_DIR1 registers can be written directly to program all bits at one time.

GPIO_DATA_DIR_SET0				FOR GPIO MODES ONLY - GPIOs GPIO_00 - GPIO_31									Addr Base+0x58			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x58	GPIO_31	GPIO_30	GPIO_29	GPIO_28	GPIO_27	GPIO_26	GPIO_25	GPIO_24	GPIO_23	GPIO_22	GPIO_21	GPIO_20	GPIO_19	GPIO_18	GPIO_17	GPIO_16
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x58	GPIO_15	GPIO_14	GPIO_13	GPIO_12	GPIO_11	GPIO_10	GPIO_09	GPIO_08	GPIO_07	GPIO_06	GPIO_05	GPIO_04	GPIO_03	GPIO_02	GPIO_01	GPIO_00
GPIO_DATA_DIR_SET1				FOR GPIO MODES ONLY - GPIOs GPIO_32 - GPIO_63									Addr Base+0x5C			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x5C	GPIO_63	GPIO_62	GPIO_61	GPIO_60	GPIO_59	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	GPIO_53	GPIO_52	GPIO_51	GPIO_50	GPIO_49	GPIO_48
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x5C	GPIO_47	GPIO_46	GPIO_45	GPIO_44	GPIO_43	GPIO_42	GPIO_41	GPIO_40	GPIO_39	GPIO_38	GPIO_37	GPIO_36	GPIO_35	GPIO_34	GPIO_33	GPIO_32

Table 11-18. GPIO Pad Direction Set Register Bit Descriptions

Bit Number	Description	Operation
0-63	GPIO_DATA_DIR_SETx[31:0] - These bits set corresponding bits in the GPIO_PAD_DIRx Registers. Note: Setting a bit enables the GPIO as an output	Write-only. Writing a 1 sets the corresponding bit in the GPIO_DATAx Register. Any existing bits set to 1 in GPIO_DATAx remain at 1.

11.7.12 GPIO Pad Direction Reset Registers (GPIO_PAD_DIR_RESET1 and GPIO_PAD_DIR_RESET0)

The GPIO Pad Direction Reset Registers are used to set specific bits to “0” in the GPIO Pad Direction Registers (set GPIO to input). Two registers (GPIO_PAD_DIR_RESET1 and GPIO_PAD_DIR_RESET0) are required to control the 64 possible GPIO.

- The pad direction reset registers are write-only.
- Any bits that are “0” in the present value of GPIO_PAD_DIR0 or GPIO_PAD_DIR1 will still be “0” after a write to its corresponding data reset register.
- Any bit written to a “1” in the pad direction reset register will cause that bit to be reset to “0” in its pad direction register.
- The final contents of the pad direction register are the logic “AND” of the previous pad direction register contents and the bitwise inverse of the pad direction reset register write data.

NOTE

The GPIO_PAD_DIR0 and GPIO_PAD_DIR1 registers can be written directly to program all bits at one time.

GPIO_DATA_DIR_RESET0				FOR GPIO MODES ONLY - GPIOs GPIO_00 - GPIO_31									Addr Base+0x64			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x60	GPIO_31	GPIO_30	GPIO_29	GPIO_28	GPIO_27	GPIO_26	GPIO_25	GPIO_24	GPIO_23	GPIO_22	GPIO_21	GPIO_20	GPIO_19	GPIO_18	GPIO_17	GPIO_16
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x60	GPIO_15	GPIO_14	GPIO_13	GPIO_12	GPIO_11	GPIO_10	GPIO_09	GPIO_08	GPIO_07	GPIO_06	GPIO_05	GPIO_04	GPIO_03	GPIO_02	GPIO_01	GPIO_00
GPIO_DATA_DIR_RESET1				FOR GPIO MODES ONLY - GPIOs GPIO_32 - GPIO_63									Addr Base+0x64			
Offset ₁₆	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
Base+0x64	GPIO_63	GPIO_62	GPIO_61	GPIO_60	GPIO_59	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	GPIO_53	GPIO_52	GPIO_51	GPIO_50	GPIO_49	GPIO_48
Offset ₁₆	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
Base+0x64	GPIO_47	GPIO_46	GPIO_45	GPIO_44	GPIO_43	GPIO_42	GPIO_41	GPIO_40	GPIO_39	GPIO_38	GPIO_37	GPIO_36	GPIO_35	GPIO_34	GPIO_33	GPIO_32

Table 11-19. GPIO Pad Direction Reset Register Bit Descriptions

Bit Number	Description	Operation
0-63	<p>GPIO_DATA_DIR_RESETx[31:0] - These bits reset corresponding bits in the GPIO_PAD_DIRx Registers.</p> <p>Note: Resetting a bit enables the GPIO as an input</p>	Write-only. Writing a 1 resets the corresponding bit in the GPIO_DATA_DIRx Register. Any existing bits set to 0 in GPIO_DATA_DIRx remain at 0.

Chapter 12

Timer Module (TMR)

12.1 Overview

Each Timer module (TMR) contains four identical counter/timer groups. Each 16 bit counter/timer group contains a prescaler, a counter, a load register, a hold register, a capture register, two compare registers, two status and control registers, and one control register. All of the registers except the prescaler are read/writable. NOTE: This document uses the terms “Timer” and “Counter” interchangeably because the counter/timers may perform either or both tasks.

The Load register provides the initialization value to the counter when the counter’s terminal value has been reached.

The Hold Register captures the counter’s value when other counters are being read. This feature supports the reading of cascaded counters.

The Capture register enables an external signal to take a “snap shot” of the counter’s current value.

The COMP1 and COMP2 registers provide the values to which the counter is compared. If a match occurs, the OFLAG signal can be set, cleared, or toggled. At match time, an interrupt is generated if enabled, and the new compare value is loaded into the COMP1 or COMP2 registers from CMPLD1 and CMPLD2 if enabled.

The Prescaler provides different time bases useful for clocking the counter/timer.

The Counter provides the ability to count internal or external events.

Within a Timer module (set of 4 timer/counters) the input pins are shareable.

12.2 Features

The TMR module design includes these distinctive features:

- Four - 16 bit counters/timers.
- Count up/down.
- Counters can be cascaded.
- Programmable count modulo.
- Max count rate equals peripheral clock/2 for external clocks.
- Max count rate equals peripheral clock for internal clocks.
- Count once or repeatedly.
- Counters can be preloaded.
- Compare Registers can be preloaded. (Available with Compare Load feature)

- Counters can share available input pins.
- Separate prescaler for each counter.
- Each counter has capture and compare capability.
- Programmable operation during debug mode.
- Input glitch filter (optional)
- Counting start can be synchronized across counters. (optional)

12.3 Customization

This section specifies where to find chip specific parameters.

Table 12-1. Customization References for Specific Chips

Feature or Parameter	Where to find
Compare Load Registers	It will be stated whether or not these registers are included in the introduction chapter of the system specification.
Base Address	The Base address for the location of the Timer's memory mapped registers. See Chapter 4, "Memory" .
Input Clock Frequency	See Chapter 3, "MC1322x System Considerations"
Debug Mode	Operation during debug mode is controlled by the DBG_EN bits in the CSCTRL reg starting in the Xtalk chip and going forward.
Glitch Filter	Includes four bit shift register filters on each external input and the FILT_EN control bit.
Synchronization	Includes Enable register to allow a synchronous start of multiple counters.

12.4 Modes of Operation

The TMR module design operates in only a single mode of operation: Functional Mode. The various counting modes are detailed in [12.15 Functional Modes Functional Description](#).

12.5 Block Diagram

Each of the timer/counter groups within the quad-timer are shown in [Figure 12-1](#).

Figure 12-1. Quad Timer Block Diagram

12.6 Signal Descriptions

12.7 Overview

The TMR module has 4 external signals TIO[3:0] that can be used as either inputs or outputs. The external interface signals are generated using the TMR_OUTPUT, TMR_OUTPUT_EN_B, and TMR_INPUT signals. The TMR also interfaces to the Peripheral Bus.

Table 12-2. Internal Signal Properties

Name	I/O Type	Function	Reset State	Notes
RD_DATA_Z[15:0]	Output	IPBus read data bus.	16'z	
TMR_INT	Output	Interrupt request	1	
TMR0_OUTPUT	Output	Output signal from counter 0	0	
TMR1_OUTPUT	Output	Output signal from counter 1	0	
TMR2_OUTPUT	Output	Output signal from counter 2	0	
TMR3_OUTPUT	Output	Output signal from counter 3	0	
TMR0_OUTPUT_EN_B	Output	Output enable signal from counter 0	1	
TMR1_OUTPUT_EN_B	Output	Output enable signal from counter 1	1	
TMR2_OUTPUT_EN_B	Output	Output enable signal from counter 2	1	
TMR3_OUTPUT_EN_B	Output	Output enable signal from counter 3	1	
TMR0_INPUT	Input	Input signal to counter 0	0	
TMR1_INPUT	Input	Input signal to counter 1	0	
TMR2_INPUT	Input	Input signal to counter 2	0	
TMR3_INPUT	Input	Input signal to counter 3	0	
CLK	Input	IPBus clock.	Driven	
RST_B	Input	Asynchronous active low reset signal.	0	
MODULE_EN_B	Input	IPBus module enable.	1	
ADDR[4:0]	Input	IPBus address bus.		
RWB	Input	IP bus transaction type indicator. 0=Write, 1=Read	1	
WR_DATA[15:0]	Input	IP bus write data bus.	16'0	
SCANTESTMODE	Input	Chip is in scan test mode.	0	
DBG_SESS_B	Input	Chip is in debug mode.	1	

Table 12-3. External Signal Properties

Name	I/O Type	Function	Reset State	Notes
TIO0	Input/Output	Timer Input/Outputs	Input	
TIO1	Input/Output	Timer Input/Outputs	Input	
TIO2	Input/Output	Timer Input/Outputs	Input	
TIO3	Input/Output	Timer Input/Outputs	Input	

12.8 External Signal Descriptions

12.8.1 TIO3-TIO0 - Timer Input/Outputs

These pins can be independently configured to be either timer input sources or output flags.

12.9 Memory Map and Registers

12.10 Overview

The base address of the TMR module will differ from chip to chip. All memory mapped registers described below have their location described in relation to the BASE Address.

NOTE

Base address for the TMR Module is 0x8000_7000

12.11 Module Memory Map

Table 12-4. Timer Memory Map

Address	Reg Name	Access
Base + \$0	Timer Channel 0 Compare Register #1 (TMR0_COMP1)	Read/Write
Base + \$2	Timer Channel 0 Compare Register #2 (TMR0_COMP2)	Read/Write
Base + \$4	Timer Channel 0 Capture Register (TMR0_CAPT)	Read/Write
Base + \$6	Timer Channel 0 Load Register (TMR0_LOAD)	Read/Write
Base + \$8	Timer Channel 0 Hold Register (TMR0_HOLD)	Read/Write
Base + \$a	Timer Channel 0 Counter Register (TMR0_CNTR)	Read/Write
Base + \$c	Timer Channel 0 Control Register (TMR0_CTRL)	Read/Write
Base + \$e	Timer Channel 0 Status and Control Register (TMR0_SCTRL)	Read/Write
Base + \$10	Timer Channel 0 Comparator Load Register 1 (TMR0_CMPLD1)	Read/Write
Base + \$12	Timer Channel 0 Comparator Load Register 2 (TMR0_CMPLD2)	Read/Write
Base + \$14	Timer Channel 0 Comparator Status and Control Register (TMR0_CSCTRL)	Read/Write
Base + \$16 - Base + \$1c	Reserved	n/a
Base + \$1e	Timer Channel Enable Register (TMR0_ENBL)	Read/Write
Base + \$20	Timer Channel 1 Compare Register #1 (TMR1_COMP1)	Read/Write
Base + \$22	Timer Channel 1 Compare Register #2 (TMR1_COMP2)	Read/Write
Base + \$24	Timer Channel 1 Capture Register (TMR1_CAPT)	Read/Write
Base + \$26	Timer Channel 1 Load Register (TMR1_LOAD)	Read/Write
Base + \$28	Timer Channel 1 Hold Register (TMR1_HOLD)	Read/Write
Base + \$2a	Timer Channel 1 Counter Register (TMR1_CNTR)	Read/Write
Base + \$2c	Timer Channel 1 Control Register (TMR1_CTRL)	Read/Write
Base + \$2e	Timer Channel 1 Status and Control Register (TMR1_SCTRL)	Read/Write
Base + \$30	Timer Channel 1 Comparator Load Register 1 (TMR1_CMPLD1)	Read/Write
Base + \$32	Timer Channel 1 Comparator Load Register 2 (TMR1_CMPLD2)	Read/Write
Base + \$34	Timer Channel 1 Comparator Status and Control Register (TMR1_CSCTRL)	Read/Write

Table 12-4. Timer Memory Map (continued)

Address	Reg Name	Access
Base + \$36 - Base + \$3c	Reserved	n/a
Base + \$40	Timer Channel 2 Compare Register #1 (TMR2_COMP1)	Read/Write
Base + \$42	Timer Channel 2 Compare Register #2 (TMR2_COMP2)	Read/Write
Base + \$44	Timer Channel 2 Capture Register (TMR2_CAPT)	Read/Write
Base + \$46	Timer Channel 2 Load Register (TMR2_LOAD)	Read/Write
Base + \$48	Timer Channel 2 Hold Register (TMR2_HOLD)	Read/Write
Base + \$4a	Timer Channel 2 Counter Register (TMR2_CNTR)	Read/Write
Base + \$4d	Timer Channel 2 Control Register (TMR2_CTRL)	Read/Write
Base + \$4e	Timer Channel 2 Status and Control Register (TMR2_SCTRL)	Read/Write
Base + \$50	Timer Channel 2 Comparator Load Register 1 (TMR2_CMPLD1)	Read/Write
Base + \$52	Timer Channel 2 Comparator Load Register 2 (TMR2_CMPLD2)	Read/Write
Base + \$54	Timer Channel 2 Comparator Status and Control Register (TMR2_CSCTRL)	Read/Write
Base + \$56 - Base + \$5c	Reserved	n/a
Base + \$60	Timer Channel 3 Compare Register #1 (TMR3_COMP1)	Read/Write
Base + \$62	Timer Channel 3 Compare Register #2 (TMR3_COMP2)	Read/Write
Base + \$64	Timer Channel 3 Capture Register (TMR3_CAPT)	Read/Write
Base + \$66	Timer Channel 3 Load Register (TMR3_LOAD)	Read/Write
Base + \$68	Timer Channel 3 Hold Register (TMR3_HOLD)	Read/Write
Base + \$6a	Timer Channel 3 Counter Register (TMR3_CNTR)	Read/Write
Base + \$6c	Timer Channel 3 Control Register (TMR3_CTRL)	Read/Write
Base + \$6e	Timer Channel 3 Status and Control Register (TMR3_SCTRL)	Read/Write
Base + \$70	Timer Channel 3 Comparator Load Register 1 (TMR3_CMPLD1)	Read/Write
Base + \$72	Timer Channel 3 Comparator Load Register 2 (TMR3_CMPLD2)	Read/Write
Base + \$74	Timer Channel 3 Comparator Status and Control Register (TMR3_CSCTRL)	Read/Write
Base + \$76 - Base + \$7e	Reserved	n/a

12.12 Register Descriptions

The address of a register is the sum of a base address and an address offset. The base address is defined at the MCU level and the address offset is defined at the module level. The base address given for each register will be TMR_BASE.

Make certain to check which quad timer is available on the chip being used, and which Timer channels have external I/O.

12.12.1 TMR Compare Register 1 (COMP1)

TMR0_COMP1 (Timer A, Channel 0 Compare #1)

TMR1_COMP1 (Timer A, Channel 1 Compare #1)

TMR2_COMP1 (Timer A, Channel 2 Compare #1)

TMR3_COMP1 (Timer A, Channel 3 Compare #1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COMPARISON VALUE [15:0]															
Write	COMPARISON VALUE [15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-2. TMR Compare Register 1 (COMP1)

This read/write register stores the value used for comparison with the counter value.

12.12.2 TMR Compare Register 2 (COMP2)

TMR0_COMP2 (Timer A, Channel 0 Compare #2)

TMR1_COMP2 (Timer A, Channel 1 Compare #2)

TMR2_COMP2 (Timer A, Channel 2 Compare #2)

TMR3_COMP2 (Timer A, Channel 3 Compare #2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COMPARISON VALUE [15:0]															
Write	COMPARISON VALUE [15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-3. TMR Compare Register 2 (COMP2)

This read/write register stores the value used for comparison with the counter value.

12.12.3 TMR Capture Register (CAPT)

TMR0_CAPT (Timer A, Channel 0 Capture)

TMR1_CAPT (Timer A, Channel 1 Capture)

TMR2_CAPT (Timer A, Channel 2 Capture)

TMR3_CAPT (Timer A, Channel 3 Capture)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	CAPTURE VALUE [15:0]															
Write	CAPTURE VALUE [15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-4. TMR Capture Register (CAPT)

This read/write register stores the value captured from the counter.

12.12.4 TMR Load Register (LOAD)

TMR0_LOAD (Timer A, Channel 0 Load)

TMR1_LOAD (Timer A, Channel 1 Load)

TMR2_LOAD (Timer A, Channel 2 Load)

TMR3_LOAD (Timer A, Channel 3 Load)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	LOAD VALUE [15:0]															
Write	LOAD VALUE [15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-5. TMR Load Register (LOAD)

This read/write register stores the value used to initialize the counter.

12.12.5 TMR Hold Register (HOLD)

TMR0_HOLD (Timer A, Channel 0 Hold)

TMR1_HOLD (Timer A, Channel 1 Hold)

TMR2_HOLD (Timer A, Channel 2 Hold)

TMR3_HOLD (Timer A, Channel 3 Hold)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HOLD VALUE [15:0]															
Write	HOLD VALUE [15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-6. TMR Hold Register (HOLD)

This read/write register stores the counter's values of specific channels whenever any of the four counters within a module is read.

12.12.6 TMR Counter Register (CNTR)

TMR0_CNTR (Timer A, Channel 0 Cntr)

TMR1_CNTR (Timer A, Channel 1 Cntr)

TMR2_CNTR (Timer A, Channel 2 Cntr)

TMR3_CNTR (Timer A, Channel 3 Cntr)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COUNTER [15:0]															
Write	COUNTER [15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-7. TMR Counter (CNTR)

This read/write register is the counter for the corresponding channel in a timer module.

12.12.7 TMR Control Registers (CTRL)

TMR0_CTRL (Timer A, Channel 0 Control)

TMR1_CTRL (Timer A, Channel 1 Control)

TMR2_CTRL (Timer A, Channel 2 Control)

TMR3_CTRL (Timer A, Channel 3 Control)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Count Mode			Primary Count Source				Secondary Source	ONCE	LENGTH	DIR	Co Init	Output Mode			
Write	Count Mode			Primary Count Source				Secondary Source	ONCE	LENGTH	DIR	Co Init	Output Mode			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-8. TMR Control Register (CTRL)

Count Mode—Bits 15–13

These bits control the basic counting and behavior of the counter.

Table 12-5. Count Mode Values

Value	Meaning
000	No Operation
001	Count rising edges of primary source ¹
010	Count rising and falling edges of primary source ²
011	Count rising edges of primary source while secondary input high active
100	Quadrature Count mode, uses primary and secondary sources
101	Count primary source rising edges, secondary source specifies direction (1 = minus) ³
110	Edge of secondary source triggers primary count till compare
111	Synchronous counter mode, up/down ⁴

¹ Rising Edges counted only when IPS = 0. Falling edges counted when IPS = 1. If primary count source is IP bus clock divide by 1 only rising edges are counted regardless of IPS value.

² IP Bus clock divide by 1 can not be used as a primary count source in edge count mode.

³ Rising Edges counted only when IPS = 0. Falling edges counted when IPS = 1.

⁴ Primary count source must be set to one of the counter outputs.

Primary Count Source—Bits 12–9

These bits select the primary count source.

Table 12-6. Primary Count Source Values

Value	Meaning
0000	Counter #0 input pin
0001	Counter #1 input pin
0010	Counter #2 input pin
0011	Counter #3 input pin
0100	Counter #0 output
0101	Counter #1 output
0110	Counter #2 output
0111	Counter #3 output

Table 12-6. Primary Count Source Values (continued)

Value	Meaning
1000	IP Bus clock divide by 1 prescaler
1001	IP Bus clock divide by 2 prescaler
1010	IP Bus clock divide by 4 prescaler
1011	IP Bus clock divide by 8 prescaler
1100	IP Bus clock divide by 16 prescaler
1101	IP Bus clock divide by 32 prescaler
1110	IP Bus clock divide by 64 prescaler
1111	IP Bus clock divide by 128 prescaler

NOTE

A timer selecting its own output for input is not a legal choice. The result is no counting.

Secondary Count Source—Bits 8–7

These bits identify the external input pin to be used as a count command, or timer command. The selected input can trigger the timer to capture the current value of the CNTR register. The selected input can also be used to specify the count direction. The polarity of the signal can be inverted by the IPS bit of the SCTRL register.

Table 12-7. Secondary Count Source Values

Value	Meaning
00	Counter #0 input pin
01	Counter #1 input pin
10	Counter #2 input pin
11	Counter #3 input pin

Count Once (ONCE)—Bit 6

This bit selects continuous or one shot counting mode.

1 = Count until compare and then stop. If *counting up*, successful compare occurs when the counter reaches a COMP1 value. If *counting down*, successful compare occurs when the counter reaches a COMP2 value

0 = Count repeatedly

Count Length (LENGTH)—Bit 5

This bit determines whether the counter counts to the compare value and then re-initializes itself to the value specified in the load register, or the counter continues counting past the compare value, to the binary roll over.

1 = Count until compare, then re initialize. If counting up, successful compare occurs when counter reaches COMP1 value. If counting down, successful compare occurs when counter reaches COMP2 value.

When output mode \$4 is used, alternating values of COMP1 and COMP2 are used to generate

successful comparisons. For example, the counter counts until COMP1 value is reached, re-initializes, then counts until COMP2 value is reached, re-initializes, then counts until COMP1 value is reached, etc.

0 = Roll Over

Count Direction (DIR)—Bit 4

This bit selects either the normal count direction *up*, or the reverse direction, *down*.

1 = Count-down

0 = Count-up

Co-Channel Initialization (Co Init)—Bit 3

This bit enables another counter/timer within the module to force the re-initialization of this counter/timer when it has an active compare event.

1 = Co-Channel counter/timers may force a re-initialization of this counter/timer

0 = Co-Channel counter/timers can not force a re-initialization of this counter/timer

Output Mode—Bits 2-0

These bits determine the mode of operation for the OFLAG output signal.

Table 12-8. Output Mode Bits

Value	Meaning
000	Asserted while counter is active
001	Clear OFLAG output on successful compare
010	Set OFLAG output on successful compare
011	Toggle OFLAG output on successful compare
100	Toggle OFLAG output using alternating compare registers
101	Set on compare, cleared on secondary source input edge
110	Set on compare, cleared on counter rollover
111	Enable gated clock output while counter is active

12.12.8 TMR Status and Control Registers (SCTRL)

TMR0_SCTRL (Timer A, Channel 0 Status and Control)

TMR1_SCTRL (Timer A, Channel 1 Status and Control)

TMR2_SCTRL (Timer A, Channel 2 Status and Control)

TMR3_SCTRL (Timer A, Channel 3 Status and Control)

	15	14	13	12	11	10	9	8	[7:6]	5	4	3	2	1	0
Read	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	Capture Mode	MSTR	EEOF	VAL	0	OPS	OEN
Write													FORCE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-9. TMR Status and Control Register (SCTRL)

TCF - Timer Compare Flag

This bit is set when a successful compare occurs. This bit is cleared by writing a zero to this bit location.

TCFIE - Timer Compare Flag Interrupt Enable

This bit (when set) enables interrupts when the TCF bit is set.

TOF - Timer Overflow Flag

This bit is set when the counter rolls over its maximum value \$FFFF or \$0000 (depending on count direction). This bit is cleared by writing a zero to this bit location.

TOFIE - Timer Overflow Flag Interrupt Enable

This bit (when set) enables interrupts when the TOF bit is set.

IEF - Input Edge Flag

This bit is set when a positive input transition occurs (on an input selected as a secondary count source) while the counter is enabled. This bit is cleared by writing a zero to this bit position.

NOTE

Setting the input polarity select (IPS) bit enables the detection of negative input edge transitions detection. Also, the control register's secondary count source determines which external input pin is monitored by the detection circuitry.

IEFIE - Input Edge Flag Interrupt Enable

This bit (when set) enables interrupts when the IEF bit is set

IPS - Input Polarity Select

This bit (when set) inverts the input signal polarity.

INPUT - External input signal

This bit reflects the current state of the external input pin selected via the secondary count source after application of the IPS bit. This is a Read Only Bit.

Capture Mode - Input Capture Mode

These bits specify the operation of the capture register as well as the operation of the input edge flag. The input source is the secondary count source.

Table 12-9. Input Capture Mode Values

Capture Mode	IPS	Meaning
00	X	Capture function is disabled.
01	0	Load Capture register on rising edge of input
	1	Load Capture register on falling edge of input
10	0	Load Capture register on falling edge of input
	1	Load Capture register on rising edge of input
11	X	Load Capture register on both edges of input

MSTR - Master Mode

This bit (when set) enables the compare function’s output to be broadcast to the other counter/timers in the module. This signal then can be used to re initialize the other counters and/or force their OFLAG signal outputs.

EEOF - Enable External OFLAG Force

This bit (when set) enables the compare from another counter/timer within the same module to force the state of this counter’s OFLAG output signal.

VAL - Forced OFLAG Value

This bit determines the value of the OFLAG output signal when a software triggered FORCE command.

FORCE- Force the OFLAG output.

This write only bit forces the current value of the VAL bit to be written to the OFLAG output. This bit always reads as a zero. The VAL and FORCE bits can be written simultaneously in a single write operation. Write to the FORCE bit only if the counter is disabled. Setting this bit while the counter is enabled may yield unpredictable results.

OPS - Output Polarity Select.

This bit determines the polarity of the OFLAG output signal.

0 = True polarity.

1 = Inverted polarity.

OEN - Output Enable.

This bit determines the direction of the external pin.

1 = OFLAG output signal will be driven on the external pin. Other timer groups using this external pin as their input will see the driven value. The polarity of the signal will be determined by the OPS bit.

0 = The external pin is configured as an input.

12.12.9 TMR Comparator Load Register 1 (CMPLD1)

TMR0_CMPLD1 (Timer A, Channel 0 CMPLD1) w/Compare Load Only

TMR1_CMPLD1 (Timer A, Channel 1 CMPLD1) w/Compare Load Only

TMR2_CMPLD1 (Timer A, Channel 2 CMPLD1) w/Compare Load Only

TMR3_CMPLD1 (Timer A, Channel 3 CMPLD1) w/Compare Load Only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COMPARATOR LOAD 1 [15:0]															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-10. TMR Comparator Load 1 (CMPLD1)

This read/write register is the Comparator one preload value for the COMP1 register for the corresponding channel in a timer module.

12.12.10 TMR Comparator Load Register 2 (CMPLD2)

TMR0_CMPLD2 (Timer A, Channel 0 CMPLD2) w/Compare Load Only

TMR1_CMPLD2 (Timer A, Channel 1 CMPLD2) w/Compare Load Only

TMR2_CMPLD2 (Timer A, Channel 2 CMPLD2) w/Compare Load Only

TMR3_CMPLD2 (Timer A, Channel 3 CMPLD2) w/Compare Load Only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	COMPARATOR LOAD 2 [15:0]															
Write	COMPARATOR LOAD 2 [15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-11. TMR Comparator Load 2 (CMPLD2)

This read/write register is the Comparator two preload value for the COMP2 register for the corresponding channel in a timer module.

12.12.11 TMR Comparator Status and Control Register (CSCTRL)

TMR0_CSCTRL (Timer A, Channel 0 CSCTRL) w/Compare Load Only

TMR1_CSCTRL (Timer A, Channel 1 CSCTRL) w/Compare Load Only

TMR2_CSCTRL (Timer A, Channel 2 CSCTRL) w/Compare Load Only

TMR3_CSCTRL (Timer A, Channel 3 CSCTRL) w/Compare Load Only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DBG_EN [1:0]		FILT_EN	0	0	0	0	0	TCF2EN	TCF1EN	TCF2	TCF1	CL2[1:0]		CL1[1:0]	
Write	DBG_EN [1:0]		FILT_EN						TCF2EN	TCF1EN	TCF2	TCF1	CL2[1:0]		CL1[1:0]	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12-12. TMR Comparator Status and Control Register (CSCTRL)

Debug Actions Enable (DBG_EN)—Bits 15-14

These bits allow the TMR module to perform certain actions in response to the chip entering debug mode.

Table 12-10. Values for DBG_EN

Value	Meaning
00	Continue with normal operation during debug mode. (default)
01	Halt TMR counter during debug mode.
10	Force TMR output to logic 0 (prior to consideration of the OPS bit).
11	Both halt counter and force output to 0 during debug mode.

FILT_EN - Input Filter Enable.

Each of the four external inputs to the TMR has a four bit shift register to act as a glitch filter. Three consecutive samples must match before the value is passed on to the counter. Assuming 32MHz operation, this will filter out pulses less than 62 nsecs wide. This control bit enables all four filters.

0 = Input filters disabled.

1 = Input filters enabled.

Reserved—Bits 12–8

Timer Compare 2 Interrupt Enable (TCF2EN)—Bit 7

An interrupt is issued when both this bit and the TCF2 bit are set.

Timer Compare 1 Interrupt Enable (TCF1EN)—Bit 6

An interrupt is issued when both this bit and the TCF1 bit are set.

Timer Compare 2 Interrupt Source (TCF2)—Bit 5

When set, this bit indicates a successful comparison of the timer and COMP2 register has occurred. This bit is sticky, and will remain set until explicitly cleared by writing a zero to this bit location.

Timer Compare 1 Interrupt Source (TCF1)—Bit 4

When set, this bit indicates a successful comparison of the timer and the COMP1 register has occurred. This bit is sticky, and will remain set until explicitly cleared by writing a zero to this bit location.

Compare Load Control 2 (CL2)—Bit 3–2

These bits control when COMP2 is preloaded with the value from CMPLD2.

Table 12-11. Values for Compare Load Control 2

Value	Meaning
00	Never preload
01	Load upon successful compare with the value in COMP1
10	Load upon successful compare with the value in COMP2
11	Reserved

Compare Load Control 1 (CL1)—Bit 1–0

These bits control when COMP1 is preloaded with the value from CMPLD1.

Table 12-12. Values for Compare Load Control 1

Value	Meaning
00	Never preload
01	Load upon successful compare with the value in COMP1
10	Load upon successful compare with the value in COMP2
11	Reserved

12.12.12 TMR Channel Enable Register (ENBL)

TMR0_ENBL (Timer A, Channel ENBL)—Address: TMR_BASE + \$F w/Synchronization option only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0	0	0	0	0	ENBL			
Write													ENBL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Figure 12-13. TMR Channel Enable Register (ENBL)

Reserved—Bits 15–4

Timer Channel Enable (ENBL)—Bits 3-0

These bits enable the prescaler (if it is being used) and counter in each channel. Multiple ENBL bits can be set at the same time to synchronize the start of separate counters. If an ENBL bit is set, then the corresponding channel will start counter as soon as the COUNT_MODE field has a value other than 0. When an ENBL bit is clear, the corresponding counter maintains its current value.

1 = Timer channel is enabled. (default)

0 = Timer channel is disabled.

12.13 Functional Description

12.14 General

The counter/timer has two basic modes of operation: It can count internal or external events, or it can count an internal clock source while an external input signal is asserted, thus timing the width of the external input signal.

- The counter can count the rising, falling, or both edges of the selected input pin.
- The counter can decode and count quadrature encoded input signals.
- The counter can count up and down using dual inputs in a “count with direction” format.
- The counter’s terminal count value (modulo) is programmable.
- The value that is loaded into the counter after reaching its terminal count is programmable.
- The counter can count repeatedly, or it can stop after completing one count cycle.
- The counter can be programmed to count to a programmed value and then immediately re initialize, or it can count through the compare value until the count “rolls over” to zero.

The external inputs to each counter/timer are shareable among each of the four counter/timers within the module. The external inputs can be used as:

- Count commands
- Timer commands
- They can trigger the current counter value to be “captured”
- They can be used to generate interrupt requests

The polarity of the external inputs are selectable.

The primary output of each timer/counter is the output signal OFLAG. The OFLAG output signal can be:

- Set, cleared, or toggled when the counter reaches the programmed value.
- The OFLAG output signal may be output to an external pin instead of having that pin serve as a timer input.
- The OFLAG output signal enables each counter to generate square waves, PWM, or pulse stream outputs.
- The polarity of the OFLAG output signal is selectable.

Any counter/timer can be assigned as a “Master”. A master’s compare signal can be broadcast to the other counter/timers within the module. The other counters can be configured to re initialize their counters and/or force their OFLAG output signals to predetermined values when a Master’s counter/timer compare event occurs.

12.15 Functional Modes

The selected external count signals are sampled at the TMR’s base clock rate and then run through a transition detector. The maximum count rate is one-half of the TMR’s base clock rate. Internal clock sources can be used to clock the counters at the TMR’s base clock rate.

If a counter is programmed to count to a specific value and then stop, the Count Mode in the CTRL register is cleared when the count terminates.

12.15.1 STOP Mode

If the Count Mode field is set to ‘000’, the counter is inert. No counting will occur. Stop mode will also disable the interrupts caused by input transitions on a selected input pin.

12.15.2 COUNT Mode

If the Count Mode field is set to ‘001’, the counter will count the rising edges of the selected clock source. This mode is useful for generating periodic interrupts for timing purposes, or counting external events such as “widgets” on a conveyor belt passing a sensor. If the selected input is inverted by setting the IPS bit (Input Polarity Select), then the negative edge of the selected external input signal is counted.

```
// (See Processor Expert PulseAccumulator bean.)
// This example uses TMR1 to count pulse (actually counts rising edges of the pulse)
// from an external source (TA3).
//
void Pulse_Init(void)
{
    /* TMR1_CTRL: CM=0,PCS=3,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
    setReg(TMR1_CTRL,0x0600);          /* Set up mode */
    /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
    setReg(TMR1_SCTRL,0x00);
    setReg(TMR1_CNTR,0x00);           /* Reset counter register */
    setReg(TMR1_LOAD,0x00);          /* Reset load register */
    setRegBitGroup(TMR1_CTRL,CM,0x01); /* Run counter */
}
```

Figure 12-14. Count Pulses from External Source

```

//      (See Processor Expert TimerInt bean.)
// This example generates an interrupt every 100ms,
// assuming the chip is operating at 60 MHz.
//
// It does this by using the IP_bus_clk divided by 128 as the counter clock source.
// The counter then counts to 46874 where it matches the COMP1 value.
// At that time an interrupt is generated, the counter is reloaded and
// the next COMP1 value is loaded from CMPLD1.
//
void TimerInt_Init(void)
{
  /* TMRA0_CTRL: CM=0,PCS=0,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA0_CTRL,0x20);          /* Stop all functions of the timer */
  /* TMRA0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA0_SCTRL,0x00);
  setReg(TMRA0_LOAD,0x00);          /* Reset load register */
  setReg(TMRA0_COMP1,46874);        /* Set up compare 1 register */
  setReg(TMRA0_CMPLD1,46874);       /* Also set the compare preload register */
  /* TMRA0_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,TCF2EN=0,TCF1EN=1,
    TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMRA0_CSCTRL,0x41);        /* Enable compare 1 interrupt and */
  /* compare 1 preload */
  setRegBitGroup(TMRA0_CTRL,PCS,0xF); /* Primary Count Source to IP_bus_clk / 128 */
  setReg(TMRA0_CNTR,0x00);          /* Reset counter register */
  setRegBitGroup(TMRA0_CTRL,CM,0x01); /* Run counter */
}

```

Figure 12-15. Generate Periodic Interrupt By Counting Internal Clocks

12.15.3 EDGE-COUNT Mode

If the Count Mode field is set to '010', the counter will count both edges of the selected external clock source. This mode is useful for counting the changes in the external environment such as a simple encoder wheel.

```

//      (See Processor Expert PulseAccumulator bean.)
// This example uses TMRA1 to count pulse (actually counts rising edges of the pulse)
// from an external source (TA3).
//
void Pulse_Init(void)
{
  /* TMRA1_CTRL: CM=0,PCS=3,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA1_CTRL,0x0600);        /* Set up mode */
  /* TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA1_SCTRL,0x00);
  setReg(TMRA1_CNTR,0x00);          /* Reset counter register */
  setReg(TMRA1_LOAD,0x00);          /* Reset load register */
  setRegBitGroup(TMRA1_CTRL,CM,0x02); /* Run counter */
}

```

Figure 12-16. Count Both Edges of External Source Signal

12.15.4 GATED-COUNT Mode

If the Count Mode field is set to ‘011’, the counter will count while the selected secondary input signal is high. This mode is used to time the duration of external events. If the selected input is inverted by setting the IPS bit (Input Polarity Select), then the counter will count while the selected secondary input is low.

```
//      (See Processor Expert PulseAccumulator bean.)
// This example uses TMRA1 to determine the duration of an external pulse.
//
// The IP_bus clock is used as the primary counter.  If the duration of the
// external pulse is longer than 0.001 seconds one of the other IP_bus clock
// dividers can be used.  If the pulse duration is longer than 0.128 seconds
// an external clock source will have to be used as the primary clock source.
//
void Pulse1_Init(void)
{
  /* TMRA1_CTRL: CM=0,PCS=8,SCS=1,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
  setReg(TMRA1_CTRL,0x1080);          /* Set up mode */

  /* TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMRA1_SCTRL,0x00);
  setReg(TMRA1_CNTR,0x00);           /* Reset counter register */
  setReg(TMRA1_LOAD,0x00);          /* Reset load register */
  setRegBitGroup(TMRA1_CTRL,CM,0x03); /* Run counter */
}
```

Figure 12-17. Capture Duration of External Pulse

12.15.5 QUADRATURE-COUNT Mode

If the Count Mode field is set to ‘100’, the counter will decode the primary and secondary external inputs as quadrature encoded signals. Quadrature signals are usually generated by rotary or linear sensors used to monitor movement of motor shafts or mechanical equipment. The quadrature signals are square waves that are 90 degrees out of phase. The decoding of quadrature signal provides both count and direction information.

Figure 12-18 shows a timing diagram illustrating the basic operation of a quadrature incremental position encoder.

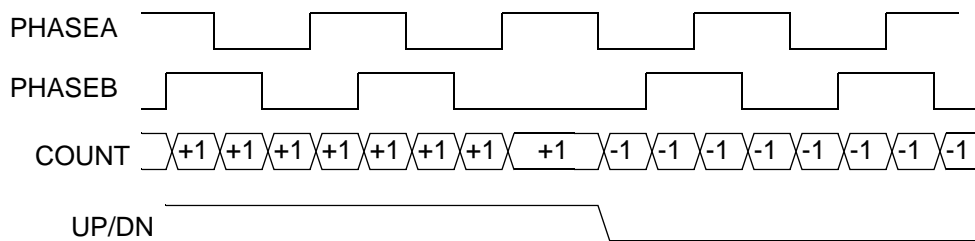


Figure 12-18. Quadrature Incremental Position Encoder

```

//      (See Processor Expert PulseAccumulator bean.)
//      This example uses TMRA0 for counting states of a quadrature position encoder.
//
//      Timer input 0 is used as the primary count source (PHASEA).
//      Timer input 1 is used as the secondary count source (PHASEB).
//
void Pulse_Init(void)
{
    /* TMRA0_CTRL: CM=0,PCS=0,SCS=1,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
    setReg(TMRA0_CTRL,0x80);          /* Set up mode */

    /* TMRA0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
    setReg(TMRA0_SCTRL,0x00);

    setReg(TMRA0_CNTR,0x00);          /* Reset counter register */
    setReg(TMRA0_LOAD,0x00);         /* Reset load register */
    setReg(TMRA0_COMP1,0xFFFF);      /* Set up compare 1 register */
    setReg(TMRA0_COMP2,0x00);        /* Set up compare 2 register */

    /* TMRA0_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
    TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
    setReg(TMRA0_CSCTRL,0x00);

    setRegBitGroup(TMRA0_CTRL,CM,0x04); /* Run counter */
}

```

Figure 12-19. Quadrature Count Mode Example

12.15.6 SIGNED-COUNT Mode

If the Count Mode field is set to '101', the counter counts the primary clock source while the selected secondary source provides the selected count direction (up/down).

```

//      (See Processor Expert PulseAccumulator bean.)
//      This example uses TMRA0 for signed mode counting.
//
//      Timer input 2 is used as the primary count source.
//      Timer input 1 is used to determine the count direction.
//
void Pulse_Init(void)
{
    /* TMRA0_CTRL: CM=0,PCS=2,SCS=1,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
    setReg(TMRA0_CTRL,0x0480);        /* Set up mode */

    /* TMRA0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
    setReg(TMRA0_SCTRL,0x1000);

    setReg(TMRA0_CNTR,0x00);          /* Reset counter register */
    setReg(TMRA0_LOAD,0x00);         /* Reset load register */

    setRegBitGroup(TMRA0_CTRL,CM,0x05); /* Run counter */
}

```

Figure 12-20. Signed Count Mode Example

12.15.7 TRIGGERED-COUNT Mode

If the Count Mode field is set to '110', the counter will begin counting the primary clock source after a positive transition (negative edge if $IPS = 1$) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. If a second input transition occurs before a terminal count was reached, counting will stop and the TCF bit (timer compare flag) will be set. Subsequent secondary input transitions will continue to restart and stop the counting until a compare event occurs.

NOTE

I am unsure why the TCF bit is set upon the second positive edge, but it has been this way as far back as I can find (at least Prague). This behavior could be "fixed" if it is determined that no customers are depending on it.

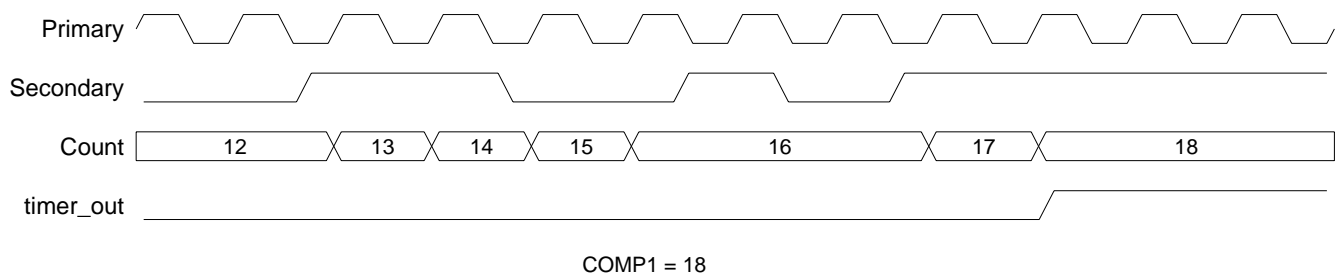


Figure 12-21. Triggered Count Mode (Length=0)

```
//      (See Processor Expert PulseAccumulator bean.)
//      This example uses TMRA1 for triggered mode counting.
//
//      Timer input 3 is used as the primary count source.
//      Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
    /* TMRA1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0 */
    setReg(TMRA1_CTRL,0x0700);          /* Set up mode */

    /* TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
    setReg(TMRA1_SCTRL,0x1000);

    setReg(TMRA1_CNTR,0x00);            /* Reset counter register */
    setReg(TMRA1_LOAD,0x00);           /* Reset load register */
    setReg(TMRA1_COMP1,0x0012);        /* Set up compare 1 register */

    /* TMRA1_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
    TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
    setReg(TMRA1_CSCTRL,0x00);

    setRegBitGroup(TMRA1_CTRL,CM,0x06); /* Run counter */
}
```

Figure 12-22. Triggered Count Mode Example

12.15.8 One-Shot Mode

If the Count Mode field is set to '110', and the counter is set to re initialize at a compare event (Count Length =1), and the OFLAG Output Mode is set to '101' (Cleared on init, set on Compare), the counter works in a "One-Shot Mode". An external events causes the counter to count, when terminal count is reached, the output is asserted. This "delayed" output can be used to provide timing delays.

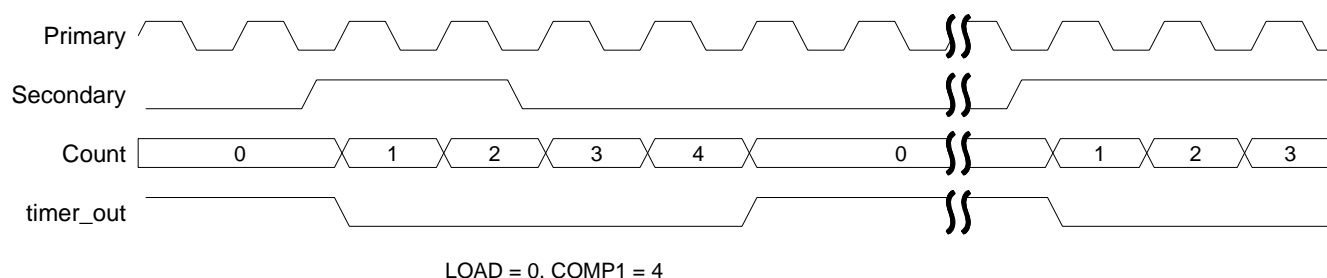


Figure 12-23. One-Shot Mode (Length=0)

```
// (See Processor Expert PulseAccumulator bean.)
// This example uses TMRA1 for one-shot mode counting.
//
// Timer input 3 is used as the primary count source.
// Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
    /* TMRA1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=5 */
    setReg(TMRA1_CTRL,0x0725);          /* Set up mode */

    /* TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
    Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
    setReg(TMRA1_SCTRL,0x1000);

    setReg(TMRA1_CNTR,0x00);           /* Reset counter register */
    setReg(TMRA1_LOAD,0x00);          /* Reset load register */
    setReg(TMRA1_COMP1,0x0004);       /* Set up compare 1 register */

    /* TMRA1_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
    TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
    setReg(TMRA1_CSCTRL,0x00);

    setRegBitGroup(TMRA1_CTRL,CM,0x06); /* Run counter */
}
```

Figure 12-24. One-Shot Mode Example

12.15.9 CASCADE-COUNT Mode

In counter modes 001 to 110, two, three or four counters can be "cascaded". This can be used to create very long time bases. In these modes each counters will operate as a large ripple counter. Each counter is still a synchronous counter but the output of one counter will ripple into the next counter. The side effect of this is that it is possible to read the four counter values and get a invalid value. So if it is required to cascade the counters to generate long time bases and it is required to be able to read all of the timers and get the correct value, then the cascade mode (111) should be used.

Cascade mode is set by setting the counter mode bit field to '111'. In this mode, the counter's input is connected to the output of previous counter. The counter will count up and down as compare events occur in the previous counter. This "Cascade" mode enables multiple counters to be cascaded to yield longer counter lengths just as in the previous mode (001 to 110), but now the counter values can be read at any time.

Up to four counters may be cascaded to create a 64 bit wide synchronous counter.

Whenever any counter is read within a counter module, all of the counters' values within the module are captured in their respective Hold registers. This action supports the reading of a cascaded counter chain. First read any counter of a cascaded counter chain, then read the hold registers of the other counters in the chain. The Cascaded counter mode is synchronous.

There are 6 possible combinations of counters in this "cascade" mode.

- Cascade 0 into 1. Counter 1 is set to cascade mode.
- Cascade 1 into 2. Counter 2 is set to cascade mode.
- Cascade 2 into 3. Counter 3 is set to cascade mode.
- Cascade 0 into 1 into 2. Counters 1 and 2 are set to cascade mode.
- Cascade 1 into 2 into 3. Counters 2 and 3 are set to cascade mode.
- Cascade 0 into 1 into 2 into 3. Counters 1, 2, and 3 are set to cascade mode.

In the previous modes (001 to 110) there are many more combinations. This is because in these ripple modes of cascading counters, the output of any counter can be used as the input to any counter except itself.


```

// (See Processor Expert TimerInt bean.)
// This example generates an interrupt every 30 seconds,
// assuming the chip is operating at 60 MHz.
//
// To do this, counter 2 is used to count 60,000 IP_bus clocks, which means it
// will compare and reload every 0.001 seconds.
// Counter 3 is cascaded and used to count the 0.001 second ticks and
// generate the desired interrupt interval.
//
void TimerInt_Init(void)
{
// Set counter 2 to count IP_bus clocks
/* TMRA2_CTRL: CM=0,PCS=8,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
setReg(TMRA2_CTRL,0x1020); /* Stop all functions of the timer */

// Set counter 3 as cascaded and to count counter 2 outputs
/* TMRA3_CTRL: CM=7,PCS=6,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=0 */
setReg(TMRA3_CTRL,0xEC20); /* Set up cascade counter mode */

/* TMRA3_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
setReg(TMRA3_SCTRL,0x00);
/* TMRA2_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
setReg(TMRA2_SCTRL,0x00);

setReg(TMRA3_CNTR,0x00); /* Reset counter register */
setReg(TMRA2_CNTR,0x00);
setReg(TMRA3_LOAD,0x00); /* Reset load register */
setReg(TMRA2_LOAD,0x00);
setReg(TMRA3_COMP1, 30000); /* milliseconds in 30 seconds */
setReg(TMRA3_CMPLD1,30000);
setReg(TMRA2_COMP1, 60000); /* Set to cycle every millisecond */
setReg(TMRA2_CMPLD1,60000);

/* TMRA3_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
TCF2EN=0,TCF1EN=1,TCF2=0,TCF1=0,CL2=0,CL1=1 */
setReg(TMRA3_CSCTRL,0x41); /* Enable compare 1 interrupt and */
/* compare 1 preload */
/* TMRA2_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=1 */
setReg(TMRA2_CSCTRL,0x01); /* Enable Compare 1 preload */

setRegBitGroup(TMRA2_CTRL,CM,0x01); /* Run counter */
}

```

Figure 12-25. Generate Periodic Interrupt Cascading Two Counters

12.15.10 PULSE-OUTPUT Mode

If the counter is setup for COUNT mode (mode = 001), and the OFLAG Output mode is set to '111' (Gated Clock Output), and the Count Once bit is set, then the counter will output a pulse stream of pulses that has the same frequency of the selected clock source, and the number of output pulses is equal to the compare value minus the init value. This mode is useful for driving step motor systems.

NOTE

This does not work if the Primary Count Source is set to 1000 (IP_bus/1).

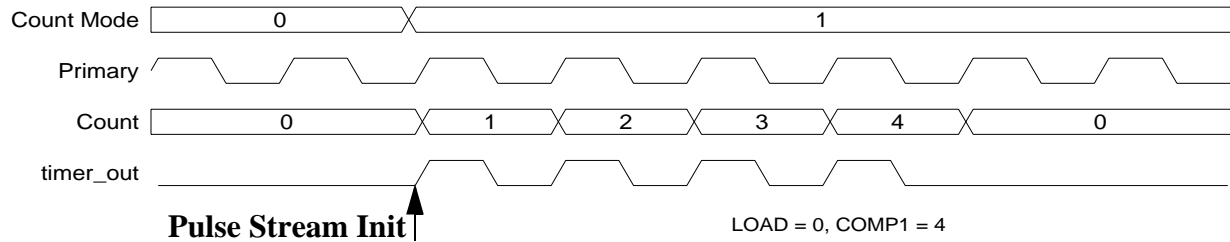


Figure 12-26. Pulse Output Mode

```
//      (See Processor Expert PulseStream bean.)
// This example generates six 10ms pulses, from TA1 output.
// Assuming the chip is operating at 60 MHz.
//
// To do this, timer 3 is used to generate a clock with a period of 10ms.
//
// Timer 1 is used to gate these clocks and count the number of pulses that have
// been generated.
//
void PulseStream_Init(void)
{
// Select IP_bus_clk/16 as the clock source for Timer A3
/* TMRA3_CTRL: CM=0,PCS=0x0C,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=3 */
setReg(TMRA3_CTRL,0x1823);          /* Set up mode */
/* TMRA3_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
setReg(TMRA3_SCTRL,0x00);
setReg(TMRA3_LOAD,0x00);           /* Reset load register */
setReg(TMRA3_COMP1,37500);         /* (16 * 37500) / 60e6 = 0.01 sec */
/* TMRA3_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
setReg(TMRA3_CSCTRL,0x00);        /* Set up comparator control register */

// Timer 3 output is the clock source for this timer.
/* TMRA1_CTRL: CM=0,PCS=7,SCS=0,ONCE=1,LENGTH=1,DIR=0,Co_INIT=0,OM=7 */
setReg(TMRA1_CTRL,0x0E67);        /* Set up mode */
/* TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=1 */
setReg(TMRA1_SCTRL,0x01);
setReg(TMRA1_CNTR,0x00);          /* Reset counter register */
setReg(TMRA1_LOAD,0x00);          /* Reset load register */
setReg(TMRA1_COMP1,0x04);         /* Set up compare 1 register */

// set to interrupt after the last pulse
/* TMRA1_CSCTRL: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,
TCF2EN=0,TCF1EN=1,TCF2=0,TCF1=0,CL2=0,CL1=0 */
setReg(TMRA1_CSCTRL,0x40);        /* Set up comparator control register */

// Finally, start the counters running
setReg(TMRA3_CNTR,0);             /* Reset counter */

setRegBitGroup(TMRA3_CTRL,CM,0x01); /* Run source clock counter */
setRegBitGroup(TMRA1_CTRL,CM,0x01); /* Run counter */
}

```

Figure 12-27. Pulse Outputs Using Two Counters

12.15.11 FIXED-FREQUENCY PWM Mode

If the counter is setup for COUNT mode (mode = 001), count through roll-over (Count Length = 0), continuous count (Count Once = 0) and the OFLAG Output mode is '110' (set on compare, cleared on counter roll-over) then the counter output yields a Pulse Width Modulated (PWM) signal with a frequency equal to the count clock frequency divided by 65,536 and a pulse width duty cycle equal to the compare value divided by 65,536. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters.

```
//      (See Processor Expert PWM bean.)
//      This example uses TMRA0 for Fixed-Frequency PWM mode timing.
//
//      The timer will count IP_bus clocks continuously until it rolls over.
//      This results in a PWM period of 65536 / 60e6 = 1092.267 usec
//
//      Initially, an output pulse width of 25 usec is generated ( 1500 / 60e6 )
//      giving a PWM ratio of 1500 / 65536 = 2.289%
//      This pulse width can be changed by changing the COMP1 register value (using CMPLD1).
//
void PWM1_Init(void)
{
    setReg(TMRA0_CNTR,0);          /* Reset counter */
    /* TMRA0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
       Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=1,OPS=0,OEN=1 */
    setReg(TMRA0_SCTRL,0x05);     /* Enable output */
    setReg(TMRA0_COMP1,1500);     /* Store initial value to the duty-compare register
*/

    /* TMRA0_CTRL: CM=1,PCS=8,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=6 */
    setReg(TMRA0_CTRL,0x3006);    /* Run counter */
}
```

Figure 12-28. Fixed-Frequency PWM Mode Example

12.15.12 VARIABLE-FREQUENCY PWM Mode

If the counter is setup for COUNT mode (mode = 001), count till compare (Count Length = 1), continuous count (Count Once = 0) and the OFLAG Output mode is '100' (toggle OFLAG and alternate compare registers) then the counter output yields a Pulse Width Modulated (PWM) signal whose frequency and pulse width is determined by the values programmed into the COMP1 and COMP2 registers, and the input clock frequency. This method of PWM generation has the advantage of allowing almost any desired PWM frequency and/or constant on or off periods. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters. The CMPLD1 and CMPLD2 registers are especially useful for this mode, as they allow the programmer time to calculate values for the next PWM cycle while the PWM current cycle is underway.

To set up the TMR to run in Variable Frequency PWM mode with compare preload please use the following setup for the specific counter you would like to use. When performing the setup it is suggested that the TMR_CTRL register be updated last as the counter will start counting if the count mode is changed to any value other than 3'b000. (assuming the primary count source is already active)

Timer Control Register (CTRL)

- Count Mode = 3'b001 (count rising edges of primary source)

Timer Module (TMR)

- Primary Count Source = 4'b1000 (IP Bus clock for best granularity for waveform timing)
- Secondary Count Source = Any (ignored in this mode)
- Count Once = 1'b0 (want to count repeatedly)
- Count Length = 1'b1 (want to count until compare value is reached and re-initialize counter register)
- Direction = Any (user's choice. The compare register values need to be chosen carefully to account for things like roll-under, etc.)
- Co-channel Init = 1'b0 (user can set this if they need this function)
- Output Mode = 3'b100 (Toggle OFLAG output using alternating compare registers)

Timer Status and Control Register (SCTRL)

- OEN = 1'b1 (Output enable to allow OFLAG output to be put on an external pin. Set this bit as needed)
- OPS = Any (user's choice)
- Make sure the rest of the bits are cleared for this register. We will enable interrupts in the Comparator Status and Control Register instead of in this register.

Comparator Status and Control Register (CSCTRL)

- TCF2 enable = 1'b1 (allow interrupt to be issued when TCF2 is set)
- TCF1 enable = 1'b0 (do not allow interrupt to be issued when TCF1 is set)
- TCF1 = 1'b0 (clear timer compare 1 interrupt source flag. This is set when counter register equals compare register 1 value and OFLAG is low)
- TCF2 = 1'b0 (clear timer compare 2 interrupt source flag. This is set when counter register equals compare register 2 value and OFLAG is high)
- CL1[1:0] = 2'b10 (load compare register when TCF2 is asserted)
- CL2[1:0] = 2'b01 (load compare register when TCF1 is asserted)

Interrupt Service Routines

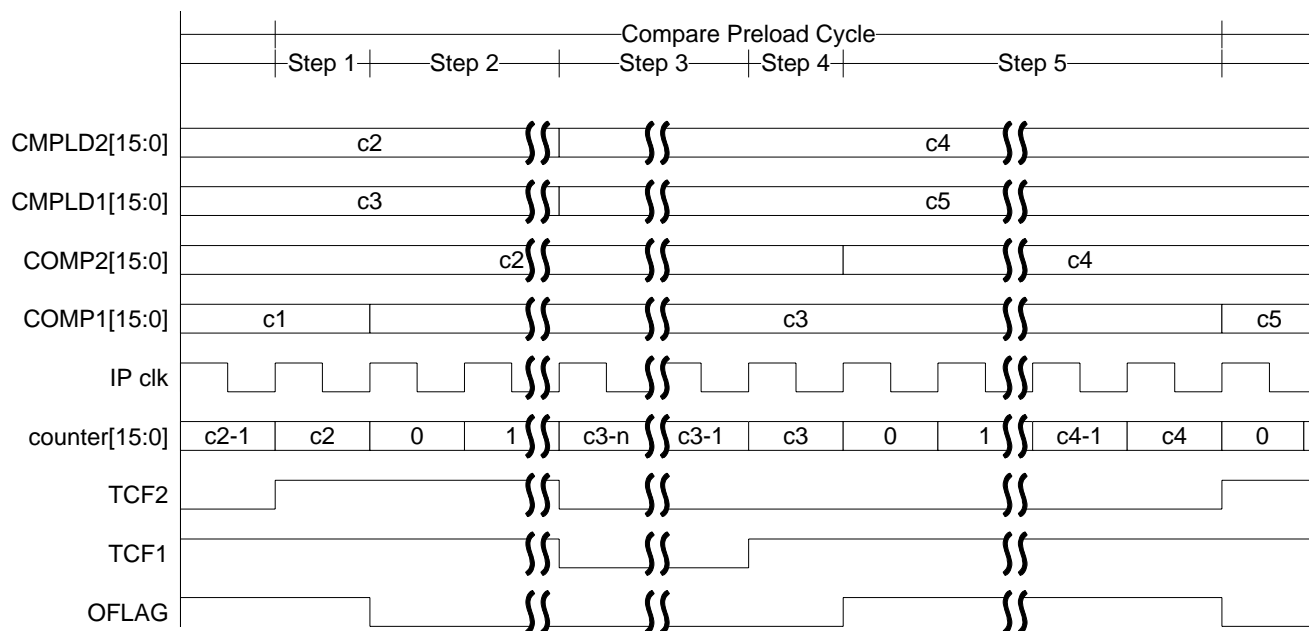
To service the TCF2 interrupts generated by the Timer, the interrupt controller must be configured to enable the interrupts for the particular timer being used. Additionally the user will need to write an interrupt service routine to do at a minimum the following:

- Clear the TCF2 and TCF1 flags.
- Calculate and write new values for both CMPLD1[15:0] and CMPLD2[15:0].

Timing

Figure 12-29 contains the timing for using the compare preload feature. The compare preload cycle begins with a compare event on COMP2 causing TCF2 to be asserted. COMP1 is loaded with the value in the CMPLD1 (c3) one IP Bus clock later. In addition an interrupt is asserted by the timer and the interrupt service routine is executed during which both comparator load registers are updated with new values (c4 and c5). When TCF1 is asserted, COMP2 is loaded with the value in CMPLD2 (c4). And on the subsequent TCF2 event, COMP1 is loaded with the value in CMPLD1 (c5). The cycle starts over again as an interrupt

is asserted and the interrupt service routine clears TCF1 and TCF2 and calculates new values for CMPLD1 and CMPLD2.



Step 1 -- CNTR matches COMP2 value. TCF2 is asserted and an interrupt request is generated.

Step 2 -- One clock later, OFLAG toggles, CMPLD1 is copied to COMP1, LOAD is copied to CNTR, the counter starts counting.

Step 3 -- The interrupt service routine clears TCF1 and TCF2 and the ISR loads CMPLD1 and CMPLD2 with the values for the next cycle. The counter continues counting until CNTR matches COMP1.

Step 4 -- TCF1 is asserted. One clock later, OFLAG toggles, CMPLD2 is copied to COMP2, LOAD is copied to CNTR and the counter starts counting.

Step 5 -- The counter continues counting until CNTR matches COMP2.

Figure 12-29. Compare Load Timing

```

//      (See Processor Expert PPG [Programmable Pulse Generator] bean.)
// This example starts with an 11 msec with a 31 msec cycle.
// Assuming the chip is operating at 60 MHz, the timer use IP_bus_clk/32 as its
// clock source.
//
// Initial pulse period:  60e6/32 clocks/sec * 31 ms = 58125 total clocks in period
// Initial pulse width:   60e6/32 clocks/sec * 11 ms = 20625 clocks in pulse
//
//
// Once the initial values of COMP1/CMPLD1 and COMP2/CMPLD2 are set the pulse width
// can be varied by load new values of CMPLD1 and CMPLD2 on each compare interrupt.
//
//
void PPG1_Init(void)
{
    setReg(TMRA0_LOAD,0);           /* Clear load register */
    setReg(TMRA0_CNTR,0);          /* Clear counter */

    /* TMRA0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
        Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=1,OPS=0,OEN=1 */
    setReg(TMRA0_SCTRL,5);         /* Set Status and Control Register */

// Set compare preload operation and enable an interrupt on compare2 events.
/* TMRA0_CSCTRL: TCF2EN=1,TCF1EN=0,TCF2=0,TCF1=0,CL21=0,CL20=1,CL11=1,CL10=0 */
setReg(TMRA0_CSCTRL,0x86);       /* Set Comparator Status and Control Register */

setReg(TMRA0_COMP1,20625);        /* Set the pulse width of the off time */
setReg(TMRA0_CMPLD1,20625);      /* Set the pulse width of the off time */
setReg(TMRA0_COMP2,58125-20625); /* Set the pulse width of the on time */
setReg(TMRA0_CMPLD2,58125-20625); /* Set the pulse width of the on time */

    /* TMRA0_CTRL: CM=1,PCS=0xD,SCS=0,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=4 */
    setRegBits(TMRA0_CTRL,0x3A24); /* Set variable PWM mode and run counter */
}

```

Figure 12-30. Variable Frequency PWM Mode

12.15.13 Usage of Compare Registers

The dual compare registers (COMP1 and COMP2) provide a bidirectional modulo count capability. The COMP1 register is used when the counter is *counting up*, and the COMP2 register is used when the counter is *counting down*. Alternating the compare mode is the only exception.

The COMP1 register should be set to the desired maximum count value or \$FFFF to indicate the maximum unsigned value prior to roll-over, and the COMP2 register should be set to the minimum count value or \$0000 to indicate the minimum unsigned value prior to roll-under.

If the output mode is set to 100, the OFLAG will toggle while using alternating compare registers. In this variable frequency PWM mode, the COMP2 value defines the desired pulse width of the on time, and the COMP1 register defines the off time. The variable frequency PWM mode is defined for positive counting only.

Use caution when changing COMP1 and COMP2 while the counter is active. If the counter has already passed the new value, it will count to \$FFFF or \$0000, roll over, then begin counting toward the new value. The check is: Count = CMPx, *not* Count > COMP1 or Count < COMP2.

With the use of the CMPLD1 and CMPLD2 registers to compare values will help to minimize this problem.

12.15.14 Usage of Compare Load Registers

The CMPLD1, CMPLD2 and CSCTRL registers offer a high degree of flexibility for loading compare registers with user-defined values on different compare events. To ensure correct functionality while using these registers we strongly suggest using the following method described in this section.

The purpose of the compare load feature is to allow quicker updating of the compare registers. In the past, a compare register could be updated using interrupts. However, because of the latency between an interrupt event occurring and the service of that interrupt, there was the possibility that the counter may have already counted past the new compare value by the time the compare register is updated by the interrupt service routine. The counter would then continue counting until it rolled over and reached the new compare value.

To address this, the compare registers are now updated in hardware in the same way the counter register is re-initialized to the value stored in the load register. The compare load feature allows the user to calculate new compare values and store them in to the comparator load registers. When a compare event occurs, the new compare values in the comparator load registers are written to the compare registers eliminating the use of software to do this.

The compare load feature is intended to be used in variable frequency PWM mode. The COMP1 register determines the pulse width for the logic low part of OFLAG and COMP2 determines the pulse width for the logic high part of OFLAG. The period of the waveform is determined by the COMP1 and COMP2 values and the frequency of the primary clock source. See [Figure 12-31](#).

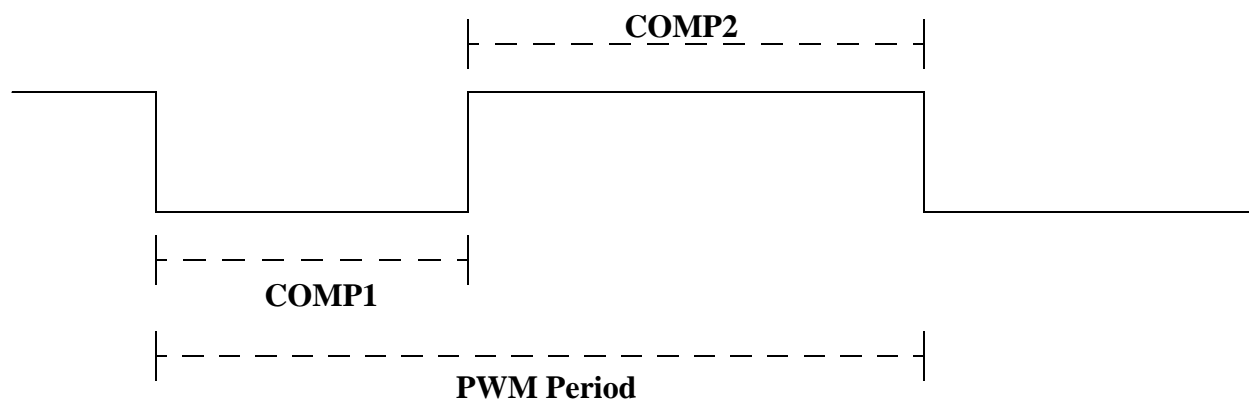


Figure 12-31. Variable PWM Waveform

Should we desire to update the duty cycle or period of the above waveform, we would need to update the COMP1 and COMP2 values using the compare load feature.

12.15.15 Usage of Capture Register

The Capture Register stores a copy of the counter's value when an input edge (positive, negative, or both) is detected. Once a capture event occurs, no further updating of the Capture register will occur until the IEF (Input Edge Flag) is cleared by writing a zero to the IEF.

12.16 Resets

12.17 General

The TMR module can only be reset by the RST_B signal. This forces all registers to their reset state and clears the OFLAG signal if it is asserted. The counter will be turned off until the settings in the Control register are changed.

Table 12-13. Reset Summary

Reset	Priority	Source	Characteristics
RST_B	n/a	Hardware Reset	Full System Reset

12.18 Clocks

12.19 General

The Timer only receives the IPBus Clock.

Table 12-14. Clock Summary

Clock	Priority	Source	Characteristics
lp_clk	n/a	IP Bus Bridge	IP Bus Clock

12.20 Interrupts

12.21 General

The TMR module can generate 20 interrupts, Five for each of the four counters/channels.

Table 12-15. Interrupt Summary

Core Interrupt	Interrupt	Description
TMR Channel 0	TMR0_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 0
	TMR0_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 0
	TMR0_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 0
	TMR0_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 0
	TMR0_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 0
TMR Channel 1	TMR1_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 1
	TMR1_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 1
	TMR1_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 1
	TMR1_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 1
	TMR1_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 1
TMR Channel 2	TMR2_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 2
	TMR2_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 2
	TMR2_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 2
	TMR2_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 2
	TMR2_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 2

Table 12-15. Interrupt Summary (continued)

Core Interrupt	Interrupt	Description
TMR Channel 3	TMR3_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 3
	TMR3_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 3
	TMR3_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 3
	TMR3_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 3
	TMR3_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 3

12.22 Description of Interrupt Operation

12.22.1 Timer Compare Interrupts

These interrupts are generated when a successful compare occurs between a counter and its compare registers while the Timer Compare Flag Interrupt Enable is set in the TMR_SCTRL. These interrupts are cleared by writing a zero to the TCF bit in the appropriate TMR_SCTRL.

When a timer compare interrupt is set in the TMR_SCTRL and the Compare Load registers are available, one of the following two interrupts will also be asserted.

12.22.1.1 Timer Compare 1 Interrupts (Available with Compare Load Feature)

These interrupts are generated when a successful compare occurs between a counter and its COMP1 register while the Timer Compare 1 Interrupt Enable is set in the CSCTRL register. These interrupts are cleared by writing a zero to the TCF1 bit in the appropriate CSCTRL.

12.22.1.2 Timer Compare 2 Interrupts (Available with Compare Load Feature)

These interrupts are generated when a successful compare occurs between a counter and its COMP2 register while the Timer Compare 2 Interrupt Enable is set in the CSCTRL register. These interrupts are cleared by writing a zero to the TCF1 bit in the appropriate CSCTRL.

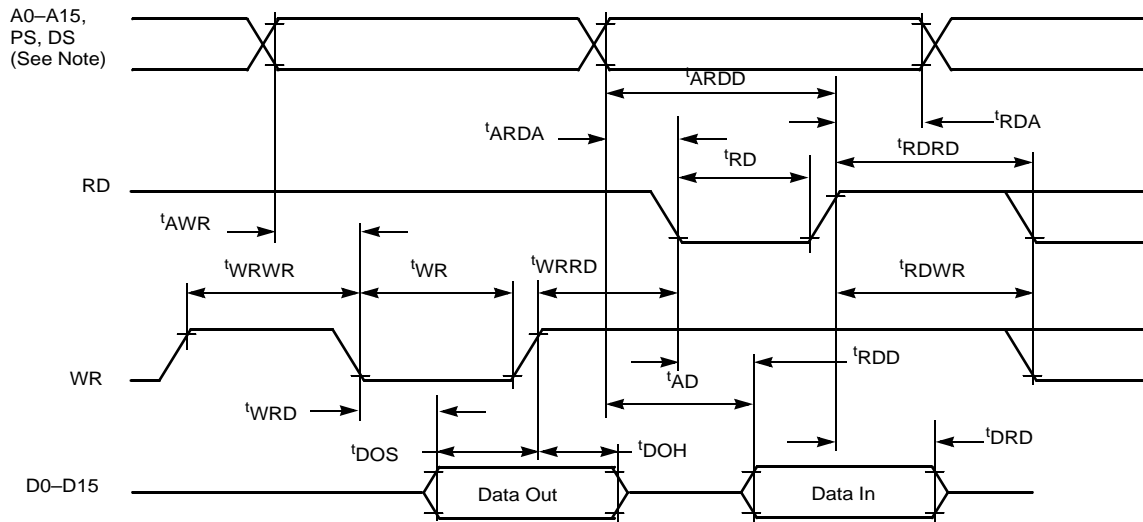
12.22.2 Timer Overflow Interrupts

These interrupts are generated when a counter rolls over its maximum value while the Timer Overflow Flag Interrupt Enable bit is set in the TMR_SCTRL. These interrupts are cleared by writing zero to the TOF bit of the appropriate TMR_SCTRL.

12.22.3 Timer Input Edge Interrupts

These interrupts are generated by a transition of the input signal (either positive or negative depending on IPS setting) while the Input Edge Flag Interrupt Enable bit is set in the TMR_SCTRL. These interrupts are cleared by writing a zero to the IEF bit of the appropriate TMR_SCTRL.

12.23 Timing Specifications



Note: During read-modify-write instructions and internal instructions, the address lines do not change state.

Figure 12-32. Timing Specifications

Chapter 13

Universal Asynchronous Receiver/Transmitter Module (UART)

13.1 Overview

This chapter describes the Universal Asynchronous Receiver Transmitter (UART) module. The MC1322x has two independent UART modules designated UART1 and UART2. These can be used to connect to traditional RS232 serial ports or to communicate with other embedded controllers.

Each UART has an independent fractional divider, baud rate generator that is clocked by the peripheral bus clock (generated from the reference oscillator divided by the CRM prescaler; typically 24 MHz max) which enables a broad range of baud rates up to 1,843.2 kbaud. Transmit and receive use a common baud rate for each module.

The UART has many advanced features useful for reliable, high-throughput serial communication. Hardware flow control, independent 32-byte receive and transmit FIFOs with level indicators, high baud rate, and error detection.

13.2 Features

The UART module features include:

- 8-bit only data
- Programmable one or two stop bits
- Programmable parity (even, odd, and none)
- Full duplex four-wire serial interface (RXD, TXD, RTS, and CTS)
- Programmable hardware flow control for RTS and CTS signals with programmable sense (high true/low true)
- Independent 32-byte receive FIFO and 32-byte transmit FIFO
- Status flags for flow control and FIFO states
- Receiver detects framing errors, start bit error, break characters, parity errors, and overrun errors.
- Voting logic for improved noise immunity (16X/8X oversampling)
- Maskable interrupt request
- Time-out interrupt timer, which times out after eight non-present characters
- Receiver and transmitter enable/disable
- Low-power modes
- Baud rate generator to provide any multiple-of-2 baud rate between 1.2 kbaud and 1,843.2 kbaud
- Software reset.

13.3 Block Diagram

The UART module is shown in Figure 13-1.

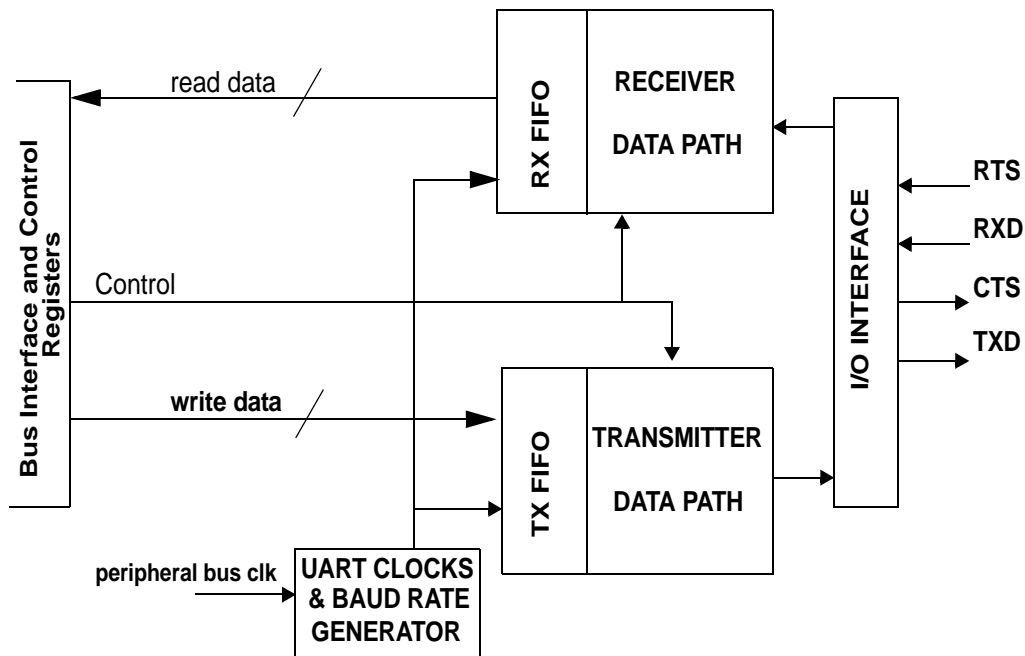


Figure 13-1. Top Level UART Block Diagram

13.4 Functional Description

The UART consists of control and status registers, a baud rate generator, TX logic with a 32-byte FIFO, and TX logic with a separate 32-byte FIFO. The transmitter and receiver operate independently, although they use the same baud rate generator. The following sections describe the function of the UART.

13.4.1 Signal Descriptions

Table 13-1 lists UART signal names and their descriptions.

Table 13-1. UART Signals

Signal Names	Direction	Active	Comments
UARTx_RTS	Input	Programmable	Request to Send flow control input
UARTx_CTS	Output	Programmable	Clear to Send flow control output
UARTx_RX	Input	Low	Receive serial data input
UARTx_TX	Output	Low	Transmit serial data output

13.4.2 Baud Rate Generation (Fractional Divider)

The high-speed UART is clocked with the peripheral bus clock (typically 24 MHz). This high frequency clock, plus an internal fractional divider, enables a wide range of frequencies calculated using the following equations:

Eqn. 13-1

$$\begin{aligned} \text{baudrateX16} &= \text{pbclk} \times \left[\frac{\text{INC} + 1}{\text{MOD}} \right] \\ \text{baudrate} &= \frac{\text{baudrateX16}}{16} \end{aligned}$$

or,

Eqn. 13-2

$$\begin{aligned} \text{baudrateX8} &= \text{pbclk} \times \left[\frac{\text{INC} + 1}{\text{MOD}} \right] \\ \text{baudrate} &= \frac{\text{baudrateX8}}{8} \end{aligned}$$

The internal baud rate generator is shown in [Figure 13-2](#).

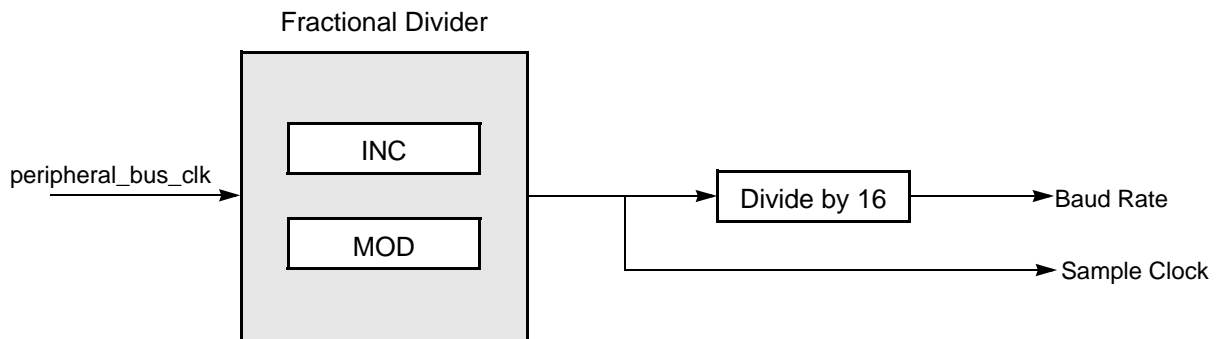


Figure 13-2. UART Baud Rate Generator (16x oversampling)

[Table 13-2](#) and [Table 13-3](#) show sample values for INC and MOD that yield standard baud rates for 8x oversampling at the various input clocks. The INC and MOD values are programmed in the UART Baud Rate Divider Register (UBR).

Table 13-2. Standard Baud Rates for 8x Oversampling (1 of 2)

Baud Rate	peripheral_bus_clk = 13Mhz			peripheral_bus_clk = 15.36Mhz			peripheral_bus_clk = 16Mhz		
	MOD	INC	FREQ	MOD	INC	FREQ	MOD	INC	FREQ
1,200	9,999	6	1137.6	9,999	5	1152.1	9,999	5	1200.1
2,400	9,999	14	2437.7	9,999	11	2304.2	9,999	11	2400.2
4,800	9,999	29	4875.5	9,999	24	4800.5	9,999	23	4800.5
9,600	9,999	58	9588.5	9,999	49	9601.0	9,999	47	9601.0
19,200	9,999	117	19176.9	9,999	99	19201.9	9,999	95	19201.9
28,800	9999	176	28765.4	9,999	149	28802.9	9,999	143	28802.9
38,400	9,999	235	38353.8	9,999	199	38403.8	9,999	191	38403.8
57,600	9,999	353	57530.8	9,999	299	57605.8	9,999	287	57605.8
115,200	9,999	708	115224.0	9,999	599	115211.5	9,999	575	115211.5
230,400	9,999	1417	230448.0	9,999	1199	230423.0	9,999	1151	230423.0
460,800	9,999	2834	460733.6	9,999	2399	460846.1	9,999	2303	460846.1
921,600	9,999	5670	921629.7	9,999	4799	921692.2	9,999	4607	921692.2
1,843,200	9,999	NA ¹	—	9999	9598	1843192.3	9,999	9214	1843184.3
Baud Rate	peripheral_bus_clk = 16.8Mhz			peripheral_bus_clk = 19.2Mhz			peripheral_bus_clk = 19.44Mhz		
	MOD	INC	FREQ	MOD	INC	FREQ	MOD	INC	FREQ
1,200	9,999	5	1260.1	9,999	4	1200.1	9,999	4	1215.1
2,400	9,999	10	2310.2	9,999	9	2400.2	9,999	9	2430.2
4,800	9,999	22	4830.5	9,999	19	4800.5	9,999	19	4860.5
9,600	9,999	45	9661.0	9,999	39	9601.0	9,999	39	9721.0
19,200	9,999	90	19111.9	9,999	79	19201.9	9,999	78	19198.9
28,800	9,999	136	28772.9	9,999	119	28802.9	9,999	118	28919.9
38,400	9,999	182	38433.8	9,999	159	38403.8	9,999	157	38397.8
57,600	9,999	273	57545.8	9,999	239	57605.8	9,999	236	57596.8
115,200	9,999	548	115301.5	9,999	479	115211.5	9,999	473	115193.5
230,400	9,999	1096	230393.0	9,999	959	230423.0	9,999	947	230387.0
460,800	9,999	2193	460786.1	9,999	1919	460846.1	9,999	1895	460774.1
921,600	9,999	4387	921572.2	9,999	3839	921692.2	9,999	3791	921548.2
1,843,200	9,999	8775	1843144.3	9,999	7678	1843144.3	9,999	7583	1843096.3

¹This baud rate is not valid running at a system clock of 13 MHz.

Table 13-3. Standard Baud Rates for 8x Oversampling (2 of 2)

Baud Rate	peripheral_bus_clk = 19.8Mhz			peripheral_bus_clk = 24Mhz			peripheral_bus_clk = 26Mhz		
	MOD	INC	FREQ	MOD	INC	FREQ	MOD	INC	FREQ
1,200	9,999	4	1237.6	9,999	3	1200.1	9,999	3	1300.1
2,400	9,999	9	2475.2	9,999	7	2400.2	9,999	6	2275.2
4,800	9,999	18	4703.0	9,999	15	4800.5	9,999	14	4875.5
9,600	9,999	38	9653.5	9,999	31	9601.0	9,999	29	9751.0
19,200	9,999	77	19306.9	9,999	63	19201.9	9,999	58	19176.9
28,800	9,999	115	28712.9	9,999	95	28802.9	9,999	88	28927.9
38,400	9,999	154	38366.3	9,999	127	38403.8	9,999	117	38353.8
57,600	9,999	232	57673.3	9,999	191	57605.8	9,999	176	57530.8
115,200	9,999	464	115099.0	9,999	383	115211.5	9,999	353	115061.5
230,400	9,999	930	230445.5	9,999	767	230423.0	9,999	708	230448.0
460,800	9,999	1861	460891.1	9,999	1535	460846.1	9,999	1417	460896.1
921,600	9,999	3722	921534.7	9,999	3071	921692.2	9,999	2834	921467.1
1,843,200	9,999	7446	1843316.8	9999	6142	1843084.3	9999	5670	1843259.3

Table 13-4 and Table 13-5 show sample values for INC and MOD that yield standard baud rates for 16x oversampling at the various input clocks.

Table 13-4. Standard Baud Rates for 16x Oversampling (1 of 2)

Baud Rate	peripheral_bus_clk = 13Mhz			peripheral_bus_clk = 15.36Mhz			peripheral_bus_clk = 16Mhz		
	MOD	INC	FREQ	MOD	INC	FREQ	MOD	INC	FREQ
1,200	9,999	14	1218.9	9,999	11	1152.1	9,999	11	1200.1
2,400	9,999	29	2437.7	9,999	24	2400.2	9,999	23	2400.2
4,800	9,999	58	4794.2	9,999	49	4800.5	9,999	47	4800.5
9,600	9,999	117	9588.5	9,999	99	9601.0	9,999	95	9601.0
19,200	9,999	235	19176.9	9,999	199	19201.9	9,999	191	19201.9
28,800	9,999	353	28765.4	9,999	299	28802.9	9,999	287	28802.9
38,400	9,999	472	38435.1	9,999	399	38403.8	9,999	383	38403.8
57,600	9,999	708	57612.0	9,999	599	57605.8	9,999	575	57605.8
115,200	9,999	1417	115224.0	9,999	1199	115211.5	9,999	1151	115211.5
230,400	9,999	2834	230366.8	9,999	2399	230423.0	9,999	2303	230423.0
460,800	9,999	5670	460814.8	9,999	4799	460846.1	9,999	4607	460846.1

Table 13-4. Standard Baud Rates for 16x Oversampling (1 of 2)

921,600	9,999	NA ¹	—	9,999	9598	921596.2	9,999	9214	921592.2
1,843,200	9,999	NA ²	—	9999	NA ³	—	9,999	NA ⁴	—

¹This baud rate is not valid running at a system clock of 13 MHz.

²This baud rate is not valid running at a system clock of 13 MHz.

³This baud rate is not valid running at a system clock of 15.36 MHz.

⁴This baud rate is not valid running at a system clock of 16 MHz.

Table 13-5. Standard Baud Rates for 16x Oversampling (2 of 2)

Baud Rate	peripheral_bus_clk = 16.8Mhz			peripheral_bus_clk = 19.2Mhz		
	MOD	INC	FREQ	MOD	INC	FREQ
1,200	9,999	10	1155.1	9,999	9	1200.1
2,400	9,999	22	2415.2	9,999	19	2400.2
4,800	9,999	45	4830.5	9,999	39	4800.5
9,600	9,999	90	9556.0	9,999	79	9601.0
19,200	9,999	182	19216.9	9,999	159	19201.9
28,800	9,999	273	28772.9	9,999	239	28802.9
38,400	9,999	365	38433.8	9,999	319	38403.8
57,600	9,999	548	57650.8	9,999	479	57605.8
115,200	9,999	1096	115196.5	9,999	959	115211.5
230,400	9,999	2193	230393.0	9,999	1919	230423.0
460,800	9,999	4387	460786.1	9,999	3839	460846.1
921,600	9,999	8775	921572.2	9,999	7678	921572.2
1,843,200	9,999	NA ¹	—	9,999	NA ²	—

¹This baud rate is not valid running at a system clock of 16.8 MHz.

²This baud rate is not valid running at a system clock of 19.2 MHz.

13.4.3 Basic Operation

The CPU accesses the UART through its data and control registers. The UART transmits and receives a characters (bytes) of 8-bit length only. For transmission, data is first written to a 32-byte deep transmitter FIFO (first in, first out). This data is passed to the TX shift register and shifted serially out on the UART TXD pin. For reception, data is received serially from the UART RXD pin and stored in a 32-byte deep receiver FIFO. The received data is read from the receiver FIFO.

Figure 13-3 shows the UART data transfer format. The format supports 8-bit data with programmable odd, even, or no parity. One or two stop bits are also programmable. During transmission, a TX FIFO overrun error can be detected, and the receiver detects framing errors, start bit error, break characters, parity errors, and RX FIFO overrun errors.

The receiver and transmitter FIFOs contain a maskable interrupt that can be configured to interrupt when it reaches a certain level.

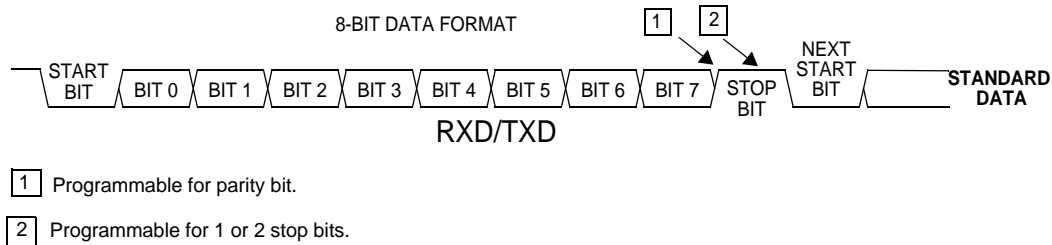


Figure 13-3. Data Transfer Format

The UART also supports hardware RTS and CTS flow control as described in [Section 13.5, “Flow Control”](#).

Each UART module supports an independent interrupt request to the CPU Interrupt Controller with a pre-determine priority level. The interrupt request can be from a number of sources.

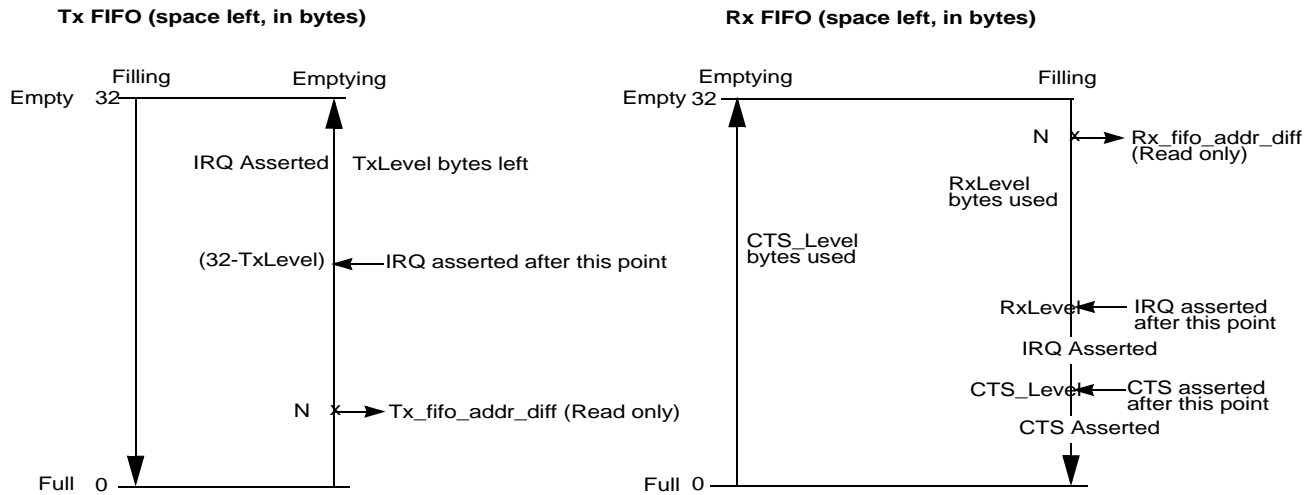
The UART generated baud rate is based upon a configurable divisor and input clock and is common to both the RX and TX blocks. The fractional divider is setup by writing an INCRement and MODulo value to two registers. It is important that the UART is disabled (RxE and TxE equal 0) before writing new values to the INC/MOD registers. After writing to the registers, the receiver and/or transmitter can be enabled (RxE and TxE equal 1).

NOTE

When the UART is disabled (RxE and TxE equal 0) both the Rx and the Tx buffers are flushed, and the status register is updated.

13.4.4 FIFO Operation

The operation of the transmit and receive FIFO counters is shown in [Figure 13-3](#).



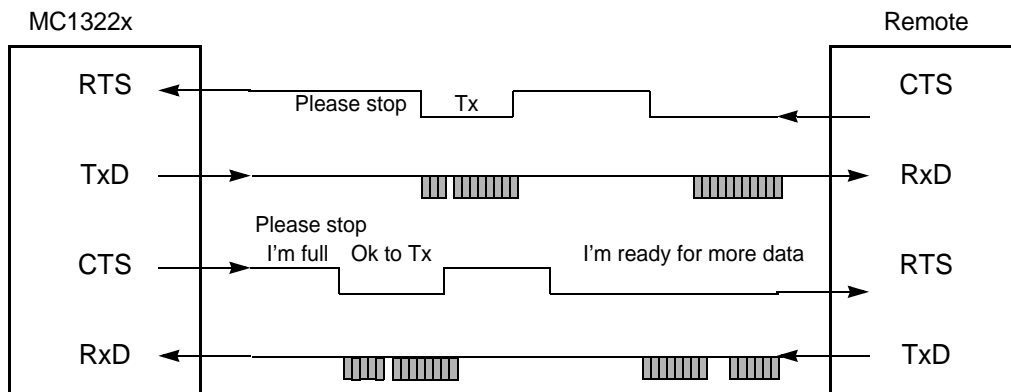
When the TX FIFO is emptying, an interrupt is generated once there are fewer than *TxLevel* bytes left.
 When the RX FIFO is filling, an interrupt is generated once there are more than *RxLevel* bytes filled.

Figure 13-4. TX and RX FIFO-Related Levels

13.5 Flow Control

The UART has hardware support for:

- RTS (Ready To Send) input - means that the MC1322x is allowed to send because the remote unit is ready.
- CTS (Clear To Send) output - means that the MC1322x is ready to accept new data, so the remote unit can transmit when it has data.
- The polarity of RTS/CTS can be changed by changing the control[FCp] bit.
- When the control[FCe]=0 the flow control is disabled. This causes the UART to ignore the RTS input and asserting the CTS pin (that is, the CTS pin is therefore 0 when control[FCp]=0).



NOTE:
 The above signals assume control (FCe) = 0 (enabled and control) and
 (FCp) = 0 (active low RTS/CTS signals).

Figure 13-5. Flow Control

13.6 RX Timeout Counter

The UART contains a counter that provides a time-out function to indicate lack of receive activity.

- The time-out counter is always active after
 - Receive mode is enabled (RxE bit is enabled), AND
 - The first RX character has been received.
- The counter will time-out after lack of 8 characters. The wait time is set to 96 total data bits, which is derived as 8 characters * (1 start + 8 data + 1 parity + 2 stop) bits/character = 96 bits. The bit-time is determined by the programmed baud rate.
- The counter is reset to zero every time a stop bit has been processed correctly, which means, once a byte of data has been received into the RX FIFO.
- On time-out, the Rx FIFO Underrun Error Status (RUE) is set, and in turn, sets the RxRdy Status if it is not already set. An interrupt request (IRQ) will be generated if enabled.
- The time-out counter cannot be used to get a time-out IRQ before the receiver gets any data because the counter is not started until the first character is received.

13.7 Interrupts

Each UART module has a single interrupt request signal that is connected to the interrupt controller. The UART IRQ can be generated from two primary sources which are logically ORed such that either can generate the interrupt request if enabled. [Table 13-6](#) lists the UART interrupt sources and their characteristics.

Table 13-6. UART Interrupt Sources

Item	Status Bit	Mask Bit	Source Description	Interrupt Clear Mechanism
1	TxRdy	MTxR	Transmitter Is Causing An Interrupt - the transmitter can cause an interrupt request from two conditions <ul style="list-style-type: none"> • TX FIFO empty space exceeds TxLevel[4:0] set in UART TxBuffer Control Register (no additional status bit) • TX FIFO Overrun Error - the incoming data overran the FIFO capacity. The TOE Status bit is also set 	Clearing the interrupt is dependent upon the source <ul style="list-style-type: none"> • TX FIFO empty interrupt request (MTxR) is disabled by writing to the TX FIFO • TX FIFO Overrun Error TOE status is cleared by reading the status register
2	RxRdy	MRxR	Receiver Is Causing An Interrupt - the receiver can cause an interrupt request from multiple conditions <ul style="list-style-type: none"> • Number of available bytes in RX FIFO exceeds the value specified by RxLevel[4:0] set in UART RxBuffer Control Register (no additional status bit) • RX FIFO Underrun Error - the RX time-out counter fired. The RUE Status bit is also set • RX FIFO Overrun Error - the incoming data overran the FIFO capacity. The ROE Status bit is also set • Frame/Stop Bit Error - the FE Status bit is also set • Parity Error - the PE Status bit is also set • Start Bit Error - the SE Status bit is also set 	Clearing the interrupt is dependent upon the source <ul style="list-style-type: none"> • RX FIFO data interrupt request (MTxR) is disabled by reading from the RX FIFO • Error status are cleared by reading the status register

13.7.1 Transmitter Sources

As shown in [Table 13-6](#), the transmitter can generate a request from two sources

- TX FIFO empty space exceeds TxLevel[4:0] set in UART TxBuffer Control Register - this interrupt is used to ask for more data in the TX FIFO to keep the UART TX channel running at max capacity. The threshold is programmable.
- TX FIFO Overrun Error - the data written to the TX FIFO overran the FIFO capacity. The TOE Status bit is set. This should not occur if application software is written properly

13.7.2 Receiver Sources

As shown in [Table 13-6](#), the receiver can generate a request from multiple sources

- Number of available received bytes in RX FIFO exceeds the value specified by RxLevel[4:0] set in UART RxBuffer Control Register - this interrupt is used to ask the CPU to fetch data from the RX FIFO. The threshold is programmable
- RX FIFO Underrun Error - the RX time-out counter fired. The RUE Status bit is also set.
- RX FIFO Overrun Error - the incoming RX data overran the FIFO capacity. The ROE Status bit is set.
- RX Frame/Stop Bit Error - the FE Status bit is set
- RX Parity Error - the PE Status bit is set
- RX Start Bit Error - the SE Status bit is set

13.8 UART Register Memory Map

Each UART module is programmed via a set of memory-mapped registers and their respective base addresses are listed in [Table 13-7](#). The register memory map for each module related to the base address is listed in [Table 13-8](#) and shows the 32-bit registers of the UART module.

The control and status registers should be accessed only as 32 bits. The data register may be accessed as 32, 16 or 8 bits.

Table 13-7. Base Addresses of UART Modules

Base Address	UART Designation
0x80005000	UART1
0x8000B000	UART2

Table 13-8. UART Module Register Memory Map

Offset	Name	Access Type	Access Width
Base + 0x00	UART Control Register (UCON)	R/W	32-bit only
Base + 0x04	UART Status Register (USTAT)	R	32-bit only
Base + 0x08	UART Data Register (UDATA)	R/W	8-bit, 16-bit, or 32-bit
Base + 0x0C	UART RxBuffer Control Register (URxCON)	R/W	32-bit only
Base + 0x10	UART TxBuffer Control Register (UTxCON)	R/W	32-bit only
Base + 0x14	UART CTS Level Control Register (UCTS)	R/W	32-bit only
Base + 0x18	UART Baud Rate Divider Register (UBR)	R/W	32-bit only

13.9 UART Registers

The following sections provide descriptions of the UART registers.

13.9.1 UART Control Register (UCON)

The UART Control Register is used to specify transmission parameters, such as flow control, stop bits, parity, etc.

UCON								Base + 0x00								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TST	mRxr	mTxr	FCe	FCp	xTIM	Res	0	Tx_oen_b	conTx	SB	ST2	EP	PEN	RxE	TxE
W																
RST	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

= writes have no effect and terminate without transfer error exception

Table 13-9. AGC Software Override Register Bit Descriptions

Bit Number	Description	Operation
16-31	Reserved	
15	TST — Test Loop-Back	1 = Test mode 0 = Normal operation
14	MRxR — Mask RxRDY Interrupt. Used to enable or disable (mask) receive interrupt request	1 = Masked 0 = Enabled
13	MTxR — Mask TxRDY Interrupt. Used to enable or disable (mask) transmit interrupt request	1 = Masked 0 = Enabled
12	FCe — Flow Control Enable	1 = Enabled 0 = Disable
11	FCp — Flow Control polarity	1 = Assert RTS to enable transmission 0 = Deassert RTS to enable transmission
10	xTIM — Times Of Oversampling	1 = 16 times oversampling 0 = 8 times oversampling
9	Reserved	
8	Reserved	

Table 13-9. AGC Software Override Register Bit Descriptions

Bit Number	Description	Operation
7	Tx_oen_b — TXD Output Enable. This bit can disable the TXD output which is useful when using loopback mode and not wanting to send data.	1 = Disable TXD output 0 = Enable TXD output (default)
6	conTx — Continuous Tx (Test Mode)	1 = Enable 0 = Disable
5	SB — Send Break	1 = Send Break 0 = No Break
4	ST2 — Stop Bits	1 = Two Stop Bits 0 = One Stop Bit
3	EP — Even Parity	1 = Odd 0 = Even
2	PEN — Parity Enable	1 = Enable 0 = Disable
1	RxE — Rx Enable	1 = Enable 0 = Disable
0	TxE — Tx Enable	1 = Enable 0 = Disable

13.9.2 UART Status Register (USTAT)

The UART Status Register indicates interrupt status and any errors that have been detected during transmission, such as FIFO buffer overflow or underflow, parity error, and frame, start, or stop bit error. This register is read-only.

NOTE

- Bits[7:6] are cleared when the respective condition allows it.
- Status bits[5:0] are cleared when the register is read.

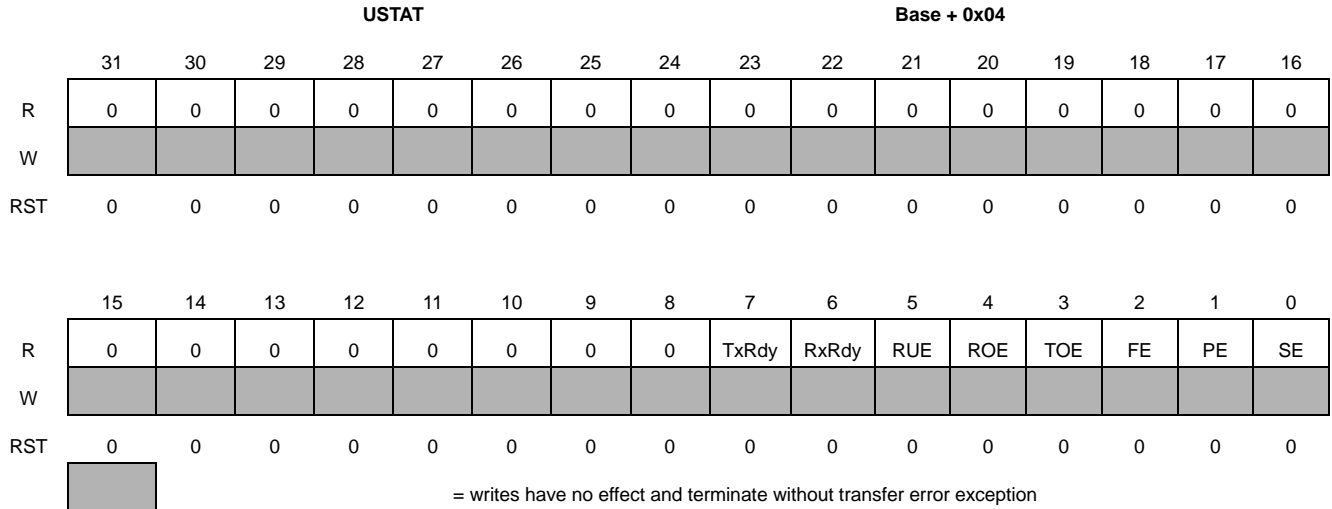


Table 13-10. USTAT Register Bit Descriptions

Bit Number	Description	Operation
8-31	Reserved	1 = Interrupt pending 0 = No interrupt
7	TxRdy — Transmitter Is Causing An Interrupt	1 = Interrupt pending 0 = No interrupt
6	RxRdy — Receiver Is Causing An Interrupt	1 = Error occurred 0 = No error
5	RUE — Rx FIFO Underrun Error	1 = Error occurred 0 = No error
4	ROE — Rx FIFO Overrun Error	1 = Error occurred 0 = No error
3	TOE — Tx FIFO Overrun Error	1 = Error occurred 0 = No error
2	FE — Frame/Stop Bit Error	1 = Error occurred 0 = No error

Table 13-10. USTAT Register Bit Descriptions

Bit Number	Description	Operation
1	PE — Parity Error	1 = Error occurred 0 = No error
0	SE — Start Bit Error	1 = Error occurred 0 = No error

13.9.3 UART Data Register (UDATA)

The UART Data Register is used to write the UART transmit FIFO buffer with bytes that are to be transmitted, and to read the received bytes from the UART receive FIFO buffer. An 8-bit, 16-bit, or 32-bit access will all yield only one byte of valid data.

UDATA																Base + 0x08																					
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	Rx_data											
W	[Shaded]																	[Shaded]								Tx_Data											
RST	Undefined																	Undefined								Undefined											

Table 13-11. UDATA Register Bit Descriptions

Bit Number	Description	Operation
8-31	Reserved	
0-7	Rx_data[7:0] — Received Byte. This bit field contains the 8-bit data that has been received.	
0-7	Tx_data[7:0] — Byte To Transmit. This bit field contains the 8-bit data to be transmitted.	

13.9.4 UART Buffer Control Registers

The UART buffers use threshold values to generate interrupts. These thresholds can be used to specify that an interrupt is generated before the transmit FIFO has become completely empty, or some time before the receive FIFO has been completely filled. These “early warnings” allow relaxed interrupt response times.

13.9.4.1 UART RxBuffer Control Register (URxCON)

The UART RxBuffer Control Register provides the level of data within the RX FIFO and also sets the threshold for the interrupt request that is generated when the receive buffer is considered sufficiently filled for retrieving data. Two fields are used

- Rx_fifo_addr_diff[5:0] (read-only) - this value indicates the number of bytes currently buffered in the receive FIFO buffer. The number of bytes is between 0 (empty) and 32 (full) bytes. If this value goes negative (i.e., exceeding FIFO capacity), the write enable is deasserted and no more writes will occur.
- RxLevel[4:0] (write-only) -When the number of bytes filled in the receive FIFO exceeds the value specified by RxLevel, an interrupt is generated.

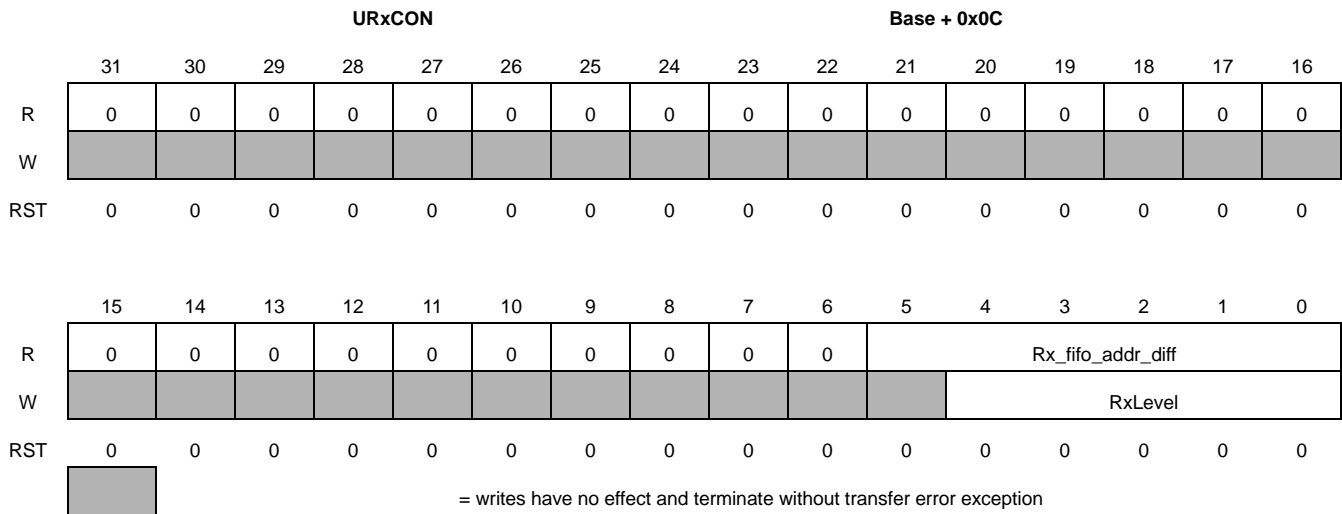


Table 13-12. URxCON Register Bit Descriptions

Bit Number	Description	Operation
6-31	Reserved	
0-5	Rx_fifo_addr_diff[5:0] — Receive buffer full level (read-only). This value indicates the number of bytes currently buffered in the receive FIFO buffer. The number of bytes is between 0 (empty) and 32 (full) bytes.	
0-4	RxLevel[4:0] — Receive buffer level (write-only). When the number of bytes filled in the receive FIFO exceeds the value specified by RxLevel, an interrupt is generated.	

13.9.4.2 UART TxBuffer Control Register (UTxCON)

The UART TxBuffer Control Register

provides the number of free bytes within the TX FIFO and also sets the threshold for the interrupt request that is generated when the transmit buffer is considered sufficiently empty for writing additional data. Two fields are used

- Tx_fifo_addr_diff[5:0] (read-only) - this value indicates the number of empty bytes currently available in the transmit FIFO buffer. The number of free bytes is between 0 (full) and 32 (empty) bytes. Data can be written to the transmit buffer while this number is non-zero. If this value goes negative (i.e., exceeding FIFO capacity), the write enable is deasserted and no more writes will occur.
- RxLevel[4:0] (write-only) -When the number of bytes filled in the receive FIFO exceeds the value specified by RxLevel, an interrupt is generated.

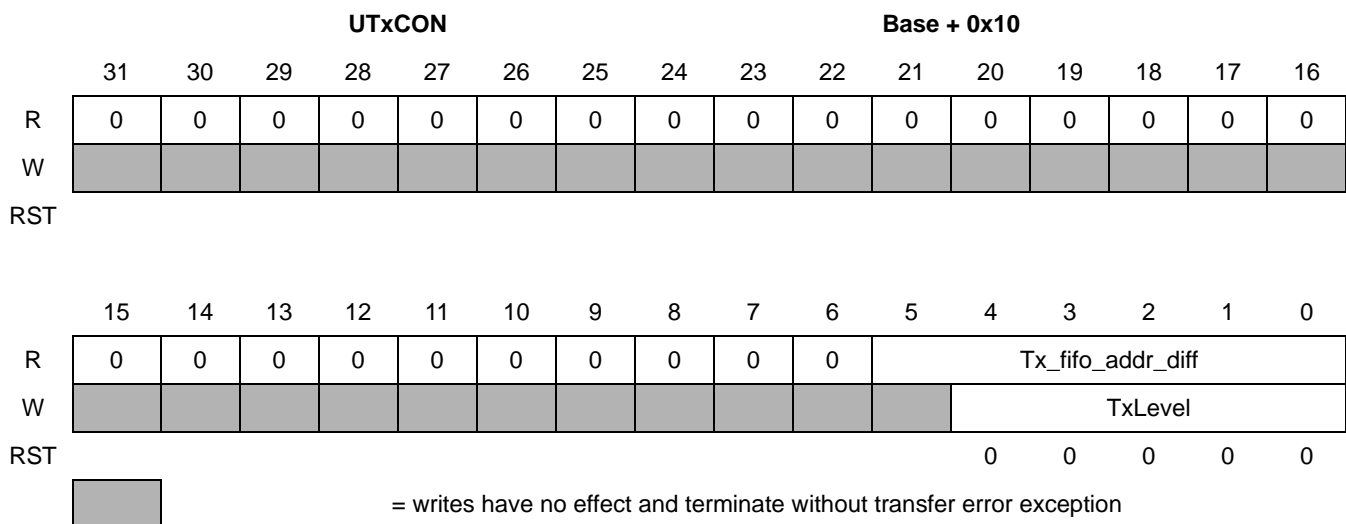


Table 13-13. UTxCON Register Bit Descriptions

Bit Number	Description	Operation
6-31	Reserved	
0-5	Tx_fifo_addr_diff[5:0] — Transmit Buffer Empty Level. This value indicates the number of empty bytes currently available in the transmit FIFO buffer. The number of free bytes is between 0 (full) and 32 (empty) bytes.	
0-4	TxLevel[4:0] — Transmit Buffer Level. When the number of free bytes in the transmit FIFO buffer exceeds the value specified by TxLevel, an interrupt is generated.	

13.9.4.3 UART CTS Level Control Register (UCTS)

UART CTS Level Control Register sets the threshold for the CTS flow control signal. If the remote UART continues to send limited amounts of data after detecting the deasserted CTS signal, the local receive buffer can handle the additional data if CTS_Level is set to a suitable value.



Table 13-14. UCTS Register Bit Descriptions

Bit Number	Description	Operation
5-31	Reserved	
0-4	CTS_Level[4:0] — CTS Buffer Level. When the number of bytes in the receiver (Rx) FIFO exceeds this value, the CTS signal is deasserted.	

13.9.5 UART Baud Rate Divider Register (UBR)

The UART Baud Rate Divider Register contains the fractional divider register fields used to compute the receive and transmit baud rate.

		UBR								Base + 0x18							
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W		UBRINC															
	RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W		UBRMOD															
	RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-15. UBR Register Bit Descriptions

Bit Number	Description	Operation
16-31	UBRINC[15;0] — INC Value. The UBRINC field is used to select the increment value for the fractional division divider. See Section 13.4.2, “Baud Rate Generation (Fractional Divider)” for an explanation of the value to be written to the register.	
0-15	UBRMOD[15;0] — MOD Value. The UBRMOD field is used to select the modulus for the fractional division divider. See Section 13.4.2, “Baud Rate Generation (Fractional Divider)” for an explanation of the value to be written to the register.	

Chapter 14

I²C Module

14.1 Overview

The inter-IC (IIC or I2C) bus is a two-wire—serial data (SDA) and serial clock (SCL)—bidirectional serial bus that provides a simple efficient method of data exchange between the system and other devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The two-wire bus minimizes the interconnections between devices. The synchronous, multi-master bus of the I²C allows the connection of additional devices to the bus for expansion and system development. The bus includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously.

NOTE

See [Section 14.8.7, “Clock Enable Register \(I2CCKER\)”](#) for a special note on programming I²C operation and enabling the I²C module.

14.2 Features

The I²C interface includes the following features:

- Two-wire interface
- Multi-master operational
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Acknowledge bit generation/detection
- Bus busy detection
- Software-programmable clock frequency
- 8-bit control registers
- Software-selectable acknowledge bit
- On-chip filtering for spikes on the bus

14.3 Block Diagram

Figure 14-1 is a block diagram of the I²C block.

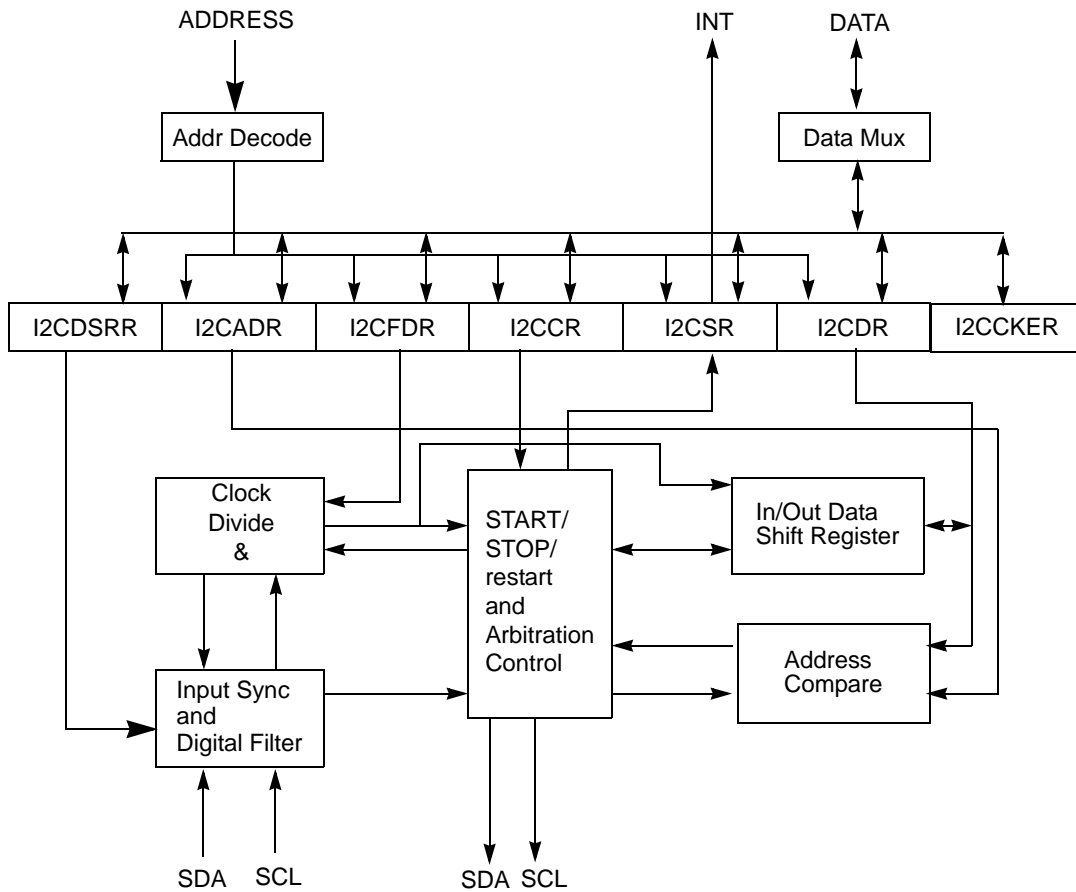


Figure 14-1. I²C Controller Block Diagram

14.4 Functional Description

The following sections describe the module functionality.

14.4.1 Signal Description

The I²C interface uses the SDA signal and SCL signal for data transfer. All devices that are connected to these two signals must have open drain or open collector outputs. A logical AND function is performed on both signals with external pull-up resistors.

Table 14-1 summarizes the signals that comprise the external I²C interface.

Table 14-1. I²C Interface Signal Properties

Signal Names	Direction	Active	Description/Comments
SCL	Input/Output	0	Bus clock. Open drain output. External pull-up required
SDA	Input/Output	0	Bus clock. Open drain output. External pull-up required

14.4.2 Modes of Operation

The following modes of operation are supported by the I²C controller:

- Master mode. The I²C is the driver of the SDA line. It cannot use its own slave address as a calling address. The I²C cannot be a master and a slave simultaneously.
- Slave mode. The I²C is not the driver of the SDA line. The module must be enabled before a START condition from a non-I²C master is detected.
- START condition. This condition denotes the beginning of a new data transfer (each data transfer contains several bytes of data) and awakens all slaves. This mode is I²C-specific.
- Repeated START condition. A START condition that is generated without a STOP condition to terminate the previous transfer. This mode is I²C-specific.
- STOP condition. The master can terminate the transfer by generating a STOP condition to free the bus. This mode is I²C-specific.
- Interrupt-driven byte-to-byte data transfer. When successful slave addressing is achieved (and SCL returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/W bit sent by the calling master. Each byte of data must be followed by an acknowledge bit, which is signalled from the receiving device. Several bytes can be transferred during a data transfer session.

14.4.3 I²C Protocol Description

A standard I²C transfer consists of the following:

- START condition
- Slave target address transmission
- Data transfer
- STOP condition

Figure 14-2 shows the interaction of these four parts with the calling address, data byte, and new calling address components of the I²C protocol. The details of the protocol are described in the following subsections.

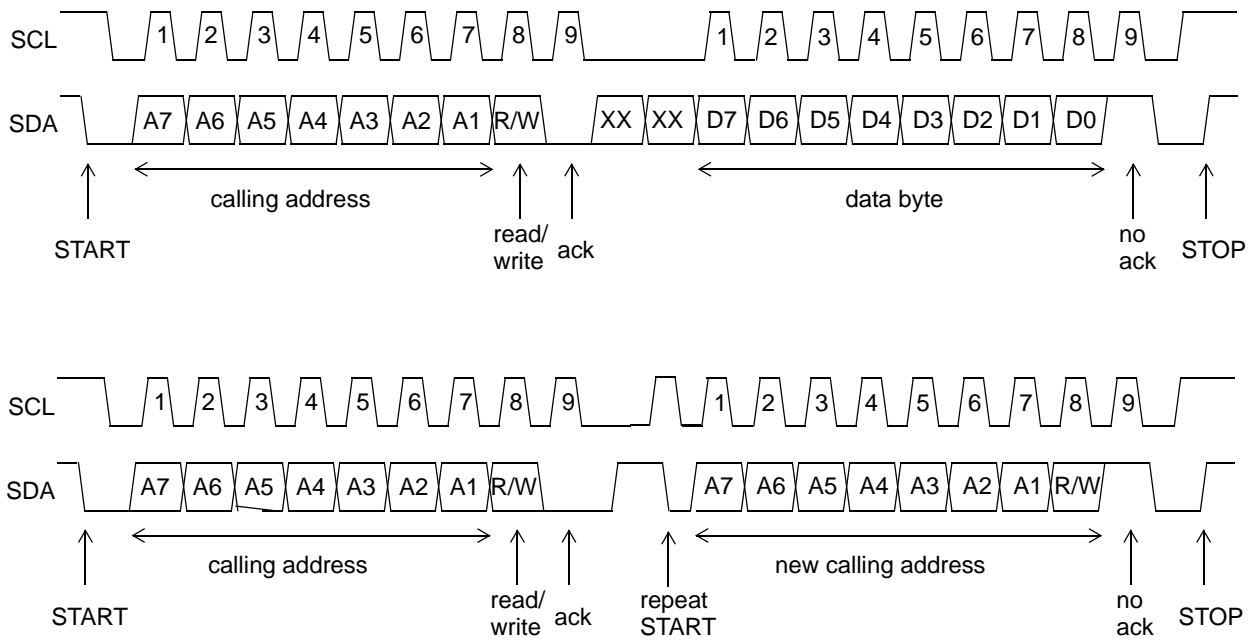


Figure 14-2. I²C Interface Transaction Protocol

14.4.3.1 START Condition

When the I²C bus is not engaged (both SDA and SCL lines are at logic high), a master can initiate a transfer by sending a START condition. As shown in Figure 14-2, a START condition is defined as a high-to-low transition of SDA while SCL is high. This condition denotes the beginning of a new data transfer. Each data transfer can contain several bytes and awakens all slaves. The START condition is initiated by a software write that sets the I2CCR[MSTA].

14.4.3.2 Slave Address Transmission

The first byte of data is transferred by the master immediately after the START condition is the slave address. This is a seven-bit calling address followed by a R/W bit, which indicates the direction of the data being transferred to the slave. Each slave in the system has a unique address. In addition, when the I²C module is operating as a master, it must not transmit an address that is the same as its slave address. An I²C device cannot be master and slave at the same time.

Only the slave with a calling address that matches the one transmitted by the master responds by returning an acknowledge bit (pulling the SDA signal low at the 9th clock) as shown in Figure 14-2. If no slave acknowledges the address, the master should generate a STOP condition or a repeated START condition.

When slave addressing is successful (and SCL returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/W bit sent by the calling master.

The I²C module responds to a general call (broadcast) command when I2CCR[BCST] is set. A broadcast address is always zero, and the R/W bit is zero.

Each data byte is 8 bits long. Data bits can be changed only while SCL is low and must be held stable while SCL is high, as shown in Figure 14-2. There is one clock pulse on SCL for each data bit, and the most

significant bit (MSB) is transmitted first. Each byte of data must be followed by an acknowledge bit, which is signalled from the receiving device by pulling the SDA line low at the ninth clock. Therefore, one complete data byte transfer takes nine clock pulses. Several bytes can be transferred during a data transfer session.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA line for the master to generate a STOP or a START condition.

14.4.3.3 Repeated START Condition

Figure 14-2 shows a repeated START condition, which is generated without a STOP condition that can terminate the previous transfer. The master uses this method to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

14.4.3.4 STOP Condition

The master can terminate the transfer by generating a STOP condition to free the bus. A STOP condition is defined as a low-to-high transition of the SDA signal while SCL is high. For more information, see Figure 14-2. Note that a master can generate a STOP even if the slave has transmitted an acknowledge bit, at which point the slave must release the bus. The STOP condition is initiated by a software write that clears I2CCR[MSTA].

As described in Section 14.4.3.3, “Repeated START Condition”, the master can generate a START condition followed by a calling address without generating a STOP condition for the previous transfer. This is called a repeated START condition.

14.4.3.5 Arbitration Procedure

The I²C interface is a true multiple master bus that allows more than one master device to be connected on it. If two or more masters simultaneously try to control the bus, each master’s (including the I²C module) clock synchronization procedure determines the bus clock—the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. A bus master loses arbitration if it transmits a logic 1 on SDA while another master transmits a logic 0. The losing masters immediately switch to slave-receive mode and stop driving the SDA line. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, the I²C unit sets the I2CSR[MAL] status bit to indicate the loss of arbitration and, as a slave, services the transaction if it is directed to itself.

Arbitration is lost (and I2CSR[MAL] is set) in the following circumstances:

- SDA sampled as low when the master drives a high during address or data-transmit cycle.
- SDA sampled as low when the master drives a high during the acknowledge bit of a data-receive cycle.
- A START condition is attempted when the bus is busy.

- A repeated START condition is requested in slave mode.

Note that the I²C module does not automatically retry a failed transfer attempt.

If the I²C module is enabled in the middle of an ongoing byte transfer, the interface behaves as follows:

- Slave mode—the I²C module ignores the current transfer on the bus and starts operating whenever a subsequent START condition is detected.
- Master mode—the I²C module cannot tell whether the bus is busy; therefore, if a START condition is initiated, the current bus cycle can be corrupted. This ultimately results in the current bus master of the I²C interface losing arbitration, after which bus operations return to normal.

14.4.3.6 Clock Synchronization

Due to the wire AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached. However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time. When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

14.4.3.7 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices can hold the SCL low after completion of a 1-byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

14.4.3.8 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven the SCL line low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period, then the resulting SCL bus signal low period is stretched.

14.4.4 Description of I²C Functional Blocks

The following sections provide additional information on the operation of the I²C module.

14.4.4.1 Clock Control

The clock control block handles requests from clock for transferring and controlling data for multiple tasks.

A 9-cycle data transfer clock is requested for the following conditions:

- Master mode
- transmit slave address after START condition
- transmit slave address after restart condition
- transmit data
- receive data
- Slave mode
- transmit data
- receive data
- receive slave address after START or restart condition

14.4.4.2 Input Synchronization and Digital Filter

The input synchronization block synchronizes the input SCL and SDA signals to the system clock and detects transitions of these signals. In addition, the SCL and SDA input pins are filtered to eliminate noise. Three consecutive samples of the SCL and SDA lines are compared with a pre-determined sampling rate. If they are all high, the output of the filter is high. If they are all low, the output is low. If there are any combination of highs and lows, the output is whatever the value of the line was in the previous clock cycle.

The sampling rate is equal to a binary value stored in the frequency register. The duration of the sampling cycle is controlled by a down counter and a signal that sets at the end of the count. This allows a software write to the frequency register to control the filtered sampling rate.

14.4.4.2.1 Transaction Monitoring

The different conditions of the I²C data transfers are monitored as follows:

- START conditions are detected when an SDA fall occurs while SCL is high.
- STOP conditions are detected when an SDA rise occurs while SCL is high.
- Data transfers in progress are cancelled when a STOP condition is detected or if there is a slave address mismatch. Cancellation of data transactions resets the clock module
- The bus is detected to be busy upon the detection of a START condition, and idle upon the detection of a STOP condition.

14.4.4.3 Arbitration Control

The arbitration control block controls the arbitration procedure of the master mode. A loss of arbitration occurs whenever the master detects a 0 on the external SDA line while attempting to drive a 1, tries to generate a START or restart at an inappropriate time, or detects an unexpected STOP request on the line.

Arbitration by the master in master mode is lost under the following conditions—

- Low detected when high expected (transmit)
- Ack bit, low detected when high expected (receive)
- A START condition is attempted when the bus is busy
- A START condition is attempted when the bus is nearly busy
- A start condition is attempted when the requesting device is not the bus owner
- Unexpected STOP condition detected

14.4.4.4 Control Transfer

The I²C contains logic that controls the output to the serial data (SDA) and serial clock (SCL) lines of the I²C. The SCL output is pulled low as determined by the internal clock generated in the clock module. The SDA output can only change at the midpoint of a low cycle of the SCL, unless performing a START, STOP, or restart condition. Otherwise, the SDA output is held constant.

The SDA signal is pulled low when one or more of the following conditions are true in either master or slave mode—

- Master mode
- data bit (transmit)
- ack bit (receive)
- START condition
- STOP condition
- Restart condition
- Slave mode
- acknowledging address match
- data bit (transmit)
- ack bit (receive)

The SCL signal corresponds to the internal SCL signal when one or more of the following conditions are true in either master or slave mode—

- Master mode
- bus owner
- lost arbitration
- START condition
- STOP condition
- Restart condition begin

- Restart condition end
- Slave mode
- address cycle
- transmit cycle
- ack cycle

14.4.4.5 In/Out Data Shift Register

This block controls the interface between the serial data line and the data register, both transmitting data to and receiving data from the I²C.

In transmit mode, a write to I2CCR[MTX] sets the direction to transmit. The contents of yicp_tx_data are loaded into a shift register, which are then loaded to yici_rx_data. The contents of yici_rx_data are then shifted out to shift_sda, which eventually is routed out to yici_sda_out.

14.4.4.6 Address Compare

Address compare block determines if a slave has been properly addressed, either by its slave address or by the general broadcast address (which addresses all slaves). The four performed address comparisons are described as follows:

- The address sent by the current master matches the general broadcast address.
- The address matches either the broadcast address or the slave address; in either case, slave mode results.
- Whether a broadcast message has been received, to update the I2CSR and to generate an interrupt.
- Whether the module has been addressed as a slave, to update the I2CSR and to generate an interrupt.

14.5 Reset

The reset state of the I²C is that the module is disabled. Once enabled, the default condition is that the interface will perform as a slave receiver, unless explicitly programmed to be a master or slave transmitter address.

14.6 Interrupts

There is a single interrupt request signal from the I²C module. The interrupt request is active if the interrupt status bit MIF (Register I2CSR, Bit 1) is set, and the request is enabled by bit MIEN enable (Register I2CCR, Bit 6).

MIF is set when one of the following events occurs (See [Section 14.8.4, “I2C Status Register \(I2CSR\)”](#) for more details):

- One byte of data is transferred (set at the falling edge of the 9th clock). Status bit MCF is set.
- The value in I2CADR matches with the calling address in slave-receive mode. Status bit MAAS is set.

- Arbitration is lost. Status bit MAL is set.

14.7 I²C Register Memory Map.

For the I2C module

- The base address is 0x8000_6000
- All registers are 8 bits wide and are accessed only as a byte-access on a long word boundary.

Table 14-2 lists the I²C-specific registers and their offsets.

Table 14-2. I²C Module Register Memory Map

Offset	[7:0]	Access Type	Access Width
+0x00	I ² C address register (I2CADR)	R/W	8-bit only
+0x04	I ² C frequency divider register (I2CFDR)	R/W	8-bit only
+0x08	I ² C control register (I2CCR)	R/W	8-bit only
+0x0C	I ² C status register (I2CSR)	R/W	8-bit only
+0x10	I ² C data register (I2CDR)	R/W	8-bit only
+0x14	Digital filter sampling rate register (I2CDFSRR)	R/W	8-bit only
+0x18	Clock Enable Register (I2CCKER)	R/W	8-bit only

14.8 I²C Registers

14.8.1 I²C Address Register (I2CADR)

Figure 14-3 shows the I2CADR register, which contains the address to which the I²C interface responds when addressed as a slave. Note that this is not the address that is sent on the bus during the address-calling cycle when the I²C module is in master mode.

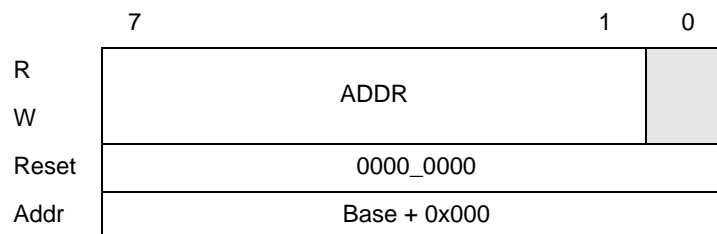


Figure 14-3. I²C Address Register (I2CADR)

Table 14-3 describes the bit settings of I2CADR.

Table 14-3. I2CADR Field Descriptions

Bits	Name	Description
7-1	ADDR	Slave address. Contains the specific slave address that is used by the I ² C interface. Note that the default mode of the I ² C interface is slave mode for an address match.
0	—	Reserved

14.8.2 I²C Frequency Divider Register (I2CFDR)

Figure 14-4 shows the sampling rate and the clock bit rate for the I²C unit.

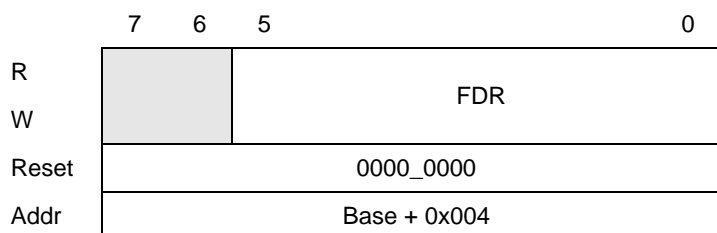


Figure 14-4. I²C Frequency Divider Register (I2CFDR)

Table 14-4 describes the bit settings of I2CFDR. It also maps the I2CFDR[FDR] field to the clock divider values.

Table 14-4. I2CFDR Field Descriptions

Bits	Name	Description
7-6	—	Reserved
5-0	FDR	Frequency divider ratio. Used to prescale the clock for bit rate selection. The serial bit clock frequency of SCL is equal to the platform clock divided by the divider. Note that the frequency divider value can be changed at any point in a program. The serial bit clock frequency divider selections are described as follows:

Table 14-5 shows the divider ratio versus FDR field setting. The incoming peripheral clock frequency to the module will be the reference frequency divided by the xtal_clkdiv (CRM).

Table 14-5. I2C Clock Divide Ratio vs. FDR Setting

FDR	Divider (Decimal)	FDR	Divider (Decimal)
0x00	288	0x20	160
0x01	320	0x21	192
0x02	384	0x22	224
0x03	480	0x23	256
0x04	576	0x24	320
0x05	640	0x25	384
0x06	768	0x26	448
0x07	960	0x27	512
0x08	1152	0x28	640
0x09	1280	0x29	768
0x0A	1536	0x2A	896
0x0B	1920	0x2B	1024
0x0C	2304	0x2C	1280
0x0D	2560	0x2D	1536

Table 14-5. I2C Clock Divide Ratio vs. FDR Setting (continued)

FDR	Divider (Decimal)	FDR	Divider (Decimal)
0x0E	3072	0x2E	1792
0x0F	3840	0x2F	2048
0x10	4608	0x30	2560
0x11	5120	0x31	3072
0x12	6144	0x32	3584
0x13	7680	0x33	4096
0x14	9216	0x34	5120
0x15	10240	0x35	6144
0x16	12288	0x36	7168
0x17	15360	0x37	8192
0x18	18432	0x38	10240
0x19	20480	0x39	12288
0x1A	24576	0x3A	14336
0x1B	30720	0x3B	16384
0x1C	36864	0x3C	20480
0x1D	40960	0x3D	24576
0x1E	49152	0x3E	28672
0x1F	61440	0x3F	32768

14.8.3 I²C Control Register (I2CCR)

Figure 14-5 shows the modes of the I²C interface that are controlled by I2CCR.

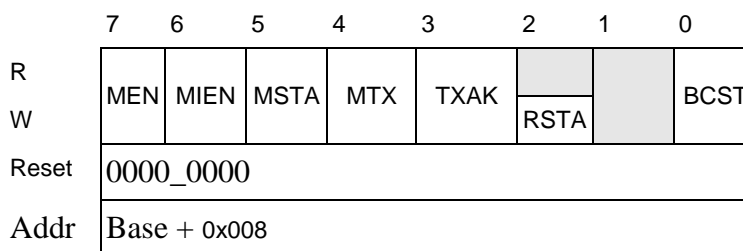


Figure 14-5. I²C Control Register (I2CCR)

Table 14-6 describes the bit settings of the I2CCR

Table 14-6. I2CCR Field Descriptions

Bits	Name	Description
7	MEN	Module enable. This bit controls the software reset of the I ² C module. 0 The module is reset and disabled. When low, the interface is held in reset but the registers can still be accessed. 1 The I ² C module is enabled. This bit must be set before any other control register bits have any effect. All I ² C registers for slave receive or master START can be initialized before setting this bit.
6	MIEN	Module interrupt enable 0 Interrupts from the I ² C module are disabled. This does not clear any pending interrupt conditions. 1 Interrupts from the I ² C module are enabled. An interrupt occurs provided I2CSR[MIF] is also set.
5	MSTA	Master/slave mode START 0 When this bit is changed from a 1 to a 0, a STOP condition is generated and the mode changes from master to slave. 1 Cleared without generating a STOP condition when the master loses arbitration. When this bit is changed from a 0 to a 1, a START condition is generated on the bus, and the master mode is selected.
4	MTX	Transmit/receive mode select. This bit selects the direction of the master and slave transfers. When configured as a slave, this bit should be set by software according to I2CSR[SRW]. In master mode, the bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. The MTX bit is cleared when the master loses arbitration. 0 Receive mode 1 Transmit mode
3	TXAK	Transfer acknowledge. This bit specifies the value driven onto the SDA line during acknowledge cycles for both master and slave receivers. The value of this bit only applies when the I ² C module is configured as a receiver, not a transmitter. It also does not apply to address cycles; when the core is addressed as a slave, an acknowledge is always sent. 0 An acknowledge signal (low value on SDA) is sent out to the bus at the 9th clock bit after receiving one byte of data. 1 No acknowledge signal response (high value on SDA) is sent.
2	RSTA	Repeat START. Setting this bit always generates a repeated START condition on the bus and provides the core with the current bus master. Attempting a repeated START at the wrong time (or if the bus is owned by another master), results in loss of arbitration. Note that this bit is not readable. 0 No START condition is generated 1 Generates repeat START condition.

Table 14-6. I2CCR Field Descriptions (continued)

Bits	Name	Description
1	—	Reserved
0	BCST	Broadcast 0 Disables the broadcast accept capability. 1 Enables the I ² C to accept broadcast messages at address zero.

14.8.4 I²C Status Register (I2CSR)

The status register, shown in Figure 14-6, is read-only with the exception of the MIF and MAL bits, which can be cleared by software. The MCF and RXAK bits are set at reset; all other I2CSR bits are cleared on reset.

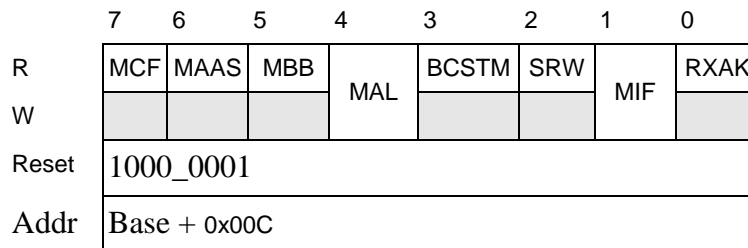
Figure 14-6. I²C Status Register (I2CSR)

Table 14-7 describes the bit settings of the I2CSR.

Table 14-7. I2CSR Field Descriptions

Bits	Name	Description
7	MCF	Data transfer. When one byte of data is transferred, the bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. 0 Byte transfer in progress. MCF is cleared under two following conditions— when I2CSR is read in receive mode or when I2CDR is written in transmit mode 1 Byte transfer is completed.
6	MAAS	Addressed as a slave. When the value in I2CDR matches with the calling address, this bit is set. The processor is interrupted, if I2CCR[MIEN] is set. Next, the processor must check the SRW bit and set I2CCR[MTX] accordingly. Writing to the I2CCR automatically clears this bit. 0 Not addressed as a slave 1 Addressed as a slave
5	MBB	Bus busy. Indicates the status of the bus. When a START condition is detected, MBB is set. If a STOP condition is detected, it is cleared. 0 I ² C bus is idle 1 I ² C bus is busy
4	MAL	Arbitration lost. Automatically set when the arbitration procedure is lost. Note that the core does not automatically retry a failed transfer attempt. 0 Arbitration is not lost. Can only be cleared by software. 1 Arbitration is lost

Table 14-7. I2CSR Field Descriptions (continued)

Bits	Name	Description
3	BCSTM	<p>Broadcast match</p> <p>0 There has not been a broadcast match.</p> <p>1 The calling address matches with the broadcast address instead of the programmed slave address. This will also be set if the I²C drives an address of all 0s and broadcast mode is enabled.</p>
2	SRW	<p>Slave read/write. When MAAS is set, SRW indicates the value of the R/W command bit of the calling address, which is sent from the master.</p> <p>0 Slave receive, master writing to slave</p> <p>1 Slave transmit, master reading from slave. This bit is valid only when both of the following conditions are true:</p> <ul style="list-style-type: none"> A complete transfer occurred and no other transfers have been initiated The I²C interface is configured as a slave and has an address match. <p>By checking this bit, the processor can select slave transmit/receive mode according to the command of the master.</p>
1	MIF	<p>Module interrupt. The MIF bit is set when an interrupt is pending, causing a processor interrupt request (provided I2CCR[MIE] is set).</p> <p>0 No interrupt is pending. Can be cleared only by software.</p> <p>1 Interrupt is pending. MIF is set when one of the following events occurs:</p> <ul style="list-style-type: none"> • One byte of data is transferred (set at the falling edge of the 9th clock) • The value in I2CADR matches with the calling address in slave-receive mode • Arbitration is lost
0	RXAK	<p>Received acknowledge. The value of SDA during the reception of acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates that an acknowledge signal has been received after the completion of eight bits of data transmission on the bus. If RXAK is high, it means no acknowledge signal has been detected at the 9th clock.</p> <p>0 Acknowledge received</p> <p>1 No acknowledge received</p>

14.8.5 I²C Data Register (I2CDR)

The I²C data register is shown in [Figure 14-7](#).

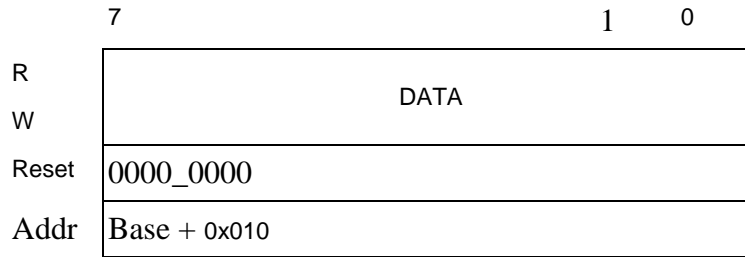


Figure 14-7. I²C Data Register (I2CDR)

[Table 14-8](#) shows the bit descriptions for I2CDR.

Table 14-8. I2CDR Field Descriptions

Bits	Name	Description
7-0	DATA	Transmission starts when an address and the R/W bit are written to the data register and the I ² C interface performs as the master. A data transfer is initiated when data is written to the I2CDR. The most significant bit is sent first in both cases. In the master receive mode, reading the data register allows the read to occur, but also allows the I ² C module to receive the next byte of data on the I ² C interface. In slave mode, the same function is available after it is addressed.

14.8.6 Digital Filter Sampling Rate Register (I2CDFSRR)

The digital filter sampling rate register (I2CDFSRR) is shown in [Figure 14-8](#).

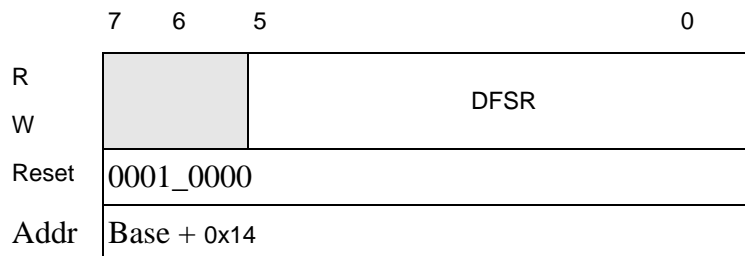


Figure 14-8. I2CDFSRR Format

[Table 14-9](#) shows the field descriptions for I2CDFSRR.

Table 14-9. I2CDFSRR Field Descriptions

Bits	Name	Description
7-6	—	Reserved
5-0	DFSR	Digital filter sampling rate. To assist in filtering out signal noise, the sample rate is programmed. This field is used to prescale the frequency at which the digital filter takes samples from the I ² C bus. The resulting sampling rate is calculated by dividing the platform frequency by the non-zero value of DFSR. If I2CDFSRR is set to zero, the I ² C bus sample points default to the reset divisor 0x10.

14.8.7 Clock Enable Register (I2CCKER)

The clock enable register (I2CCKER) is shown in [Figure 14-9](#).

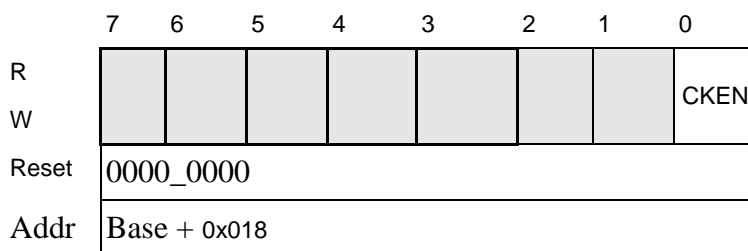


Figure 14-9. Clock Enable Register (I2CCKER)

[Table 14-10](#) shows the field descriptions for I2CCKER.

NOTE

Bit CKEN must be written high before any other I2C register accesses or any I2C activity.

Table 14-10. I2CCKER Field Descriptions

Bits	Name	Description
7-1	—	Reserved
0	CKEN	Clock Enable bit. When asserted, the clock to I2C is running continuously from the CRM. When de-asserted, the clock to I2C is gated off by the CRM unless an IP bus write to the I2C is detected by the CRM. This bit must be written high by software before any other I2C register writes, and kept high during normal I2C operation.

14.9 Initialization/Application Information

This section describes some programming guidelines recommended for the I²C interface. It also includes [Figure 14-10](#), a recommended flow chart for the I²C interrupt service routines.

The I²C registers in this chapter are shown in big-endian format. If the system is in little-endian mode, software must swap the bytes appropriately. Also, an msync assembly instruction should be executed after each I²C register read/write access to guarantee in-order execution.

The I²C controller does not guarantee its recovery from all illegal I²C bus activity. In addition, a malfunctioning device may hold the bus captive. A good programming practice is for software to rely on a watchdog timer to help recover from I²C bus hangs. The recovery routine should also handle the case when the status bits returned after an interrupt are not consistent with what was expected due to illegal I²C bus protocol behavior.

14.9.1 Initialization Sequence

A hard reset initializes all the I²C registers to their default states. The following initialization sequence initializes the I²C unit:

- All I²C registers must be located in a cache-inhibited page.
- Update I2CFDR[FDR] and select the required division ratio to obtain the SCL frequency from the platform clock.
- Update I2CADR to define the slave address for this device.
- Modify I2CCR to select master/slave mode, transmit/receive mode, and interrupt-enable or disable.
- Set the I2CCR[MEN] to enable the I²C interface.

14.9.2 Generation of START

After initialization, the following sequence can be used to generate START:

- If the device is connected to a multi-master I²C system, test the state of I2CSR[MBB] to check whether the serial bus is free (I2CSR[MBB] = 0) before switching to master mode.
- Select master mode (set I2CCR[MSTA]) to transmit serial data and select transmit mode (set I2CCR[MTX]) for the address cycle.
- Write the slave address being called into the data register (I2CDR). The data written to I2CDR comprises the slave calling address. I2CCR[MTX] indicates the direction of transfer (transmit/receive) required from the slave.

This scenario assumes that the I²C interrupt bit (I2CSR[MIF]) is cleared. If I2CSR[MIF] = 1 at any time, the I²C interrupt handler should immediately handle the interrupt.

14.9.3 Post-Transfer Software Response

Transmission or reception of a byte automatically sets the data transferring bit (I2CSR[MCF]), which indicates that one byte has been transferred. The I²C interrupt bit (I2CSR[MIF]) is also set; an interrupt is generated to the processor if the interrupt function is enabled during the initialization sequence (I2CCR[MEN] = 1). In the interrupt handler, software must do the following:

- Clear I2CSR[MIF]
- Read the contents of the I²C data register (I2CDR) in receive mode or write to I2CDR in transmit mode. Note that this causes I2CSR[MCF] to be cleared.

When an interrupt occurs at the end of the address cycle, the master remains in transmit mode. If master receive mode is required, I2CCR[MTX] should be toggled at this stage.

If the interrupt function is disabled, software can service the I2CDR in the main program by monitoring I2CSR[MIF]. In this case, I2CSR[MIF] should be polled rather than I2CSR[MCF] because MCF behaves differently when arbitration is lost. Note that interrupt or other bus conditions may be detected before the I²C signals have time to settle. Thus, when polling I2CSR[MIF] (or any other I2CSR bits), software delays may be needed (in order to give the I²C signals sufficient time to settle).

During slave-mode address cycles ($I2CSR[MAAS] = 1$), $I2CSR[SRW]$ should be read to determine the direction of the subsequent transfer and $I2CCR[MTX]$ should be programmed accordingly. For slave-mode data cycles ($I2CSR[MAAS] = 0$), $I2CSR[SRW]$ is not valid and $I2CCR[MTX]$ should be read to determine the direction of the current transfer.

14.9.4 Generation of STOP

A data transfer ends with a STOP condition generated by the master device. A master transmitter can generate a STOP condition after all the data has been transmitted.

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data, which is done by setting the transmit acknowledge ($I2CCR[TXAK]$) bit before reading the next-to-last byte of data. At this time, the next-to-last byte of data has already been transferred on the I²C interface, so the last byte will not receive the data acknowledge when $I2CCR[TXAK]$ is set. For 1-byte transfers, a dummy read should be performed by the interrupt service routine. Before the interrupt service routine reads the last byte of data, a STOP condition must first be generated.

The I²C automatically generates a STOP if $I2CCR[TXAK] = 1$. Therefore, $I2CCR[TXAK]$ must be set to a 1 before allowing the I²C module to receive the last data byte on the I²C bus. Eventually, $I2CCR[TXAK]$ will need to be cleared again for subsequent I²C transactions. This can be accomplished when setting up the $I2CCR$ for the next transfer.

14.9.5 Generation of Repeated START

At the end of a data transfer, if the master still wants to communicate on the bus, it can generate another START condition followed by another slave address without first generating a STOP condition by setting $I2CCR[RSTA]$.

14.10 Generation of SCL When SDA Low

In some cases it is necessary to force the I²C module to become the I²C bus master out of reset and drive the SCL signal (even though SDA may already be driven, which indicates that the bus is busy). This can occur when a system reset does not cause all I²C devices to be reset. Thus, the SDA signal can be driven low by another I²C device while the I²C module is coming out of reset and will stay low indefinitely. The following procedure can be used to force the I²C module to generate SCL so that the device driving SDA can finish its transaction:

- Disable the I²C and set the master bit by setting $I2CCR$ to 0x20.
- Enable the I²C by setting $I2CCR$ to 0xA0.
- Read the $I2CDR$.
- Return the I²C module to slave mode by setting $I2CCR$ to 0x80.

14.10.1 Slave Mode Interrupt Service Routine

In the slave interrupt service routine, the module addressed as a slave should be tested to check if a calling of its own address has been received. If $I2CSR[MAAS] = 1$, software should set the transmit/receive mode select bit ($I2CCR[MTX]$) according to the R/\overline{W} command bit ($I2CSR[SRW]$). Writing to $I2CCR$ clears $I2CSR[MAAS]$ automatically. The only time $I2CSR[MAAS]$ is read as set is from the interrupt handler at the end of that address cycle where an address match occurred; interrupts resulting from subsequent data transfers will have $I2CSR[MAAS] = 0$. A data transfer can then be initiated by writing to $I2CDR$ for slave transmits or dummy reading from $I2CDR$ in slave-receive mode. The slave drives SCL low between byte transfers. SCL is released when the $I2CDR$ is accessed in the required mode.

14.10.1.1 Slave Transmitter and Received Acknowledge

In the slave transmitter routine, the received acknowledge bit ($I2CSR[RXAK]$) must be tested before sending the next byte of data. The master signals an end-of-data by not acknowledging the data transfer from the slave. When no acknowledge is received ($I2CSR[RXAK] = 1$), the slave transmitter interrupt routine must clear $I2CCR[MTX]$ to switch the slave from transmitter to receiver mode. A dummy read of $I2CDR$ then releases SCL so that the master can generate a STOP condition.

14.10.1.2 Loss of Arbitration and Forcing of Slave Mode

When a master loses arbitration the following conditions all occur:

- $I2CSR[MAL]$ is set
- $I2CCR[MSTA]$ is cleared (changing the master to slave mode)
- An interrupt occurs (if enabled) at the falling edge of the 9th clock of this transfer

Thus, the slave interrupt service routine should test $I2CSR[MAL]$ first, and the software should clear $I2CSR[MAL]$ if it is set.

14.10.2 Interrupt Service Routine Flow Chart

Figure 14-10 shows an example algorithm for an I²C interrupt service routine. Deviation from the flow chart may result in unpredictable I²C bus behavior. However, in the slave receive mode (not shown in the flow chart), the interrupt service routine may need to set $I2CCR[TXAK]$ when the next-to-last byte is to be accepted. It is recommended that a sync instruction follow each I²C register read or write to guarantee in-order instruction execution.

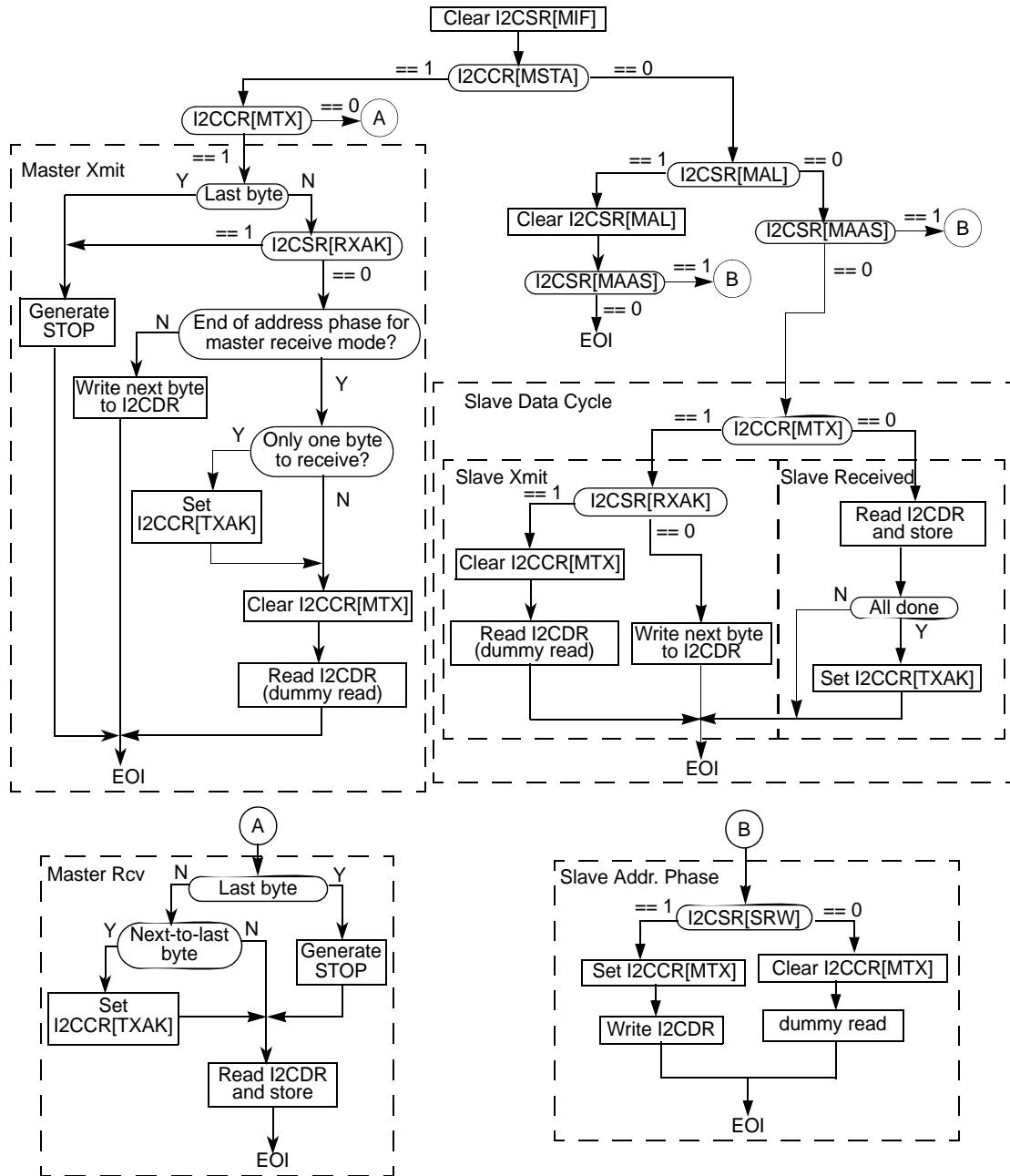


Figure 14-10. Example I²C Interrupt Service Routine Flow Chart

Chapter 15

Serial Peripheral Interface Module (SPI)

15.1 Overview

This chapter describes the Serial Peripheral Interface (SPI) module for external use. The MC1322x SPI is a high-speed synchronous serial data input/output port used for interfacing with serial memories, peripheral devices, or other processors. The SPI allows a serial bit stream of a programmed length (1 to 32 bits) to be shifted simultaneously into and out of the device at a programmed bit-transfer rate (called 4-wire mode). There are four pins associated with the SPI port (SPI_SCK, SPI_MOSI, SPI_MISO, and SPI_SS).

The SPI module can be programmed for master or slave operation. It also supports a 3-wire mode where for master mode, the MOSI becomes MOMI, a bidirectional data pin, and for slave mode the MISO becomes SISO, a bidirectional data pin. In 3-wire mode, data is only transferred in one direction at a time.

The SPI bit clock is derived from the peripheral reference clock (typically 24 MHz with a maximum of 26 MHz). A prescaler divides the peripheral reference clock with a programmed divide ratio from 2 to 256. Typical bit clock range will be from 12 MHz to 93.75 kHz.

NOTE

Although only one dedicated SPI module is present on the MC1322x, the SSI module can also be programmed as a SPI functional block. See [Chapter 16, “Synchronous Serial Interface Module \(SSI\)”](#)

15.2 Features

The SPI module features include:

- Master or slave mode operation
- Transfer length programmable from 1 to 32 bits
- MSB-first shifting
- Data buffer is 4 bytes (32 bits) in length (not double buffered)
- Programmable transmit bit rate (typically 12 MHz max)
- Serial clock phase and polarity options
- Full-duplex (4-wire) or bidirectional data (3-wire) operation
- SPI transaction can be polled or interrupt driven
- Maskable interrupt request
- Slave select input/output
- Low Power (SPI Master uses gated clocks. Slave clock derived completely from SPI_SCK.)
- SPI Slave does not need system clock to be active

15.3 SPI Module Used for FLASH Interface (SPIF)

A second separate SPI FLASH Module is used for serial interface with the internal 128 Kbyte FLASH memory. Only the SPI module Master logic is used for the SPIF module with the following differences:

- The SPIF communicates through four signals. It does not support 3-wire operation.
- The SPIF MOSI output can not be tri-stated.

The SPIF Module is described in detail in [Section 4.6.1, “FLASH Access”](#).

15.4 Block Diagrams

15.4.1 SPI System Block Diagram

This section shows a typical simple SPI system level 4-wire block diagram.

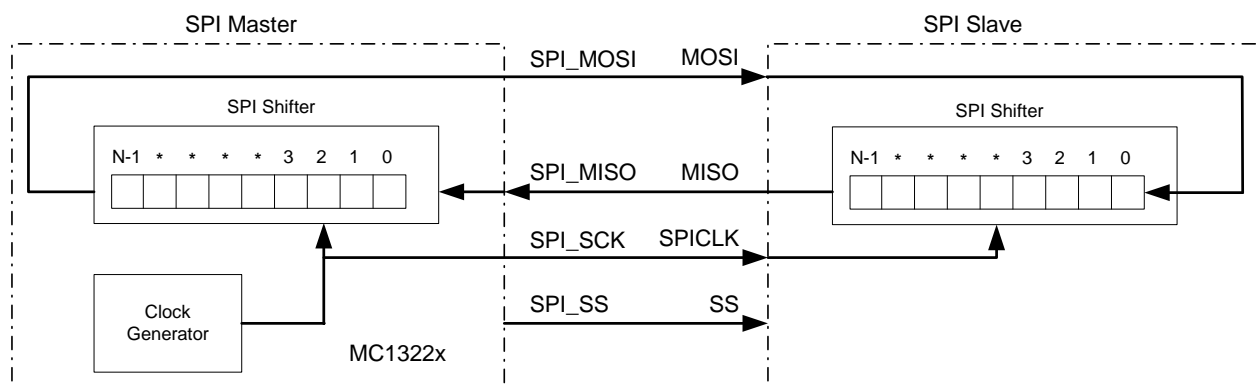


Figure 15-1. SPI System Block Diagram

[Figure 15-1](#) shows the MC1322x as the SPI master and an external device as a SPI slave. The MCU (master) initiates all SPI transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The SPI interface supports simultaneous data exchange between master and slave, although some devices may be read or write only. The SPI_SCK signal is a clock output from the master and an input to the slave. The slave device must be enabled by a low level on the slave select input.

Most SPI devices are 8 bits in width for the SPI shifter. The MC1322x SPI module is programmable and supports up to a 32-bit width. Also, some SPI ports are capable of MSB-first or LSB-first serial shifting, however the SPI module supports only MSB-first shifting.

15.4.2 SPI Module Block Diagram

The SPI module is shown in [Figure 15-2](#).

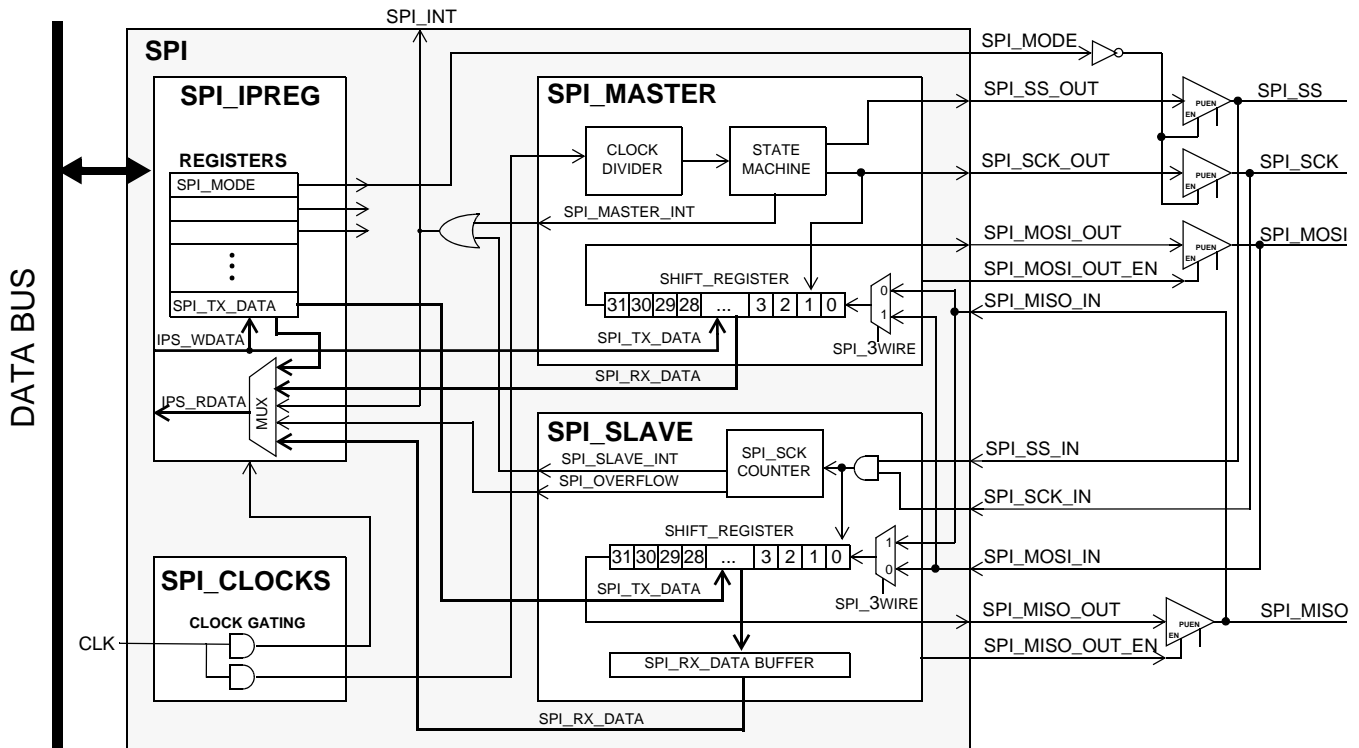


Figure 15-2. Top Level SPI Block Diagram

15.5 Functional Description

The SPI consists of control and status registers, clock generation, and data buffer registers. The master and slave blocks operate separately although they use the same clock. The following sections describe the function of the SPI.

15.5.1 Signal Descriptions

[Table 15-1](#) lists SPI signal names and their descriptions.

Table 15-1. SPI Signals

Signal Names	Direction	Active	Comments
SPI_SCK	Digital Input/Output	Programmable	SPI port clock.
SPI_MOSI	Digital Input/Output	High	SPI Port Master Out Slave In (MOSI) data signal.

Table 15-1. SPI Signals (continued)

Signal Names	Direction	Active	Comments
SPI_MISO	Digital Input/Output	High	SPI Port Master In Slave Out (MISO) data signal.
SPI_SS	Digital Input/Output	Low	SPI Port Slave Select (SS) signal.

15.5.2 Clock Generation

Figure 15-2 shows the SPI block diagram. The SPI module generates the SPI_SCK only in master mode. The SPI bit clock is derived from the peripheral reference clock which is the same as the CPU clock (see Section 5.1.3.2, “Main System Clock Distribution”). The peripheral reference clock is the reference oscillator frequency divided by a 1-64 prescaler located in the CRM.

In the SPI module the peripheral reference clock is further divided by a second prescaler with a programmed divide ratio which is set by the field SPI_SCK_FREQ[2:0] in SPI Register SPI_SETUP. The 3-bit SPI_SCK_FREQ[2:0] is programmable from 0 to 7, where:

$$\text{SPI_SCK} = (\text{peripheral reference clock}) / (2^{(\text{SPI_SCK_FREQ} + 1)})$$

Table 15-2 provides the SPI_SCK frequency versus SPI_SCK_FREQ[2:0] program value with an input reference frequency of 24 Mhz.

Table 15-2. SPI_SCK Frequency versus SPI_SCK_FREQ[2:0]

SPI_SCK_FREQ[2:0]	Divide Ratio	SPI_SCK Frequency (reference frequency = 24 MHz)
0	2	12 MHz
1	4	6 MHz
2	8	3 MHz
3	16	1.5 MHz
4	32	750 kHz
5	64	375 kHz
6	128	187.5 kHz
7	256	93.75 kHz

In slave mode, the SPI_SCK is determined by the SPI master (the other device).

15.5.3 Basic Operation

The SPI module is designed to operate either in master mode (SPI_MODE = 0) or slave mode (SPI_MODE = 1). In either mode data can be shifted in at the same time that data is being shifted out unless the device is configured as a 3-wire connection (SPI_3WIRE = 1). Programming of the SPI is done through a 32-bit IP Bus interface, which can also be accessed in half words and bytes. The following diagram shows the structure of the SPI module.

NOTE

Freescall recommends enabling the pad hysteresis for the SPI_SS and SPI_SCK inputs in slave mode to avoid any glitches propagating into the SPI logic.

15.5.4 SPI Shift Clock Formats

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

The [Figure 15-3](#) shows the clock formats when SPI_SCK_PHASE = 1. At the top of the figure, the bit times are shown for reference with bit 1 starting at the first SPI_SCK edge and the last bit ending one-half SPI_SCK cycle after the last SPI_SCK edge. The MSB first line shows the order of SPI data bits. Notice that "n" is the number of bits in a transfer (provided by SPI_DATA_LENGTH). Both variations of SPI_SCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in SPI_SCK_POL. The SAMPLE IN waveform applies to the SPI_MOSI input of a slave or the SPI_MISO input of a master. The SPI_MOSI waveform applies to the SPI_MOSI output pin from a master and the SPI_MISO waveform applies to the SPI_MISO output from a slave. The SPI_SS OUT waveform applies to the slave select output from a master (provided SPI_SS_SETUP = 1). The master SPI_SS output goes to active low at least one SPI_SCK cycle before the start of the transfer and goes back high at least one SPI_SCK cycle after the end of the last bit time of the transfer (provided that SPI_SCK_COUNT is set equal to SPI_DATA_LENGTH + 1). The SPI_SS IN waveform applies to the slave select input of a slave.

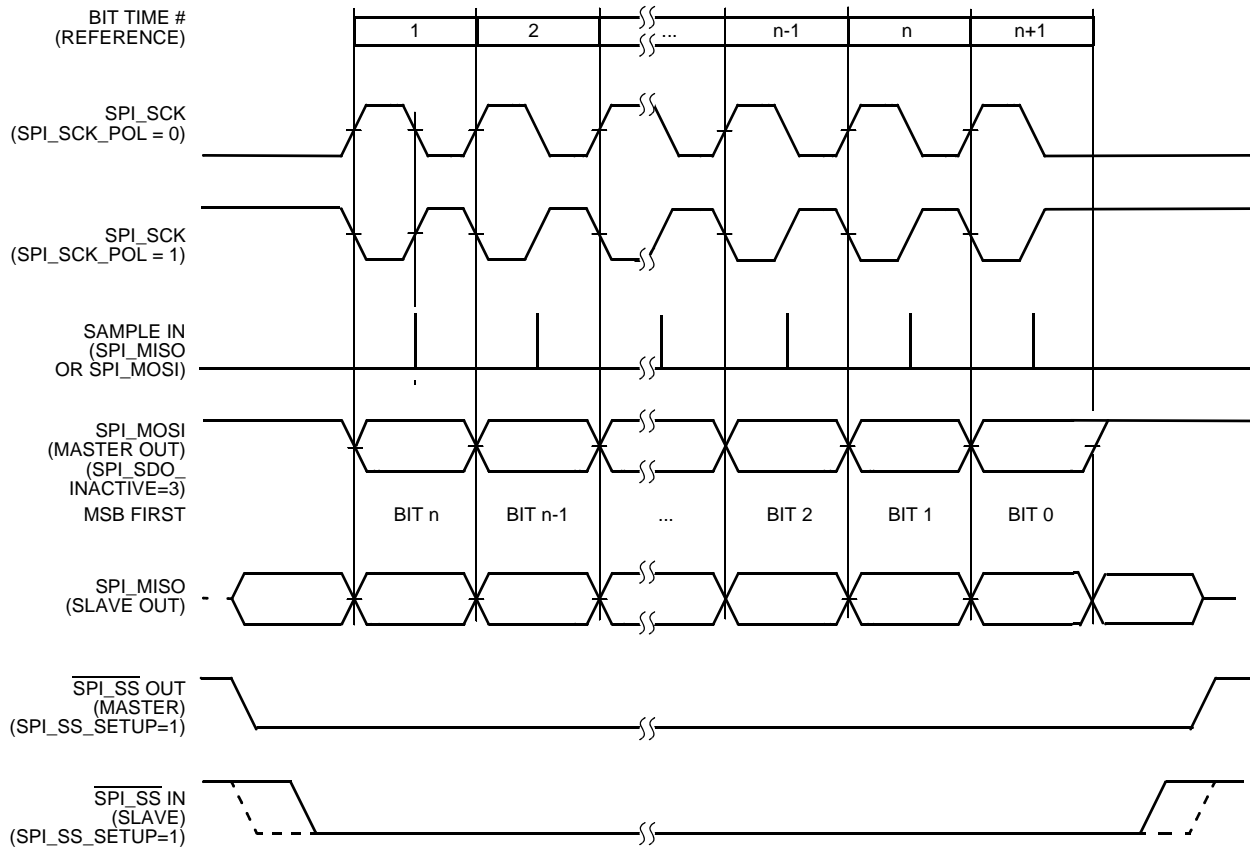


Figure 15-3. SPI Clock Formats (SPI_SCK_PHASE = 1)

When SPI_SCK_PHASE = 1, the slave begins to drive its SPI_MISO output when SPI_SS goes active, but the data is not defined until the first SPI_SCK edge. The first SPI_SCK edge shifts the first bit of data from the shifter onto the SPI_MOSI output of the master and the SPI_MISO output of the slave. The next SPI_SCK edge causes both the master and the slave to sample the data bit values on their SPI_MISO and SPI_MOSI inputs, respectively. At the third SPI_SCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the SPI_MOSI and SPI_MISO outputs of the master and slave, respectively.

The Figure 15-4 shows the clock formats when SPI_SCK_PHASE = 0. At the top of the figure, the bit times are shown for reference with bit 1 starting as the slave is selected (SPI_SS IN goes low), and the last bit ends at the last SPI_SCK edge. The MSB first line shows the order of SPI data bits. Both variations of SPI_SCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in SPI_SCK_POL. The SAMPLE IN waveform applies to the SPI_MOSI input of a slave or the SPI_MISO input of a master. The SPI_MOSI waveform applies to the SPI_MOSI output pin from a master and the SPI_MISO waveform applies to the SPI_MISO output from a slave. The SPI_SS OUT waveform applies to the slave select output from a master (provided SPI_SS_SETUP = 1). The master SPI_SS output goes active at the start of the first bit time of the transfer and goes inactive at least one SPI_SCK cycle after the end of the last bit time of the transfer (provide SPI_SCK_COUNT is set equal to SPI_DATA_LENGTH + 1). The SPI_SS IN waveform applies to the slave select input of a slave.

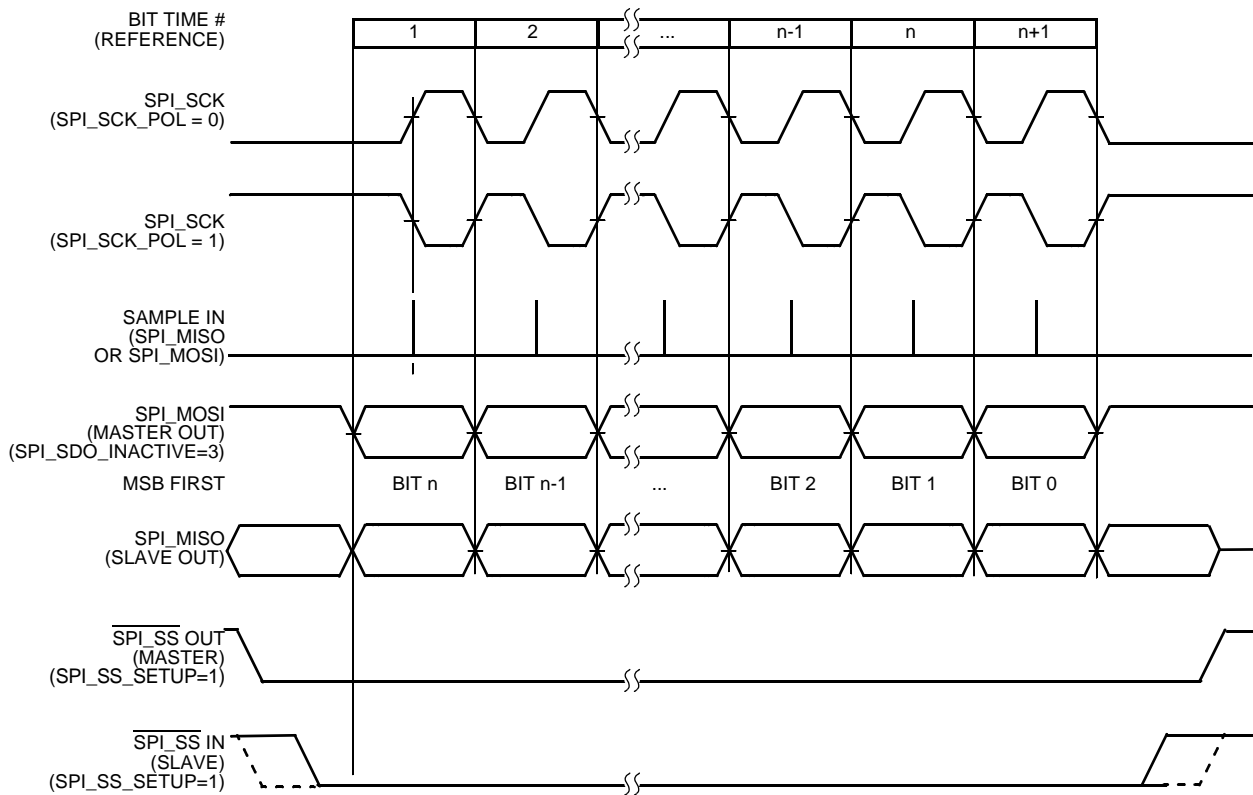


Figure 15-4. SPI Clock Formats ($SPI_SCK_PHASE = 0$)

When $SPI_SCK_PHASE = 0$, the slave begins to drive its SPI_MISO output with the first data bit value when SPI_SS goes active. The first SPI_SCK edge causes both the master and the slave to sample the data bit values on their SPI_MISO and SPI_MOSI inputs, respectively. At the second SPI_SCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the SPI_MOSI and SPI_MISO outputs of the master and slave, respectively.

15.5.5 SPI Interrupts

The SPI module has a single interrupt/status done bit, i.e., SPI_INT , Register SPI_STATUS . The SPI_INT is set when either the SPI is done sending data in master mode or is done receiving data in slave mode. There is no local mask bit for the SPI_INT . The interrupt for the module is enabled/disabled via the Interrupt Controller (ITC).

Completion of a master or slave transfer is controlled by the $SPI_DATA_LENGTH[6:0]$ field of Register SPI_CLK_CTRL :

- In master mode, the SPI_DATA_LENGTH value controls for how many SPI_SCK periods the SPI shifts out data to SPI_MOSI .
 - The number of data shifts is equal to SPI_DATA_LENGTH .

- The SPI_DATA_LENGTH field auto decrements, and when it reaches zero it changes SPI_MOSI to its inactive state (the inactive state is programmed in the SPI_SETUP Register) and SPI_INT is set high.
- This register must be reprogrammed for each SPI transaction.
- If SPI_DATA_LENGTH is greater than 32 (not allowed), then the data received from the external slave SPI device is shifted out after the 32nd shift.
- In slave mode, the SPI_DATA_LENGTH register defines the length in bits (or clocks) of the receive data.
 - The number of bits to receive is equal to SPI_DATA_LENGTH + 1.
 - The internal SPI_SCK count is automatically set to zeros when SPI_SS is deasserted or when SPI_DATA_LENGTH is reached.
 - Each time SPI_SCK count reaches SPI_DATA_LENGTH, the received data is saved in the SPI_RX_DATA buffer and the SPI_INT signal is set high.
 - If SPI_DATA_LENGTH is greater than 31(not allowed), only the last 32 bits received are saved in the SPI_RX_DATA buffers.

The SPI_INT status bit can be cleared by writing it to a “1”. It is also cleared with the start of a master SPI transfer which is caused by setting the SPI_START bit in the SPI_CLK_CTRL Register.

15.6 SPI Register Memory Map

The SPI module is programmed via a set of memory-mapped registers

- The base address is **0x8000_2000**.
- The register memory map for the module related to the base address is listed in [Table 15-3](#) and shows the 32-bit registers of the SPI module.

The registers should be accessed only as 32 bits.

Table 15-3. SPI Module Register Memory Map

Offset	[31:24]	[23:16]	[15:8]	[7:0]	Access Type	Access Width
+ 0x00	SPI_TX_DATA (SPI transmit data)				R/W	32
+ 0x04	SPI_RX_DATA (SPI received data)				R/W	32
+ 0x08	SPI_CLK_CTRL (SPI Status and Clock Control)				R/W	32
+ 0x0C	SPI_SETUP (SPI setup)				R/W	32
+ 0x10	SPI_STATUS (SPI status)				R/W	32

15.7 SPI Registers and Control Bits

15.7.1 Transmit Data Register (SPI_TX_DATA)

Transmit Data Register (SPI_TX_DATA) is a 32-bit shift register which is used for transmitting data. During master mode operation the data are shifted left (MSB-first), where SPI_TX_DATA[31] is the first bit to come out on SPI_MOSI. During slave mode operation, SPI_TX_DATA[31] is the first bit to come out on SPI_MISO.

SPI_TX_DATA				SPI Transmit Data									Addr Base+0x00			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
SPI_TX_DATA[31:16]																
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
SPI_TX_DATA[15:0]																
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-4. SPI Transmit Data Register Bit Descriptions

Bit Number	Description	Operation
0-31	SPI transmit data —	Transmit shift register data.

15.7.2 SPI Received Data Register (SPI_RX_DATA)

SPI Received Data Register (SPI_RX_DATA) is a 32-bit shift register which is used for receiving data. During master mode operation the data are shifted left (MSB-first), where SPI_RX_DATA[0] is the first bit to come in on SPI_MISO. During slave mode operation, SPI_TX_DATA[0] is the first bit to come in on SPI_MOSI.

SPI_RX_DATA				SPI Received Data									Addr Base+0x04			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
SPI_RX_DATA[31:16]																
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
SPI_RX_DATA[15:0]																
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-5. SPI Receive Data Register Bit Descriptions

Bit Number	Description	Operation
0-31	SPI receive data —	Receive shift register data.

15.7.3 SPI Clock Control Register (SPI_CLK_CTRL)

SPI_CLK_CTRL				Clock Control Register									Addr Base+0x08			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	SPI_SCK_COUNT								SPI_START	SPI_DATA_LENGTH						
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-6. SPI Clock Control Register Bit Descriptions

Bit Number	Description	Operation
16-31	Reserved	
8-15	SPI_SCK_COUNT[7:0] — In master mode this register determines how many SPI_SCK periods to generate in the following SPI transaction.	The number of SPI_SCK periods is equal to SPI_SCK_COUNT + 1. The SPI_SCK_COUNT register auto decrements during the SPI transaction and has to be set for each transaction. This register has no effect in slave mode.
7	SPI_START — In master mode this bit is a write-only bit used to initiate the SPI transaction.	By writing a '1' to SPI_START the SPI will exit its IDLE state which automatically turns on the gated clock in the SPI device and start the transaction. This bit has no effect in slave mode.

Table 15-6. SPI Clock Control Register Bit Descriptions

Bit Number	Description	Operation
0-6	<p>SPI_DATA_LENGTH[6:0] — In master mode the SPI_DATA_LENGTH register controls for how many SPI_SCK periods the SPI shifts out data from the Shift Register to SPI_MOSI. The number of data shifts is equal to SPI_DATA_LENGTH.</p>	<p>The SPI_DATA_LENGTH register auto decrements and when it reaches 0 it changes SPI_MOSI to its inactive state. The exact behavior of the inactive state is programmed in the SPI_SETUP Register. This Register must be reprogrammed for each SPI transaction. Notice that if SPI_DATA_LENGTH is greater than 32, then the data received from the external slave SPI device is shifted out after the 32nd shift.</p> <p>In slave mode the SPI_DATA_LENGTH register defines the length in bits of the receive data. The number of bits to receive is equal to SPI_DATA_LENGTH + 1. A internal SPI_SCK count is automatically set to zeros when SPI_SS is deasserted or when SPI_DATA_LENGTH is reached. Each time SPI_SCK count reaches SPI_DATA_LENGTH, the received data is saved in the SPI_RX_DATA buffer and the SPI_INT signal is set high. Notice that if SPI_DATA_LENGTH is greater than 31, only the last 32 bits received are saved in the SPI_RX_DATA buffer.</p>

15.7.4 SPI Setup Register (SPI_SETUP)

SPI_SETUP				SPI Setup Register									Addr Base+0x0C			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
															SPI_3WIRE	SPI_MODE
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
		SPI_SCK_FREQ				SPI_MISO_PHASE	SPI_SCK_PHASE	SPI_SCK_POL			SPI_SDO_INACTIVE_ST		SPI_SS_DELAY		SPI_SS_SETUP	
TYPE	r	rw	rw	rw	r	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-7. SPI Setup Register Bit Descriptions

Bit Number	Description	Operation
18-31	Reserved	
17	SPI_3WIRE — The SPI_3WIRE bit chooses between a SPI 3-wire or a SPI 4-wire connection.	When SPI_3WIRE is set high, the SPI serial data output and input will share the same port. When SPI_3WIRE is set low, the SPI serial data output and input have independent ports, SPI_MISO and SPI_MOSI. When the SPI 3-wire connection is selected, in master mode the SPI uses the MOSI (MOMI) pin for bidirectional SPI transfers and in slave mode the SPI uses the MISO (SISO) pin for bidirectional SPI transfers. The enable or disable of the MISO or MOSI output drivers are controlled by the SPI_SDO_INACTIVE_ST bits.
16	SPI_MODE — This bit determines if the SPI is a master device or slave device.	When SPI_MODE is low, the SPI is in master mode. When SPI_MODE is set high, the SPI is in slave mode.
15	Reserved	

Table 15-7. SPI Setup Register Bit Descriptions

Bit Number	Description	Operation
12-14	SPI_SCK_FREQ[2:0] — In master mode the SPI_SCK_FREQ register controls the SPI_SCK frequency: $\text{SPI_SCK} = \text{CLK} / (2^{(\text{SPI_SCK_FREQ} + 1)})$ This register has no effect in slave mode.	
11	Reserved	
10	SPI_MISO_PHASE — In master mode the SPI_MISO_PHASE determines which edge of SPI_SCK is used to sample the SPI_MISO input signal. This allows the communication to SPI devices that latches data on one clock edge and clocks out data either on the same edge or on the opposite edge.	When SPI_MISO_PHASE is low SPI_MISO is latched with the opposite SPI_SCK edge as it clocks out SPI_MOSI. This is accomplished by pre-latching into a register on the preceding clock edge before being clocked into the Shift Register. When SPI_MISO_PHASE is high SPI_MISO is latched with the same SPI_SCK edge as it clocks out SPI_MOSI. The figure in the next section illustrates this. (Note: This figure is only a illustration and does not show the real implementation). The SPI_MISO_PHASE register has no effect in slave mode.
9	SPI_SCK_PHASE — The SPI_SCK_PHASE register determines the phase of SPI_SCK with respect to SPI_MISO and SPI_MOSI.	
8	PI_SCK_POL — The SPI_SCK_POL register is used to control the SPI_SCK polarity.	
6-7	Reserved	
4-5	SPI_SDO_INACTIVE_ST — In master mode SPI_SDO_INACTIVE_ST controls the state of the SPI_MOSI_OUT and SPI_MOSI_OUT_EN output signals when the SPI device is not shifting data out of the Shift Register (inactive state). If SPI_DATA_LENGTH is set to zeros, the SPI Master is always in its inactive state. The SPI_3WIRE bit has no effect to this behavior in master mode. While the SPI Master is active, SPI_MOSI_OUT is connected to the serial data output and SPI_MOSI_OUT_EN is set high. In slave mode SPI_SDO_INACTIVE_ST controls the state of the SPI_MISO_OUT and SPI_MISO_OUT_EN output signals when SPI_SS is deasserted or SPI_3WIRE has been set high. While SPI_SS is asserted and SPI_3WIRE is reset low, SPI_MISO_OUT is connected to the serial data output and SPI_MISO_OUT_EN is set high.	

Table 15-7. SPI Setup Register Bit Descriptions

Bit Number	Description	Operation
2-3	SPI_SS_DELAY — In master mode and SPI_SS set in Auto Mode (SPI_SS_SETUP = 0 or 1), SPI_SS_DELAY determines when the SPI_SS is asserted and deasserted. SPI_SS is asserted SPI_SS_DELAY + 1 SPI_SCK periods before the first shift and deasserted SPI_SS_DELAY + 1 SPI_SCK period after the last shift. This register has no effect in slave mode.	
0-1	SPI_SS_SETUP — In master mode the SPI_SS_SETUP determines the exact behavior of the SPI_SS_OUT signal and in slave mode it determines the behavior of the SPI_SS_IN signal. In master mode if a SPI_SS is setup to be automatic, it will automatically select the SPI device prior to the first SPI_SCK edge and automatically de-select after the last SPI_SCK edge.	

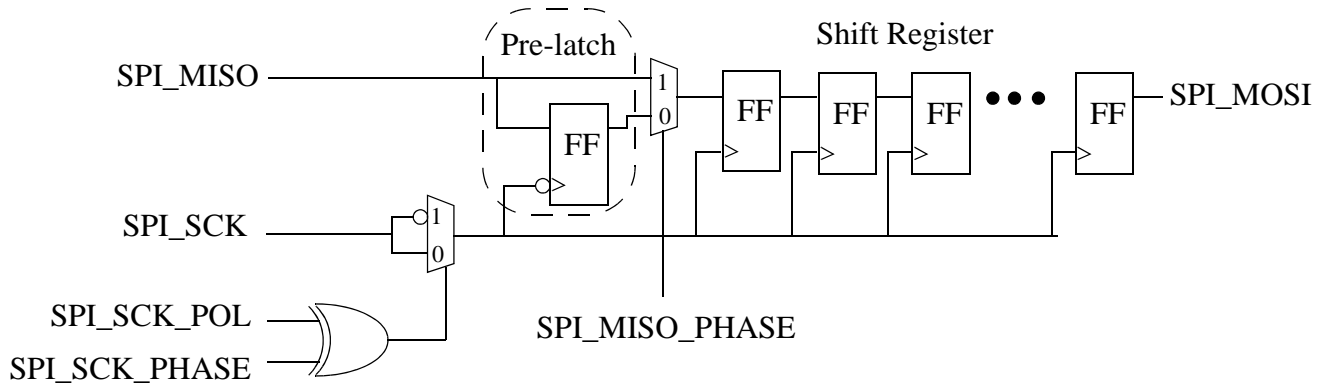


Figure 15-5. SPI Master MISO Sampling

Table 15-8. SPI_MOSI Inactive State (Master Mode)

SPI_MODE	SPI_SDO_IN ACTIVE_ST[1:0]	SPI_MOSI_OUT	SPI_MOSI_OUT_EN	PAD
0	00	0	0	'z'
0	01	1	0	'z'
0	10	0	1	0
0	11	1	1	1
1	XX	0	0	'z'

Table 15-9. SPI_MISO Inactive State (Slave Mode)

SPI_MODE	SPI_SS	SPI_3WIRE	SPI_SDO_IN ACTIVE_ST[1:0]	SPI_MISO_OUT	SPI_MISO_OUT_EN	PAD
1	deasserted	X	00	0	0	'z'
1	deasserted	X	01	1	0	'z'
1	deasserted	X	10	0	1	0
1	deasserted	X	11	1	1	1
1	asserted	0	XX	SDO	1	SDO
1	asserted	1	0X	SDO	0	'z'
1	asserted	1	1X	SDO	1	SDO
0	X	X	XX	0	0	'z'

Table 15-10. SPI_SS Behavior

SPI_SS_SETUP	SPI_SS_OUT (Master Mode)	SPI_SS_IN (Slave Mode)
00	Auto/Active High	Active High
01	Auto/Active Low	Active Low
10	Low	Active High
11	High	Active Low

15.7.5 SPI Status Register (SPI_STATUS)

SPI_STATUS				SPI Status Register									Addr Base+0x10			
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
								SPI_FIRST_DATA				SPI_OVERFLOW				SPI_INT
TYPE	rw	rw	rw	rw	rw	rw	rw	r	w	rw	rw	r,w1c	rw	rw	rw	r,w1c
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-11. SPI Status Register Bit Descriptions

Bit Number	Description	Operation
9-31	Reserved	
8	SPI_FIRST_DATA — This bit is a read-only status bit generated by the slave SPI device.	This bit is set high when the SPI has received data for the first transfer after SPI_SS went active. For all remaining transfers, while SPI_SS is kept active, SPI_FIRST_DATA is reset to zero. This bit is not used in master mode.
5-7	Reserved	
4	SPI_OVERFLOW — This bit is a read-only and write-one-to-clear SPI overflow status bit generated by the slave SPI device.	This bit is set when the SPI has received data in slave mode and the SPI interrupt from the previous received data has not been cleared. The SPI_OVERFLOW bit can be cleared by writing a “1” to it. This bit is not used in master mode.
1-3	Reserved	
0	SPI_INT — This bit (also referred as SPI done) is a read-only and write-one-to-clear interrupt bit generated by the master or slave SPI.	This bit is set high either when the SPI is done sending data in master mode or the SPI is done receiving data in slave mode. This bit can be cleared by writing a “1” to it. This bit is also cleared with the start of a master SPI transfer which is caused by setting the SPI_START bit in the SPI_CLK_CTRL register.

15.8 Timing Information

The SPI timing information is shown in the figure below. The SPI Master logic is clocked with the MCU clock which highest frequency is 26 Mhz. Thus the highest SPI Master SPI_SCK frequency is 13 Mhz (1/2 MCU clock). The SPI Slave logic is clocked by the incoming SPI_SCK and asynchronously reset by the deassertion of SPI_SS. The timing diagram is only applicable to the SPI I/O when they are in their input mode. Notice that the setup and hold times is with respect to the capturing SPI_SCK edge which could be the rising or falling edge and not just the rising edge as shown in this diagram. See the two SPI Clock Formats diagrams for more information on the capturing SPI_SCK edge. Also keep in mind that the setup and hold time of SPI_SS is with respect to its assertion which could be either active low or active high.

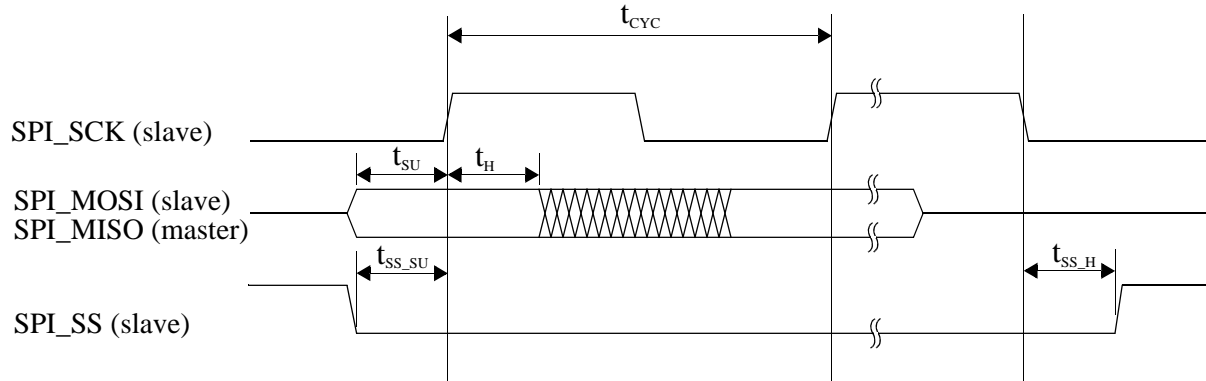


Figure 15-6. SPI Timing Diagram

Table 15-12. SPI Timing Limits

PARAMETER	MNEMONIC	SPECIFICATION			UNITS
		MIN	TYP	MAX	
Slave SPI_SCK Period	t_{cyc}		38		nsec
Slave SPI_MOSI & Master SPI_MISO1 Setup Time	t_{su}		10		nsec
Slave SPI_MOSI & Master SPI_MISO1 Hold Time	t_h		10		nsec
Slave SPI_SS Setup Time	t_{ss_su}		10		nsec
Slave SPI_SS Hold Time	t_{ss_h}		10		nsec

Chapter 16

Synchronous Serial Interface Module (SSI)

This chapter describes the Synchronous Serial Interface (SSI) architecture, programming model, operating modes, and initialization.

16.1 Overview

The SSI is a full-duplex, serial port that allows the MC1322x to communicate with a variety of serial devices. These serial devices can be standard CODer-DECoder (CODECs), Digital Signal Processors (DSPs), microprocessors, peripherals, and popular industry audio CODECs that implement the inter-IC sound bus standard (I²S) and Intel AC97 standard.

SSI is typically used to transfer samples in a periodic manner. The SSI consists of independent transmitter and receiver sections with independent clock generation and frame synchronization.

16.1.1 Features

The SSI includes the following features:

- Independent (asynchronous) or shared (synchronous) transmit and receive sections with separate or shared internal/external clocks and frame syncs, operating in Master or Slave mode.
- Normal mode operation using frame sync
- Network mode operation allowing multiple devices to share the port with as many as thirty-two time slots
- Gated Clock mode operation requiring no frame sync
- 2 sets of Transmit and Receive FIFOs. Each of the four FIFOs is 8x32 bits. The two sets of Tx/Rx FIFOs can be used in Network mode to provide 2 independent channels for transmission and reception
- Programmable data interface modes such like I²S, LSB, MSB aligned
- Programmable word length (8, 10, 12, 16, 18, 20, 22 or 24 bits)
- Program options for frame sync and clock generation
- Programmable I²S modes (Master, Slave or Normal). Oversampling clock, `ccm_ssi_clk` available as output from SRCK in I²S Master mode
- Completely separate clock and frame sync selections for the receive and transmit sections. In AC97 standard, the clock is taken from an external source and frame sync is generated internally.
- External `ccm_ssi_clk` input for use in I²S Master mode. Programmable oversampling clock (`SYS_CLK/ccm_ssi_clk`) of the sampling frequency available as output in master mode at SRCK, when operated in sync mode.
- Programmable internal clock divider

- Time Slot Mask Registers for reduced CPU overhead (for Tx and Rx both)
- SSI power-down feature
- Programmable wait states for CPU accesses
- IP Interface for register accesses, compliant to SRS 3.0.2 standard

16.2 Block Diagram

Figure 16-1 shows the SSI block diagram. The SSI consists of control registers to set up the port, status register, separate transmit and receive circuits with FIFO registers, and separate serial clock and frame sync generation for the transmit and receive sections. The second set of Tx and Rx FIFOs replicates the logic used for the first set of FIFOs.

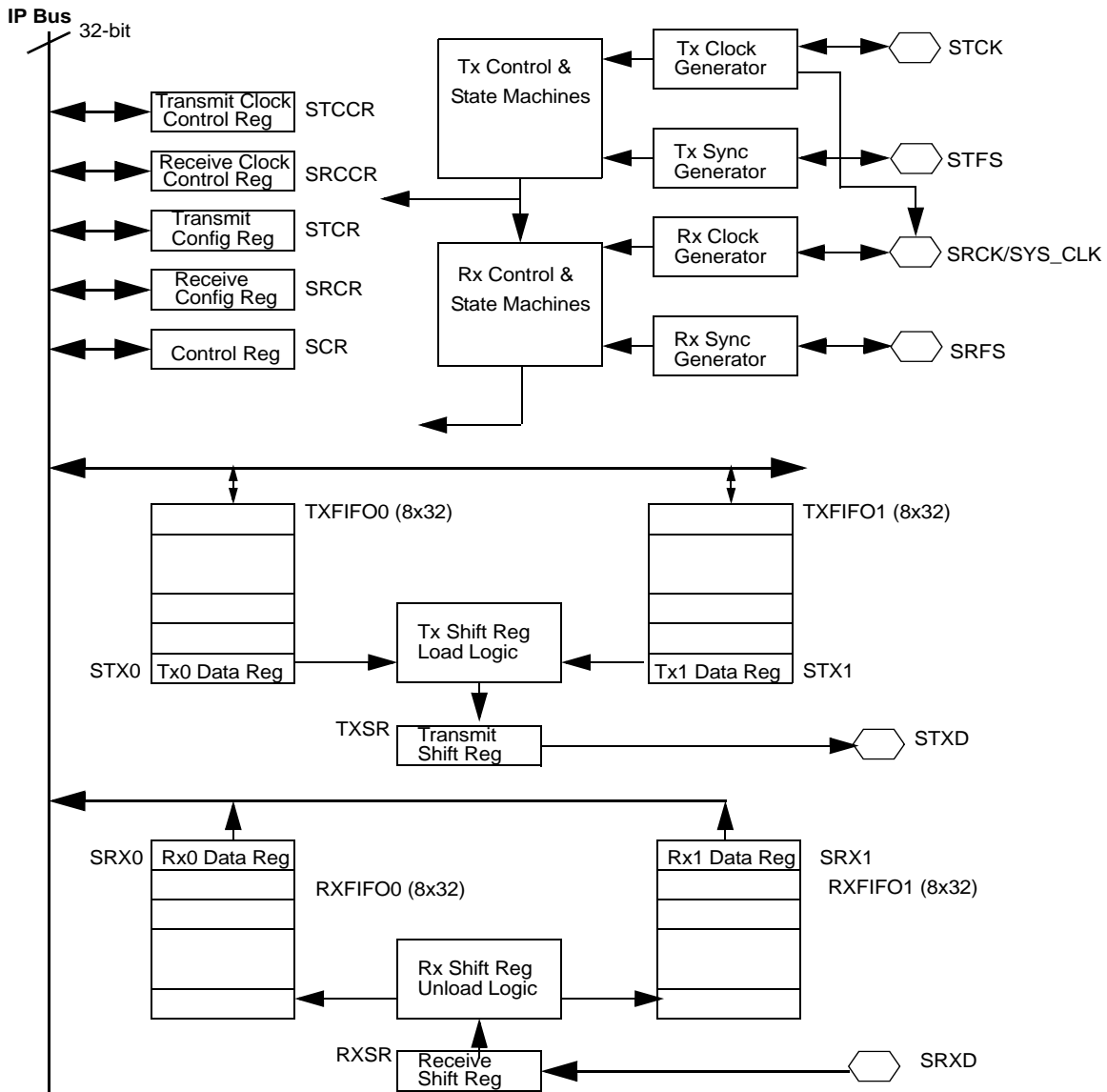


Figure 16-1. SSI Block Diagram

16.3 Modes of Operation

The SSI has the following basic operating modes.

- Normal mode
 - Asynchronous protocol
 - Synchronous protocol
- Network mode
 - Asynchronous protocol
 - Synchronous protocol
- Gated Clock mode
 - Synchronous protocol only

These modes can be programmed by several bits in the SSI control registers. See [Table 16-1](#) for the list of SSI operating modes and some of the typical applications in which they can be used:

Table 16-1. SSI Operating Modes

TX, RX Sections	Serial Clock	Mode	Typical Application
Asynchronous	Continuous	Normal	Multiple synchronous CODECs
Asynchronous	Continuous	Network	TDM CODEC or DSP networks
Synchronous	Continuous	Normal	Multiple synchronous CODECs
Synchronous	Continuous	Network	TDM CODEC or DSP network
Synchronous	Gated	Normal	SPI-type devices; DSP to MCU

The transmit and receive sections of the SSI has two modes:

Synchronous mode The transmitter and the receiver use a common clock and frame synchronization signal. The RXBIT0 and RSHFD bits in SRCR still affect shifting-in of received data in synchronous mode.

Asynchronous mode The transmitter and receiver each has its own clock and frame synchronization signals. Continuous or Gated Clock mode can be selected. In Continuous mode, the clock runs continuously. In Gated Clock mode, the clock is only functioning during transmission.

The SSI also had a Normal and a Network mode.

Normal mode The SSI functions with one data word of I/O per frame.

Network mode Any number from two to thirty-two data words of I/O per frame can be used. Network mode is typically used in star time division multiplex networks with other processors or CODECs, allowing interface to time division multiplexed networks without additional logic. Use of the gated clock is not allowed in Network mode.

These modes allow the SSI to communicate with a wide variety of devices.

The SSI supports both Normal and Network modes which can be selected independently of whether the transmitter and receiver are in Synchronous or Asynchronous mode. Typically, these protocols are used in a periodic manner, where data is transferred at regular intervals, such as at the sampling rate of an external CODEC. Both modes use the concept of a frame. The beginning of the frame is marked with a frame sync when programmed with continuous clock. The frame sync occurs at a periodic interval. The length of the frame is determined by the DC[4:0] bits in either the SRCCR or STCCR register, depending on whether data is being transmitted or received. The number of words transferred per frame depends on the mode of the SSI.

In Normal mode, one data word is transferred per frame. In Network mode, the frame is divided into anywhere between two and thirty-two time slots, where in each time slot one data word can optionally be transferred.

Apart from the above basic modes of operation, SSI supports the following modes which require some specific programming.

- I²S mode
- AC97 mode
 - AC97 Fixed mode
 - AC97 Variable mode

In (non-I²S) slave modes (external frame sync), the programmed word length setting of the SSI should be equal to the word length setting of the master. In I²S slave mode, the programmed word length setting of the SSI can be lesser than or equal to the word length setting of the I²S master (external CODEC).

In slave modes, the programmed frame length setting (DC bits) of the SSI can be lesser than or equal to the frame length setting of the master (external CODEC).

16.3.1 Normal Mode

Normal mode is the simplest mode of the SSI. It transfers data in one time slot per frame. A time slot is a unit of data and the WL[3:0] bits define the number of bits in a time slot. In Continuous Clock mode, a frame sync occurs at the beginning of each frame. The length of the frame is determined by the following factors:

- The period of the Serial Bit Clock (DIV2, PSR, PM[7:0] bits for internal clock or the frequency of the external clock on the STCK port)
- The number of bits per time slot (WL[3:0] bits)
- The number of time slots per frame (DC[4:0] bits)

If Normal mode is configured with more than one time slot per frame, data is transferred only in the first time slot. No data is transferred in subsequent time slots. In Normal mode, DC[4:0] values corresponding to more than a single time slot in a frame, only result in lengthening the frame. Data transfer only takes place during the first time slot of the frame.

16.3.1.1 Normal Mode Transmit

The conditions for data transmission from the SSI in Normal mode are:

- SSI enabled (SSIEN = 1)
- Enable FIFO and configure Transmit and Receive Watermark if FIFO is used.
- Write data to Transmit Data Register (STX)
- Transmitter enabled (TE = 1)
- Frame sync active (for continuous clock case)
- Bit clock begins (for gated clock case)

When these conditions occur in Normal mode, the next data word is transferred into the Transmit Shift Register (TXSR) from the Transmit Data Register 0 (STX0), or from the Transmit FIFO 0 Register, if transmit FIFO 0 is enabled. The new data word is transmitted immediately.

If transmit FIFO 0 is not enabled and the transmit data register empty (TDE0) bit is set, transmit interrupt 0 occurs if the transmit interrupt enable (TIE) and TDE0_EN bits are set.

The Transmit FIFO Empty 0 (TFE0) bit is set if the Transmit FIFO 0 reaches the selected threshold. If transmit FIFO 0 is enabled and the Transmit FIFO Empty (TFE0) bit is set, transmit interrupt 0 occurs if the transmit interrupt enable (TIE) and TFE0_EN bits are set. If transmit FIFO 0 is enabled and filled with data, 8 data words can be transferred before the core must write new data to the STX0 register.

The STXD port is disabled except during the data transmission period. For a continuous clock, the optional frame sync output and clock outputs are not disabled, even if both receiver and transmitter are disabled.

16.3.1.2 Normal Mode Receive

The conditions for data reception from the SSI are:

- SSI enabled (SSIEN = 1)
- Receiver enabled (RE = 1)
- Frame sync active (for continuous clock case)
- Bit clock begins (for gated clock case)

In Normal mode with a continuous clock and the listed conditions met, each time the frame sync signal is generated (or detected) a data word is clocked in. With a gated clock and the listed conditions met, each time the clock begins, a data word is clocked in.

If receive FIFO 0 is not enabled, the received data word is transferred from the Receive Shift Register (RXSR) to the Receive Data Register 0 (SRX0), the Receive Data Ready 0 (RDR0) flag is set. Receive Interrupt 0 occurs if RIE and RDR0_EN bits are set.

If receive FIFO 0 is enabled, the received data word is transferred to the Receive FIFO 0. The Receive FIFO Full 0 (RFF0) flag is set if the Receive Data Register (SRX0) is full and Receive FIFO 0 reaches the selected threshold. Receive Interrupt 0 occurs if Receive Interrupt Enable (RIE) and RFF0_EN bits are set.

The core program must read the data from the Receive Data Register 0 (SRX0) before a new data word is transferred from the Receive Shift Register (RXSR), otherwise the Receive Overrun Error 0 (ROE0) bit is

set. If receive FIFO 0 is enabled, the Receive Overrun Error 0 (ROE0) bit is set when the Receive FIFO 0 data level reaches the selected threshold and a new data word is ready to be transferred to the Receive FIFO 0.

Figure 16-2 shows the transmitter and receiver timing for an 8-bit word in the first time slot in Normal mode, continuous clock with a late word length frame sync. The Tx Data register is loaded with the data to be transmitted. On arrival of the clock, this data is transferred to the Transmit Shift Register which gets transmitted on arrival of the frame-sync on the STXD output. Simultaneously, the Receive Shift Register shifts in the received data available on the SRXD input and at the end of the time slot, this data is transferred to the Rx Data Register.

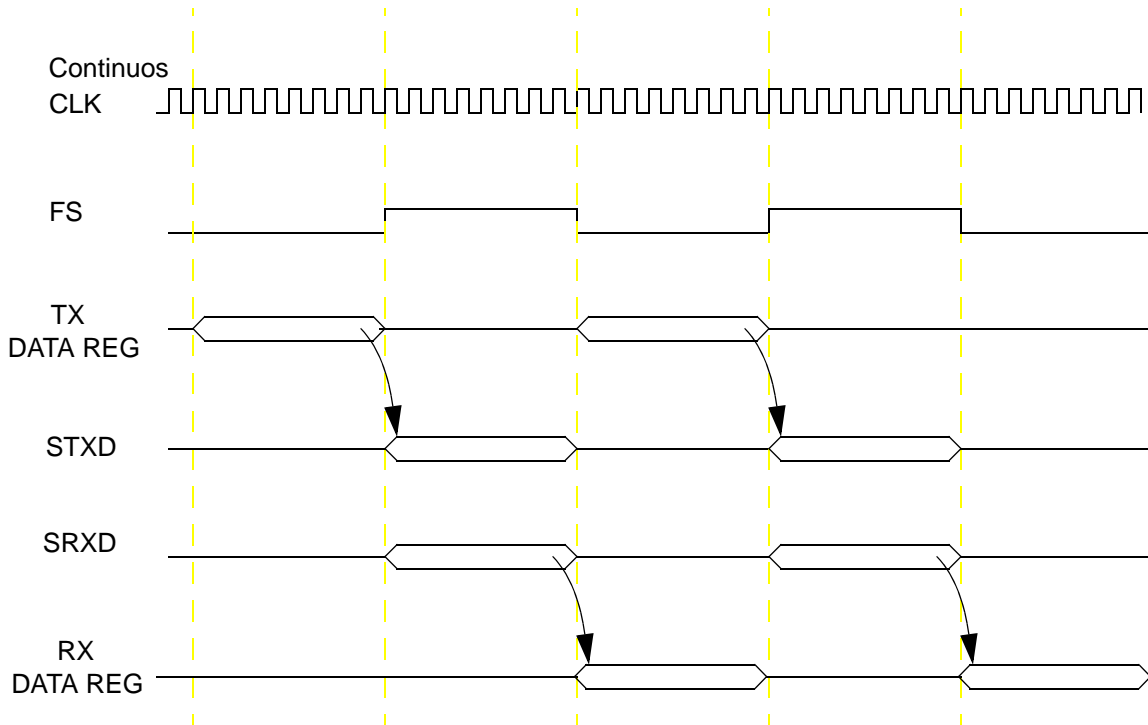


Figure 16-2. Normal Mode Timing - Continuous Clock

NOTE

A pull-down resistor is required in the gated clock case because the clock port is disabled between transmissions.

The Tx Data register is loaded with the data to be transmitted. On arrival of the clock, this data is transferred to the Transmit Shift Register which gets transmitted on arrival of the frame-sync on the STXD output. Simultaneously, the Receive Shift Register shifts in the received data available on the SRXD input and at the end of the time slot, this data is transferred to the Rx Data Register. In case of Internal Gated clock mode, the Tx Data line and clock output port are put in the high-impedance state at the end of transmission of the last bit (at the completion of the complete clock cycle), whereas, in External Gated clock mode, the Tx Data line is tri-stated at the last inactive edge of the incoming bit clock (during the last bit in a data word).

Figure 16-3 shows an example for internal (SSI generates clock) gated clock mode.

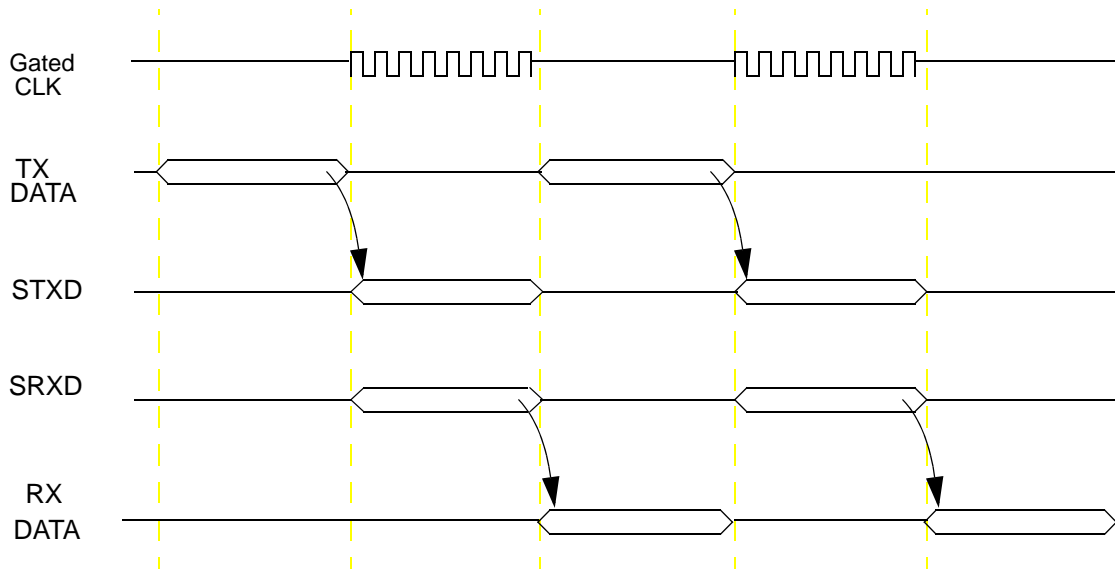


Figure 16-3. Normal Mode Timing - Internal Gated Clock

Figure 16-4 shows an example using external (SSI receives clock) gated clock mode.

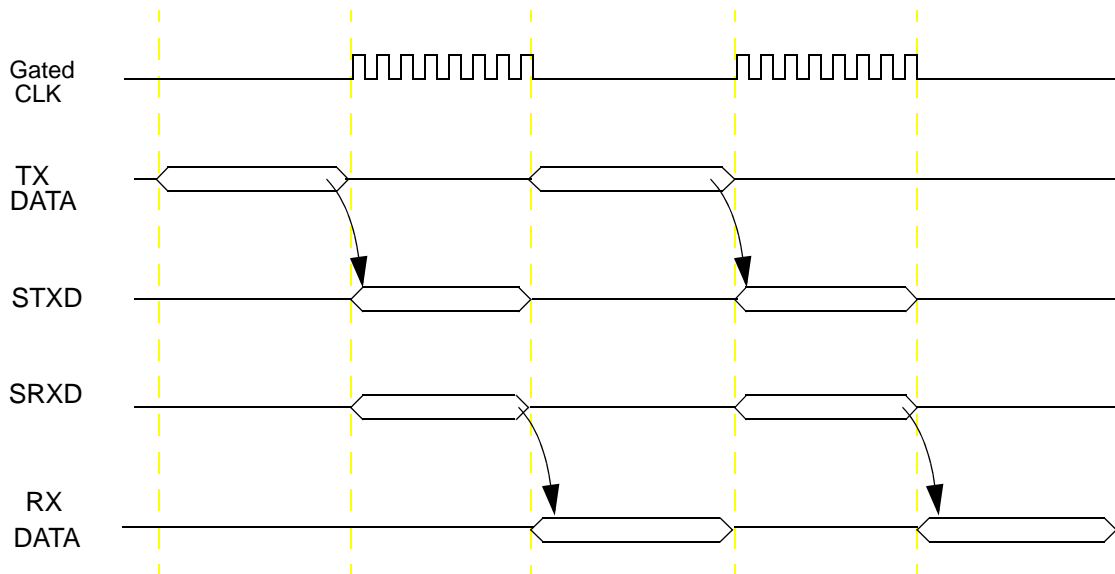


Figure 16-4. Normal Mode Timing - External Gated Clock

16.3.2 Network Mode

Network mode creates a Time Division Multiplexed (TDM) network, such as a TDM CODEC network or a network of DSPs. In Continuous Clock mode, a frame sync occurs at the beginning of each frame. In this mode, the frame is divided into more than one time slot. During each time slot, one data word can be transferred. Each time slot is then assigned to an appropriate CODEC or DSP on the network. The DSP can be a master device that controls its own private network, or a slave device that is connected to an existing TDM network and occupies a few time slots.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots and transmission and/or reception of one data word can occur in each time slot (rather than in just the frame sync time slot as in Normal mode). The frame rate dividers, controlled by the DC[4:0] bits, select two to thirty-two time slots per frame. The length of the frame is determined by the following factors:

- The period of the serial bit clock (PSR, PM[7:0] bits for internal clock, or the frequency of the external clock on the STCK port)
- The number of bits per sample (WL[3:0] bits)
- The number of time slots per frame (DC[4:0] bits)

In Network mode, data can be transmitted in any time slot. The distinction of the Network mode is that each time slot is identified with respect to the frame sync (data word time). This time slot identification allows the option of transmitting data during the time slot by writing to the STX registers or ignoring the time slot as determined by STMSK register bits. The receiver is treated in the same manner and received data is only transferred to the receive data register/fifo if the corresponding time slot is enabled (through SRMSK).

By utilizing the STMSK and SRMSK registers, software only has to service the SSI during valid time slots. This eliminates any overhead associated with unused time slots. See [Section 16.7.3.18, “SSI Transmit Time Slot Mask Register \(STMSK\)”](#) and [Section 16.7.3.19, “SSI Receive Time Slot Mask Register \(SRMSK\)”](#) for more information on STMSK and SRMSK.

In the Two-Channel mode of operation, the second set of Transmit and Receive FIFOs and Data Registers are used to create two separate channels. These channels are completely independent, with a their own set of Core interrupts and DMA requests, which are identical to the ones available for the default channel. In this mode, data is transmitted/received in enabled time slots alternately from/to FIFO 0 and FIFO 1, starting from FIFO 0. The first data word is taken from FIFO 0 and transmitted in the first enabled time slot and subsequently, data is loaded from FIFO 1 and FIFO 0 alternately and transmitted. Similarly, the first received data is sent to FIFO 0 and subsequent data is sent to FIFO 1 and FIFO 0 alternately. Time slots can be selected through the Transmit and Receive Time Slot Mask registers (STMSK and SRMSK). For using this mode of operation, the TCH_EN bit (SCR[8]) needs to be set.

16.3.2.1 Network Mode Transmit

The transmit portion of SSI is enabled when the SSIEN and the TE bits in the SCR are both set. However, for continuous clock, when the TE bit is set, the transmitter is enabled only after detection of a new frame sync (transmission starts from the next frame boundary).

Normal start-up sequence for transmission is to perform the following:

1. Write the data to be transmitted to the STX register. This clears the TDE flag.
2. Set the TE bit to enable the transmitter on the next word boundary (for continuous clock case).
3. Enable transmit interrupts.

Alternatively, the programmer may decide not to transmit in a time slot by configuring the STMSK. The TDE flag is not cleared, but the STXD port remains disabled during the time slot. When the frame sync is detected or generated (continuous clock), the first enabled data word is transferred from the STX register to the TXSR and is shifted out (transmitted). When the STX register is empty, the TDE bit is set, which causes a transmitter interrupt (in case the FIFO is disabled) to be sent if the TIE bit is set. Software can poll the TDE bit or use interrupts to reload the STX register with new data for the next time slot. Failing to reload the STX register before the TXSR is finished shifting (empty) causes a transmitter underrun and the TUE error bit is set. In case the FIFO is enabled, the TFE flag is set in accordance with the watermark setting and this flag causes the transmitter interrupt to occur.

The operation of clearing the TE bit disables the transmitter after completion of transmission of the current frame. Setting the TE bit enables transmission from the next frame. During that time the STXD port is disabled. The TE bit should be cleared after the TDE bit is set to ensure that all pending data is transmitted.

To summarize, the Network mode transmitter generates interrupts every enabled time slot and requires the core program to respond to each enabled time slot. These responses from the core are one of the following:

- Write data in data register to enable transmission in the next time slot.
- Configure the time slot register to disable transmission in the next time slot (unless time slot is already masked by STMSK register bit).
- Do nothing—transmit underrun occurs at the beginning of the next time slot and the previous data is re-transmitted.

In the Two-Channel mode of operation, both the channels (Data Registers, FIFOs, Interrupts and DMA requests) operate in the same manner, as described above. The only difference in case of the second channel is that the Interrupts related to this channel are generated only in case this mode of operation is selected (TDE1 is low by default).

16.3.2.2 Network Mode Receive

The receiver portion of the SSI is enabled when both the SSIEN and the RE bits in the SCR are set. However, the receive enable only takes place during that time slot if RE is enabled before the second to last bit of the word. If the RE bit is cleared, the receiver is disabled at the end of the current frame. SSI is capable of finding the start of the next frame automatically. When the word is completely received, it is transferred to the SRX register, which sets the RDR bit (Receive Data Ready). Setting the RDR bit causes a receive interrupt to occur if the receiver interrupt is enabled (the RIE bit is set). The second data word (second time slot in the frame), begins shifting in immediately after the transfer of the first data word to

the SRX register. The core program has to read the data from the Receive Data Register (which clears RDR) before the second data word is completely received (ready to transfer to RX data register) or a receive overrun error occurs (the ROE bit is set).

An interrupt can occur after the reception of each enabled data word or the programmer can poll the RDR flag. The core program response can be one of the following:

- Read RX and use the data.
- Read RX and ignore the data.
- Do nothing—the receiver overrun exception occurs at the end of the current time slot.

NOTE

For a continuous clock, the optional frame sync output and clock output signals are not affected, even if the transmitter or receiver is disabled. TE and RE do not disable the bit clock or the frame sync generation. To disable the bit clock and the frame sync generation, the SSIEN bit in the SCR can be cleared, or the port control logic external to the SSI (for example, in the IOMUX) can be reconfigured.

In the Two-Channel mode of operation, both the channels (Data Registers, FIFOs, Interrupts and DMA requests) operate in the same manner, as described above. The only difference in case of the second channel is that the Interrupts related to this channel are generated only in case this mode of operation is selected.

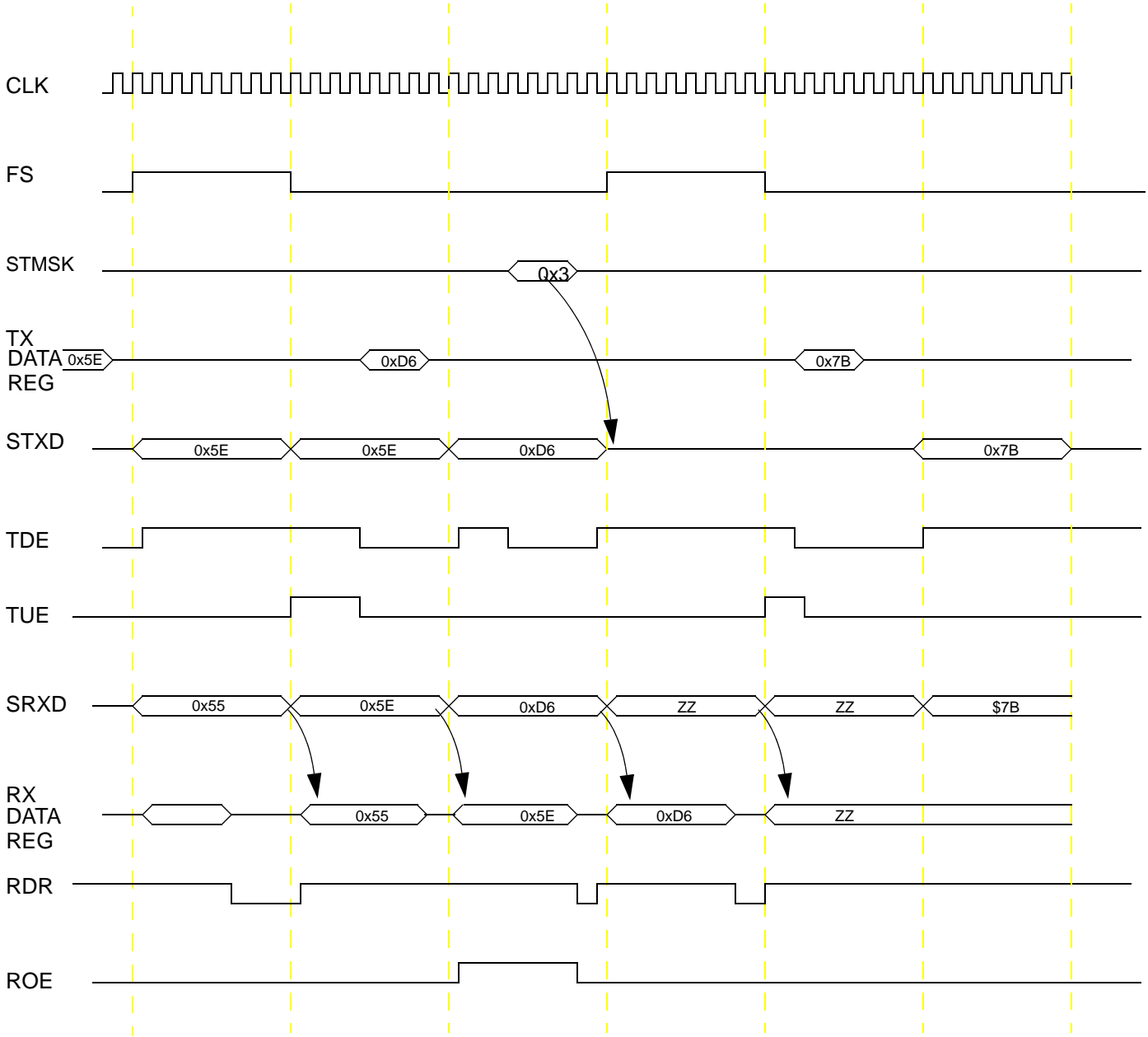
The transmitter and receiver timing for an 8-bit word with continuous clock, FIFO disabled, three words per frame sync in Network mode is shown in [Figure 16-5 \(“Network Mode Timing - Continuous Clock”\)](#).

NOTE

The transmitter repeats the value 0x5E because of an underrun condition

For the transmit section, the STMSK value is updated in the last time slot of frame 1, to mask the first two time slots (0x3). This value takes effect from the next time slot and consequently, the next frame transmits data in the third time slot only.

For the receive section, data received on the SRXD pin gets transferred to the Rx Data register at the end of each time slot. If the FIFO is disabled, the RDR flag gets set and causes a receiver interrupt if RE, RIE and RDR_EN bits are set. If the FIFO is enabled, then the RFF flag is used for interrupt generation (this flag is set in accordance with the watermark settings). Here all time slots are enabled. The receive data ready flag is set after reception of the first data (0x55). Since the flag is not cleared (Rx Data Register is not read by core), the Receive Overrun Error (ROE) flag is set on reception of the next data (0x5E). ROE flag is cleared on writing ‘1’ to the corresponding interrupt status bit in SSI Status Register.



Note: Processor must write '1' to the corresponding TUE/ROE Interrupt status bit in SISR to clear TUE/ROE Interrupt.

Figure 16-5. Network Mode Timing - Continuous Clock

16.3.2.3 Gated Clock Mode

Gated Clock mode is often used to hook up to SPI-type interfaces on Microcontroller Units (MCUs) or external peripheral chips. In Gated Clock mode, the presence of the clock indicates that valid data is on the STXD or SRXD ports. For this reason, no frame sync is needed in this mode. Once transmission of data has completed, the clock is pulled to the inactive state. Gated clocks are allowed for both the transmit and receive sections with either internal or external clock in Normal mode. Gated clocks are not allowed in Network mode.

The clock runs when the TE bit and/or the RE bit are appropriately enabled. For the case of internally generated clock, all internal bit clocks, word clocks, and frame clocks continue to operate. When a valid time slot occurs (such as the first time slot in Normal mode), the internal bit clock is enabled onto the appropriate clock port. This allows data to be transferred out in periodic intervals in Gated Clock mode. With an external clock, the SSI waits for a clock signal to be received. Once the clock begins, valid data is shifted in. Care should be taken to clear all DC bits (0x00000) when SSI is used in Gated mode.

For Gated clock operated in external clock mode, a proper clock signalling must be applied to the SSI STCK in order for it to function properly. If the SSI uses rising edge transition to clock data (TSCKP=0) and the falling edge transition to latch data (RSCKP=0), the clock must be in an active low state when idle. If the SSI uses falling edge transition to clock data (TSCKP=1) and the rising edge transition to latch data (RSCKP=1), the clock must be in an active high state when idle. Figure 16-6, through Figure 16-9 illustrate the different edge clocking/latching.

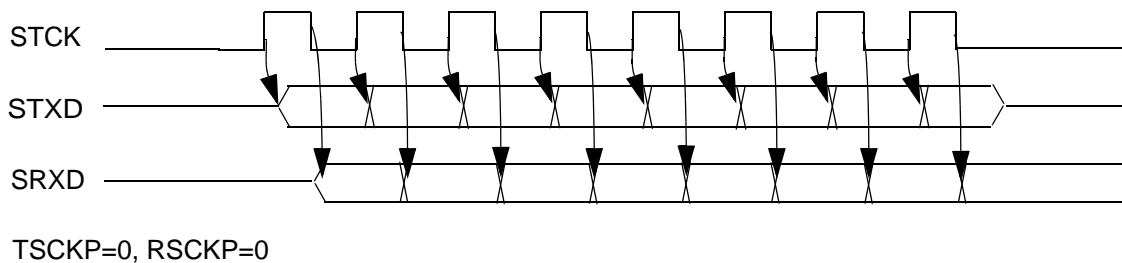


Figure 16-6. Internal Gated Mode Timing - Rising Edge Clocking / Falling Edge Latching

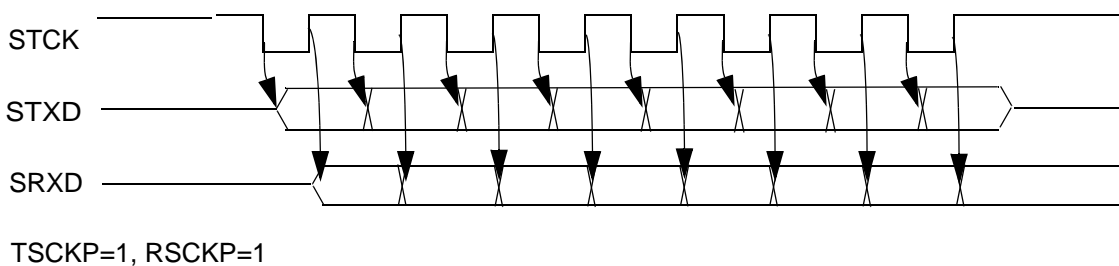
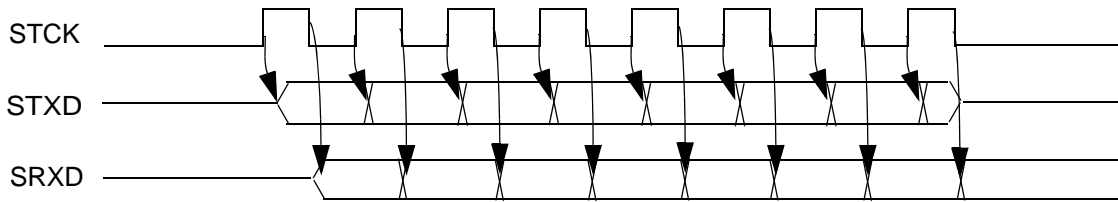
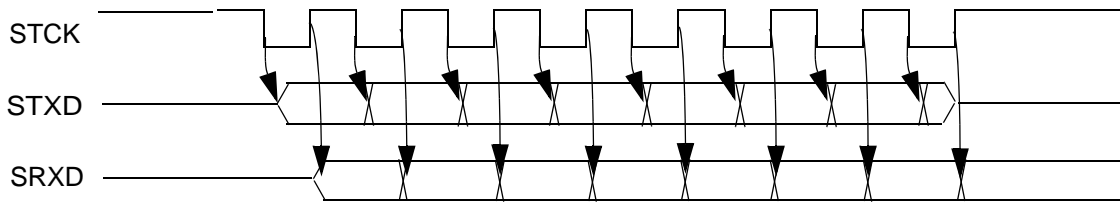


Figure 16-7. Internal Gated Mode Timing - Falling Edge Clocking / Rising Edge Latching



TSCKP=0, RSCKP=0

Figure 16-8. External Gated Mode Timing - Rising Edge Clocking / Falling Edge Latching



TSCKP=1, RSCKP=1

Figure 16-9. External Gated Mode Timing - Falling Edge clocking / Rising Edge Latching

NOTE

- The bit clock ports must be kept free of timing glitches. If a single glitch occurs, all ensuing transfers will be out of synchronization.
- In case of External Gated Mode, even though the Tx Data line is put in the high-impedance state at the last non-active edge of the bit clock, the round trip delay should be sufficient to take care of hold time requirements at the external receiver.

16.3.3 I²S Mode

The SSI is compliant to I²S bus specification from Philips Semiconductors (February 1986, Revised June 5, 1996). See [Figure 16-10](#) on for an illustration of the basic I²S protocol timing.

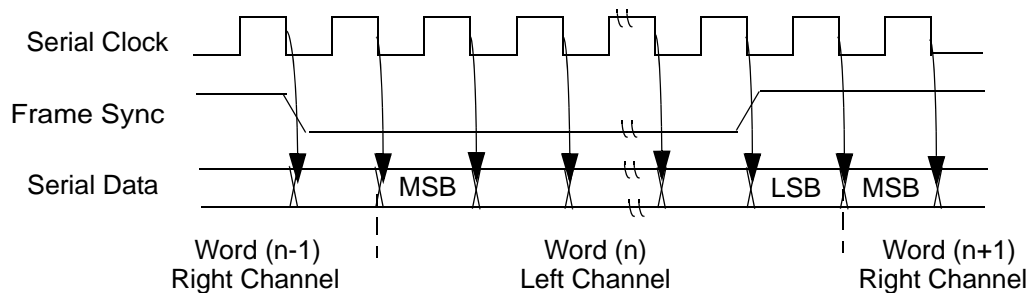


Figure 16-10. I²S Mode Timing - Serial Clock, Frame Sync and Serial Data

Select I²S mode using the options listed in [Table 16-2](#).

Table 16-2. I²S Mode Selection

I2S_MODE[1]	I2S_MODE[0]	Mode Type
0	0	Normal mode
0	1	I ² S master mode
1	0	I ² S slave mode
1	1	Normal mode

In normal mode operation, no register bits are forced to any particular state internally and the user can program the SSI to work in any operating condition.

When I²S modes are entered (I²S master (01) or I²S slave (10)), the following settings are recommended:

- Sync mode (SCR[4]=1)
- Tx shift direction: MSB transmitted first (STCR[4]=0)
- Rx shift direction: MSB received first (SRCR[4]=0)
- Tx data clocked at falling edge of the clock (STCR[3]=1)
- Rx data latched at rising edge of the clock (SRCR[3]=1)
- Tx frame sync active low (STCR[2]=1)
- Rx frame sync active low (SRCR[2]=1)
- Tx frame sync initiated one bit before data is transmitted (STCR[0]=1)
- Rx frame sync initiated one bit before data is received (SRCR[0]=1)

In I²S master mode (SCR[6:5]=01), the following additional settings are recommended:

- TXDIR bit (STCR[5]) set to 1 to select internal generated bit clock
- TFDIR bit (STCR[6]) set to 1 to select internal generated frame sync

In I²S slave mode (SCR[6:5]=10), the following settings are internally overridden by the hardware:

- Network mode is selected (SCR[3]=1)
- Tx frame sync length set to one-word-long-frame (STCR[1]=0)
- Rx frame sync length set to one-word-long-frame (SRCR[1]=0)
- Tx shifting w.r.t. bit 0 of TXSR (STCR[9]=1)
- Rx shifting w.r.t. bit 0 of RXSR (SRCR[9]=1)

The user needs to set the following control bits to configure the bit clock and frame sync:

- PM (STCCR[7:0])
- PSR (STCCR[17])
- DIV2 (STCCR[18])
- WL (STCCR[16:13])
- DC (STCCR[12:8])

The word length is fixed to 32 in I²S Master mode and the WL bits determine the number of bits that will contain valid data (out of the 32 transmitted/received bits in each channel). The fixing of word duration as 32 simplifies the relation between oversampling clock (ccm_ssi_clk) and Frame Sync (ccm_ssi_clk becomes an integer multiple of Frame Sync).

In I²S slave mode(SCR[6:5]=10), the following additional settings are recommended:

- TXDIR bit(STCR[5]) set to 0 to select external generated bit clock
- TFDIR bit(STCR[6]) set to 0 to select external generated frame sync

In I²S slave mode(SCR[6:5]=10), the following settings are done internally overridden by the hardware :

- Normal mode is selected (SCR[3]=0)
- Tx frame sync length set to one-bit-long-frame (STCR[1]=1)
- Rx frame sync length set to one-bit-long-frame (SRCR[1]=1)
- Tx shifting w.r.t. bit 0 of TXSR (STCR[9]=1)
- Rx shifting w.r.t. bit 0 of RXSR (SRCR[9]=1)

The user needs to set the following control bits to configure the data transmission:

- WL (STCCR[16:13])
- DC (STCCR[12:8])

The word length is variable in I²S slave mode and the WL bits determine the number of bits that will contain valid data. The actual word length is determined by the external CODEC. The external I²S Master still sends frame sync according to the I²S protocol (early, word wide and active low), the SSI internally operates so that each frame sync transition is the start of a new frame (the WL bits determine the number of bits to be transmitted/received). After one data word has been transferred, the SSI waits for the next frame sync transition to start operation in the next time slot. Transmit (STMSK) and receive (SRMSK) mask bits should not be used in I²S Slave mode of operation.

The SACCST, SACCEN and SACCDIS registers helps in determining which transmit slots are active. This information is used to ensure that SSI does not transmit data for powered-down/inactive channels.

16.3.4 External Frame and Clock Operation

When applying external frame sync and clock signals to SSI, there should be at least 4-bit clock cycles between the enabling of the transmit or receive section and the rising edge of the corresponding frame sync signal. The transition of STFS or SRFS should be synchronized with the rising edge of external clock signal, STCK or SRCK.

16.3.5 Data Alignment Formats Supported

The SSI supports three data formats in order to provide flexibility with handling data. These formats dictate how data is written to (and read from) the data registers. Therefore, data can appear in different places in STX0/1 and SRX0/1 based on the data format and the number of bits per word. Independent data formats are supported for both the transmitter and receiver (that is, the transmitter and receiver can use different data formats).

The supported data formats are:

- MSB alignment
 - Zero-extended (receive data only)
 - Sign-extended (receive data only)
- LSB alignment

With MSB alignment, the most significant byte is bits 31 through 24 of the data register if the word length is larger than or equal to 16 bits. If the word length is less than 16 bits and MSB alignment is chosen, the most significant byte is bits 15 through 8. With LSB alignment, the least significant byte is bits 7 through 0. Data alignment is controlled by the TXBIT0 bit in the STCR and the RXBIT0 bit in the SRCR. See [Table 16-3](#) for the bit assignment for all the data formats supported by the SSI.

Table 16-3. Data Alignment

Format	Bit Number																																																		
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																			
8-bit LSB Aligned																																	7	6	5	4	3	2	1	0											
8-bit MSB Aligned																																																			
10-bit LSB Aligned																																																			
10-bit MSB Aligned																																																			
12-bit LSB Aligned																																																			
12-bit MSB Aligned																																																			
16-bit LSB Aligned																																																			
16-bit MSB Aligned																																																			
18-bit LSB Aligned																																																			
18-bit MSB Aligned																																																			
20-bit LSB Aligned																																																			
20-bit MSB Aligned																																																			
22-bit LSB Aligned																																																			
22-bit MSB Aligned																																																			

Table 16-3. Data Alignment (continued)

24-bit LSB Aligned										2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0		
24-bit MSB Aligned	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0												
	3	2	1	0	9	8	7	6	5	4	3	2	1	0																							

In addition, receive data can either be zero-extended or sign-extended if LSB alignment is selected. With zero-extension, all bits above the most significant bit are 0's. This format is useful when data is stored in a pure integer format. With sign-extension, all bits above the most significant bit are equal to the most significant bit. This format is useful when data is stored in a fixed-point integer format (which implies fractional values). Receive data extension is controlled by the RXEXT bit in the SRCR. Transmit data used with LSB alignment has no concept of sign/zero-extension. Unused bits above the most significant bit are simply ignored.

When configured in I²S or AC97 mode, the SSI forces the selection of LSB alignment. However, RXEXT still permits a choice between zero-extension and sign-extension.

See [Section 16.7.3.10, “SSI Transmit Configuration Register \(STCR\)”](#) and [Section 16.7.3.11, “SSI Receive Configuration Register \(SRCR\)”](#) for more details on the relevant bits in the STCR and SRCR registers.

16.4 External Signal Description

The SSI has no external signals because its serial I/O signals are connected to the ADUMUX. See [Chapter 17, “Analog to Digital Converter Module \(ADC\)”](#) for information on the external audio signals.

16.5 Functional Description

16.5.1 Architecture

The Synchronous Serial Interface (SSI) is connected to chip pads through the Digital Audio Mux (AUDMUX) module. The AUDMUX can be configured to connect the SSI module to the chip pads in various ways. See [Figure 16-1](#) for a block diagram of the SSI.

16.5.2 Clocking

The SSI uses the following clocks:

- Bit clock — Used to serially clock the data bits in and out of the SSI port. This clock is either generated internally (from `ccm_ssi_clk`) or taken from external clock source (through the Tx/Rx clock ports).
- Word clock — Used to count the number of data bits per word (8, 10, 12, 16, 18, 20, 22 or 24 bits). This clock is generated internally from the bit clock.
- Frame clock (Frame Sync) — Used to count the number of words in a frame. This signal can be generated internally from the bit clock, or taken from external source (from the Tx/Rx frame sync ports).

- Sys clock —In master mode, this is an integer multiple of frame clock. This is `ccm_ssi_clk`. It is used in cases when SSI has to provide the clock.

NOTE

Ensure that the bit clock frequency (either internally generated by dividing the `ccm_ssi_clk` or sourced from external device through Tx/Rx clock ports) is never greater than 1/5 of the `ipg_clk` frequency.

In Normal mode (`SCR[6:5]=00`), the bit clock, used to serially clock the data, is visible on the Serial Transmit Clock (STCK) and Serial Receive Clock (SRCK) ports. The word clock is an internal clock used to determine when transmission of an 8, 10, 12, 16, 18, 20, 22 or 24 bit word has completed. The word clock in turn then clocks the frame clock, which counts the number of words in the frame. The frame clock can be viewed on the STFS and SRFS frame sync ports, because a frame sync is generated after the correct number of words in the frame have passed. In master and synchronous mode, the unused port SRCK is used as Serial System Clock (SYS_CLK) enabled by the SCR register bit 15, `SYS_CLK_EN`. This Serial System Clock is an oversampling clock of the frame sync clock (STFS). In this mode, the word length (WL), Prescaler Range (PSR), Prescaler Modulus (PM) and Frame rate (DC) selects the ratio of `SYS_CLK` to sampling clock STFS. In case of I²S mode, the oversampling clock `ccm_ssi_clk` can be made available on this port if the `SYS_CLK_EN` bit is set. The relationship between the clocks and the dividers is shown in Figure 16-11 (“SSI Clocking”).

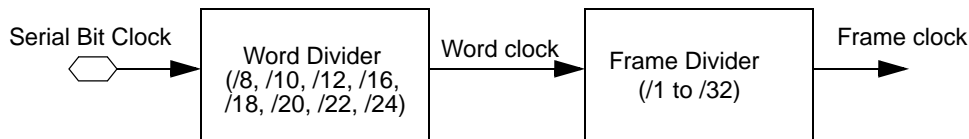


Figure 16-11. SSI Clocking

16.5.2.1 SSI Clock and Frame Sync Generation

Data clock and frame sync signals can be generated internally, or can be obtained from external sources. If internally generated, the SSI clock generator is used to derive bit clock and frame sync signals from the `ccm_ssi_clk` clock. The SSI clock generator consists of a selectable, fixed prescaler and a programmable prescaler for bit rate clock generation. In Gated Clock mode, the data clock is valid only when data is being transmitted. Otherwise the clock port is pulled to the inactive state. A programmable frame rate divider and a word length divider are used for frame rate sync signal generation.

Figure 16-12 shows a block diagram of the clock generator for the transmit section. The serial bit clock can be internal or external, depending on the Transmit Direction (TXDIR) bit in the SSI Transmit Configuration Register (STCR). The receive section contains an equivalent clock generator circuit.

The bit clock can be received from an SSI clock port or can be generated from the `ccm_ssi_clk` through a divider, as shown in Figure 16-12.

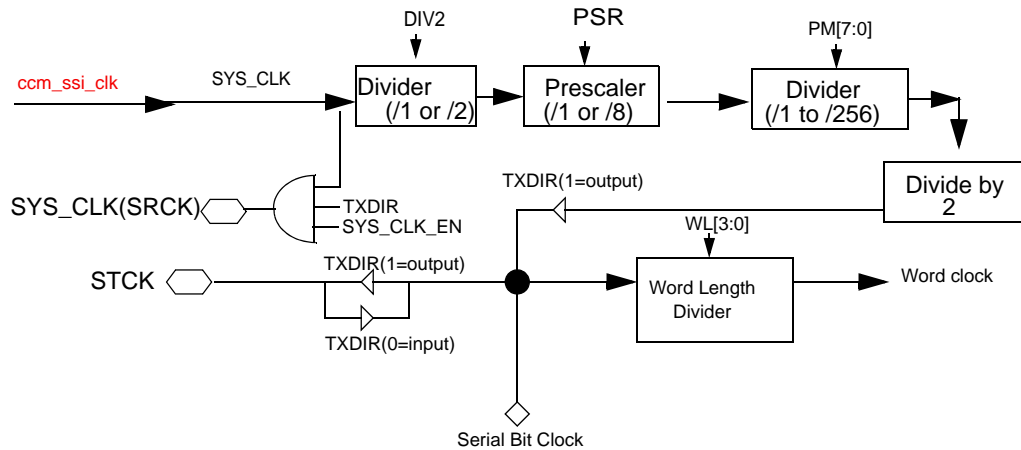


Figure 16-12. SSI Transmit Clock Generator Block Diagram

See Figure 16-13 shows the Frame Sync Generator block for the transmit section. When internally generated, both receive and transmit frame sync are generated from the word clock and are defined by the Frame Rate Divider (DC[4:0]) bits and the Word Length (WL[3:0]) bits of the SSI Transmit Clock Control Register (STCCR). The receive section contains an equivalent circuit for the Frame Sync Generator.

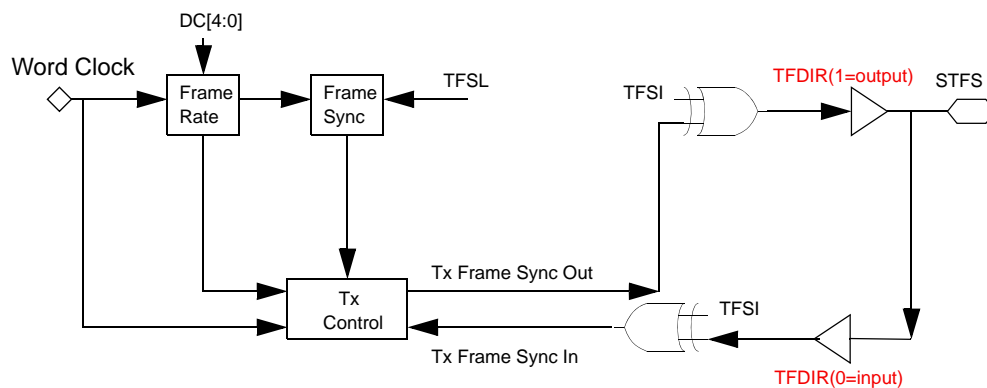


Figure 16-13. SSI Transmit Frame Sync Generator Block Diagram

16.5.2.2 DIV2, PSR and PM Bit Description

The bit clock frequency can be calculated from the SSI Serial System Clock (ccm_ssi_clk) using the equation in Figure 16-14.

NOTE

Users must ensure that the bit-clock frequency must be 5 times the ipg_clk frequency. The oversampling clock frequency can go up to ipg_clk frequency. Bits DIV2, PSR and PM should not be all set to zero at the same time.

$$f_{\text{INT_BIT_CLK}} = f_{\text{SYS_CLK}} / [(DIV2 + 1) \times (7 \times PSR + 1) \times (PM + 1) \times 2]$$

where $PM = PM[7:0]$

$$f_{\text{FRAME_SYN_CLK}} = (f_{\text{INT_BIT_CLK}}) / [(DC + 1) \times WL]$$

where $DC = DC[4:0]$ and $WL = 8, 10, 12, 16, 18, 20, 22, 24$

Figure 16-14. SSI Bit Clock Equation

For example, if the SSI working clock SYS_CLK (ccm_ssi_clk) is 12.288, in 8-bit word Normal mode with $DC[4:0]$ set to 1 (00001), $PM[7:0]$ set to 47 (0010 1111), the PSR bit cleared, $DIV2$ bit set to 1, a bit clock rate of $12.288 \text{ Mhz} / [1 \times 4 \times 48] = 64 \text{ kHz}$ is generated. Since the 8-bit word rate is equal to one (i.e. normal mode), the sampling rate (FS rate) would then be $64 \text{ kHz} / [1 * 8] = 8 \text{ kHz}$.

In next example, the SYS_CLK (ccm_ssi_clk) clock is 11.2896 Mhz. A 16-bit word Network mode with $DC[4:0]$ set to 1 (00001), $PM[7:0]$ set to 3 (0000 0011), the PSR bit is set to 0, $DIV2$ bit set to 0, and a 11.2896 MHz SYS_CLK clock, a bit clock rate of $11.2896 \text{ Mhz} / [1 \times 2 \times 4] = 1.4112 \text{ MHz}$ is generated. Since the 16-bit word rate is equal to two, the sampling rate (FS rate) would be $1.4112 \text{ MHz} / [2 * 16] = 44.1 \text{ kHz}$.

Table 16-4 shows programming examples for the clock dividers in the CRM and the SSI to support various bit clock (STCK) frequencies.

Table 16-4. SSI Bit Clock and Frame Rate as a Function of PSR, PM, and DIV2

Bits/ Word	Words / Frame	Ideal Frame Rate (kHz)	PLL Freq (Mhz)	SSDIV (in CRM)	MCLK/ ccm_ssi_clk Freq (Mhz)	DIV 2	PS R	PM	WL	DC	Actual Bit_Clk Freq (kHz) STCK	Target Bit_Clk Freq (kHz) STCK	Error (Hz)
16	1	8	294.912	48	12.288	0	0	47	7	0	128	128	0
16	2	8	294.912	48	12.288	0	0	23	7	1	256	256	0
16	4	8	294.912	48	12.288	0	0	11	7	3	512	512	0
16	1	12	294.912	48	12.288	0	0	31	7	0	192	192	0
16	2	12	294.912	48	12.288	0	0	15	7	1	384	384	0
16	4	12	294.912	48	12.288	0	0	7	7	3	768	768	0
16	1	16	294.912	48	12.288	0	0	23	7	0	256	256	0
16	2	16	294.912	48	12.288	0	0	11	7	1	512	512	0
16	4	16	294.912	48	12.288	0	0	5	7	3	1024	1024	0
16	1	24	294.912	48	12.288	0	0	15	7	0	384	384	0
16	2	24	294.912	48	12.288	0	0	7	7	1	768	768	0
16	4	24	294.912	48	12.288	0	0	3	7	3	1536	1536	0
16	1	32	294.912	48	12.288	0	0	11	7	0	512	512	0
16	2	32	294.912	48	12.288	0	0	5	7	1	1024	1024	0
16	4	32	294.912	48	12.288	0	0	2	7	3	2048	2048	0
16	1	48	294.912	48	12.288	0	0	15	7	0	768	768	0
16	2	48	294.912	48	12.288	0	0	3	7	1	1536	1536	0
16	4	48	294.912	48	12.288	0	0	1	7	3	3072	3072	0
16	1	11.025	270.9504	48	11.2896	0	0	31	7	0	176.4	176.4	0

Table 16-4. SSI Bit Clock and Frame Rate as a Function of PSR, PM, and DIV2

Bits/ Word	Words / Frame	Ideal Frame Rate (kHz)	PLL Freq (Mhz)	SSIDIV (in CRM)	MCLK/ ccm_ssi_clk Freq (Mhz)	DIV 2	PS R	PM	WL	DC	Actual Bit_Clk Freq (kHz) STCK	Target Bit_Clk Freq (kHz) STCK	Error (Hz)
16	2	11.025	270.9504	48	11.2896	0	0	15	7	1	352.8	352.8	0
16	4	11.025	270.9504	48	11.2896	0	0	7	7	3	705.6	705.6	0
16	1	22.05	270.9504	48	11.2896	0	0	15	7	0	352.8	352.8	0
16	2	22.05	270.9504	48	11.2896	0	0	7	7	1	705.6	705.6	0
16	4	22.05	270.9504	48	11.2896	0	0	3	7	3	1411.2	1411.2	0
16	1	44.1	270.9504	48	11.2896	0	0	7	7	0	705.6	705.6	0
16	2	44.1	270.9504	48	11.2896	0	0	3	7	1	1411.2	1411.2	0
16	4	44.1	270.9504	48	11.2896	0	0	1	7	3	2822.4	2822.4	0

NOTE

Table 16-4 describes how various frame rates can be achieved with the PLL0/1 supplying a frequency of 294.912 MHz and 270.9504 MHz (with WL and DC settings as shown). Using PLL2 requires that these input frequencies be lowered by a factor of 2 (and the dividers be changed accordingly).

Using the MRCG allows the input frequency to be lowered by a factor of 4 (provided the dividers are changed accordingly). These clocks are recommended as convenient starting points but the system allows for other input clock frequencies as well.

Table 16-5 below shows programming of the CRM and SSI dividers in order to generate the appropriate SYS_CLK and BIT_CLK frequencies for various sampling rates. In these examples, the master mode is selected either by setting I²S master bit (SCR[6:5]=01) or individually programming the SSI in network, synchronous, transmit internal mode (the table specifically illustrates the I²S mode frequencies/sample rates). The SYS_CLK clock is ccm_ssi_clk.

Note that the I²S master mode requires that a word length of 32 bits be used (regardless of the actual data type). Consequently, the fixed I²S frame rate of 64 bits per frame (word length (WL) can be any value) and DC of 1 are assumed.

Table 16-5. SSI Sys Clock, Bit Clock, Frame Clock in Master Mode

Bits / Word	Words / Frame	Ideal Sampling Rate (kHz)	Over Sampling Rate	PLL Freq (Mhz)	SSIDIV (in CRM)	Target MCLK Freq (Mhz)	Actual MCLK Freq (Mhz)	Bits / Word	Words/ Frame	Actual Sampling Rate (kHz) STFS	Error (Hz)
32	2	22.05	512	270.9504	48	11.2896	11.2896	32	2	44.117	17.65
32	2	24	512	294.912	48	12.288	12.288	32	2	22.058	8.82
32	2	32	384	294.912	48	12.288	12.288	32	2	11.029	4.41
32	2	32	512	294.912	36	16.384	16.384	32	2	48.000	0

NOTE

Table 16-5 describes how various frame rates can be achieved with the PLL0/1 supplying a frequency of 294.912 MHz and 270.9504 MHz. Using PLL2 requires that these input frequencies be lowered by a factor of 2 (and the dividers be changed accordingly). These clocks are recommended as convenient starting points but the system allows for other input clock frequencies as well.

16.5.3 Receive Interrupt Enable Bit Description

When the RIE and RE bit are set, the processor is interrupted when either of the SSI Receive FIFO Full (RFF0/1) bits in SISR is set (if the corresponding Receive FIFO is enabled). If the Receive FIFO is not enabled, the interrupt is generated when the corresponding SSI Receive Data Ready (RDR0/1) bit in the SISR is set. When the receive FIFO is enabled, a maximum of eight values are available to be read (8 values per channel in Two-Channel mode). If not enabled, then one value can be read from the SRX register (one each in case of Two-Channel mode). If the RIE bit is cleared, these interrupts are disabled. However, the RFF0/1 and RDR0/1 bits still indicate the receive data register full condition. Reading the SRX registers clears the RDR bits, thus clearing the pending interrupt. Two receive data interrupts (two per channel in case of Two-Channel mode) are available: receive data with exception status and receive data without exception. Table 16-6 and Table 16-7 show the conditions under which these interrupts are generated.

Table 16-6. SSI Receive Data 1 Interrupts

Interrupt	RIE	ROE0	RFF0/RDR0
Receive Data 1 (with Exception Status)	1	1	1
Receive Data 1 (without exception)	1	0	1

Table 16-7. SSI Receive Data 0 Interrupts

Interrupt	RIE	ROE1	RFF1/RDR1
Receive Data 0 (with Exception Status)	1	1	1
Receive Data 0 (without exception)	1	0	1

16.5.4 Transmit Interrupt Enable Bit Description

The SSI Transmit Interrupt Enable (TIE) control bit determines whether the processor is interrupted when the SSI transmitter needs to be serviced. When the TIE and TE bits are set, the program controller is interrupted when either of the SSI Transmit FIFO Empty (TFE0/1) flags in SISR are set (if corresponding Transmit FIFO is enabled). If the corresponding Transmit FIFO is not enabled, an interrupt is generated when the corresponding SSI Transmit Data Register Empty (TDE0/1) flag in the SISR is set and Transmit Enable (TE) bit is set.

When Transmit FIFO 0 is enabled, a maximum of eight values can be written to the SSI (8 per channel in case of Two-Channel mode, using Tx FIFO 1). If not enabled, then one value can be written to the STX0

register (one per channel in case of Two-Channel mode using STX1). When the TIE bit is cleared, all transmit interrupts are disabled. However, the TDE0/1 bits always indicate the corresponding STX register empty condition, even when the transmitter is disabled by the Transmit Enable (TE) bit (in the SCR). Writing data to the STX clears the corresponding TDE bit, thus clearing the interrupt. Two transmit data interrupts are available (four in case of Two-Channel mode, two per channel): transmit data with exception status and transmit data without exceptions. Table 16-8 and Table 16-9 show the conditions under which these interrupts are generated.

Table 16-8. SSI Transmit Data 1 Interrupts

Interrupt	TIE	TUE1	TFE1/TDE1
Transmit Data 1 (with Exception Status)	1	1	1
Transmit Data 1 (without exception)	1	0	1

Table 16-9. SSI Transmit Data 0 Interrupts

Interrupt	TIE	TUE0	TFE0/TDE0
Transmit Data 0 (with Exception Status)	1	1	1
Transmit Data 0 (without exception)	1	0	1

16.5.5 Internal Frame and Clock Shutdown

During transmit/receive operation, disabling TE/RE will ensure that data transmission/reception stops after current frame ends following which TFRC/RFRC Status bits will get set to indicate the Frame Completion State. If TFR_CLK_DIS/RFR_CLK_DIS bit is set in the current or any of the previous frames, SSI will stop driving the STFS/SRFS and STCK/SRCK signals after the current frame ends.

If TFR_CLK_DIS/RFR_CLK_DIS bit is not set, SSI will continue generating STFS/SRFS and STCK/SRCK signals (in case direction is from SSI), which then can be disabled by writing '1' to TFS_CLK_DIS/RFR_CLK_DIS bit. SSI will then stop driving these signals after end of frame is reached following which TFRC/RFRC status bits will get set to indicate the Frame Completion State.

Figure 16-15 is an illustration of transmission case where TXDIR and TFDIR are both set to '1'. In this case TE is disabled with TFS_CLK_DIS bit set in current or any of the previous frames.

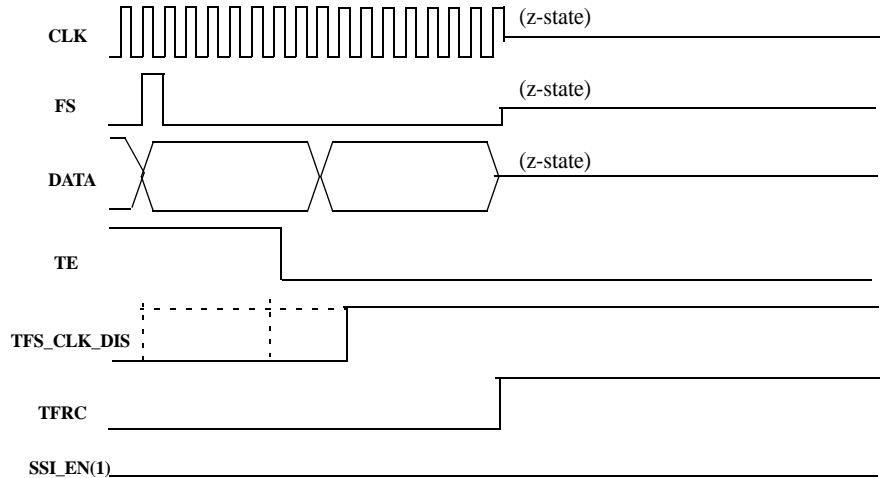


Figure 16-15. TFS_CLK_DIS assertion in current or previous frame as TE disable

Figure 16-16 is an illustration of transmission case where TXDIR and TFDIR are both set to ‘1’. In this case TFS_CLK_DIS bit is set after few frames of disabling TE. TFRC (Transmit Frame Complete) is set at frame boundary after TE is cleared. Once software services this interrupt and sets TFR_CLK_DIS bit later, TFRC bit is again set at next frame boundary.

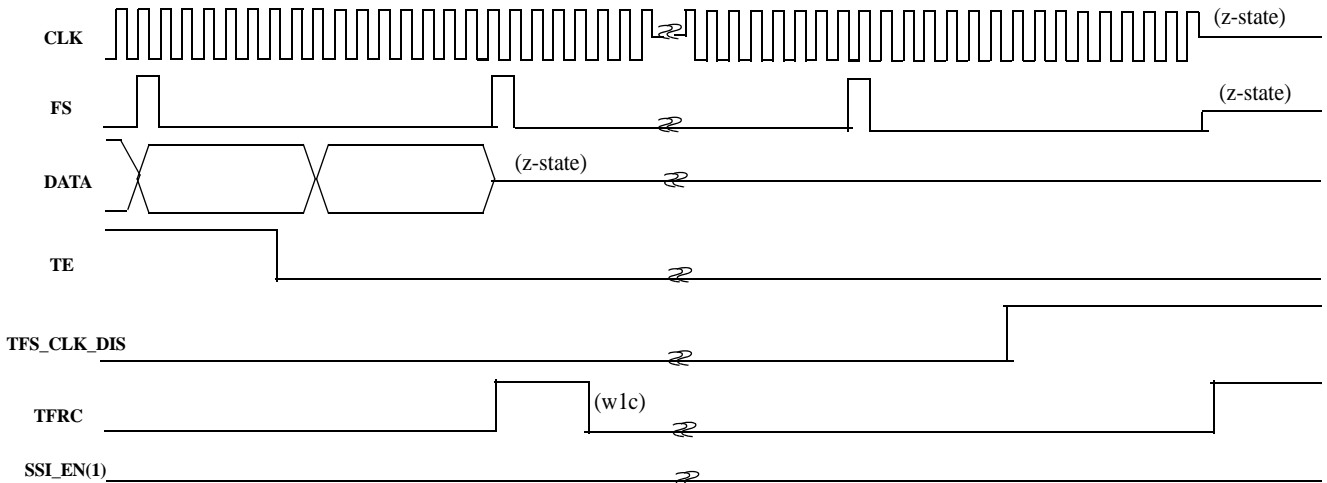


Figure 16-16. TFS_CLK_DIS assertion in subsequent frame after disabling TE

16.5.6 IP Bus Interface

The SSI has an IP Bus interface compliant with SRS 3.0.2 in order to provide a control and data interface. This interface is used by both the processor and DMA controller.

16.5.6.1 Transfer Lengths Supported

The IP Bus interface of the SSI only supports 32-bit transfers with all SSI registers other than STX0, STX1, SRX0, and SRX1 (that is, the data registers). With the exception of the data registers, using 8-bit and 16-bit transactions could result in undesired behavior but will not result in a transfer bus error. The data registers (STX0, STX1, SRX0, and SRX1) support 8-bit, 16-bit, and 32-bit transfer lengths without restrictions.

16.5.6.2 Transfer Bus Errors

Transfer bus errors are generated upon response to the following:

- Write transfer to a read-only register.
- Read or write access to a register space beyond the last populated register of the SSI in its memory map (up until the end of the allocated memory address range of the SSI).

16.5.6.3 Clock Rate

The IP Bus clock frequency must be at least four times the serial bit clock frequency.

16.6 Initialization/Application Information

The SSI is affected by the following types of reset:

- Power-on Reset—The Power-on reset is generated by asserting the RESET port. The Power-on reset clears the SSIEN bit in SCR, which disables the SSI. All other status and control bits in the SSI are affected as described in SSI Programming Model in [Section 16.3](#).
- SSI Reset—The SSI reset is generated when the SSIEN bit in the SCR is cleared. The SSI status bits are preset to the same state produced by the Power-on reset. The SSI control bits are unaffected. The control bits in the SCR are also unaffected. The SSI reset is useful for selective reset of the SSI without changing the present SSI control bits and without affecting the other peripherals.

The correct sequence to initialize the SSI is as follows:

1. Issue a Power-on or SSI reset (SCR[SSIEN]=0).
2. Disable SSI clocks (ipg_clk, ccm_ssi_clk).
3. Set all control bits for configuring the SSI (see [Table 16-10](#) for the list of “SSI Control Bits Requiring SSI to be Disabled Before Change”).
4. Enable appropriate interrupts/DMA requests through SIER.
5. Set the SCR[SSIEN] bit (=1) to enable the SSI.
6. Enable SSI clocks (ipg_clk, ccm_ssi_clk), as required.
7. In case of AC97 mode, set the SACNT[AC97EN] bit after programming the SATAG register (if needed, for AC97 Fixed mode).
8. Set SCR[TE/RE] bits.
9. To ensure proper operation of the SSI, use the “Power-on or SSI reset before changing any of the SSI Control” bits listed in [Table 16-10](#)).

NOTE

These control bits should not be changed when SSI is enabled

Table 16-10. SSI Control Bits Requiring SSI to be Disabled Before Change

Control Register	Bit
SCR	[9]=CLK_IST [8]=TCH_EN [7]=SYS_CLK_EN [6:5]=I2S_MODE [4]=SYN [3]=NET
SIER	[22]=RDMAE [20]=TDMAE
SRCCR STCCR	[9]=RXBIT0 [9]=TXBIT0 [8]=RFEN1 [8]=TFEN1 [7]=RFEN0 [7]=TFEN0 [6]=RFDIR [6]=TFDIR [5]=RXDIR [5]=TXDIR [4]=RSHFD [4]=TSHFD [3]=RSCKP [3]=TSCKP [2]=RFSI [2]=TFSI [1]=RFSL [1]=TFSL [0]=REFS [0]=TEFS
SRCCR STCCR	[16]=WL3 [15]=WL2 [14]=WL1 [13]=WL0
SACNT	[1]=FV [10:5]=FRDIV

16.7 Memory Map and Register Definition

Section 16.7.3, “Register Descriptions” provides the detailed descriptions for all of the SSI registers.

16.7.1 SSI Memory Map

Table 16-11 shows the SSI memory map.

Table 16-11. SSI Memory Map

Address	Register	Access	Reset Value	Section/Page
0xBASE_00 (STX0_)	SSI Transmit Data Register	R/W	0x0000_0000	16.7.3.1/16-32
0xBASE_08 (SRX0_)	SSI Receive Data Register	Read-only	0x0000_0000	16.7.3.4/16-35
0xBASE_10 (SCR)	SSI Control Register	R/W	0x0000_0000	16.7.3.7/16-38
0xBASE_14 (SISR)	SSI Interrupt Status Register	Read-only	0x0000_3003	16.7.3.8/16-40
0xBASE_18 (SIER)	SSI Interrupt Enable Register	R/W	0x0000_3003	16.7.3.9/16-45
0xBASE_1C (STCR)	SSI Transmit Configuration Register	R/W	0x0000_0200	16.7.3.10/16-46
0xBASE_20 (SRCCR)	SSI Receive Configuration Register	R/W	0x0000_0200	16.7.3.11/16-48
0xBASE_24 (STCCR)	SSI Transmit Clock Control Register	R/W	0x0004_0000	16.7.3.12/16-50
0xBASE_28 (SRCCR)	SSI Receive Clock Control Register	R/W	0x0004_0000	
0xBASE_2C (SFCSR)	SSI FIFO Control/Status Register	R/W	0x0081_0081	16.7.3.13/16-52
0xBASE_30 (STR)	SSI Test Register ¹	R/W	0x0000_1111	
0xBASE_34 (SOR)	SSI Option Register ²	R/W	0x0000_0000	
0xBASE_48 (STMSK)	SSI Transmit Time Slot Mask Register	R/W	0x0000_0000	16.7.3.18/16-60
0xBASE_4C (SRMSK)	SSI Receive Time Slot Mask Register	R/W	0x0000_0000	16.7.3.19/16-61

¹SSI Test Register is intended for debugging purposes only and is not visible to the end user.

²SSI Option Register intended for internal use only and is not visible to the end user.

16.7.2 Register Summary

Figure 16-17 shows the key to the register fields and Table 16-12 shows the register figure conventions.

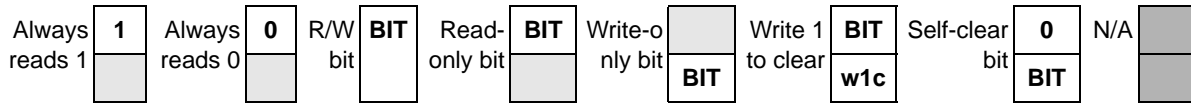


Figure 16-17. Key to Register Fields

Table 16-12. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
sfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[<i>signal_name</i>]	Reset value is determined by polarity of indicated signal.

Table 16-13 shows the SSI register summary.

Table 16-13. Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBASE_00 (STX0_)	R	STX0[31:16]															
	W																
	R	STX0[15:0]															
	W																
0xBASE_08 (SRX0_)	R	SRX0[31:16]															
	W																
	R	SRX0[15:0]															
	W																
0xBASE_10 (SCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	RFR_CLK_DIS	TFR_CLK_DIS	CLK_IST	TCH_EN	SYS_CLK_EN	I ² S MODE[1:0]		0	NET	RE	TE	SSIE N
	W																
0xBASE_14 (SISR)	R	0	0	0	0	0	0	0	RFR_C	TFRC	0	0	0	0	0	0	0
	W																
	R	0	RDR0	0	TDE0	0	ROE0	0	TUE0	TFS	RFS	TLS	RLS	0	RFF0	0	TFE0
	W																
0xBASE_18 (SIER)	R	0	0	0	0	0	0	0	RFR_C_EN	TFRC_EN	0	RIE	0	TIE	0	0	0
	W																
	R	0	RDR0_EN	0	TDE0_EN	0	ROE0_EN	0	TUE0_EN	TFS_EN	RFS_EN	TLS_EN	RLS_EN	0	RFF0_EN	0	TFE0_EN
	W																
0xBASE_1C (STCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	TXBIT0	0	TFEN0	TFDIR	TXDIR	TSHFD	TSCKP	TFSI	TFSL	TEFS
	W																
0xBASE_20 (SRCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	RXEXT	RXBIT0	RFEN1	RFEN0	RFDIR	RXDIR	RSHFD	RSCKP	RFSI	RFSL	REFS
	W																
0xBASE_24 (STCCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
	W																
	R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
	W																

Table 16-13. Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBASE_28 (SRCCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
	W																
	R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
	W																
0xBASE_2C (SFCSR)	R	RFCNT1[3:0]				TFCNT1[3:0]				RFWM1[3:0]				TFWM1[3:0]			
	W																
	R	RFCNT0[3:0]				TFCNT0[3:0]				RFWM0[3:0]				TFWM0[3:0]			
	W																
0xBASE_30 (STR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	TEST	RCK2 TCK	RFS2 RFS	RXSTATE[4:0]				TXD2 RXD	TCK2 RCK	TFS2 RFS	TXSTATE[4:0]					
	W																
0xBASE_34 (SOR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	CLKO FF	RX_C LR	TX_C LR	INIT	WAIT[1:0]		SYN RST
	W																
0xBASE_38 (SACNT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	FRDIV[5:0]				WR	RD	TIF	FV	AC97 EN		
	W																
0xBASE_3C (SACADD)	R	0	0	0	0	0	0	0	0	0	0	0	0	SACADD[18:16]			
	W																
	R	SACADD[15:0]															
	W																
0xBASE_40 (SACDAT)	R	0	0	0	0	0	0	0	0	0	0	0	0	SACDAT[19:16]			
	W																
	R	SACDAT[15:0]															
	W																
0xBASE_44 (SATAG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SATAG[15:0]															
	W																

Table 16-13. Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0xBASE_48 (STMSK)	R	STMSK[31:0]																
	W																	
	R																	
	W																	
0xBASE_4C (SRMSK)	R	SRMSK[31:0]																
	W																	
	R																	
	W																	
0xBASE_50 (SACCST)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	SACCST[9:0]										
	W																	
0xBASE_54 (SACCEN)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W							SACCEN[9:0]										
0xBASE_58 (SACCDIS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																	
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W							SACCDIS[9:0]										

16.7.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams in bit order.

16.7.3.1 SSI Transmit Data Registers 0 & 1 (STX0/1)

See [Table 16-14](#) for description of the bit fields in the register.

0xBASE_00 (STX0_)				Access: User read/write												
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	STX0[31:16]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STX0[15:0]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

See [Table 16-14](#) for description of the bit fields in the register.

0xBASE_04 (STX1_)				Access: User read/write												
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	STX1[31:16]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STX1[15:0]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-14. SSI Transmit Data Register Field Descriptions

Field	Description
31–0 STX0 STX1	SSI Transmit Data. These bits store the data to be transmitted by the SSI. These are implemented as the first word of their respective Tx FIFOs. Data written to these registers is transferred to the Transmit Shift Register (TXSR), when shifting of the previous data is complete. If both FIFOs are in use, data is alternately transferred from STX0 and STX1, to TXSR. Multiple writes to the STX registers will not result in the previous data being over-written by the subsequent data. STX1 can only be used in Two-Channel mode of operation. Protection from over-writing is present irrespective of whether the transmitter is enabled or not. Example 1: If Tx FIFO0 is in use and user writes Data1...Data9 to STX0, Data9 will not over-write Data1. Data1...Data8 are stored in the FIFO while Data9 is discarded. Example 2: If Tx FIFO0 is not in use and user writes Data1, Data2 to STX0, then Data2 will not over-write Data1 and will be discarded.

NOTE

Enable SSI (SSIEN=1) before writing to SSI Transmit Data Registers.

16.7.3.2 SSI Transmit FIFO 0 & 1 Registers

The SSI Transmit FIFO registers are 8x32-bit registers. These registers are not directly accessible by the end user (except in SSI test mode). Transmit Shift Register (TXSR) receives its values from these FIFO registers. Transmitted data is first-in-first-out. When the Transmit Interrupt Enable (TIE) bit in the SIER and either of the Transmit FIFO Empty (TFE0 or 1) bits in the SISR are set, the core is interrupted whenever the data level in either of the SSI Transmit FIFOs falls below the selected threshold.

16.7.3.3 SSI Transmit Shift Register (TXSR)

The SSI Transmit Shift Register (TXSR) is a 24-bit shift register that contains the data being transmitted. This register is not directly accessible by the end user. When a continuous clock is used, data is shifted out to the Serial Transmit Data (STXD) port by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted out to the STXD port by the selected (internal/external) gated clock. The Word Length control bits (WL[3:0]) in the STCCR (described in SSI Transmit and Receive Clock Control Registers) determine the number of bits to be shifted out of the TXSR before it is considered empty and can be written to again. This word length can be 8, 10, 12, 16, 18, 20, 22 or 24 bits. The data to be transmitted occupies the most significant portion of the shift register if TXBIT0 is '0', otherwise it occupies the least significant portion. The unused portion of the register is ignored. Data is always shifted out of this register with the Most Significant Bit (MSB) first when the SHFD bit of the STCR is cleared. If this bit is set, the Least Significant Bit (LSB) is shifted out first. [Figure 16-20](#) through [Figure 16-21](#) show the transmitter loading and shifting operation. The figures show the working for some WL values, the same can be extended for other values.

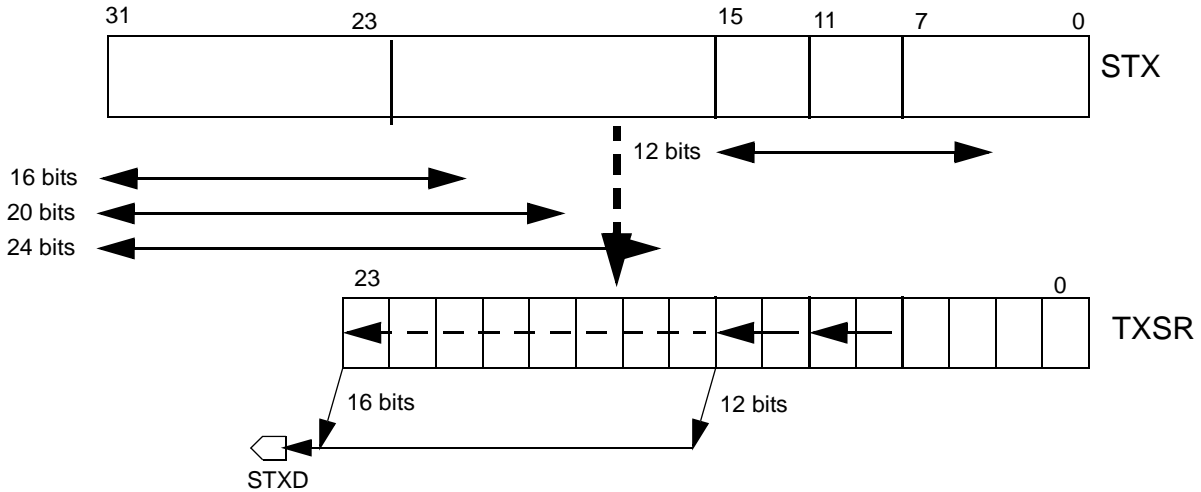


Figure 16-18. Transmit Data Path (TXBIT0=0, TSHFD=0) (MSB Alignment)

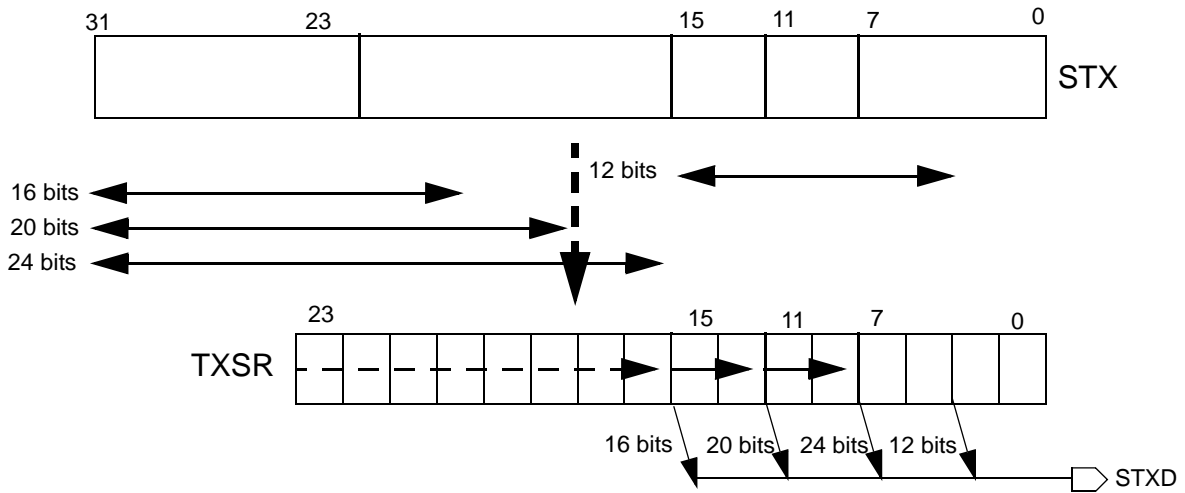


Figure 16-19. Transmit Data Path (TXBIT0=0, TSHFD=1) (MSB Alignment)

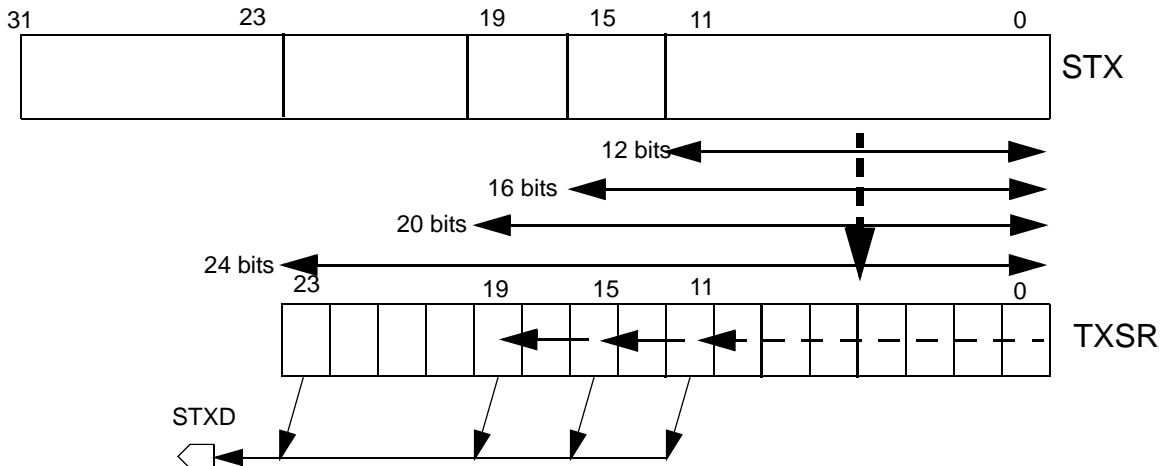


Figure 16-20. Transmit Data Path (TXBIT0=1, TSHFD=0) (LSB Alignment)

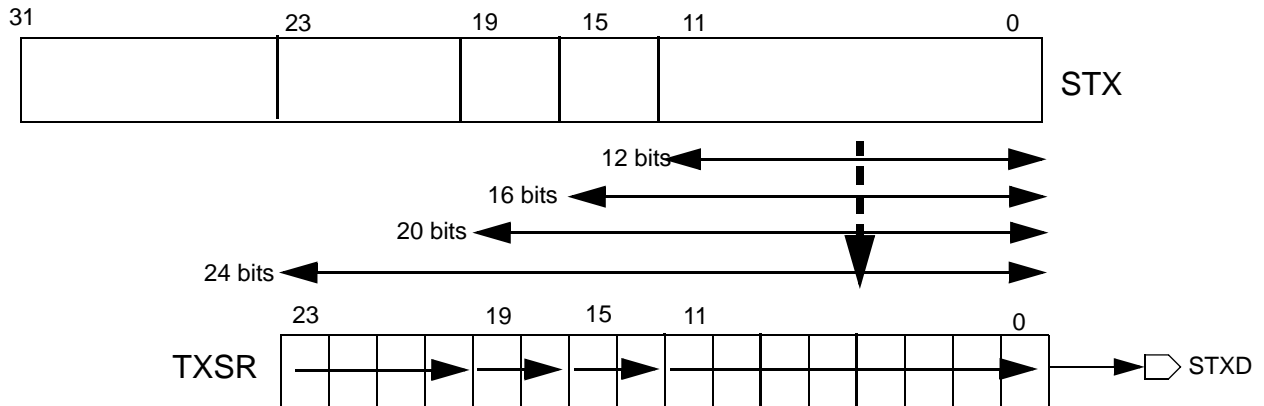


Figure 16-21. Transmit Data Path (TXBIT0=1, TSHFD=1) (LSB Alignment)

16.7.3.4 SSI Receive Data Registers 0 & 1 (SRX0/1)

See [Table 16-15](#) for description of the bit fields in the register.

0xBASE_08 (SRX0_)		Access: User read-only														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRX0[31:16]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRX0[15:0]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

See [Table 16-15](#) for description of the bit fields in the register.

0xBASE_0C (SRX1_)		Access: User read-only															
0xBASE_		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRX1[31:16]																
W																	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	SRX1[15:0]																
W																	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 16-15. SSI_1 Receive Data Register Field Descriptions

Field	Description
31–0 SRX0 SRX1	SSI Receive Data. These bits store the data received by the SSI. These are implemented as the first word of their respective Rx FIFOs. These bits receive data from the RXSR depending on the mode of operation. In case both FIFOs are in use, data is transferred to each data register alternately. SRX1 can only be used in Two-Channel mode of operation.

16.7.3.5 SSI Receive FIFO 0 & 1 Registers

The SSI Receive FIFO Registers are 8x32-bit registers. These registers are not directly accessible by the end user (except in SSI test mode). They always accept data from the Receive Shift Register (RXSR). The core is interrupted when the data level in either of the SSI Receive FIFOs reaches the selected threshold, if the associated interrupt is enabled.

16.7.3.6 SSI Receive Shift Register (RXSR)

The SSI Receive Shift Register (RXSR) is a 24-bit, shift register that receives incoming data from the serial receive data SRXD port. This register is not directly accessible by the end user. When a continuous clock is used, data is shifted in by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted in by the selected (internal/external) gated clock. Data is assumed to be received MSB first if the SHFD bit of the SRCR is cleared. If this bit is set, the data is received LSB first. Data is transferred to the appropriate SSI Receive Data Register (SRX0/1) or Receive FIFOs (if the receive FIFO is enabled and the corresponding SRX is full) after 8, 10, 12, 16, 18, 20, 22 or 24 bits have been shifted in depending on the WL[3:0] control bits. For receiving less than 24 bits of data, LSB bits are appended with zero. The following figures show the receiver loading and shifting operation. The figures show the working for some WL values, the same can be extended for other values.

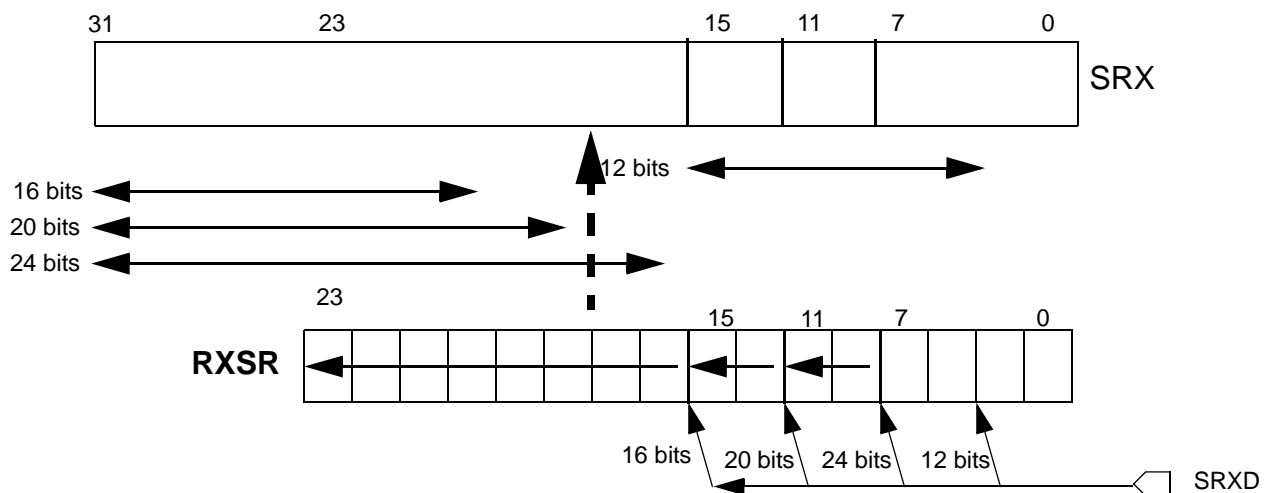


Figure 16-22. Receive Data Path (RXBIT0=0, RSHFD=0) (MSB Alignment)

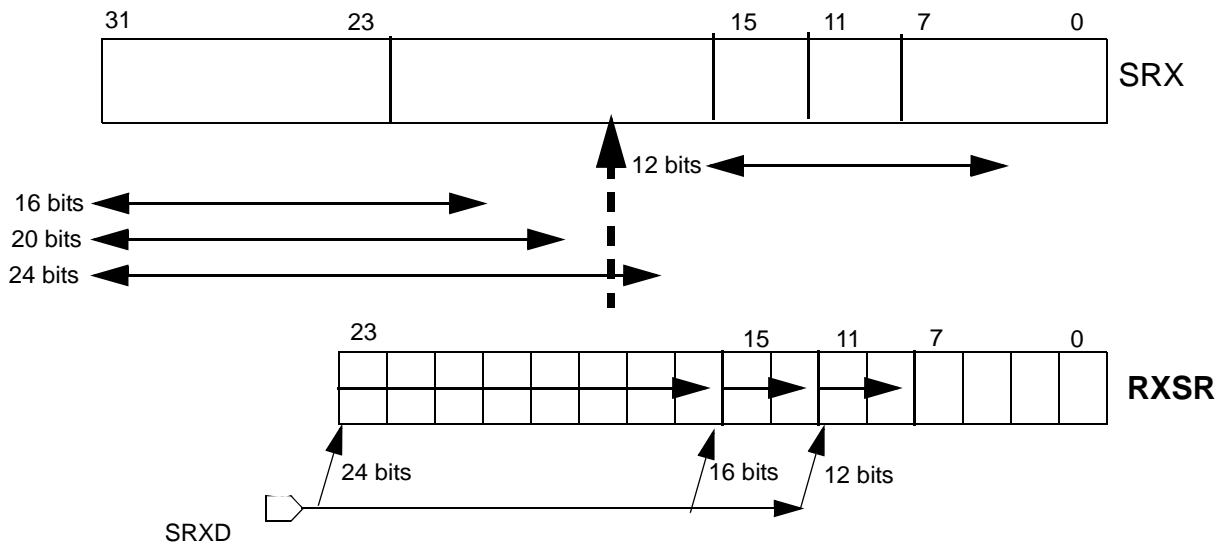


Figure 16-23. Receive Data Path (RXBIT0=0, RSHFD=1) (MSB Alignment)

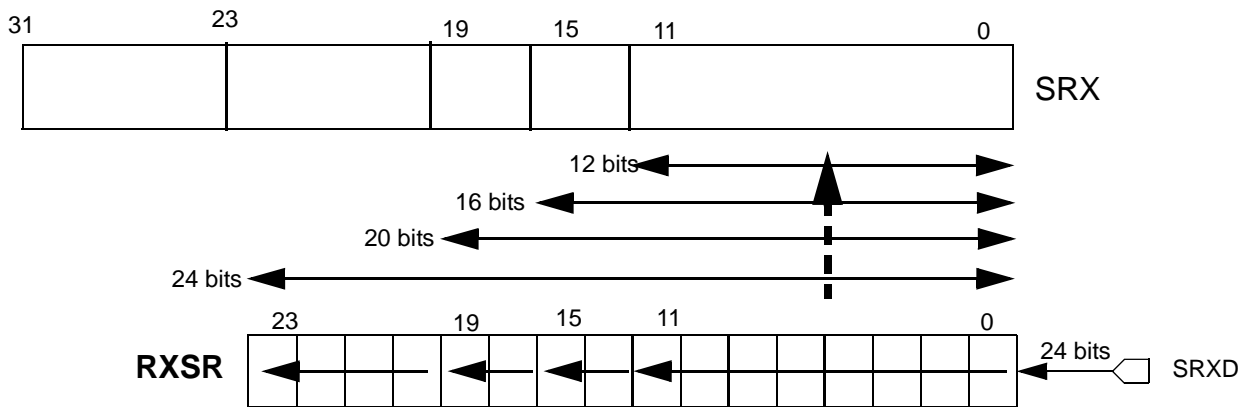


Figure 16-24. Receive Data Path (RXBIT0=1, RSHFD=0) (LSB Alignment)

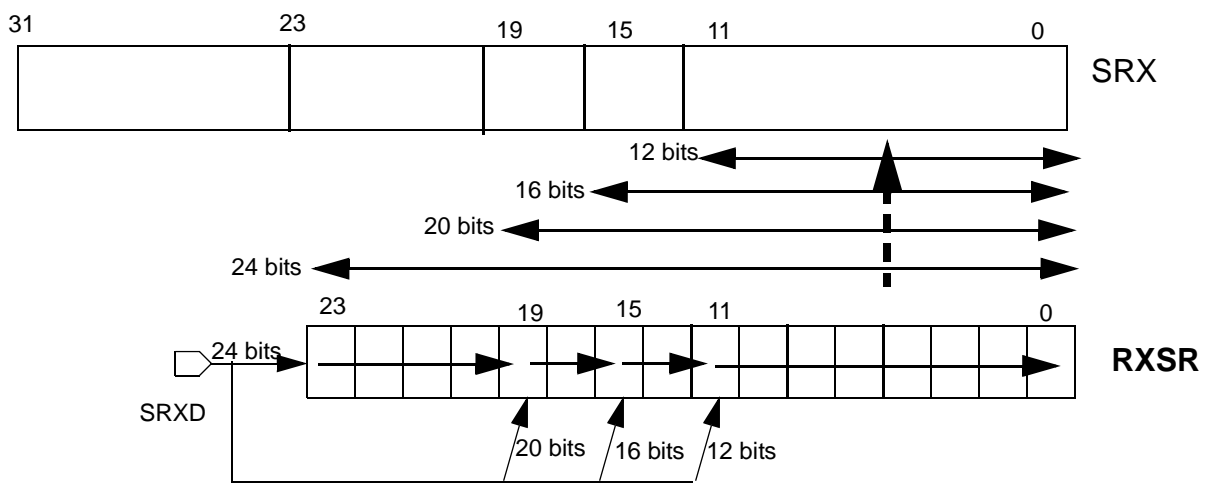


Figure 16-25. Receive Data Path (RXBIT0=1, RSHFD=1) (LSB Alignment)

16.7.3.7 SSI Control Register (SCR)

The SSI Control Register (SCR) is a 10-bit register used to set up the SSI. SSI reset is controlled by bit 0 in the SCR. SSI operating modes are also selected in this register (except AC97 mode which is selected in SACNT register).

See [Table 16-16](#) for description of the bit fields in the register.

0xBASE_10 (SCR)																Access: User read/write			
0xBASE_																			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
W																			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RFR_	TFR_	CLK_I	TCH_	SYS_	I ² S		SYN	NET	RE	TE	SSIE
W					CLK_	CLK_	ST	EN	CLK_E	MODE[1:0]						N
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-16. SSI Control Register Field Descriptions

Field	Description
31–10	Reserved
11 RFR_CLK_DIS	<p>Receive Frame Clock Disable.</p> <p>This bit provide option to keep the Frame-sync and Clock enabled or disabled after current receive frame, in which receiver is disabled by clearing RE bit. Writing to this bit has effect only when RE is disabled.</p> <p>0 Continue Frame-sync/Clock generation after current frame during which RE is cleared. This may be required when Fram-sync and Clocks are required from SSI , even when no data is to be received.</p> <p>1 Stop Frame-sync/Clock generation at next frame boundary. This will be effective also in case where receiver is already disabled in current or previous frames.</p>
10 TFR_CLK_DIS	<p>Transmit Frame Clock Disable.</p> <p>This bit provide option to keep the Frame-sync and Clock enabled or disabled after current transmit frame, in which transmitter is disabled by clearing TE bit. Writing to this bit has effect only when TE is disabled.</p> <p>0 Continue Frame-sync/Clock generation after current frame during which TE is cleared. This may be required when Fram-sync and Clocks are required from SSI , even when no data is to be received.</p> <p>1 Stop Frame-sync/Clock generation at next frame boundary. This will be effective also in case where transmitter is already disabled in current or previous frames.</p>
9 CLK_IST	<p>Clock Idle State. This bit controls the idle state of the transmit clock port during SSI internal gated mode.</p> <p>0 Clock idle state is '0'.</p> <p>1 Clock idle state is '1'.</p>

Table 16-16. SSI Control Register Field Descriptions (continued)

Field	Description
8 TCH_EN	Two-Channel Operation Enable. This bit allows SSI to operate in the two-channel mode. In this mode, 2 time slots should be used out of the possible 32. Any 2 time slots (from 0 to 31) can be selected. The data in the two time slots is alternately handled by the two data registers (0 and 1). While receiving, the RXSR transfers data to SRX0 and SRX1 alternately and while transmitting, data is alternately transferred from STX0 and STX1 to TXSR. If more than 2 slots are to be enabled, then for odd number of slots Two-Channel Operation is deprecated and TCH_EN should be cleared. This will ensure that all data is received and transmitted from one fifo, FIFO0. For an even number of slots, Two-Channel Operation can be enabled to optimize usage of both FIFOs or disabled as in the case of odd number of active slots. 0 Two-channel mode disabled. 1 Two-channel mode enabled.
7 SYS_CLK_EN	System Clock Enable. When set, this bit allows the SSI to output the SYS_CLK (ccm_ssi_clk) at the SRCK port, provided that network mode, synchronous mode, and transmit internal clock mode are set. The relationship between bit clock and SYS_CLK is determined by DIV2, PSR, and PM bits. This bit can be used to output the oversampling clock on SRCK port in I ² S Master mode. 0 SYS_CLK not output on SRCK port. 1 SYS_CLK output on SRCK port.
6–5 I ² S MODE[1:0]	I ² S Mode Select. These bits allow the SSI to operate in Normal, I ² S Master or I ² S Slave mode. See Section 16.3.3 for a detailed description of I ² S Mode of operation. See Table 16-2 (“Mode Selection”) for details regarding settings.
4 SYN	Synchronous Mode. This bit controls whether SSI is in synchronous mode or not. In synchronous mode, the transmit and receive sections of SSI share a common clock port (STCK) and frame sync port (STFS). 0 Asynchronous mode selected. 1 Synchronous mode selected.
3 NET	Network Mode. This bit controls whether SSI is in network mode or not. 0 Network mode not selected. 1 Network mode selected.
2 RE	Receive Enable. This control bit enables the receive section of the SSI. When this bit is enabled, data reception starts with the arrival of the next frame sync. If data is being received when this bit is cleared, data reception continues until the end of the current frame and then stops. If this bit is set again before the second to last bit of the last time slot in the current frame, then reception continues without interruption. 0 Receive section disabled. 1 Receive section enabled.

Table 16-16. SSI Control Register Field Descriptions (continued)

Field	Description
1 TE	<p>Transmit Enable. This control bit enables the transmit section of the SSI. It enables the transfer of the contents of the STX registers to the TXSR and also enables the internal transmit clock. The transmit section is enabled when this bit is set and a frame boundary is detected. When this bit is cleared, the transmitter continues to send data until the end of the current frame and then stops. Data can be written to the STX registers with the TE bit cleared (the corresponding TDE bit will be cleared). If the TE bit is cleared and then set again before the second to last bit of the last time slot in the current frame, data transmission continues without interruption. The normal transmit enable sequence is to write data to the STX register(s) and then set the TE bit. The normal disable sequence is to clear the TE and TIE bits after the TDE bit is set.</p> <p>In gated clock mode, clearing the TE bit results in the clock stopping after the data currently in TXSR has shifted out. When the TE bit is set, the clock starts immediately (for internal gated clock mode).</p> <p>0 Transmit section disabled. 1 Transmit section enabled.</p>
0 SSIEN	<p>SSIEN — SSI Enable</p> <p>This bit is used to enable/disable the SSI. When disabled, all SSI status bits are preset to the same state produced by the power-on reset, all control bits are unaffected, the contents of Tx and Rx FIFOs are cleared. When SSI is disabled, all internal clocks are disabled (except register access clock).</p> <p>0 SSI is disabled. 1 SSI is enabled.</p>

16.7.3.8 SSI Interrupt Status Register (SISR)

The SSI Interrupt Status Register (SISR) is used to monitor the SSI. This register is used by the core to interrogate the status of the SSI. The status bits are described in the following table. See [Figure 16-26](#) for a list of SSI interrupt sources. All Receiver related interrupts are generated only if the Receiver is enabled (SCR[2]=1) and all Transmitter related interrupts are generated only if the Transmitter is enabled (SCR[1]=1).

SSI Interrupt Sources

- SSI Receive Data with Exception Status 0/1
- SSI Receive Data 0/1
- SSI Receive Last Time Slot
- SSI Transmit Data with Exception Status 0/1
- SSI Transmit Data 0/1
- SSI Transmit Last Time Slot
- SSI AC97 Command Address Updated
- SSI AC97 Command Data Updated
- SSI AC97 Receive Tag Updated
- SSI Receive Frame Sync
- SSI Transmit Frame Sync
- SSI Receive Frame Complete
- SSI Transmit Frame Complete

Figure 16-26. SSI Interrupts

NOTE

- SSI Status flags are valid when SSI is enabled.
- See [Section 16.5.3](#) and [Section 16.5.4](#) for interrupt source mapping.
- All the flags in the SISR are updated after the first bit of the next SSI word has completed transmission or reception. Certain status bits (ROE0/1 and TUE0/1) are cleared by writing 1 to the corresponding interrupt status bit in SISR.

See [Table 16-17](#) for description of the bit fields in the register.

0xBASE_14 (SISR)																Access: User read-only			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
R	0	0	0	0	0	0	0	RFRC	TRFC	0	0	0	0	CMDA U	CMDDU	RXT			
W	█								█										
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
R	RDR1	RDR0	TDE1	TDE0	ROE1	ROE0	TUE1	TUE0	TFS	RFS	TLS	RLS	RFF1	RFF0	TFE1	TFE0			
W	█				█				█				█						
RESET	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1			

Table 16-17. SSI Interrupt Status Register Field Descriptions

Field	Description
31–19	Reserved
24 RFRC	Receive Frame Complete. This flag is set at the end of the frame during which Receiver is disabled. If Receive Frame & Clock are not disabled in the same frame, this flag is also set at the end of the frame in which Receive Frame & Clock are disabled. See description of RFR_CLK_DIS bit for more details on how to disable Receiver Frame & Clock or keep them enabled after receiver is disabled . 0 End of Frame not reached 1 End of frame reached after disabling RE or disabling RFR_CLK_DIS, when receiver is already disabled.
23 TFRC	Transmit Frame Complete. This flag is set at the end of the frame during which Transmitter is disabled. If Transmit Frame & Clock are not disabled in the same frame, this flag is also set at the end of the frame in which Transmit Frame & Clock are disabled. See description of TFR_CLK_DIS bit for more details on how to disable Receiver Frame & Clock or keep them enabled after receiver is disabled . 0 End of Frame not reached 1 End of frame reached after disabling TE or disabling TFR_CLK_DIS, when transmitter is already disabled.
18 CMDAU	Command Address Register Updated. This bit causes the Command Address Updated interrupt (when CMDAU_EN bit is set). This status bit is set each time there is a difference in the previous and current value of the received Command Address. This bit is cleared on reading the SACADD register. 0 No change in SACADD register. 1 SACADD register updated with different value.

Table 16-17. SSI Interrupt Status Register Field Descriptions (continued)

Field	Description
17 CMDDU	Command Data Register Updated. This bit causes the Command Data Updated interrupt (when CMDDU_EN bit is set). This status bit is set each time there is a difference in the previous and current value of the received Command Data. This bit is cleared on reading the SACDAT register. 0 No change in SACDAT register. 1 SACDAT register updated with different value.
16 RXT	Receive Tag Updated. This status bit is set each time there is a difference in the previous and current value of the received tag. It causes the Receive Tag Interrupt (if RXT_EN bit is set). This bit is cleared on reading the SATAG register. 0 No change in SATAG register. 1 SATAG register updated with different value.
15 RDR1	Receive Data Ready 1. This flag bit is set when SRX1 or Rx FIFO 1 is loaded with a new value and Two-Channel mode is selected. RDR1 is cleared when the Core reads the SRX1 register. If Rx FIFO 1 is enabled, RDR1 is cleared when the FIFO is empty. If RIE and RDR1_EN are set, a Receive Data 1 interrupt request is issued on setting of RDR1 bit in case Rx FIFO1 is disabled, if the FIFO is enabled, the interrupt is issued on RFF1 assertion. The RDR1 bit is cleared by POR and SSI reset. 0 No new data for Core to read. 1 New data for Core to read.
14 RDR0	Receive Data Ready 0. This flag bit is set when SRX0 or Rx FIFO 0 is loaded with a new value. RDR0 is cleared when the Core reads the SRX0 register. If Rx FIFO 0 is enabled, RDR0 is cleared when the FIFO is empty. If RIE and RDR0_EN are set, a Receive Data 0 interrupt request is issued on setting of RDR0 bit in case Rx FIFO0 is disabled, if the FIFO is enabled, the interrupt is issued on RFF0 assertion. The RDR0 bit is cleared by POR and SSI reset. 0 No new data for Core to read. 1 New data for Core to read.
13 TDE1	Transmit Data Register Empty 1. This flag is set whenever data is transferred to TXSR from STX1 register and Two-Channel mode is selected. If Tx FIFO1 is enabled, this occurs when there is at least one empty slot in STX1 or Tx FIFO1. If Tx FIFO1 is not enabled, this occurs when the contents of STX1 are transferred to TXSR. The TDE1 bit is cleared when the Core writes to STX1. If TIE and TDE1_EN are set, an SSI Transmit Data 1 interrupt request is issued on setting of TDE1 bit. The TDE1 bit is cleared by POR and SSI reset. 0 Data available for transmission. 1 Data needs to be written by the Core for transmission.
12 TDE0	Transmit Data Register Empty 0. This flag is set whenever data is transferred to TXSR from STX0 register. If Tx FIFO 0 is enabled, this occurs when there is at least one empty slot in STX0 or Tx FIFO 0. If Tx FIFO 0 is not enabled, this occurs when the contents of STX0 are transferred to TXSR. The TDE0 bit is cleared when the Core writes to STX0. If TIE and TDE0_EN are set, an SSI Transmit Data 0 interrupt request is issued on setting of TDE0 bit. The TDE0 bit is cleared by POR and SSI reset. 0 Data available for transmission. 1 Data needs to be written by the Core for transmission.

Table 16-17. SSI Interrupt Status Register Field Descriptions (continued)

Field	Description
11 ROE1	Receiver Overrun Error 1. This flag is set when the RXSR is filled and ready to transfer to SRX1 register or to Rx FIFO 1 (when enabled) and these are already full and Two-Channel mode is selected. If Rx FIFO 1 is enabled, this is indicated by RFF1 flag, else this is indicated by the RDR1 flag. The RXSR is not transferred in this case. The ROE1 flag causes an interrupt if RIE and ROE1_EN are set. The ROE1 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. Clearing the RE bit does not affect the ROE1 bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
10 ROE0	Receiver Overrun Error 0. This flag is set when the RXSR is filled and ready to transfer to SRX0 register or to Rx FIFO 0 (when enabled) and these are already full. If Rx FIFO 0 is enabled, this is indicated by RFF0 flag, else this is indicated by the RDR0 flag. The RXSR is not transferred in this case. The ROE0 flag causes an interrupt if RIE and ROE0_EN are set. The ROE0 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. Clearing the RE bit does not affect the ROE0 bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
9 TUE1	Transmitter Underrun Error 1. This flag is set when the TXSR is empty (no data to be transmitted), the TDE1 flag is set, a transmit time slot occurs and the SSI is in Two-Channel mode. When a transmit underrun error occurs, the previous data is retransmitted. In Network mode, each time slot requires data transmission (unless masked through STMSK register), when the transmitter is enabled (TE is set). The TUE1 flag causes an interrupt if TIE and TUE1_EN are set. The TUE1 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
8 TUE0	Transmitter Underrun Error 0. This flag is set when the TXSR is empty (no data to be transmitted), the TDE0 flag is set and a transmit time slot occurs. When a transmit underrun error occurs, the previous data is retransmitted. In Network mode, each time slot requires data transmission (unless masked through STMSK register), when the transmitter is enabled (TE is set). The TUE0 flag causes an interrupt if TIE and TUE0_EN are set. The TUE0 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
7 TFS	Transmit Frame Sync. This flag indicates the occurrence of transmit frame sync. Data written to the STX registers during the time slot when the TFS flag is set, is sent during the second time slot (in Network mode) or in the next first time slot (in Normal mode). In Network mode, the TFS bit is set during transmission of the first time slot of the frame and is then cleared when starting transmission of the next time slot. In Normal mode, this bit is always high. This flag causes an interrupt if TIE and TFS_EN are set. The TFS bit is cleared by POR and SSI reset. 0 No Occurrence of Transmit frame sync. 1 Transmit frame sync occurred during transmission of last word written to STX registers.
6 RFS	Receive Frame Sync. This flag indicates the occurrence of receive frame sync. In Network mode, the RFS bit is set when the first slot of the frame is being received. It is cleared when the next slot begins to be received. In Normal mode, this bit is always high. This flag causes an interrupt if RIE and RFS_EN are set. The RFS bit is cleared by POR and SSI reset. 0 No Occurrence of Receive frame sync. 1 Receive frame sync occurred during reception of next word in SRX registers.

Table 16-17. SSI Interrupt Status Register Field Descriptions (continued)

Field	Description
5 TLS	<p>Transmit Last Time Slot. This flag indicates the last time slot in a frame. When set, it indicates that the current time slot is the last time slot of the frame. TLS is set at the start of the last transmit time slot and causes the SSI to issue an interrupt (if TIE and TLS_EN are set). TLS is cleared when the SISR is read with this bit set. The TLS bit is cleared by POR and SSI reset.</p> <p>0 Current time slot is not last time slot of frame. 1 Current time slot is the last transmit time slot of frame.</p>
4 RLS	<p>Receive Last Time Slot. This flag indicates the last time slot in a frame. When set, it indicates that the current time slot is the last receive time slot of the frame. RLS is set at the end of the last time slot and causes the SSI to issue an interrupt (if RIE and RLS_EN are set). RLS is cleared when the SISR is read with this bit set. The RLS bit is cleared by POR and SSI reset.</p> <p>0 Current time slot is not last time slot of frame. 1 Current time slot is the last receive time slot of frame.</p>
3 RFF1	<p>Receive FIFO Full 1. This flag is set when Rx FIFO1 is enabled, the data level in Rx FIFO1 reaches the selected Rx FIFO WaterMark 1 (RFWM1) threshold and the SSI is in Two-Channel mode. The setting of RFF1 only causes an interrupt when RIE and RFF1_EN are set, Rx FIFO1 is enabled and the Two-Channel mode is selected. RFF1 is automatically cleared when the amount of data in Rx FIFO1 falls below the threshold. The RFF1 bit is cleared by POR and SSI reset.</p> <p>When Rx FIFO1 contains 8 words, the maximum it can hold, all further data received (for storage in this FIFO) is ignored until the FIFO contents are read.</p> <p>0 Space available in Receive FIFO1. 1 Receive FIFO1 is full.</p>
2 RFF0	<p>Receive FIFO Full 0. This flag is set when Rx FIFO0 is enabled and the data level in Rx FIFO0 reaches the selected Rx FIFO WaterMark 0 (RFWM0) threshold. The setting of RFF0 only causes an interrupt when RIE and RFF0_EN are set and Rx FIFO0 is enabled. RFF0 is automatically cleared when the amount of data in Rx FIFO0 falls below the threshold. The RFF0 bit is cleared by POR and SSI reset.</p> <p>When Rx FIFO0 contains 8 words, the maximum it can hold, all further data received (for storage in this FIFO) is ignored until the FIFO contents are read.</p> <p>0 Space available in Receive FIFO0. 1 Receive FIFO0 is full.</p>
1 TFE1	<p>Transmit FIFO Empty 1. This flag is set when Tx FIFO1 is enabled, the data level in Tx FIFO1 falls below the selected Tx FIFO WaterMark 1 (TFWM1) threshold and the Two-Channel mode is selected. The setting of TFE1 only causes an interrupt when TIE and TFE1_EN are set, Tx FIFO1 is enabled and Two-Channel mode is selected. The TFE1 bit is automatically cleared when the data level in Tx FIFO1 becomes more than the amount specified by the watermark bits. The TFE1 bit is set by POR and SSI reset.</p> <p>0 Transmit FIFO1 has data for transmission. 1 Transmit FIFO1 is empty.</p>
0 TFE0	<p>Transmit FIFO Empty 0. This flag is set when Tx FIFO0 is enabled and the data level in Tx FIFO0 falls below the selected Tx FIFO WaterMark 0 (TFWM0) threshold. The setting of TFE0 only causes an interrupt when TIE and TFE0_EN are set and Tx FIFO0 is enabled. The TFE0 bit is automatically cleared when the data level in Tx FIFO0 becomes more than the amount specified by the watermark bits. The TFE0 bit is set by POR and SSI reset.</p> <p>0 Transmit FIFO0 has data for transmission. 1 Transmit FIFO0 is empty.</p>

16.7.3.9 SSI Interrupt Enable Register (SIER)

The SSI Interrupt Enable Register (SIER) is a 25-bit register used to set up the SSI interrupts and DMA requests. See [Table 16-18](#) for description of the bit fields in the register.

0xBASE_18 (SIER)																Access: User read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
R	0	0	0	0	0	0	0	RFRC	TFRC	RDMA	RIE	TDMA	TIE	CMDA	CMDDU	RXT			
W								_EN	_EN	E		E		U_EN	_EN	EN			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
R	RDR1	RDR0	TDE1	TDE0	ROE1	ROE0	TUE1	TUE0	TFS_E	RFS_E	TLS_E	RLS_E	RFF1	RFF0	TFE1	TFE0			
W	_EN	_EN	EN	_EN	_EN	_EN	EN	_EN	N	EN	N	N	EN	EN	N	_EN			
RESET	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1			

Table 16-18. SSI Interrupt Enable Register Field Descriptions

Field	Description
31–23	Reserved
24-23 Enable Bits	Enable Bit. Each bit controls whether the corresponding status bit in SISR can issue an interrupt to the Core or not. 0 Status bit cannot issue interrupt. 1 Status bit can issue interrupt.
22 RDMAE	Receive DMA Enable. This bit allows SSI to request for DMA transfers. When enabled, DMA requests are generated when any of the RFF0/1 bits in the SISR are set and if the corresponding RFEN bit is also set. If the corresponding FIFO is disabled, a DMA request is generated when the corresponding RDR bit is set. 0 SSI Receiver DMA requests disabled. 1 SSI Receiver DMA requests enabled.
21 RIE	Receive Interrupt Enable. This control bit allows the SSI to issue receiver related interrupts to the Core. See Section 16.5.3 for a detailed description of this bit. 0 SSI Receiver Interrupt requests disabled. 1 SSI Receiver Interrupt requests enabled.
20 TDMAE	Transmit DMA Enable. This bit allows SSI to request for DMA transfers. When enabled, DMA requests are generated when any of the TFE0/1 bits in the SISR are set and if the corresponding TFEN bit is also set. If the corresponding FIFO is disabled, a DMA request is generated when the corresponding TDE bit is set. 0 SSI Transmitter DMA requests disabled. 1 SSI Transmitter DMA requests enabled.
19 TIE	Transmit Interrupt Enable. This control bit allows the SSI to issue transmitter data related interrupts to the Core. See Section 16.5.4 for a detailed description of this bit. 0 SSI Transmitter Interrupt requests disabled. 1 SSI Transmitter Interrupt requests enabled.
18–0 Enable Bits	Enable Bit. Each bit controls whether the corresponding status bit in SISR can issue an interrupt to the Core or not. 0 Status bit cannot issue interrupt. 1 Status bit can issue interrupt.

16.7.3.10 SSI Transmit Configuration Register (STCR)

The SSI Transmit Configuration Register (STCR) is a read/write control registers used to direct the transmit operation of the SSI. STCR controls the direction of the bit clock and frame sync ports, STCK and STFS. Interrupt enable bit for the transmit sections is provided in this control register. The Power-on reset clears all STCR bits. However, SSI reset does not affect the STCR bits. The STCR bits are described in the following paragraphs. See [Table 16-11](#) for the programming model of the SSI. The SSI Control Register (SCR) must first be set to enable interrupts. Next, the SSI interrupt bit in the Interrupt Enable Register (SIER) must be set to enable the interrupt. Finally, the interrupt can be enabled from within the SSI.

See [Table 16-19](#) for description of the bit fields in the register.

0xBASE_1C (STCR)																Access: User read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
W																			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
R	0	0	0	0	0	0	TXBIT	TFEN1	TFENO	TFDIR	TXDIR	TSHF	TSCK	TFSI	TFSL	TEFS			
W							0					D	P						
RESET	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0			

Table 16-19. SSI Transmit Configuration Register Field Descriptions

Field	Description
31–10	Reserved
9 TXBIT0	Transmit Bit 0. This control bit allows SSI to transmit the data word from bit position 0 or 15/31 in the transmit shift register. The shifting data direction can be MSB or LSB first, controlled by the TSHFD bit. 0 Shifting with respect to bit 31 (if word length = 16, 18, 20, 22 or 24) or bit 15 (if word length = 8, 10 or 12) of transmit shift register (MSB aligned). 1 Shifting with respect to bit 0 of transmit shift register (LSB aligned).
8 TFEN1	Transmit FIFO Enable 1. This bit enables transmit FIFO 1. When enabled, the FIFO allows 8 samples to be transmitted by the SSI (per channel) (a 9th sample can be shifting out) before TDE1 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is transferred to the transmit shift register (provided the interrupt is enabled). 0 Transmit FIFO 1 disabled. 1 Transmit FIFO 1 enabled.
7 TFENO	Transmit FIFO Enable 0. This bit enables transmit FIFO 0. When enabled, the FIFO allows eight samples to be transmitted by the SSI per channel (a 9th sample can be shifting out) before TDE0 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is transferred to the transmit shift register (provided the interrupt is enabled). 0 Transmit FIFO 0 disabled. 1 Transmit FIFO 0 enabled.

Table 16-19. SSI Transmit Configuration Register Field Descriptions (continued)

Field	Description
6 TFDIR	Transmit Frame Direction. This bit controls the direction and source of the transmit frame sync signal. Internally generated frame sync signal is sent out through the STFS port and external frame sync is taken from the same port. 0 Frame Sync is external. 1 Frame Sync generated internally.
5 TXDIR	Transmit Clock Direction. This bit controls the direction and source of the clock signal used to clock the TXSR. Internally generated clock is output through the STCK port. External clock is taken from this port. 0 Transmit Clock is external. 1 Transmit Clock generated internally.
4 TSHFD	Transmit Shift Direction. This bit controls whether the MSB or LSB will be transmitted first in a sample. The CODEC device labels the MSB as bit 0, whereas the Core labels the LSB as bit 0. Therefore, when using a standard CODEC, Core MSB (CODEC LSB) is shifted in first (TSHFD cleared). 0 Data transmitted MSB first. 1 Data transmitted LSB first.
3 TCKP	Transmit Clock Polarity. This bit controls which bit clock edge is used to clock out data for the transmit section. 0 Data clocked out on rising edge of bit clock. 1 Data clocked out on falling edge of bit clock.
2 TFSI	Transmit Frame Sync Invert. This bit controls the active state of the frame sync I/O signal for the transmit section of SSI. 0 Transmit frame sync is active high. 1 Transmit frame sync is active low.
1 TFSL	Transmit Frame Sync Length. This bit controls the length of the frame sync signal to be generated or recognized for the transmit section. The length of a word-long frame sync is same as the length of the data word selected by WL[3:0]. 0 Transmit frame sync is one-word long. 1 Transmit frame sync is one-clock-bit long.
0 TEFS	Transmit Early Frame Sync. This bit controls when the frame sync is initiated for the transmit section. The frame sync signal is deasserted after one bit-for-bit length frame sync and after one word-for-word length frame sync. In case of synchronous operation, the frame sync can also be initiated on receiving the first bit of data. 0 Transmit frame sync initiated as the first bit of data is transmitted. 1 Transmit frame sync is initiated one bit before the data is transmitted.

16.7.3.11 SSI Receive Configuration Register (SRCR)

The SSI Receive Configuration Register (SRCR) is a read/write control registers used to direct the receive operation of the SSI. SRCR controls the direction of the bit clock and frame sync ports, SRCK and SRFS. Interrupt enable bit for the transmit sections is provided in this control register. The Power-on reset clears all SRCR bits. However, SSI reset does not affect the SRCR bits. See [Table 16-20](#) for description of the bit fields in the register.

0xBASE_20 (SRCR)																Access: User read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
W																			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
R	0	0	0	0	0	RXEX T	RXBIT 0	RFEN 1	RFEN 0	RFDIR	RXDIR	RSHF D	RSCK P	RFSI	RFSL	REF S			
W																			
RESET	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0			

Table 16-20. SSI Receive Configuration Register Field Descriptions

Field	Description
31–11	Reserved
10 RXEXT	Receive Data Extension. This control bit allows SSI to store the received data word in sign extended form. This bit affects data storage only in case received data is LSB aligned (SRCR[9]=1) 0 Sign extension turned off. 1 Sign extension turned on.
9 RXBIT0	Receive Bit 0. This control bit allows SSI to receive the data word at bit position 0 or 15/31 in the receive shift register. The shifting data direction can be MSB or LSB first, controlled by the RSHFD bit. 0 Shifting with respect to bit 31 (if word length = 16, 18, 20, 22 or 24) or bit 15 (if word length = 8, 10 or 12) of receive shift register (MSB aligned). 1 Shifting with respect to bit 0 of receive shift register (LSB aligned).
8 RFEN1	Receive FIFO Enable 1. This bit enables receive FIFO 1. When enabled, the FIFO allows eight samples to be received by the SSI per channel (a 9th sample can be shifting in) before RDR1 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is received by the SSI (provided the interrupt is enabled). 0 Receive FIFO 1 disabled. 1 Receive FIFO 1 enabled.
7 RFEN0	Receive FIFO Enable 0. This bit enables receive FIFO 0. When enabled, the FIFO allows 8 samples to be received by the SSI (per channel) (a 9th sample can be shifting in) before RDR0 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is received by the SSI (provided the interrupt is enabled). 0 Receive FIFO 0 disabled. 1 Receive FIFO 0 enabled.

Table 16-20. SSI Receive Configuration Register Field Descriptions (continued)

Field	Description
6 RFDIR	Receive Frame Direction. This bit controls the direction and source of the receive frame sync signal. Internally generated frame sync signal is sent out through the SRFS port and external frame sync is taken from the same port. 0 Frame Sync is external. 1 Frame Sync generated internally.
5 RXDIR	Receive Clock Direction. This bit controls the direction and source of the clock signal used to clock the RXSR. Internally generated clock is output through the SRCK port. External clock is taken from this port. 0 Receive Clock is external. 1 Receive Clock generated internally.
4 RSHFD	Receive Shift Direction. This bit controls whether the MSB or LSB will be received first in a sample. The CODEC device labels the MSB as bit 0, whereas the Core labels the LSB as bit 0. Therefore, when using a standard CODEC, Core MSB (CODEC LSB) is shifted in first (RSHFD cleared). 0 Data received MSB first. 1 Data received LSB first.
3 RSCKP	Receive Clock Polarity. This bit controls which bit clock edge is used to latch in data for the receive section. 0 Data latched on falling edge of bit clock. 1 Data latched on rising edge of bit clock.
2 RFSI	Receive Frame Sync Invert. This bit controls the active state of the frame sync I/O signal for the receive section of SSI. 0 Receive frame sync is active high. 1 Receive frame sync is active low.
1 RFSL	Receive Frame Sync Length. This bit controls the length of the frame sync signal to be generated or recognized for the receive section. The length of a word-long frame sync is same as the length of the data word selected by WL[3:0]. 0 Receive frame sync is one-word long. 1 Receive frame sync is one-clock-bit long.
0 REFS	Receive Early Frame Sync. This bit controls when the frame sync is initiated for the receive section. The frame sync is disabled after one bit-for-bit length frame sync and after one word-for-word length frame sync. 0 Receive frame sync is initiated one bit before the data is received. 1 Receive frame sync initiated as the first bit of data is received.

16.7.3.12 SSI Transmit and Receive Clock Control Registers (STCCR & SRCCR)

The SSI Transmit and Receive Control (STCCR and SRCCR) registers are 19-bit, read/write control registers used to direct the operation of the SSI. The Clock and Reset Module (CRM) can source the SSI clock (`ccm_ssi_clk`) from multiple sources and perform fractional division to support commonly used audio bit rates. The CRM can maintain the `ccm_ssi_clk` frequency at a constant rate even in cases where the `ipg_clk` frequency changes. These registers control the SSI clock generator, bit and frame sync rates, word length, and number of words per frame for the serial data. The STCCR register is dedicated to the transmit section, and the SRCCR register is dedicated to the receive section except in Synchronous mode, in which the STCCR register controls both the receive and transmit sections. Power-on reset clears all STCCR and SRCCR bits. SSI reset does not affect the STCCR and SRCCR bits. The control bits are described in the following paragraphs. Although the bit patterns of the STCCR and SRCCR registers are the same, the contents of these two registers can be programmed differently.

See [Table 16-21](#) for description of the bit fields for both SSI Transmit and Receive Clock Control registers.

0xBASE_24 (STCCR)														Access: User read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3	
W																	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

See [Table 16-21](#) for description of the bit fields for both SSI Transmit and Receive Clock Control registers.

0xBASE_28 (SRCCR)														Access: User read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3	
W																	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-21. SSI Transmit and Receive Clock Control Register Field Descriptions

Field	Description
31–19	Reserved
18 DIV2	Divide By 2. This bit controls a divide-by-two divider in series with the rest of the prescalers. 0 Divider bypassed. 1 Divider used to divide clock by 2.
17 PSR	Prescaler Range. This bit controls a fixed divide-by-eight prescaler in series with the variable prescaler. It extends the range of the prescaler for those cases where a slower bit clock is required. 0 Prescaler bypassed. 1 Prescaler used to divide clock by 8.
16–13 WL3–WL0	Word Length Control. These bits are used to control the length of the data words being transferred by the SSI. These bits control the Word Length Divider in the Clock Generator. They also control the frame sync pulse length when the FSL bit is cleared. In I ² S Master mode, the SSI works with a fixed word length of 32, and the WL bits are used to control the amount of valid data in those 32 bits. See Table 16-22 for details of data word lengths supported by SSI. In AC97 Mode of operation, if word length is set to any value other than 16 bits, it will result in a word length of 20 bits.
12–8 DC4–DC0	Frame Rate Divider Control. These bits are used to control the divide ratio for the programmable frame rate dividers. The divide ratio works on the word clock. In Normal mode, this ratio determines the word transfer rate. In Network mode, this ratio sets the number of words per frame. The divide ratio ranges from 1 to 32 in Normal mode and from 2 to 32 in Network mode. In Normal mode, a divide ratio of 1 (DC=00000) provides continuous periodic data word transfer. A bit-length frame sync must be used in this case. These bits can be programmed with values ranging from “00000” to “11111” to control the number of words in a frame.
7–0 PM7–PM0	Prescaler Modulus Select. These bits control the prescale divider in the clock generator. This prescaler is used only in Internal Clock mode to divide the internal clock (ccm_ssi_clk). The bit clock output is available at the clock port. A divide ratio from 1 to 256 (PM[7:0] = 0x00 to 0xFF) can be selected. See Section 16.5.2.2 for details regarding settings.

Table 16-22. SSI Data Length

WL3	WL2	WL1	WL0	Number of Bits/Word	Supported in Implementation
0	0	0	0	2	No
0	0	0	1	4	No
0	0	1	0	6	No
0	0	1	1	8	Yes
0	1	0	0	10	Yes
0	1	0	1	12	Yes
0	1	1	0	14	No
0	1	1	1	16	Yes
1	0	0	0	18	Yes
1	0	0	1	20	Yes
1	0	1	0	22	Yes

Table 16-22. SSI Data Length (continued)

WL3	WL2	WL1	WL0	Number of Bits/Word	Supported in Implementation
1	0	1	1	24	Yes
1	1	0	0	26	No
1	1	0	1	28	No
1	1	1	0	30	No
1	1	1	1	32	No

16.7.3.13 SSI FIFO Control/Status Register (SFCSR)

See [Table 16-23](#) for description of the bit fields in the register.

0xBASE_2C (SFCSR)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RFCNT1[3:0]				TFCNT1[3:0]				RFWM1[3:0]				TFWM1[3:0]			
W																
RESET	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RFCNT0[3:0]				TFCNT0[3:0]				RFWM0[3:0]				TFWM0[3:0]			
W																
RESET	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

Table 16-23. SSI FIFO Control/Status Register Field Descriptions

Field	Description
31–28 RFCNT1[3:0]	Receive FIFO Counter1. These bits indicate the number of data words in Receive FIFO 1. See Table 16-24 for details regarding settings for receive FIFO counter bits.
27–24 TFCNT1[3:0]	Transmit FIFO Counter1. These bits indicate the number of data words in Transmit FIFO. See Table 16-26 for details regarding settings for transmit FIFO counter bits.
23–20 RFWM1[3:0]	Receive FIFO Full WaterMark 1. These bits control the threshold at which the RFF1 flag will be set. The RFF1 flag is set whenever the data level in Rx FIFO 1 reaches the selected threshold. See Table 16-27 for details regarding settings for receive FIFO watermark bits.
19–16 TFWM1[3:0]	Transmit FIFO Empty WaterMark 1. These bits control the threshold at which the TFE1 flag will be set. The TFE1 flag is set whenever the data level in Tx FIFO 1 falls below the selected threshold. See Table 16-25 for details regarding settings for transmit FIFO watermark bits.
15–12 RFCNT0[3:0]	Receive FIFO Counter 0. These bits indicate the number of data words in Receive FIFO 0. See Table 16-24 for details regarding settings for receive FIFO counter bits .
11–8 TFCNT0[3:0]	Transmit FIFO Counter 0. These bits indicate the number of data words in Transmit FIFO 0. See Table 16-26 for details regarding settings for transmit FIFO counter bits.

Table 16-23. SSI FIFO Control/Status Register Field Descriptions (continued)

Field	Description
7–4 RFWM0[3:0]	Receive FIFO Full WaterMark 0. These bits control the threshold at which the RFF0 flag will be set. The RFF0 flag is set whenever the data level in Rx FIFO 0 reaches the selected threshold. See Table 16-27 for details regarding settings for receive FIFO watermark bits.
3–0 TFWM0[3:0]	Transmit FIFO Empty WaterMark 0. These bits control the threshold at which the TFE0 flag will be set. The TFE0 flag is set whenever the data level in Tx FIFO 0 falls below the selected threshold. See Table 16-25 for details regarding settings for transmit FIFO watermark bits.

Table 16-24. Receive FIFO Counter Bit Description

Bits	Description
0000	0 data word in receive FIFO
0001	1 data word in receive FIFO
0010	2 data word in receive FIFO
0011	3 data word in receive FIFO
0100	4 data word in receive FIFO
0101	5 data word in receive FIFO
0110	6 data word in receive FIFO
0111	7 data word in receive FIFO
1000	8 data word in receive FIFO

Table 16-25. Transmit FIFO WaterMark Bit Description

Bits	Description
0000	Reserved
0001	TFE set when there are more than or equal to 1 empty slots in Transmit FIFO. (default) Transmit FIFO empty is set when TxFIFO <= 7 data.
0010	TFE set when there are more than or equal to 2 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 6 data.
0011	TFE set when there are more than or equal to 3 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 5 data.
0100	TFE set when there are more than or equal to 4 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 4 data.
0101	TFE set when there are more than or equal to 5 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 3 data.
0110	TFE set when there are more than or equal to 6 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 2 data.
0111	TFE set when there are more than or equal to 7 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 1 data.
1000	TFE set when there are 8 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO=0 data.

Table 16-26. Transmit FIFO Counter Bit Description

Bits	Description
0000	0 data word in transmit FIFO
0001	1 data word in transmit FIFO
0010	2 data word in transmit FIFO
0011	3 data word in transmit FIFO
0100	4 data word in transmit FIFO
0101	5 data word in transmit FIFO
0110	6 data word in transmit FIFO
0111	7 data word in transmit FIFO
1000	8 data word in transmit FIFO

Table 16-27. Receive FIFO WaterMark Bit Description

Bits	Description
0000	Reserved
0001	RFF set when at least one data word have been written to the Receive FIFO Set when RxFIFO = 1,2,3,4,5,6,7,8 data words
0010	RFF set when more than or equal to 2 data word have been written to the Receive FIFO. Set when RxFIFO = 2,3,4,5,6,7,8 data words
0011	RFF set when more than or equal to 3 data word have been written to the Receive FIFO. Set when RxFIFO = 3,4,5,6,7,8 data words
0100	RFF set when more than or equal to 4 data word have been written to the Receive FIFO. Set when RxFIFO = 4,5,6,7,8 data words
0101	RFF set when more than or equal to 5 data word have been written to the Receive FIFO. Set when RxFIFO = 5,6,7,8 data words
0110	RFF set when more than or equal to 6 data word have been written to the Receive. Set when RxFIFO = 6,7,8 data words
0111	RFF set when more than or equal to 7 data word have been written to the Receive FIFO. Set when RxFIFO = 7,8 data words
1000	RFF set when 8 data word have been written to the Receive FIFO (default). Set when RxFIFO = 8 data words

Table 16-28 indicates the status of the Transmit FIFO Empty flag, with different settings of the Transmit FIFO WaterMark bits and varying amounts of data in the Tx FIFO.

Table 16-28. Status of Transmit FIFO Empty Flag

Transmit FIFO Watermark (TFWM)	Number of data in Tx-Fifo							
	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	0
2	1	1	1	1	1	1	0	0
3	1	1	1	1	1	0	0	0
4	1	1	1	1	0	0	0	0
5	1	1	1	0	0	0	0	0
6	1	1	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

16.7.3.14 SSI AC97 Control Register (SACNT)

See Table 16-29 for description of the bit fields for the register.

0xBASE_38 (SACNT)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	FRDIV[5:0]						WR	RD	TIF	FV	AC97 EN
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-29. SSI AC97 Control Register Field Descriptions

Field	Description
31–11	Reserved
10–5 FRDIV[5:0]	Frame Rate Divider. These bits control the frequency of AC97 data transmission/reception. They are programmed with the number of frames for which the SSI should be idle, after operating in one frame. Through these bits, AC97 frequency of operation, from 48 KHz (000000) to 1 KHz (101111) can be achieved. Sample Value: 001010 (10 Decimal) = SSI will operate once every 11 frames.

Table 16-29. SSI AC97 Control Register Field Descriptions (continued)

Field	Description
4 WR	<p>Write Command. This bit specifies whether the next frame will carry an AC97 Write Command or not. The programmer should take care that only one of the bits (WR or RD) is set at a time. When this bit is set, the corresponding tag bits (corresponding to Command Address and Command Data slots of the next Tx frame) are automatically set. This bit is automatically cleared by the SSI after completing transmission of a frame.</p> <p>0 Next frame will not have a Write Command. 1 Next frame will have a Write Command.</p>
3 RD	<p>Read Command. This bit specifies whether the next frame will carry an AC97 Read Command or not. The programmer should take care that only one of the bits (WR or RD) is set at a time. When this bit is set, the corresponding tag bit (corresponding to Command Address slot of the next Tx frame) is automatically set. This bit is automatically cleared by the SSI after completing transmission of a frame.</p> <p>0 Next frame will not have a Read Command. 1 Next frame will have a Read Command.</p>
2 TIF	<p>Tag in FIFO. This bit controls the destination of the information received in AC97 tag slot (Slot #0).</p> <p>0 Tag info stored in SATAG register. 1 Tag info stored in Rx FIFO 0.</p>
1 FV	<p>Fixed/Variable Operation. This bit selects whether the SSI is in AC97 Fixed mode or AC97 Variable mode.</p> <p>0 AC97 Fixed Mode. 1 AC97 Variable Mode.</p>
0 AC97EN	<p>AC97 Mode Enable. This bit is used to enable SSI AC97 operation. See Section 16.7.3.14 for details of AC97 operation.</p> <p>0 AC97 mode disabled. 1 SSI in AC97 mode.</p>

16.7.3.15 SSI AC97 Command Address Register (SACADD)

See [Table 16-30](#) for description of the bit fields for the register.

0xBASE_3C (SACADD)													Access: User read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	SACADD[18:16]		
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SACADD[15:0]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-30. SSI AC97 Command Address Register Field Descriptions

Field	Description
31–19	Reserved
18–0 SACADD	AC97 Command Address. These bits store the Command Address Slot information (bit 19 of the slot is sent in accordance with the Read and Write Command bits in SACNT register). These bits can be updated by a direct write from the Core. They are also updated with the information received in the incoming Command Address Slot. If the contents of these bits change due to an update, the CMDAU bit in SISR is set.

16.7.3.16 SSI AC97 Command Data Register (SACDAT)

See [Table 16-31](#) for description of the bit fields for the register.

0xBASE_40 (SACDAT)												Access: User read/write				
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	SACDAT[19:16]			
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SACDAT[15:0]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-31. SSI AC97 Command Data Register

Field	Description
31–20	Reserved
19–0 SACDAT	AC97 Command Data. The outgoing Command Data Slot carries the information contained in these bits. These bits can be updated by a direct write from the Core. They are also updated with the information received in the incoming Command Data Slot. If the contents of these bits change due to an update, the CMDDU bit in SISR is set. These bits are transmitted only during AC97 Write Command. During AC97 Read Command, 0x00000 is transmitted in time slot #2.

16.7.3.17 SSI AC97 Tag Register (SATAG)

See [Table 16-32](#) for description of the bit fields for the register.

0xBASE_44 (SATAG)																Access: User read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
W																			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
R	SATAG[15:0]																		
W																			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 16-32. SSI AC97 Tag Register Field Descriptions

Field	Description
31–16	Reserved
15–0 SATAG	AC97 Tag Value. Writing to this register (by the Core) sets the value of the Tx-Tag in AC97 fixed mode of operation. On a read, the Core gets the Rx-Tag Value received (in the last frame) from the Codec. If TIF bit in SACNT register is set, the TAG value is also stored in Rx-FIFO in addition to SATAG register. When the received Tag value changes, the RXT bit in SISR register is set. Bits SATAG[1:0] convey the Codec-ID. In current implementation only Primary Codecs are supported. Thus writing value 2'b00 to this field is mandatory.

16.7.3.18 SSI Transmit Time Slot Mask Register (STMSK)

See Table 16-33 for description of the bit fields for the register.

0xBASE_48 (STMSK)																Access: User read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
R	STMSK[31:16]																		
W																			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
R	STMSK[15:0]																		
W																			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 16-33. SSI Transmit Time Slot Mask Register Field Descriptions

Field	Description
31–0 STMSK	Transmit Mask. These bits indicate which slot has been masked in the current frame. The Core can write to this register to control the time slots in which the SSI transmits data. Each bit has info corresponding to the respective time slot in the frame. If a change is made to the register contents, the transmission pattern is updated from the next time slot. Transmit mask bits should not be used in I ² S Slave mode of operation. 0 Valid Time Slot. 1 Time Slot masked (no data transmitted in this time slot).

16.7.3.19 SSI Receive Time Slot Mask Register (SRMSK)

See [Table 16-34](#) for description of the bit fields for the register.

0xBASE_4C (SRMSK)														Access: User read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	SRMSK[31:16]																
W																	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	SRMSK[15:0]																
W																	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 16-34. SSI Receive Time Slot Mask Register Field Descriptions

Field	Description
31–0 SRMSK	Receive Mask. These bits indicate which slot has been masked in the current frame. The Core can write to this register to control the time slots in which the SSI receives data. Each bit has info corresponding to the respective time slot in the frame. If a change is made to the register contents, the reception pattern is updated from the next time slot. Receive mask bits should not be used in I ² S Slave mode of operation. 0 Valid Time Slot. 1 Time Slot masked (no data received in this time slot).

16.7.3.20 SSI AC97 Channel Status Register (SACCST)

See Table 16-35 for description of the bit fields for the register.

0xBASE_50 (SACCST)																Access: User read			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
W																			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	SACCST[9:0]									
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-35. SSI AC97 Channel Status Register Field Descriptions

Field	Description
9–0 SACCST	<p>AC97 Channel Status. These bits indicate which data slot has been enabled in AC97 variable mode operation. This register is updated in case the core enables/disables a channel through a write to SACCEN/sACCDIS register or the external codec enables a channel by sending a '1' in the corresponding SLOTREQ bit. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). The contents of this register only have relevance while the SSI is operating in AC97 variable mode. Writes to this register result in an error response on the IP interface.</p> <p>0 Data channel disabled. 1 Data channel enabled.</p>

16.7.3.21 SSI AC97 Channel Enable Register (SACCEN)

See [Table 16-36](#) for description of the bit fields for the register.

0xBASE_54 (SACCEN)																Access: User write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
W																			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
W									SACCEN[9:0]										
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 16-36. SSI AC97 Channel Enable Register Field Descriptions

Field	Description
9–0 SACCEN	AC97 Channel Enable. The Core writes a ‘1’ to these bits to enable an AC97 data channel. Writing a ‘0’ has no effect. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). Writes to these bits only have effect in the AC97 Variable mode of operation. These bits are always read as ‘0’ by the Core. 0 Write Has no effect. 1 Write Enables the corresponding data channel.

16.7.3.22 SSI AC97 Channel Disable Register (SACCDIS)

See [Table 16-37](#) for description of the bit fields for the register.

0xBASE_58 (SACCDIS)																Access: User write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
W																			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
W					SACCDIS[9:0]														
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 16-37. SSI AC97 Channel Disable Register Field Descriptions

Field	Description
9–0 SACCDIS	AC97 Channel Disable. The Core writes a '1' to these bits to disable an AC97 data channel. Writing a '0' has no effect. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). Writes to these bits only have effect in the AC97 Variable mode of operation. These bits are always read as '0' by the Core. 0 Write Has no effect. 1 Write Disables the corresponding data channel.

Chapter 17

Analog to Digital Converter Module (ADC)

17.1 Overview

The ADC module manages the interface to external sensors, scans multiple channels, and controls the warm-up of the analog portions of the analog to digital system. The ADC module can be set to interrupt the ARM core based on programmed compare values that can be set for up to 8 channels. The primary A to D can be programmed to convert/monitor 9 channels (8 GP-ADC pins plus battery). The secondary A to D can be programmed to convert/monitor the 8 GP-ADC pins.

17.2 Features

The ADC has the following features:

- 12 bit resolution. Effective number of bits 10.5
- Input voltage range: Vref_high, Vref_low
- Maximum input of 3.6v
- Minimum input of 0.0v
- Integral non-linearity 0.25 bit
- Differential non-linearity 0.5 bit
- Maximum current: 4ma
- Sample rate maximum 20us
- 8 bit prescaler to provide the time base for the 32 bit timers
- Two independent channels, each with a 32 bit timer
- Primary ADC has 9 channels: 8 GPIO plus battery
- Secondary ADC has 8 channels: 8 GPIO
- Active channels for each A to D are programmable
- A maximum of 8 active monitors can generate a IRQ trigger
- A 8 deep FIFO for recording data
- IRQs can be generated by the channel compare values, FIFO status, 32 bit timers

17.3 Block Diagram

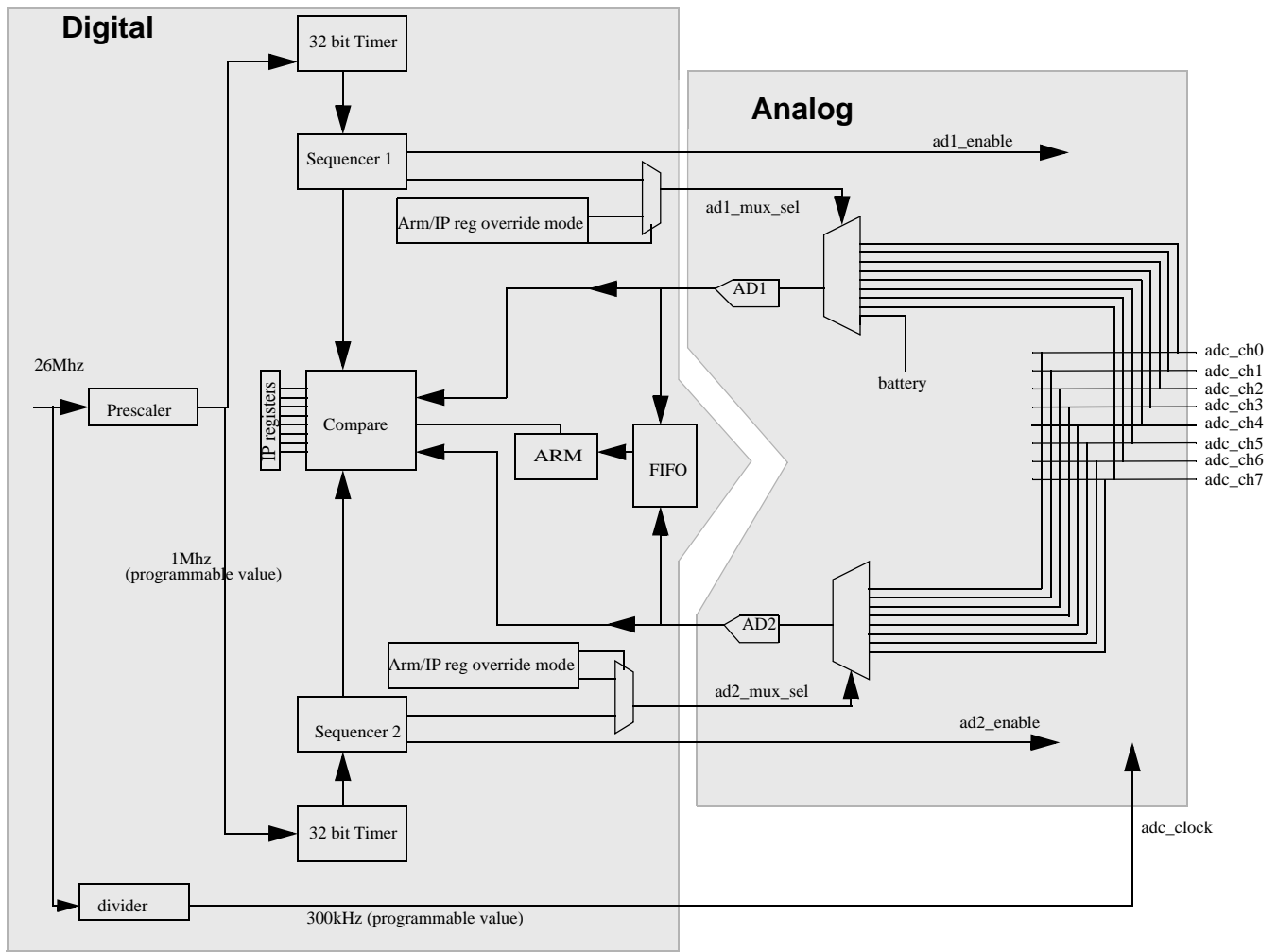


Figure 17-1. ADC Block Diagram

17.4 ADC Registers

Table 17-1. ADC Registers

Address	Register Name	Access
Base + 0x00	adc_comp_0	R/W
Base + 0x02	adc_comp_1	R/W
Base + 0x04	adc_comp_2	R/W
Base + 0x06	adc_comp_3	R/W
Base + 0x08	adc_comp_4	R/W
Base + 0x0a	adc_comp_5	R/W
Base + 0x0c	adc_comp_6	R/W
Base + 0x0e	adc_comp_7	R/W
Base + 0x10	adc_bat_comp_over	R/W

Table 17-1. ADC Registers (continued)

Base + 0x12	adc_bat_comp_under	R/W
Base + 0x14	adc_seq_1	R/W
Base + 0x16	adc_seq_2	R/W
Base + 0x18	adc_control	R/W
Base + 0x1a	adc_triggers	R
Base + 0x1c	adc_prescale	R/W
Base + 0x1e		
Base + 0x20	adc_fifo_read	R
Base + 0x22	adc_fifo_control	R/W
Base + 0x24	adc_fifo_status	R
Base + 0x26		
Base + 0x28		
Base + 0x2a		
Base + 0x2c		
Base + 0x2e		
Base + 0x30	adc_1_sr_high	R/W
Base + 0x32	adc_1_sr_low	R/W
Base + 0x34	adc_2_sr_high	R/W
Base + 0x36	adc_2_sr_low	R/W
Base + 0x38	adc_on_time	R/W
Base + 0x3a	adc_convert_time	R/W
Base + 0x3c	adc_clock_div	R/W
Base + 0x3e		
Base + 0x40	adc_override	R/W
Base + 0x42	adc_irq	R/W
Base + 0x44	adc_mode	R/W
Base + 0x46	ad1_result	R
Base + 0x48	ad2_result	R

17.5 Register Access

All registers in the ADC register map can only be accessed as 16-bit half-words. No byte accesses are allowed.

17.5.1 adc_comp_0

This register contains information to set the ADC module to monitor a GP-ADC pin for changes.

adc_comp_0													Addr Base+0x00			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GL	channel			value											
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GL (Greater/Less Than)

If (value on channel < value) and (sequencer active channel == channel) then trigger a compare flag.

If (value on channel > value) and (sequencer active channel == channel) then trigger a compare flag.

channel

This is a three bit value which controls which GP-ADC pin is being monitored. Values can be 0,1,2,3,4,5,6,7 which correlate to ADC_CHAN0 to ADC_CHAN7.

value

This is a 12 bit value that is the compare threshold.

17.5.2 adc_comp_1

This register contains information to set the ADC module to monitor a GP-ADC pin for changes.

adc_comp_1													Addr Base+0x02			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GL	channel			value											
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GL (Greater/Less Than)

1 = If (value on channel < value) and (sequencer active channel == channel) then trigger a compare flag.

0 = If (value on channel > value) and (sequencer active channel == channel) then trigger a compare flag.

channel

This is a three bit value which controls which GP-ADC pin is being monitored. Values can be 0,1,2,3,4,5,6,7 which correlate to ADC_CHAN0 to ADC_CHAN7.

value

This is a 12 bit value that is the compare threshold.

17.5.3 adc_comp_2

This register contains information to set the ADC module to monitor a GP-ADC pin for changes.

adc_comp_2													Addr Base+0x04			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GL	channel			value											
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GL (Greater/Less Than)

- 1 = If (value on channel < value) and (sequencer active channel == channel) then trigger a compare flag.
- 0 = If (value on channel > value) and (sequencer active channel == channel) then trigger a compare flag.

channel

This is a three bit value which controls which GP-ADC pin is being monitored. Values can be 0,1,2,3,4,5,6,7 which correlate to ADC_CHAN0 to ADC_CHAN7.

value

This is a 12 bit value that is the compare threshold.

17.5.4 adc_comp_3

This register contains information to set the ADC module to monitor a GP-ADC pin for changes.

adc_comp_3													Addr Base+0x06			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GL	channel			value											
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GL (Greater/Less Than)

1 = If (value on channel < value) and (sequencer active channel == channel) then trigger a compare flag.

0 = If (value on channel > value) and (sequencer active channel == channel) then trigger a compare flag.

channel

This is a three bit value which controls which GP-ADC pin is being monitored. Values can be 0,1,2,3,4,5,6,7 which correlate to ADC_CHAN0 to ADC_CHAN7.

value

This is a 12 bit value that is the compare threshold.

17.5.5 adc_comp_4

This register contains information to set the ADC module to monitor a GP-ADC pin for changes.

adc_comp_4													Addr Base+0x08			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GL	channel			value											
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GL (Greater/Less Than)

- 1 = If (value on channel < value) and (sequencer active channel == channel) then trigger a compare flag.
- 0 = If (value on channel > value) and (sequencer active channel == channel) then trigger a compare flag.

channel

This is a three bit value which controls which GP-ADC pin is being monitored. Values can be 0,1,2,3,4,5,6,7 which correlate to ADC_CHAN0 to ADC_CHAN7.

value

This is a 12 bit value that is the compare threshold.

17.5.6 adc_comp_5

This register contains information to set the ADC module to monitor a GP-ADC pin for changes.

adc_comp_5													Addr Base+0x0a			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GL	channel			value											
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GL (Greater/Less Than)

1 = If (value on channel < value) and (sequencer active channel == channel) then trigger a compare flag.

0 = If (value on channel > value) and (sequencer active channel == channel) then trigger a compare flag.

channel

This is a three bit value which controls which GP-ADC pin is being monitored. Values can be 0,1,2,3,4,5,6,7 which correlate to ADC_CHAN0 to ADC_CHAN7.

value

This is a 12 bit value that is the compare threshold.

17.5.7 adc_comp_6

This register contains information to set the ADC module to monitor a GP-ADC pin for changes.

adc_comp_6													Addr Base+0x0c			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GL	channel			value											
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GL (Greater/Less Than)

- 1 = If (value on channel < value) and (sequencer active channel == channel) then trigger a compare flag.
- 0 = If (value on channel > value) and (sequencer active channel == channel) then trigger a compare flag.

channel

This is a three bit value which controls which GP-ADC pin is being monitored. Values can be 0,1,2,3,4,5,6,7 which correlate to ADC_CHAN0 to ADC_CHAN7.

value

This is a 12 bit value that is the compare threshold.

17.5.8 adc_comp_7

This register contains information to set the ADC module to monitor a GP-ADC pin for changes.

adc_comp_7													Addr Base+0x0e			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GL	channel			value											
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GL (Greater/Less Than)

1 = If (value on channel < value) and (sequencer active channel == channel) then trigger a compare flag.

0 = If (value on channel > value) and (sequencer active channel == channel) then trigger a compare flag.

channel

This is a three bit value which controls which GP-ADC pin is being monitored. Values can be 0,1,2,3,4,5,6,7 which correlate to ADC_CHAN0 to ADC_CHAN7.

value

This is a 12 bit value that is the compare threshold.

17.5.9 adc_bat_comp_over

Sets the battery voltage upper trip point.

adc_bat_comp_over													Addr Base+0x10			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	unused				value											
TYPE	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

value

This is a 12 bit value that is the compare threshold.

17.5.10 adc_bat_comp_under

Sets the battery voltage lower trip point.

adc_bat_com_under													Addr Base+0x12			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	unused				value											
TYPE	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

value

This is a 12 bit value that is the compare threshold.

17.5.11 adc_seq_1

Selects the channels (GP-ADC pins) to be monitored.

adc_seq_1													Addr Base+0x14			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	seq_mode	unused						batt	ch7	ch6	ch5	ch4	ch3	ch2	ch1	ch0
TYPE	rw	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

batt, ch7,ch6,ch5,ch4,ch3,ch2,ch1,ch0

Controls channel activation

1 = channel active

0 = channel not active

seq_mode

This bit controls how the channels are sequenced.

1 = Sequence to next channel based on timer event.

0 = Sequence to next channel based on convert time (20us).

17.5.12 adc_seq_2

Selects the channels (GP-ADC pins that will be monitored).

adc_seq_1													Addr Base+0x16				
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0	
	seq_mode	unused							ch7	ch6	ch5	ch4	ch3	ch2	ch1	ch0	
TYPE	rw	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

ch7,ch6,ch5,ch4,ch3,ch2,ch1,ch0

Controls channel activation

1 = channel active

0 = channel not active

seq_mode

This will bit controls how the channels are sequenced.

1 = Sequence to next channel based on timer event.

0 = Sequence to next channel based on convert time (20us).

17.5.13 adc_control

Primary module control register.

adc_control													Addr Base+0x18				
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0	
	fifo_irq_mask	seq2_irq_mask	seq1_irq_mask	compare_irq_mask	unused							ad2_vrefhl_en	ad1_vrefhl_en	soft_reset	timer2_on	timer1_on	on
TYPE	rw	rw	rw	rw	r	r	r	r	r	r	r	r	rw	rw	rw	rw	
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	

fifo_irq_mask, seq2_irq_mask, seq1_irq_mask, compare_irq_mask

Controls if IRQs are allowed.

1 = IRQ active

0 = IRQ not active

ad2_vrefhl_en

0 = Battery as ref voltage source for A/D 2

1 = External ref voltage source for A/D 2

ad1_vrefhl_en

0 = Battery as ref voltage source for A/D 1

1 = External ref voltage source for A/D 1

Soft_reset

The software can use this bit to reset the ADC module.

1 = ADC module held in reset

0 = ADC module released from reset.

timer2_on, timer1_on

Controls activation of the 32 bit timers

1 = Timer is active

0 = Timer is not active.

on

Controls activation of the ADC module

1 = ADC module is on

0 = ADC module is not on.

17.5.14 adc_triggers

This register is read-only. It displays which channel are triggered.

adc_triggers													Addr Base+0x1a				
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0	
	unused						bat_ over	bat_ under	ch7	ch6	ch5	ch4	ch3	ch2	ch1	ch0	
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

bat_over, bat_under, ch7, ch6, ch5, ch4, ch3, ch2, ch1, ch0

1 = triggered

0 = not triggered

17.5.15 adc_prescale

Sets the bus clock divide ratio. The clock that drives the timers & sequencers is determined by this register.

adc_prescale													Addr Base+0x1c			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	unused								prescale							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

prescale

The 8 bit prescale value.

The divide_ratio = prescale + 1. Prescale = 0 means no divide factor, running of 26Mhz. A prescale value of 12 implies users are doing a divide by 13. This implies a clock frequency of 2Mhz. A prescale value of 25 implies users are doing a divide by 26. This implies a clock frequency of 1 Mhz.

17.5.16 adc_fifo_read

Each read of this register updates the FIFO pointers and returns the value stored in the FIFO.

adc_fifo_read													Addr Base+0x20			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	channel				value											
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

channel

This is a 4 bit value which indicates which GP-ADC pin the sample was taken on. Values can be 0,1,2,3,4,5,6,7,8 which correlate to ADC_CHAN0 to ADC_CHAN7 and BAT.

Table 17-2. Channel to Pin Values

Channel	GP-ADC pin
0000	ADC_CHAN0
0001	ADC_CHAN1
0010	ADC_CHAN2
0011	ADC_CHAN3
0100	ADC_CHAN4
0101	ADC_CHAN5
0110	ADC_CHAN6
0111	ADC_CHAN7
1000	BATTERY

value

This is a 12 bit value that is the sample stored in FIFO.

17.5.17 adc_fifo_control

A write to this register sets the level at which the FIFO generates an IRQ.

adc_fifo_control													Addr Base+0x22			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
													level			
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

level

This is the value which sets the fill level that the FIFO must attain before generating a IRQ.

17.5.18 adc_fifo_status

This register provides the status of the ADC FIFO.

adc_fifo_status													Addr Base+0x24			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	unused											empty	full	level		
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

empty

1 = FIFO is empty

0 = FIFO is not empty

full

1 = FIFO is full

0 = FIFO is not full

level

This is a three bit value which indicates how many values are stored in the FIFO.

17.5.19 adc_sr_1_high

A write to this register sets the upper 16 bits of the timer 1 compare value.

NOTE

Timers run on the prescale clock. The time unit for these register is a function of the prescale frequency.

adc_sr_1_high													Addr Base+0x30			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	sample rate 1 high															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.5.20 adc_sr_1_low

A write to this register sets the lower 16 bits of the timer 1 compare value.

NOTE

Timers run on the prescale clock. The time unit for these register is a function of the prescale frequency.

adc_sr_1_low													Addr Base+0x32			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	sample rate 1 low															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.5.21 adc_sr_2_high

A write to this register sets the upper 16 bits of the timer 2 compare value.

NOTE

Timers run on the prescale clock. The time unit for these register is a function of the prescale frequency.

adc_sr_2_high													Addr Base+0x34			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	sample rate 2 high															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.5.22 adc_sr_2_low

A write to this register sets the lower 16 bits of the timer 2 compare value.

NOTE

Timers run on the prescale clock. The time unit for these register is a function of the prescale frequency.

adc_sr_2_low													Addr Base+0x36			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	sample rate 2 low															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.5.23 adc_on_time

A write to this register sets the turn on time for the analog A to D. The value in this register is a function of the prescale clock. For example, if the prescale clock is programmed for 1Mhz, then a value of 8 in this register results in 8 μ sec.

adc_on_time													Addr Base+0x38			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	on time															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.5.24 adc_convert_time

A write to this register sets the A to D convert time. This should not be set to a value of less than 20 μ s. The analog AD requires no less than 20 μ sec to convert data. The value in this register is a function of the prescaled clock. For example, if the prescaled clock is programmed for 1Mhz, then a value of 20 in this register results in 20 μ sec. If a value less than 20 μ sec is used, the digital ADC samples data from analog before it has completed the data conversion. If this occurs, the data is not accurate.

adc_convert_time													Addr Base+0x3a			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	convert time															
TYPE	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.5.25 adc_clock_divider

A write to this register sets the clock divider that provides the clock to the analog portion of the ADC module.

adc_clock_divider													Addr Base+0x3c			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	unused								clock divider							
TYPE	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This should be set to a value that provides a 300 khz clock. For example, to generate a value for a 300 KHz clock, then $26 \text{ Mhz} / 300\text{khz} = 86.66$. Thus, a value of 86 would be written to this divider. If this register is set to 0, ADC clock to analog is off.

17.5.26 adc_override

This register provides a way for the software to manually control the ADC module. For this register to take control, the override bit must be set in the adc_mode register.

adc_override													Addr Base+0x40			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	unused						ad2_on	ad1_on	mux2				mux1			
TYPE	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ad2_on

- 1 = ad2 is on
- 0 = ad2 is off

ad1_on

- 1 = ad1 is on
- 0 = ad1 is off

mux2

Set the GP-ADC selector for the secondary A to D.

Table 17-3. MUX2 Value Pin Select

MUX2 value	GP-ADC pin selected
0000	ADC_CHAN0
0001	ADC_CHAN1

Table 17-3. MUX2 Value Pin Select (continued)

MUX2 value	GP-ADC pin selected
0000	ADC_CHAN0
0010	ADC_CHAN2
0011	ADC_CHAN3
0100	ADC_CHAN4
0101	ADC_CHAN5
0110	ADC_CHAN6
0111	ADC_CHAN7

mux1

Set the GP-ADC selector for the primary A to D.

Table 17-4. MUX1 Value GP-ADC Pin Select

MUX1 value	GP-ADC pin selected
0000	ADC_CHAN0
0001	ADC_CHAN1
0010	ADC_CHAN2
0011	ADC_CHAN3
0100	ADC_CHAN4
0101	ADC_CHAN5
0110	ADC_CHAN6
0111	ADC_CHAN7
1000	BATTERY

17.5.27 adc_irq

This register can be read to determine the active IRQs. A write of 1 to the respective bit clears the IRQ source.

NOTE

If using compare IRQ, read the adc-triggers register for information on which channel generated the compare IRQ. Read the FIFO status register to if using the FIFO IRQ.

adc_irq													Addr Base+0x42			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	fifo	seq2	seq1	comp are	unused											
TYPE	rw	rw	rw	rw	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

17.5.28 adc_mode

This register control the ADC mode.

adc_mode													Addr Base+0x44			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	unused															overri de
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

override

- 1 = The adc_override register can be used to manually control the ADC.
- 0 = The ADC module will be automatically controlled.

17.5.29 adc_ad1_result

A read of this register will return the result of the primary A to D.

adc_ad1_result													Addr Base+0x46			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	unused			ad1_result												
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.5.30 adc_ad2_result

A read of this register will return the result of the secondary A to D.

adc_ad2_result													Addr Base+0x48			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	unused			ad2_result												
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.6 Detailed description of Digital ADC functionality

17.6.1 Prescale Clock

Prescaled clock = IP_clock / prescale register value. Both the timers and sequencers function on the prescaled clock.

17.6.2 Clock Divider

adc_clock (to analog) = IP_clock / clock_divider

17.6.3 Sequencers

The digital adc has multiple configurations in which it can run. In general we can narrow it to 2 modes: override mode (controlled via arm), and automated mode (via sequencers). The automated mode is used by selecting desired active channels and a seq_mode in the adc_seq_1 and adc_seq_2 registers. There are 2 ADs, register adc_seq_1 is for AD1 and adc_seq_2 is for AD2. When using a sequencer, the capture mechanism is selected by the seq_mode bit. The sequencer will "sequence" through all active channels to sample & capture data based on this mode.

NOTE

When using seq_mode set to timer events, the timer sample_rate high and low registers must also be programmed. If users are sequencing between channels based on convert time, there is a minimum of 20 μ sec inaccessibly for analog AD to convert the data into digital values. When using the ADC in automated mode, Freescale recommends using the available sequencer and FIFO IRQs to manage data-flow control.

It is also important to note that the number of channels active and the convert-time will affect total cycle time for a sequencer. The cycle time between sequencer samples on the same channel can be approximated to the number of channels active multiplied by convert-time. Example, if 8 channels are active, and convert time is 20usec, it will be $8*20=160$ usec before the same channel is sampled again. See section 16.5.6 for timing.

17.6.4 Comparators

Both ADs share the comparators and their associated registers (adc_comp_0 - adc_comp_7). Based on what is programmed in the sample_rate_high and sample_rate_low registers for each AD, the timers will trigger a compare IRQ if a desired threshold is reached.

NOTE

AD1 and AD2 have separate sample_rate registers.

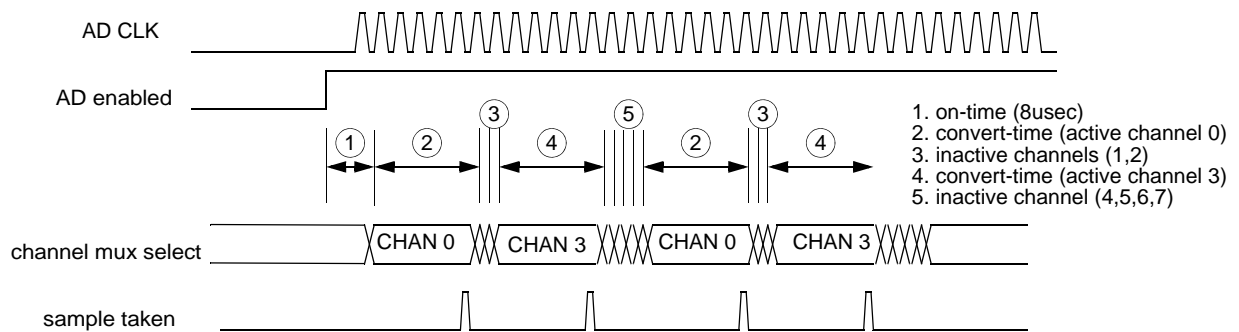
17.6.5 Fifo

The status can be found by reading the `adc_fifo_status` register. There is no overflow bit so FIFO must be correctly managed.

NOTE

If the FIFO is full, new samples will not be stored until previous samples are read out from the FIFO. The ADC continues to run and samples data as it is programmed to do, but it will not be stored in the FIFO. When the FIFO IRQ is handled, the FIFO needs to be cleared for new data. It is also important to note that the FIFO is only 8 bits deep.

17.6.6 Timing



NOTE: This timing scenario represents the case where ADC channels 0 and 3 are active and 1,2,4,5,6,7 are inactive.

Figure 17-2. Sample Timing

Chapter 18

Development and Debug Support

The MC1322x family is supported by a full set of hardware/software evaluation and development tools.

18.1 Hardware Development Interfaces

The ARM debug environment supports both a JTAG debug interface and an extended capability Nexus interface.

18.1.1 JTAG Hardware Debug Port

The JTAG port is the simpler and more common debug port for the ARM core. A standard 20-pin connector as described in [Section 2.3.1, “ARM JTAG Interface Connector”](#), is connected to the TDI, TMS, TCK, TDO, and RTCK signals of the MC1322x. Through the JTAG serial interface, standard debug and development activities such as accessing memory and registers, control of the CPU, download of FLASH memory, and software debug can be accomplished.

18.1.2 A7S Nexus3 (NEX) ARM7 Core Development Interface

The development and debug environment of the ARM7TDMI-S core is based on the A7S Nexus3 interface (compliant with a Class 3 device of the IEEE-ISTO 5001 standard for real-time embedded system design). This interface allows expansion of the development features of the JTAG port (through the addition of auxiliary signals, see [Section 2.3.2, “Nexus Mictor Interface Connector”](#)). Development features include:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to (selected) internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Run-time access to the memory map via the JTAG port. This allows for enhanced download/upload capabilities
- Watchpoint Messaging (WPM) via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- Auxiliary interface for higher data input/output

- Registers for Program Trace, Ownership Trace, Watchpoint Trigger, and Read/Write Access
- Programmable processor stall function to mitigate message queue overrun risk
- All features controllable and configurable via the JTAG port

18.2 Software Development Tools

An Integrated Development Environment (IDE) is available to facilitate the development of embedded applications targeting the MC1322x platform. Features of the IDE include:

- Project management tools and code editor
- Highly optimizing ARM compiler supporting C and C++
- Extensive JTAG and RDI debugger support
- Run-time libraries including source code
- Relocating ARM assembler
- Linker and librarian tools
- Debugger with ARM simulator, JTAG support and support for RTOS-aware debugging on hardware
- RTOS plug-ins available
- Code templates for commonly used code constructs
- Sample projects for evaluation boards
- User and reference guides, both printed and in PDF format
- Context-sensitive online help

The IDE is complemented by BeeKit™. BeeKit is a stand alone software application targeting Windows® operating systems. BeeKit™ provides a graphical user interface (GUI) in which the users can create, modify, save, and update wireless networking solutions. With the solution explorer property list windows, users will be able to set configuration parameters that will control the setup and execution behavior of the wireless link within their application. The configuration parameters can be validated inside BeeKit™ to ensure all values provided are within acceptable ranges prior to generation of a workspace. All this functionality provides a mechanism for developers to configure and validate their network parameters without having to navigate through multiple source files to configure the same parameters. BeeKit™ supports Freescale's Simple MAC (SMAC), IEEE 802.15.4-compliant MAC, and BeeStack™.

18.3 Development Hardware

Several different development modules and kits will be available to allow evaluation of ZigBee and IEEE 802.15.4 applications. The modules will provide capabilities for Coordinator, Router, and End Device nodes. Reference designs will be available for RF design and low power applications including 2-layer and 4-layer PCBs.

Appendix A

MC1322x Register Address Map

Table A-1. Register Address Memory Map

Base Address	Register Address	Mnemonic	Name	Type	Width (Bits)
0x8000_0000			GPIO		
	+ 0x00	GPIO_PAD_DIR0	GPIO Pad Direction for GPIO 00-31	R/W	32
	+ 0x04	GPIO_PAD_DIR1	GPIO Pad Direction for GPIO 32-63	R/W	32
	+ 0x08	GPIO_DATA0	GPIO Data for GPIO 00-31	R/W	32
	+ 0x0C	GPIO_DATA1	GPIO Data for GPIO 32-63	R/W	32
	+ 0x10	GPIO_PAD_PU_EN0	GPIO Pad Pull-up Enable for GPIO 00-31	R/W	32
	+ 0x14	GPIO_PAD_PU_EN1	GPIO Pad Pull-up Enable for GPIO 32-63	R/W	32
	+ 0x18	GPIO_FUNC_SEL0	GPIO Function Select for GPIO 00-15	R/W	32
	+ 0x1C	GPIO_FUNC_SEL1	GPIO Function Select for GPIO 16-31	R/W	32
	+ 0x20	GPIO_FUNC_SEL2	GPIO Function Select for GPIO 32-47	R/W	32
	+ 0x24	GPIO_FUNC_SEL3	GPIO Function Select for GPIO 48-63	R/W	32
	+ 0x28	GPIO_DATA_SEL0	GPIO Data Select for GPIO 00-31	R/W	32
	+ 0x2C	GPIO_DATA_SEL1	GPIO Data Select for GPIO 32-63	R/W	32
	+ 0x30	GPIO_PAD_PU_SEL0	GPIO Pad Pull-up Select for GPIO 00-31	R/W	32
	+ 0x34	GPIO_PAD_PU_SEL1	GPIO Pad Pull-up Select for GPIO 32-63	R/W	32
	+ 0x38	GPIO_PAD_HYST_EN0	GPIO Pad Hysteresis Enable for GPIO 00-31	R/W	32
	+ 0x3C	GPIO_PAD_HYST_EN1	GPIO Pad Hysteresis Enable for GPIO 32-63	R/W	32
	+ 0x40	GPIO_PAD_KEEP0	GPIO Pad Keeper Enable for GPIO 00-31	R/W	32
	+ 0x44	GPIO_PAD_KEEP1	GPIO Pad Keeper Enable for GPIO 32-63	R/W	32
	+ 0x48	GPIO_DATA_SET0	GPIO Data Set for GPIO 00-31	W	32
	+ 0x4C	GPIO_DATA_SET1	GPIO Data Set for GPIO 32-63	W	32
	+ 0x50	GPIO_DATA_RESET0	GPIO Data Reset for GPIO 00-31	W	32
	+ 0x54	GPIO_DATA_RESET1	GPIO Data Reset for GPIO 32-63	W	32
	+ 0x58	GPIO_PAD_DIR_SET0	GPIO Pad Direction Set for GPIO 00-31	W	32
	+ 0x5C	GPIO_PAD_DIR_SET1	GPIO Pad Direction Set for GPIO 32-63	W	32
	+ 0x60	GPIO_PAD_DIR_RESET0	GPIO Pad Direction Reset for GPIO 00-31	W	32

Table A-1. Register Address Memory Map (continued)

Base Address	Register Address	Mnemonic	Name	Type	Width (Bits)
	+ 0x64	GPIO_PAD_DIR_RESET1	GPIO Pad Direction Reset for GPIO 32-63	W	32
0x8000_1000			SSI		
	+ 0x00	SSI_STX0	SSI Transmit Data register 0	R/W	32
	+ 0x04	SSI_STX1	SSI Transmit Data register 1	R/W	32
	+ 0x08	SSI_SRX0	SSI Receive Data register 0	R	32
	+ 0x0C	SSI_SRX1	SSI Receive Data register 1	R	32
	+ 0x10	SSI_SCR	SSI Control register	R/W	32
	+ 0x14	SSI_SISR	SSI Interrupt Status register	R	32
	+ 0x18	SSI_SIER	SSI Interrupt Enable register	R/W	32
	+ 0x1C	SSI_STCR	SSI Transmit Configuration register	R/W	32
	+ 0x20	SSI_SRCR	SSI Receive Configuration register	R/W	32
	+ 0x24	SSI_STCCR	SSI Transmit Clock Control register	R/W	32
	+ 0x28	SSI_SRCCR	SSI Receive Clock Control register	R/W	32
	+ 0x2C	SSI_SFCSR	SSI FIFO Control/Status register	R/W	32
	+ 0x30		Reserved		
	+ 0x34	SSI_SOR	SSI Option register	R/W	32
	+ 0x38	SSI_SACNT	SSI AC97 Control register	R/W	32
	+ 0x3C	SSI_SACADD	SSI AC97 Command Address register	R/W	32
	+ 0x40	SSI_SACDAT	SSI AC97 Command Data register	R/W	32
	+ 0x44	SSI_SATAG	SSI AC97 Tag register	R/W	32
	+ 0x48	SSI_STMSK	SSI Transmit Time Slot Mask register	R/W	32
	+ 0x4C	SSI_SRMSK	SSI Receive Time Slot Mask register	R/W	32
	+ 0x50	SSI_SACCST	SSI AC97 Channel Status register	R	32
	+ 0x54	SSI_SACCEN	SSI AC97 Channel Enable register	W	32
	+ 0x58	SSI_SACCDIS	SSI AC97 Channel Disable register	W	32
0x8000_2000			SPI		
	+ 0x00	SPI_TX_DATA	SPI Transmit Data register	R/W	32
	+ 0x04	SPI_RX_DATA	SPI Received Data register	R/W	32
	+ 0x08	SPI_CLK_CTRL	SPI Status and Clock Control register	R/W	32
	+ 0x0C	SPI_SETUP	SPI Setup register	R/W	32
	+ 0x10	SPI_STATUS	SPI Status register	R/W	32

Table A-1. Register Address Memory Map (continued)

Base Address	Register Address	Mnemonic	Name	Type	Width (Bits)
0x8000_3000			Clock/Reset/Power Module (CRM)		
	+ 0x00	SYS_CNTL	CRM System Control	R/W	32
	+ 0x04	WU_CNTL	CRM Wake up Control	R/W	32
	+ 0x08	SLEEP_CNTL	CRM Sleep Control	R/W	32
	+ 0x0C	BS_CNTL	CRM Bus Stealing Control	R/W	32
	+ 0x10	COP_CNTL	CRM COP Control	R/W	32
	+ 0x14	COP_SERVICE	CRM COP Service	R/W	32
	+ 0x18	STATUS	CRM Event Status	R/W	32
	+ 0x1C	MOD_STATUS	CRM Module Enable Status	R	32
	+ 0x20	WU_COUNT	CRM wake-up Count	R	32
	+ 0x24	WU_TIMEOUT	CRM wake-up Time-out	R/W	32
	+ 0x28	RTC_COUNT	CRM Real Time Count	R/W	32
	+ 0x2C	RTC_TIMEOUT	CRM RTC Periodic Wake up Time-out	R/W	32
	+ 0x30		Reserved		
	+ 0x34	CAL_CNTL	CRM Calibration Control	R/W	32
	+ 0x38	CAL_COUNT	CRM Calibration XTAL Count	R	32
	+ 0x3C	RINGOSC_CNTL	CRM 2kHz Ring Oscillator Control	R/W	32
	+ 0x40	XTAL_CNTL	CRM Reference XTAL Control	R/W	32
	+ 0x44	XTAL32_CNTL	CRM 32kHz XTAL Control	R/W	32
	+ 0x48	VREG_CNTL	CRM Voltage Regulator Control	R/W	32
	+ 0x4C		Reserved		
	+ 0x50	SW_RST	CRM Software Reset	R/W	32
	+ 0x54 to 0x5C		Reserved		
0x8000_4000			MAC Accelerator (MACA)		
	+ 0x00	MACA_VERSION	MACA Version Number	R	32
	+ 0x04	MACA_RESET	MACA Reset	R/W	32
	+ 0x08	MACA_RANDOM	MACA Random Number	R/W	32
	+ 0x0C	MACA_CONTROL	MACA Control	W	32
	+ 0x10	MACA_STATUS	MACA Status	R	32
	+ 0x14	MACA_FRMPND	MACA Frame Pending	R/W	32
	+ 0x18	MACA_FREQ	MACA Frequency Channel	R/W	32

Table A-1. Register Address Memory Map (continued)

Base Address	Register Address	Mnemonic	Name	Type	Width (Bits)
	+ 0x1C	MACA_EDVALUE	MACA Energy Detect Result	R	32
	+ 0x20 to 0x3C		Reserved		
	+ 0x40	MACA_TMREN	MACA Enable Timers	R/W	32
	+ 0x44	MACA_TMRDIS	MACA Disable Timers	R/W	32
	+ 0x48	MACA_CLK	MACA Clock	R/W	32
	+ 0x4C	MACA_STARTCLK	MACA Start Clock	R/W	32
	+ 0x50	MACA_CPLCLK	MACA Complete Clock	R/W	32
	+ 0x54	MACA_SFTCLK	MACA Soft Time-out Clock	R/W	32
	+ 0x58	MACA_CLKOFFSET	MACA Clock offset	R/W	32
	+ 0x5C	MACA_RELCLK	MACA Relative clock	R/W	32
	+ 0x60	MACA_CPLTIM	MACA Action Complete Timestamp	R	32
	+ 0x64	MACA_SLOTOFFSET	MACA Tx Slot Offset Adjustment	R/W	32
	+ 0x68	MACA_TIMESTAMP	MACA Receive Time Stamp	R	32
	+ 0x6C to 0x7C		Reserved		
	+ 0x80	MACA_DMARX	MACA DMA Rx Data Pointer	R/W	32
	+ 0x84	MACA_DMATX	MACA DMA Tx Data Pointer	R/W	32
	+ 0x88	MACA_DMAPOLL	MACA DMA Tx Poll Response Pointer	R/W	32
	+ 0x8C	MACA_TXLEN	MACA Tx Length	R/W	32
	+ 0x90	MACA_TXSEQNR	MACA Tx Sequence Number	R/W	32
	+ 0x94	MACA_SETRXLVL	MACA Set Rx Level Interrupt	R/W	32
	+ 0x98	MACA_GETRXLVL	MACA Read Number of Received Bytes	R	32
	+ 0x9C to 0xBC		Reserved		
	+ 0xC0	MACA_IRQ	MACA Interrupt	R	32
	+ 0xC4	MACA_CLRIRQ	MACA Interrupt Clear	W	32
	+ 0xC8	MACA_SETIRQ	MACA Interrupt Set	W	32
	+ 0xCC	MACA_MASKIRQ	MACA Interrupt Mask	W	32
	+ 0xD0 to 0xFC		Reserved		
	+ 0x100	MACA_MACPANID	MACA PAN ID	R/W	32
	+ 0x104	MACA_MAC16ADDR	MACA Short Address	R/W	32
	+ 0x108	MACA_MAC64HI	MACA High Extended Address	R/W	32

Table A-1. Register Address Memory Map (continued)

Base Address	Register Address	Mnemonic	Name	Type	Width (Bits)
	+ 0x10C	MACA_MAC64LO	MACA Low Extended Address	R/W	32
	+ 0x110	MACA_FLTREJ	MACA Filter Rejection Mask	R/W	32
	+ 0x114	MACA_CLKDIV	MACA Clock Divider	R/W	32
	+ 0x118	MACA_WARMUP	MACA Warmup	R/W	32
	+ 0x11C	MACA_PREAMBLE	MACA Preamble	R/W	32
	+ 0x120	MACA_WHITESEED	Reserved	R/W	32
	+ 0x124	MACA_FRAMESYNC0	MACA Frame Sync Word 0	R/W	32
	+ 0x128	MACA_FRAMESYNC1	MACA Frame Sync Word 1	R/W	32
	+ 0x12C to 0x13C		Reserved		
	+ 0x140	MACA_TXACKDELAY	MACA Tx Acknowledgement Delay	R/W	32
	+ 0x144	MACA_RXACKDELAY	MACA Rx Acknowledgement Delay	R/W	32
	+ 0x148	MACA_EOFDELAY	MACA End of Frame Delay	R/W	32
	+ 0x14C	MACA_CCADELAY	MACA CCA Delay	R/W	32
	+ 0x150	MACA_RXEND	MACA Rx End	R/W	32
	+ 0x154	MACA_TXCCADELAY	MACA Tx CCA Delay	R/W	32
	+ 0x158	MACA_KEY3	MACA Key3	R	32
	+ 0x15C	MACA_KEY2	MACA Key2	R	32
	+ 0x160	MACA_KEY1	MACA Key1	R	32
	+ 0x164	MACA_KEY0	MACA Key0	R	32
	+ 0x180	MACA_OPTIONS	MACA Options	W	32
0x8000_5000			UART1		
	+ 0x00	UART1_UCON	UART Control	R/W	8
	+ 0x04	UART1_USTAT	UART Status	R	8
	+ 0x08	UART1_UDATA	UART Data	R/W	8
	+ 0x0C	UART1_URXCON	UART RxBuffer Control	R/W	8
	+ 0x10	UART1_UTXCON	UART TxBuffer Control	R/W	8
	+ 0x14	UART1_UCTS	UART CTS Level Control	R/W	8
	+ 0x18	UART1_UBRCNT	UBRINC — Fractional Divider/ UBRMOD — Fractional Divider	R/W	8
0x8000_6000			I2C		
	+ 0x00	I2C_ADR	I2C Address	R/W	8

Table A-1. Register Address Memory Map (continued)

Base Address	Register Address	Mnemonic	Name	Type	Width (Bits)
	+ 0x04	I2C_FDR	I2C Frequency Divider	R/W	8
	+ 0x08	I2C_CR	I2C Control	R/W	8
	+ 0x0C	I2C_SR	I2C Status	R/W	8
	+ 0x10	I2C_DR	I2C Data	R/W	8
	+ 0x14	I2C_DFSRR	I2C Digital Filter Sampling Rate	R/W	8
	+ 0x18	I2C_CKER	I2C Clock Enable Register	R/W	8
0x8000_7000			TIMER		
	+ 0x00	TMR0_COMP1	TMR Channel 0 Compare Register #1	R/W	16
	+ 0x02	TMR0_COMP2	TMR Channel 0 Compare Register #2	R/W	16
	+ 0x04	TMR0_CAPT	TMR Channel 0 Capture Register	R/W	16
	+ 0x06	TMR0_LOAD	TMR Channel 0 Load Register	R/W	16
	+ 0x08	TMR0_HOLD	TMR Channel 0 Hold Register	R/W	16
	+ 0x0A	TMR0_CNTR	TMR Channel 0 Counter Register	R/W	16
	+ 0x0C	TMR0_CTRL	TMR Channel 0 Control Register	R/W	16
	+ 0x0E	TMR0_SCTRL	TMR Channel 0 Status and Control Register	R/W	16
	+ 0x10	TMR0_CMPLD1	TMR Channel 0 Comparator Load Register 1	R/W	16
	+ 0x12	TMR0_CMPLD2	TMR Channel 0 Comparator Load Register 2	R/W	16
	+ 0x14	TMR0_CSCTRL	TMR Channel 0 Comparator Status and Control Register	R/W	16
	+ 0x16 to + 0x1C		Reserved		
	+ 0x1E	TMR0_ENBL	TMR Channel Enable Register	R/W	16
	+ 0x20	TMR1_COMP1	TMR Channel 1 Compare Register #1	R/W	16
	+ 0x22	TMR1_COMP2	TMR Channel 1 Compare Register #2	R/W	16
	+ 0x24	TMR1_CAPT	TMR Channel 1 Capture Register	R/W	16
	+ 0x26	TMR1_LOAD	TMR Channel 1 Load Register	R/W	16
	+ 0x28	TMR1_HOLD	TMR Channel 1 Hold Register	R/W	16
	+ 0x2A	TMR1_CNTR	TMR Channel 1 Counter Register	R/W	16
	+ 0x2C	TMR1_CTRL	TMR Channel 1 Control Register	R/W	16
	+ 0x2E	TMR1_SCTRL	TMR Channel 1 Status and Control Register	R/W	16
	+ 0x30	TMR1_CMPLD1	TMR Channel 1 Comparator Load Register 1	R/W	16
	+ 0x32	TMR1_CMPLD2	TMR Channel 1 Comparator Load Register 2	R/W	16

Table A-1. Register Address Memory Map (continued)

Base Address	Register Address	Mnemonic	Name	Type	Width (Bits)
	+ 0x34	TMR1_CSCTRL	TMR Channel 1 Comparator Status and Control Register	R/W	16
	+ 0x36 to + 0x3C		Reserved		
	+ 0x40	TMR2_COMP1	TMR Channel 2 Compare Register #1	R/W	16
	+ 0x42	TMR2_COMP2	TMR Channel 2 Compare Register #2	R/W	16
	+ 0x44	TMR2_CAPT	TMR Channel 2 Capture Register	R/W	16
	+ 0x46	TMR2_LOAD	TMR Channel 2 Load Register	R/W	16
	+ 0x48	TMR2_HOLD	TMR Channel 2 Hold Register	R/W	16
	+ 0x4A	TMR2_CNTR	TMR Channel 2 Counter Register	R/W	16
	+ 0x4C	TMR2_CTRL	TMR Channel 2 Control Register	R/W	16
	+ 0x4E	TMR2_SCTRL	TMR Channel 2 Status and Control Register	R/W	16
	+ 0x50	TMR2_CMPLD1	TMR Channel 2 Comparator Load Register 1	R/W	16
	+ 0x52	TMR2_CMPLD2	TMR Channel 2 Comparator Load Register 2	R/W	16
	+ 0x54	TMR2_CSCTRL	TMR Channel 2 Comparator Status and Control Register	R/W	16
	+ 0x56 to + 0x5C		Reserved		
	+ 0x60	TMR3_COMP1	TMR Channel 3 Compare Register #1	R/W	16
	+ 0x62	TMR3_COMP2	TMR Channel 3 Compare Register #2	R/W	16
	+ 0x64	TMR3_CAPT	TMR Channel 3 Capture Register	R/W	16
	+ 0x66	TMR3_LOAD	TMR Channel 3 Load Register	R/W	16
	+ 0x68	TMR3_HOLD	TMR Channel 3 Hold Register	R/W	16
	+ 0x6A	TMR3_CNTR	TMR Channel 3 Counter Register	R/W	16
	+ 0x6C	TMR3_CTRL	TMR Channel 3 Control Register	R/W	16
	+ 0x6E	TMR3_SCTRL	TMR Channel 3 Status and Control Register	R/W	16
	+ 0x70	TMR3_CMPLD1	TMR Channel 3 Comparator Load Register 1	R/W	16
	+ 0x72	TMR3_CMPLD2	TMR Channel 3 Comparator Load Register 2	R/W	16
	+ 0x74	TMR3_CSCTRL	TMR Channel 3 Comparator Status and Control Register	R/W	16
	+ 0x76 to + 0x7E		Reserved		
0x8000_8000			Advanced Security Module (ASM)		
	+ 0x00	ASM_KEY0	128-BIT ENCRYPTION KEY (1 of 4)	R/W	32

Table A-1. Register Address Memory Map (continued)

Base Address	Register Address	Mnemonic	Name	Type	Width (Bits)
	+ 0x04	ASM_KEY1	128-BIT ENCRYPTION KEY (2 of 4)	R/W	32
	+ 0x08	ASM_KEY2	128-BIT ENCRYPTION KEY (3 of 4)	R/W	32
	+ 0x0C	ASM_KEY3	128-BIT ENCRYPTION KEY (4 of 4)	R/W	32
	+ 0x10	ASM_DATA0	128-BIT DATA Register (1 of 4)	R/W	32
	+ 0x14	ASM_DATA1	128-BIT DATA Register (2 of 4)	R/W	32
	+ 0x18	ASM_DATA2	128-BIT DATA Register (3 of 4)	R/W	32
	+ 0x1C	ASM_DATA3	128-BIT DATA Register (4 of 4)	R/W	32
	+ 0x20	ASM_CTR0	128-BIT COUNTER Register (1 of 4)	R/W	32
	+ 0x24	ASM_CTR1	128-BIT COUNTER Register (2 of 4)	R/W	32
	+ 0x28	ASM_CTR2	128-BIT COUNTER Register (3 of 4)	R/W	32
	+ 0x2C	ASM_CTR3	128-BIT COUNTER Register (4 of 4)	R/W	32
	+ 0x30	ASM_CTR0_RESULT	128-BIT COUNTER RESULT Register (1 of 4)	R	32
	+ 0x34	ASM_CTR1_RESULT	128-BIT COUNTER RESULT Register (2 of 4)	R	32
	+ 0x38	ASM_CTR2_RESULT	128-BIT COUNTER RESULT Register (3 of 4)	R	32
	+ 0x3C	ASM_CTR3_RESULT	128-BIT COUNTER RESULT Register (4 of 4)	R	32
	+ 0x40	ASM_CBC0_RESULT	128-BIT CBC MAC RESULT Register (1 of 4)	R	32
	+ 0x44	ASM_CBC1_RESULT	128-BIT CBC MAC RESULT Register (2 of 4)	R	32
	+ 0x48	ASM_CBC2_RESULT	128-BIT CBC MAC RESULT Register (3 of 4)	R	32
	+ 0x4C	ASM_CBC3_RESULT	128-BIT CBC MAC RESULT Register (4 of 4)	R	32
	+ 0x50	ASM_CONTROL0	CONTROL 0 Register (Self Clearing)	W	32
	+ 0x54	ASM_CONTROL1	CONTROL 1 Register	R/W	32
	+ 0x58	ASM_STATUS	STATUS Register	R	32
	+ 0x5C		Reserved		
	+ 0x60	ASM_MAC0	128-BIT CBC MAC Register (1 of 4)	R/W	32
	+ 0x64	ASM_MAC1	128-BIT CBC MAC Register (2 of 4)	R/W	32
	+ 0x68	ASM_MAC2	128-BIT CBC MAC Register (3 of 4)	R/W	32
	+ 0x6C	ASM_MAC3	128-BIT CBC MAC Register (4 of 4)	R/W	32
0x8000_9000			Modem Radio Functions		
0x8000_9000			Modem Clock Synthesizer		
See Note	+ 0x00	SYN_ENABLE	Enable and Override Register	R/W	32
	+ 0x04		Reserved		
	+ 0x08	SYN_REFDIV	Reference Loop Divider Register	R/W	32

Table A-1. Register Address Memory Map (continued)

Base Address	Register Address	Mnemonic	Name	Type	Width (Bits)
	+ 0x0C	SYN_VCODIV	VCO Loop Divider Register	R/W	32
	+ 0x10	SYN_VCOLOCK	VCO Lock Register	R/W	32
	+ 0x14 to + 0x28		Reserved		
0x8000_9200			Modem Transceiver Sequence Manager		
	+ 0x00 to + 0x1FC		Reserved		
0x8000_9400			Modem Radio Receiver Functions		
	+ 0x00 to + 0x14		Reserved		
	+ 0x18	RX_SFD_CTRL	RX SFD Control Register	R/W	32
	+ 0x1C to + 0x5C		Reserved		
	+ 0x60	RX_AGC_CCA_ED	RX AGC CCA and ED Control Register	R/W	32
	+ 0x64	RX_AGC_RSSI	RX AGC RSSI Parameters Register	R/W	32
	+ 0x68 to +0xFC		Reserved		
0x8000_9600			Modem Radio Transmitter Functions		
	+ 0x00 to +0x0C		Reserved		
0x8000_9800			Modem RF Synthesizer		
	+ 0x00	RFSYN_ENABLE	Enable and Override Register (Reserved)	R/W	32
	+ 0x04 to +0x08		Reserved		
	+ 0x0C	RFSYN_VCODIV_INTG	VCO Loop Divider Integer	R/W	32
	+ 0x10	RFSYN_VCODIV_FRAC	VCO Loop Divider Fractional	R/W	32
	+ 0x14	RFSYN_VCOLOCK	VCO Lock	R	32
	+ 0x18 to +0x38		Reserved		
0x8000_9A00			Modem Tracking Oscillator Controller (TOC)		
	+ 0x00 to 0x10		Reserved		
0x8000_A000			Radio Analog Functions		

Table A-1. Register Address Memory Map (continued)

Base Address	Register Address	Mnemonic	Name	Type	Width (Bits)
0x8000_A000			Radio Analog Write Functions		
	+ 0x00 to +0x04		Reserved		
	+ 0x20	PA_CTRL	Transmit Power Amp Control	W	32
	+ 0x24 to +0x50		Reserved		
	+ 0x54	PA_ADJ	Transmit Power Adjust	W	32
	+ 0x58 to +0x1BC		Reserved		
0x8000_A000			Radio Analog Read Functions		
	+ 0x1C0 to 0x1FC		Reserved		
0x8000_B000			UART2		
	+ 0x00	UART2_UCON	UART2 Control	R/W	8
	+ 0x04	UART2_USTAT	UART2 Status	R	8
	+ 0x08	UART2_UDATA	UART2 Data	R/W	8
	+ 0x0C	UART2_URXCON	UART2 RxBuffer Control	R/W	8
	+ 0x10	UART2_UTXCON	UART2 TxBuffer Control	R/W	8
	+ 0x14	UART2_UCTS	UART2 TxBuffer Control	R/W	8
	+ 0x18	UART2_UBRCNT	UBRINC — Fractional Divider/ UBRMOD — Fractional Divider	R/W	8
0x8000_C000			SPI FLASH Module (SPIF)		
	+ 0x00	SPIF_TX_DATA	SPIF Transmit Data	R/W	32
	+ 0x04	SPIF_RX_DATA	SPIF Received Data	R/W	32
	+ 0x08	SPIF_CLK_CTRL	SPIF Status and Clock Control	R/W	32
	+ 0x0C	SPIF_SETUP	SPIF Setup	R/W	32
	+ 0x10	SPIF_STATUS	SPIF Status	R/W	32
0x8000_D000			ADC		
	+ 0x00	ADC_COMP_0	Monitor GP-ADC Pins Threshold detection w/o MCU intervention to generate IRQ	R/W	16
	+ 0x02	ADC_COMP_1	Monitor GP-ADC Pins Threshold	R/W	16
	+ 0x04	ADC_COMP_2	Monitor GP-ADC Pins Threshold	R/W	16
	+ 0x06	ADC_COMP_3	Monitor GP-ADC Pins Threshold	R/W	16

Table A-1. Register Address Memory Map (continued)

Base Address	Register Address	Mnemonic	Name	Type	Width (Bits)
	+ 0x08	ADC_COMP_4	Monitor GP-ADC Pins Threshold	R/W	16
	+ 0x0A	ADC_COMP_5	Monitor GP-ADC Pins Threshold	R/W	16
	+ 0x0C	ADC_COMP_6	Monitor GP-ADC Pins Threshold	R/W	16
	+ 0x0E	ADC_COMP_7	Monitor GP-ADC Pins Threshold	R/W	16
	+ 0x10	ADC_BAT_COMP_OVER	Battery Voltage Upper Trip Point (ADC1 only)	R/W	16
	+ 0x12	ADC_BAT_COMP_UNDER	Battery Voltage Lower Trip Point (ADC1 only)	R/W	16
	+ 0x14	ADC_SEQ_1	Monitor GP-ADC Pins for ADC1	R/W	16
	+ 0x16	ADC_SEQ_2	Monitor GP-ADC Pins for ADC2	R/W	16
	+ 0x18	ADC_CONTROL	Primary Module Control Register	R/W	16
	+ 0x1A	ADC_TRIGGERS	Triggered Channels Register	R	16
	+ 0x1C	ADC_PRESCALE	Bus Clock Divide Register (8-bit prescaler for use with 26MHz)	R/W	16
	+ 0x1E		Reserved		
	+ 0x20	ADC_FIFO_READ	ADC FIFO Read (8 deep FIFO)	R	16
	+ 0x22	ADC_FIFO_CONTROL	ADC Interrupt Level Control	R/W	16
	+ 0x24	ADC_FIFO_STATUS	ADC FIFO Status	R	16
	+ 0x26 to 0x2E		Reserved		
	+ 0x30	ADC_1_SR_HIGH	ADC 1 Sample Rate Low	R/W	16
	+ 0x32	ADC_1_SR_LOW	ADC 1 Sample Rate High	R/W	16
	+ 0x34	ADC_2_SR_HIGH	ADC 2 Sample Rate Low	R/W	16
	+ 0x36	ADC_2_SR_LOW	ADC 2 Sample Rate High	R/W	16
	+ 0x38	ADC_ON_TIME	ADC TurnOn Time	R/W	16
	+ 0x3A	ADC_CONVERT_TIME	ADC Convert Time	R/W	16
	+ 0x3C	ADC_CLOCK_DIV	ADC Clock Divider	R/W	16
	+ 0x3E		Reserved		
	+ 0x40	ADC_OVERRIDE	ADC Manual Control	R/W	16
	+ 0x42	ADC_IRQ	ADC Read/Clear Active Interrupts	R/W	16
	+ 0x44	ADC_MODE	ADC Mode	R/W	16
	+ 0x46	ADC_1_RESULT	ADC 1 Result	R	16
	+ 0x48	ADC_2_RESULT	ADC 2 Result	R	16
0x8002_0000			Interrupt Controller		
	+ 0x00	INTCNTL	ITC Interrupt Controller	R/W	32

Table A-1. Register Address Memory Map (continued)

Base Address	Register Address	Mnemonic	Name	Type	Width (Bits)
	+ 0x04	NIMASK	ITC Normal Interrupt Mask	R/W	32
	+ 0x08	INTENNUM	ITC Interrupt Enable Number	W	32
	+ 0x0c	INTDISNUM	ITC Interrupt Disable Number	W	32
	+ 0x10	INTENABLE	ITC Interrupt Enable	R/W	32
	+ 0x14	INTTYPE	ITC Interrupt Type	R/W	32
	+ 0x18		Reserved	N/A	32
	+ 0x1C		Reserved	N/A	32
	+ 0x20		Reserved	N/A	32
	+ 0x24		Reserved	N/A	32
	+ 0x28	NIVECTOR	ITC Normal Interrupt	R	32
	+ 0x2C	FIVECTOR	ITC Fast Interrupt	R	32
	+ 0x30	INTSRC	ITC Interrupt Source	R	32
	+ 0x34	INTFRC	ITC Interrupt Force	R/W	32
	+ 0x38	NIPEND	ITC Normal Interrupt Pending	R	32
	+ 0x3C	FIPEND	ITC Fast Interrupt Pending	R	32

Appendix B

MC1322x Software Driver Utilities

B.1 Overview

The MC1322x is a full Platform-in-Package device having a complex set of transceiver and MCU modules and peripherals. To simplify software development, Freescale has written and provides an extensive set of driver utilities for the platform.

B.2 Driver Summary

The drivers are documented in the *MC1322x Software Driver Reference Manual*, (22XDVRRM). Those drivers described in the reference manual include:

- Clock and Reset Module (CRM) Driver
- GPIO Drivers
- Interrupt Controller (ITC) Driver
- Non-Volatile Memory (NVM) Driver
- UART Driver
- Timer (TMR) Driver
- Inter-integrated Circuit (I2C) Driver
- Synchronous Serial Interface (SSI) Driver
- Analog to Digital Converter (ADC) Driver

NOTE

The user is directed to the *MC1322x Software Driver Reference Manual* for information on calling and utilizing the driver functions.

B.3 Using the Drivers

The MC1322x drivers are written in C-code and are provided as library functions. The Freescale BeeKit development suite allows access to the drivers, and the library functions are available either through MC1322x onboard ROM or through compiled library functions that reside in RAM as executables.

As the MC1322x platform matures, more drivers/utilities will become available and will be moved into ROM space to make more RAM space available for application.

Appendix C

Bootloader Reference

C.1 Overview

The MC1322x has a ROM-based bootloader that is described in [Section 3.9, “Bootloader”](#). Common usage employs the onboard FLASH and the file image format used is transparent to the user. The FLASH image is generated through development tools and loaded through the debug port, typically the JTAG port.

The bootloader does allow alternative means of supplying the RAM boot image. This appendix provides format and protocol information to utilize these other sources for the RAM target code image.

C.2 Alternative Load Ports

The bootstrap flow chart as shown in [Figure 3-15](#) and its accompanying explanation describes use of the following boot ports:

- Load the target image from UART1 port
- Load the target image from the SPI port (as slave) attached to a master device
- Initiate the SPI port as a master and load target image from external slave (serial FLASH)
- Load the target image from I²C (serial EEPROM)

C.3 Booting from UART1

If it has been determined that onboard FLASH does not contain valid code, the bootstrap will first try to boot from UART1.

C.3.1 Procedure

The UART1 boot procedure includes:

1. The UART1 signals get initiated as:
 - GPIO14 becomes UART1_TX output
 - GPIO15 becomes UART1_RX input
 - GPIO16 becomes UART1_CTS output
 - GPIO17 becomes UART1_RTS input
2. Control input UART1_RTS is tested and must be driven low. This indicates a device is attached ready to supply data.

NOTE

If UART1_RTS is tested and is not driven low, the procedure exits and moves to the SPI boot flow, [Section C.4, “Booting using SPI on the MC1322x”](#).

3. Auto-baud Detection - In order to properly boot with any baudrate selected by the attached host (up to 2M baud), the bootstrap determines the host baudrate based on a “zero” character.
 - The host must first transmit a single ASCII numeric “zero” byte. The host can repeat this single byte transmission until the MC1322x responds.
 - Once the MC1322x has determined the baud rate, it responds back with the message “CONNECT”. This serves two purposes:
 - Confirms the baud rate
 - Notifies the host device that it can now continue.
4. The host sends the length field and RAM data in the format defined in [Section C.3.2, “Data Format”](#).

[Figure C-1](#) shows the message sequence chart for the interaction between the host and the MC1322x.

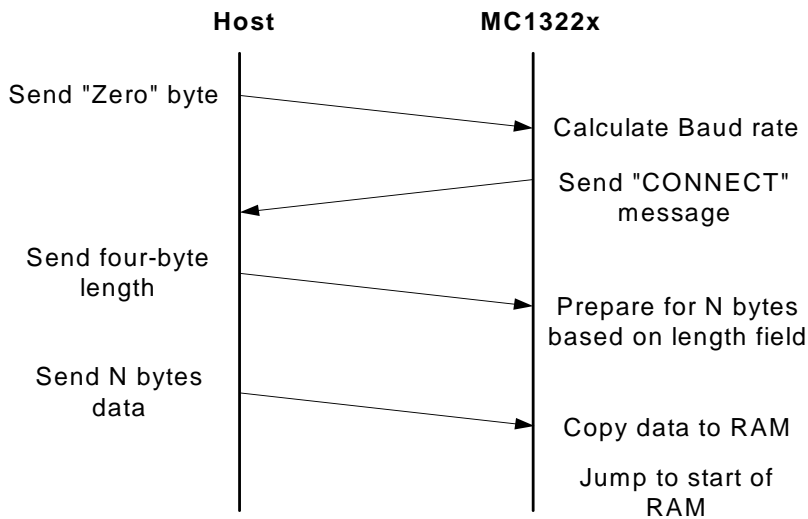


Figure C-1. Host/MC1322x UART1 Boot Sequence

C.3.2 Data Format

After completing the connection sequence (transmitting the “CONNECT” message):

1. The MC1322x first expects a four-byte length field (32-bit little-endian) from the host. This four-byte value determines the number of expected bytes from the host that will get loaded into RAM starting at address 0x0040_0000.
2. The host then sends the stream of bytes which are loaded into RAM (also little-endian).

After the MC1322x gets the expected number of bytes, the bootloader jumps to the RAM start address to begin program execution.

C.4 Booting using SPI on the MC1322x

If UART1_RTS is tested and is not driven low, the procedure exits UART1 consideration and moves to the SPI boot flow. At this point the SPI_SS is initiated as an input and tested for its state:

- If SPI_SS is low - It is assumed an external SPI master is present, the MC1322x configures its SPI to slave mode, and waits for the SPI master to clock the SPI (SPI_SS is asserted).
- If SPI_SS is high - It is assumed that an external SPI slave is present (typically a serial FLASH device), the MC1322x configures its SPI to master mode, and tries to boot from the external SPI device.

NOTE

SPI data is always transferred as bytes with MSB first.

C.4.1 MC1322x Is SPI Slave

The MC1322x SPI slave boot procedure includes:

1. The SPI signals are initiated as:
 - GPIO4 is SPI_SS input
 - GPIO5 is SPI_MISO output
 - GPIO6 is SPI_MOSI input
 - GPIO77 is SPI_SCK input
2. SPI_SS is presently asserted (low) and the MC1322x SPI gets configured as a slave and waits for the external SPI master to clock the port to transfer data (8-bit transfers are assumed). Also during configuration, the slave data buffer gets loaded with the four ASCII bytes for “RDY!”
3. The master transfers four bytes; sending the binary equivalent of the four-byte length field (32-bit little endian) and receiving the “RDY!” message.
4. The master continues to transfer the RAM data.

Figure C-2 shows the message sequence chart for the interaction between the SPI master and the MC1322x.

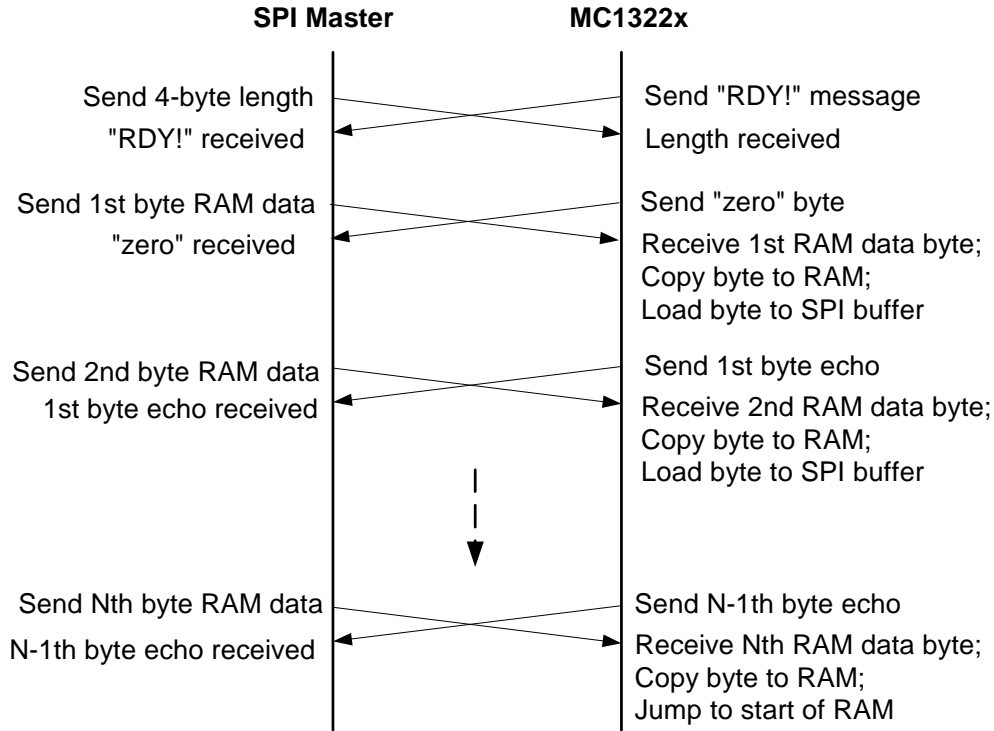


Figure C-2. SPI Master /MC1322x Boot Sequence

C.4.2 MC1322x Is SPI Slave Data Format

The exchange format is as follows:

1. Complete the connection sequence - Master transmits the length field and receives “RDY!” from slave (four ASCII characters “R”, “D”, “Y”, “!” in the order shown from left to right)
2. The MC1322x as Slave loads the data buffer with a binary zero (byte = 0x00) in anticipation of receiving the RAM data stream.
3. The external master sends the first RAM data byte and this zero-data is transmitted to the master.
4. On each subsequent byte received from the master, the MC1322x will transmit the previously received byte. This is used as a simple "acknowledgment", and validates the just received data to be correct.
5. The external master sends the expected number of bytes as determined by the length field

The bootloader finishes loading the RAM, and then jumps to the RAM start address to begin program execution.

NOTE

If SPI master fails to send expected number of bytes, the MC1322x will “hang” waiting for the RAM data transfer to complete.

C.4.3 MC1322x Is SPI Master (Boot from External FLASH)

If the SPI_SS has not tested as low, the next trial assumes that an external SPI slave is present (typically a serial FLASH device)

NOTE

The MC1322x supports FLASH memories formatted similar to the devices below:

- 1 Mbit, serial FLASH memory M25P10-A from ST Microelectronics
- 1 Mbit, serial FLASH memory AT25F1024 from Atmel Corp

The MC1322x as SPI master boot procedure includes:

1. The SPI signals are initiated as:
 - GPIO4 is SPI_SS input
 - GPIO5 is SPI_MISO input
 - GPIO6 is SPI_MOSI output
 - GPIO7 is SPI_SCK output
2. The MC1322x configures the onboard SPI as master and asserts SPI_SS (low).
3. The MC1322x reads the first four bytes from the external SPI slave and validates that the contents are the ASCII string “OKOK” (addr 0x0 = “O”, addr 0x1 = “K”, addr 0x2 = “O”, addr 0x3 = “K”)

NOTE

If any other content is read, the FLASH is assumed to be corrupted or not present. The boot flow then moves to look for an I²C EEPROM.

4. Upon reading the validation code, the bootstrap will load the subsequent four bytes (32-bit, little-endian), and use this as the length field indicator for how many bytes to load into RAM.
5. The MC1322x continues reading bytes from FLASH and loading them into RAM starting with the first RAM address (0x0040_0000).

Once all code is loaded into RAM, the bootstrap will jump to the first RAM address (0x00400000) and continue from there. [Figure C-3](#) shows the interaction between SPI master and external FLASH.

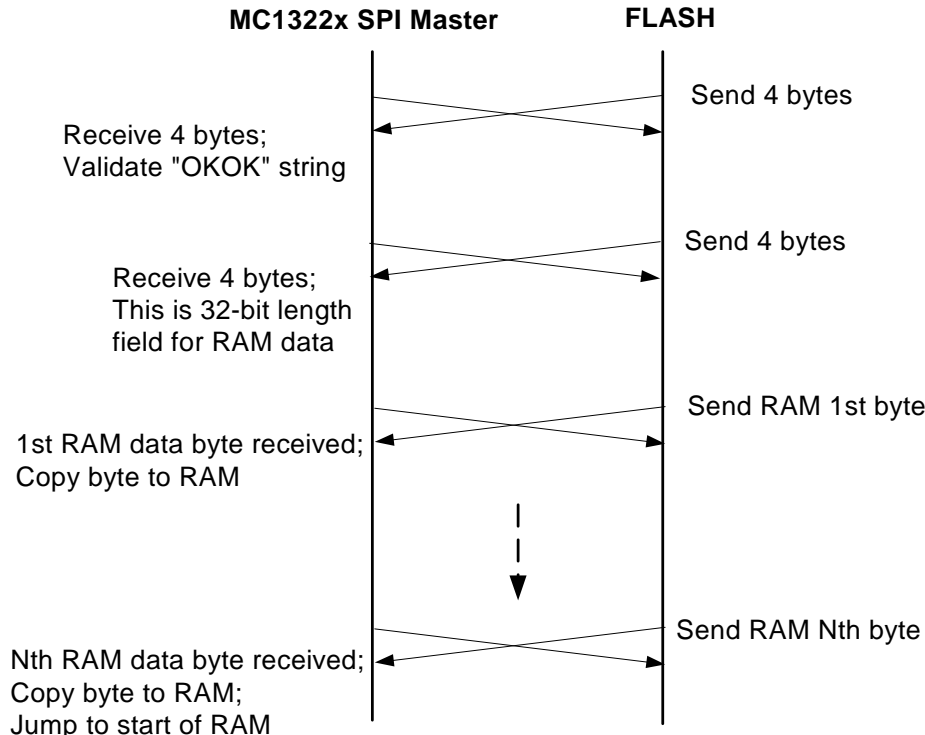


Figure C-3. MC1322x SPI Master and external FLASH Boot Sequence

C.4.4 MC1322x Is SPI Master Data Format

Similar to other sequences, the 4-byte length field and RAM data are supplied in little-endian format.

C.5 Booting from I²C (Boot from External Serial EEPROM)

If it has been determined that an external FLASH does not contain valid code (or is non-existent), the bootstrap will finally try to boot from a serial EEPROM through the I²C port. The port is configured as bus master

C.5.1 Procedure

The I²C boot procedure includes:

1. The I²C signals get initiated as:
 - GPIO12 becomes I2C_SCL input/output
 - GPIO13 becomes I2C_SDA input/output
2. The MC1322x configures the I²C as master.
3. The MC1322x reads the first four bytes from the external I²C slave (EEPROM) and validates that the contents are the ASCII string "OKOK" (addr 0x0 = "O", addr 0x1 = "K", addr 0x2 = "O", addr 0x3 = "K")

NOTE

- The first byte of the I²C slave address is the designated as the Device Select Code or Device Address. The MC1322x addresses the EEPROM with the following described content. **The EEPROM must respond to this addressing byte.**

It consists of the following fields:

- Device Type Identifier [B7:B4] = 4b1010; for a serial EEPROM
 - Lower device address or chip enable [B3:B1] = 3b000; device must be wired to be addressed at this lowest address.
 - R/W bit [B0]; programmed as read
- If any other content is read, the EEPROM is assumed to be corrupted or not present. The boot flow then moves to an endless loop. Either a debug port intervention or reset is required to recover control.
4. Upon reading the validation code, the bootstrap will load the subsequent four bytes (32-bit, little-endian), and use this as the length field indicator for how many bytes to load into RAM.
 5. The MC1322x continues reading bytes from FLASH and loading them into RAM starting with the first RAM address (0x0040_0000).

Once all code is loaded into RAM , the bootstrap will jump to the first RAM address (0x00400000) and continue from there.

C.5.2 MC1322x Is I²C Master Data Format

Similar to other sequences, the 4-byte length field and RAM data are supplied in little-endian format.

C.6 Information Available to the User Program

After the boot process completes and the user program starts to run, the following information is available in the CPU registers:

- The boot source in r0 - The boot source can take one of the values:
 - 0 for secured internal flash
 - 1 for non-secured internal flash
 - 2 for UART
 - 3 for SPI slave
 - 4 for SPI master
 - 5 for I²C
- The program size in r1

NOTE

Bootstrap does not change the ARM mode. The boot source and the program source and size will be returned in r0 and r1 of the current ARM mode. So, if the boot process occurs after a POR event the ARM will be in Supervisor mode. If the boot process occurs after a jump to 0x00000000 event, the ARM will be in the mode set before the jump instruction.