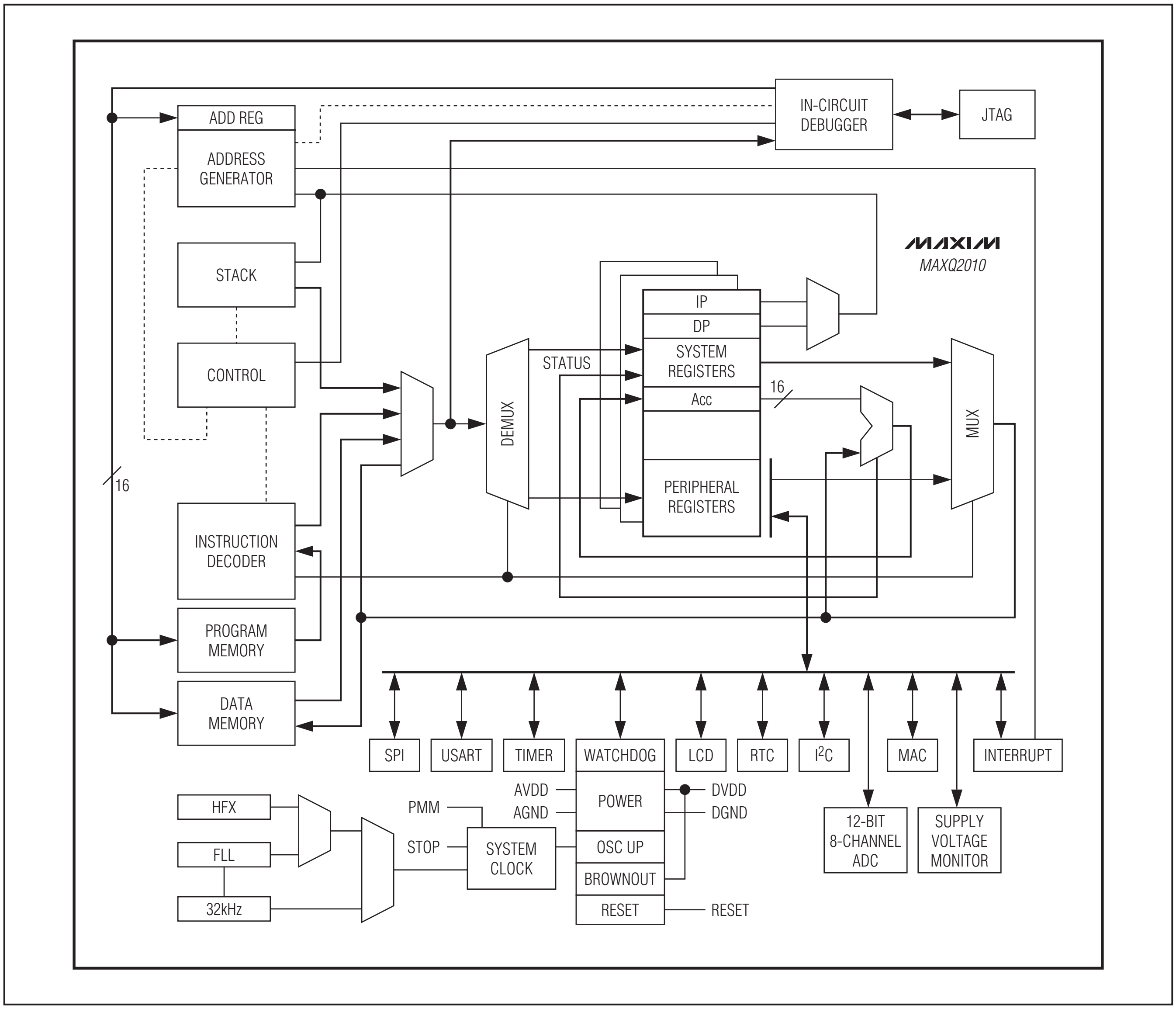
Rev 0; 6/09

MAXQ FAMILY USER'S GUIDE: MAXQ2010 SUPPLEMENT



For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim's website at www.maxim-ic.com.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES (continued)

ADDENDUM TO SECTION 1: OVERVIEW

This document is provided as a supplement to the *MAXQ Family User's Guid*e, covering new or modified features specific to the MAXQ2010. This document must be used in conjunction with the *MAXQ Family User's Guid*e, available on our website at **www.maxim-ic.com/MAXQUG**. Addenda are arranged by section number, which correspond to sections in the *MAXQ Family User's Guid*e. Additions and changes, with respect to the *MAXQ Family User's Guide*, are contained in this document, and updates/additions are added when available.

The MAXQ2010 is a low-power, high-performance, 16-bit, RISC microcontroller based on the MAXQ® architecture design. It includes support for integrated, in-system-programmable, flash memory and a wide range of peripherals including an 8-channel, 12-bit successive-approximation analog-to-digital converter (SAR ADC) and an LCD driver supporting up to 1/4-duty multiplexed displays. The MAXQ2010 is uniquely suited for any application that requires high performance and low-power operation.

1.1 References

Refer to the *MAXQ Family User's Guide* for the following information:

- Description of the core architecture, instruction set, and memory mapping common to all MAXQ microcontrollers.
- Definitions and functions of the common system register set, including accumulators, data pointers, loop counters, and general-purpose registers.
- Descriptions of common clock generation, interrupt handling, and reset/power-management modes.
- Descriptions and programming examples for common MAXQ peripherals found on the MAXQ2010 including the serial universal synchronous/asynchronous receiver-transmitter (USART), SPI™ interface, and hardware multiplier.
- Description of the test access port (TAP) and in-circuit debug interface.
- Description of the in-system programming mode.

The MAXQ2010 data sheet, which contains electrical/timing specifications and pin descriptions, is available at **www.maxim-ic.com/MAXQ2010**.

Errata sheets for the MAXQ2010 and other MAXQ micros are available at **www.maxim-ic.com/errata**.

For more information on other MAXQ microcontrollers, development hardware and software, frequently asked questions, and software examples, visit the MAXQ page at **www.maxim-ic.com/MAXQ**.

MAXQ is a registered trademark of Maxim Integrated Products, Inc.

SPI is a trademark of Motorola, Inc.

ADDENDUM TO SECTION 2: ARCHITECTURE

The MAXQ2010 shares the common architecture features with other members of the MAXQ microcontroller family. Details are discussed in the following sections.

2.1 Instruction Set

This device uses the standard 16-bit MAXQ20 core instruction set as described in the *MAXQ Family User's Guid*e.

2.2 Harvard Memory Architecture

Program memory, data memory, and register space follow the Harvard architecture model. Each type of memory is kept separate and is accessed by a separate bus, allowing different word lengths for different types of memory. Registers can be either 8 bits or 16 bits in width. Program memory is 16 bits in width to accommodate the standard MAXQ20 16-bit instruction set. Data memory is also 16 bits in width, but can be accessed in 8-bit or 16-bit modes for maximum flexibility.

The MAXQ2010 includes a flexible memory-management unit (MMU) that allows code to be executed from either the program flash, the utility ROM, or the internal data SRAM. Any of these three memory spaces can also be accessed in data space at any time, with the single restriction that the physical memory area that is currently being used as program space cannot be simultaneously read from in data space. In the event that it is necessary to read data from the program segment that is currently in use (for example, when executing code from program flash that utilizes a lookup table also located in program flash), standardized data transfer functions provided in the utility ROM can be used to do so. See *Section 24: Utility ROM* for more details.

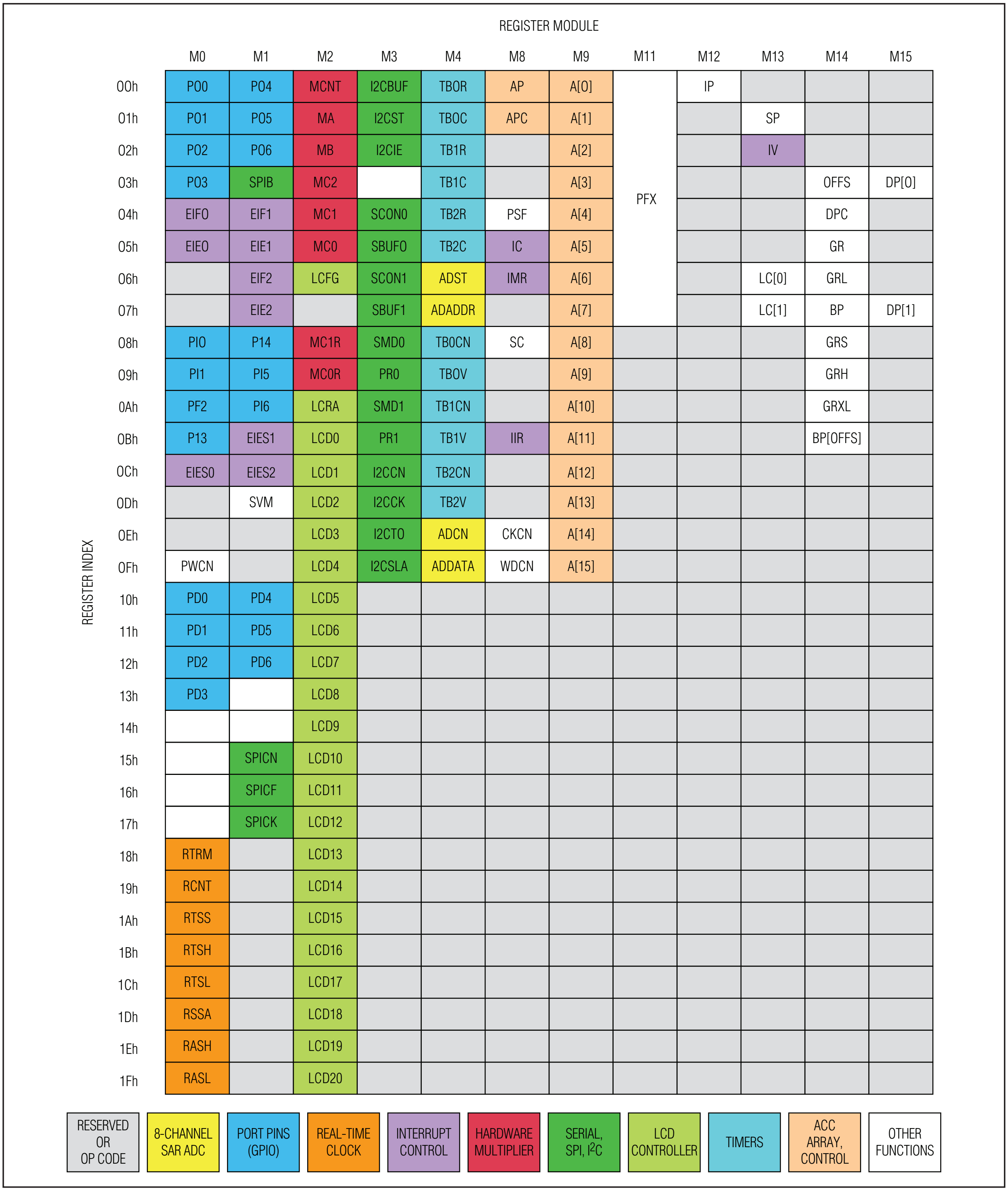
2.3 Register Space

The MAXQ2010 contains the standard set of MAXQ20 system registers as described in the *MAXQ Family User's Guide*, but with differences noted in this guide where they exist.

Peripheral register space (modules 0 to 4) contains registers that are used to access the following peripherals:

- 12-bit SAR ADC converter with up to eight single-ended or four differential input channels
- General-purpose 8-bit I/O ports (P0 to P6)
- External interrupts (up to 23)
- Three programmable Type B timer/counters
- Two serial USART interfaces
- I^{2}C interface
- SPI interface
- Hardware multiplier
- Real-time clock (RTC)
- LCD controller

The lower 8 bits of all registers in modules 0 to 4 (as well as the AP module M8) are bit addressable.



REGISTER MODULE

REGISTER INDEX

	M0	M1	M2	M3	M4	M8	M9	M11	M12	M13	M14	M15
00h	P00	P04	MCNT	I2CBUF	TB0R	AP	A[0]	PFX	IP			
01h	P01	P05	MA	I2CST	TB0C	APC	A[1]			SP		
02h	P02	P06	MB	I2CIE	TB1R		A[2]			IV		
03h	P03	SPIB	MC2		TB1C		A[3]				OFFS	DP[0]
04h	EIF0	EIF1	MC1	SCON0	TB2R	PSF	A[4]				DPC	
05h	EIE0	EIE1	MC0	SBUF0	TB2C	IC	A[5]				GR	
06h		EIF2	LCFG	SCON1	ADST	IMR	A[6]			LC[0]	GRL	
07h		EIE2		SBUF1	ADADDR		A[7]			LC[1]	BP	DP[1]
08h	PI0	P14	MC1R	SMD0	TB0CN	SC	A[8]				GRS	
09h	PI1	PI5	MC0R	PR0	TB0V		A[9]				GRH	
0Ah	PF2	PI6	LCRA	SMD1	TB1CN		A[10]				GRXL	
0Bh	P13	EIES1	LCD0	PR1	TB1V	IIR	A[11]				BP[OFFS]	
0Ch	EIES0	EIES2	LCD1	I2CCN	TB2CN		A[12]					
0Dh		SVM	LCD2	I2CCK	TB2V		A[13]					
0Eh			LCD3	I2CTO	ADCN	CKCN	A[14]					
0Fh	PWCN		LCD4	I2CSLA	ADDATA	WDCN	A[15]					
10h	PD0	PD4	LCD5									
11h	PD1	PD5	LCD6									
12h	PD2	PD6	LCD7									
13h	PD3		LCD8									
14h			LCD9									
15h		SPICN	LCD10									
16h		SPICF	LCD11									
17h		SPICK	LCD12									
18h	RTRM		LCD13									
19h	RCNT		LCD14									
1Ah	RTSS		LCD15									
1Bh	RTSH		LCD16									
1Ch	RTSL		LCD17									
1Dh	RSSA		LCD18									
1Eh	RASH		LCD19									
1Fh	RASL		LCD20									

RESERVED OR OP CODE	8-CHANNEL SAR ADC	PORT PINS (GPIO)	REAL-TIME CLOCK	INTERRUPT CONTROL	HARDWARE MULTIPLIER	SERIAL, SPI, I^{2}C	LCD CONTROLLER	TIMERS	ACC ARRAY, CONTROL	OTHER FUNCTIONS

Figure 2-1. MAXQ2010 System and Peripheral Register Map

2.4 Memory Organization

As with all MAXQ microcontrollers, the MAXQ2010 contains logically separate program and data memory spaces. All memory is internal, and physical memory segments (other than the stack and register memories) can be accessed as either program memory or as data memory, but not both at once.

The MAXQ2010 contains the following physical memory segments.

2.4.1 Register Space

As described in the *MAXQ Family User's Guide*, register space on MAXQ microcontrollers consists of 16 register modules, each of which could contain up to 32 registers. Of these possible 16 register modules, only 12 are used on the MAXQ2010: seven for system registers and five for peripheral registers.

2.4.2 Program Stack

The MAXQ2010 provides a 16 x 16 hardware stack to support subroutine calls and system interrupts. This stack is used automatically by CALL/RET and PUSH/POP instructions, and can also be accessed directly through the SP register as described in the *MAXQ Family User's Gui*de.

When using the in-circuit debugging features, one word of the stack must be reserved to store the return location when execution branches into the debugging routines in the utility ROM. If in-circuit debug is not used, the entire stack is available for application use.

2.4.3 Data SRAM

The MAXQ2010 contains up to 1024 words (2KB) of on-chip data SRAM, which can be mapped into either program or data space. The contents of this SRAM are indeterminate after power-on reset, but are maintained during stop mode and across non-POR resets.

When using the in-circuit debugging features, the highest 19 bytes of the SRAM must be reserved for saved state storage and working space for the debugging routines in the utility ROM. If in-circuit debug is not used, the entire SRAM is available for application use.

2.4.4 Program Flash

The MAXQ2010 contains 32KWords (32K x 16) of flash memory, which normally serves as program memory. When executing from the data SRAM or utility ROM, this memory is mapped to data space (as 32KWords or 64KB) and can be used for lookup tables and similar functions.

Since program memory is mapped into data space starting at address 8000h, only half the available program memory can be mapped into data space at one time when operating in byte-access mode. The CDA0 (code data access) bit is used to control which half of program memory is available in data space as shown in Figure 2-3 and Figure 2-4, and as described in the *MAXQ Family User's Guid*e. When operating in word-access mode, the entire 32KWord program memory can be mapped into data space at once.

Flash memory mapped into data space can be read from directly, like any other type of data memory. However, writing to flash memory must be done indirectly by calling the in-application functions provided by the utility ROM. See *Section 24: Utility ROM* for more details.

2.5 Program and Data Memory Mapping

Figures 2-2, 2-3, and 2-4 show the mapping of physical memory segments into the program and data memory space. The mapping of memory segments into program space is always the same. The mapping of memory segments into data space varies depending on which memory segment is currently being executed from.

In all cases, whichever memory segment is currently being executed from in program space may not be accessed in data space.

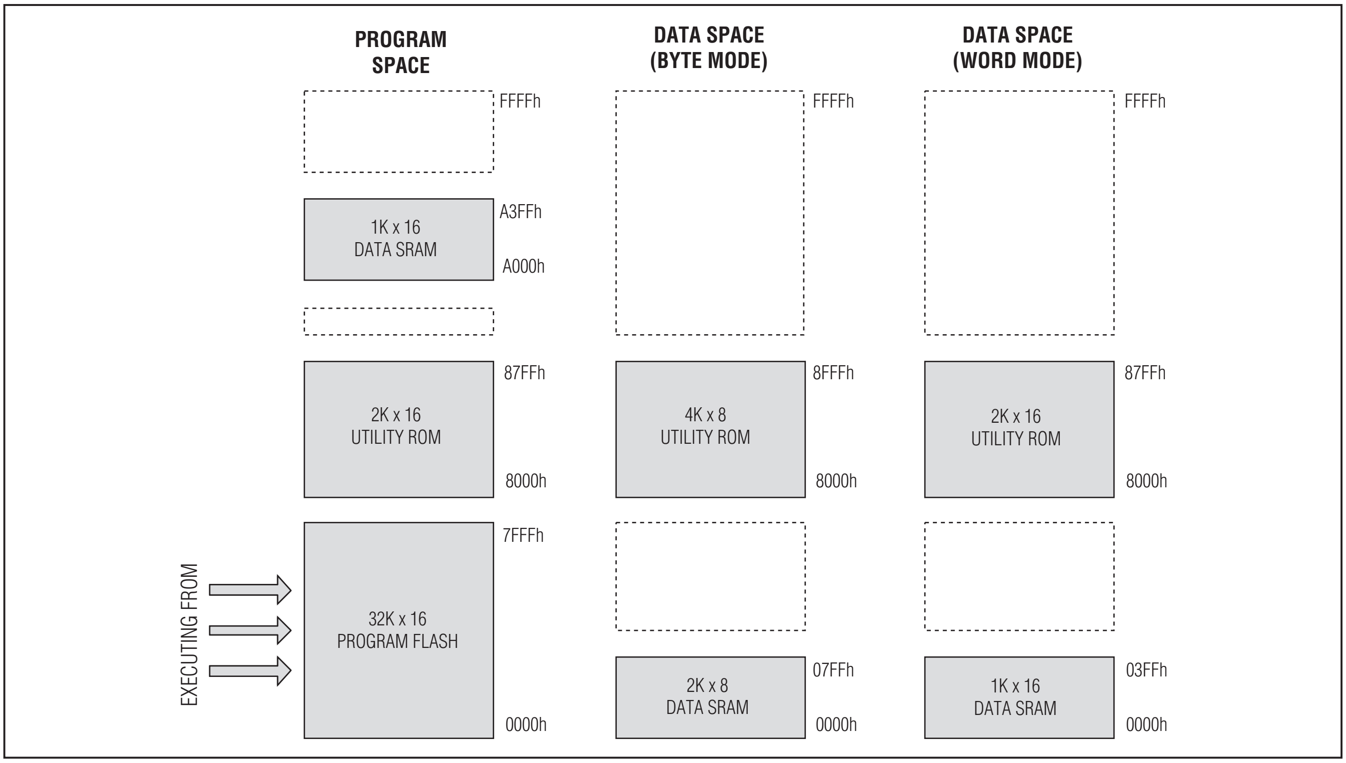


Figure 2-2. Memory Map When Executing from Program Flash Memory

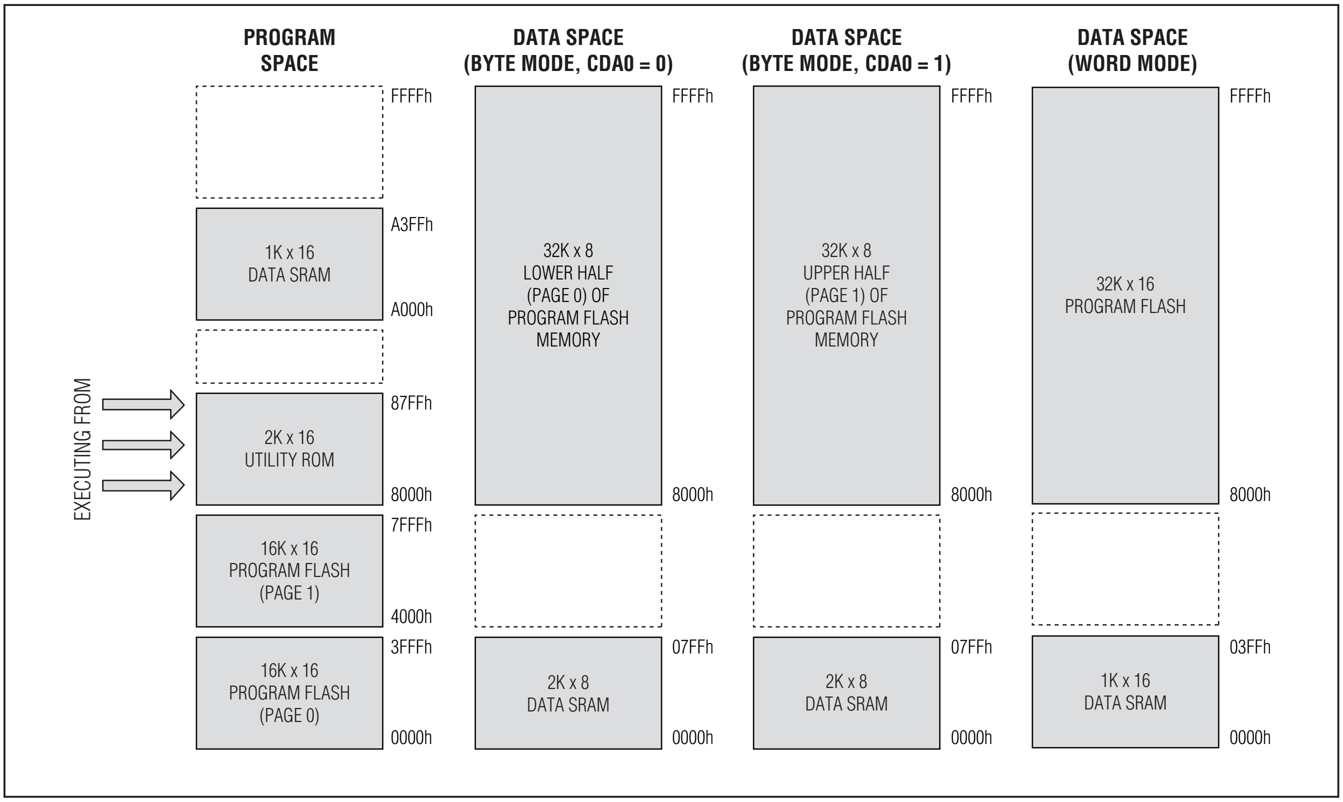


Figure 2-3. Memory Map When Executing from Utility ROM

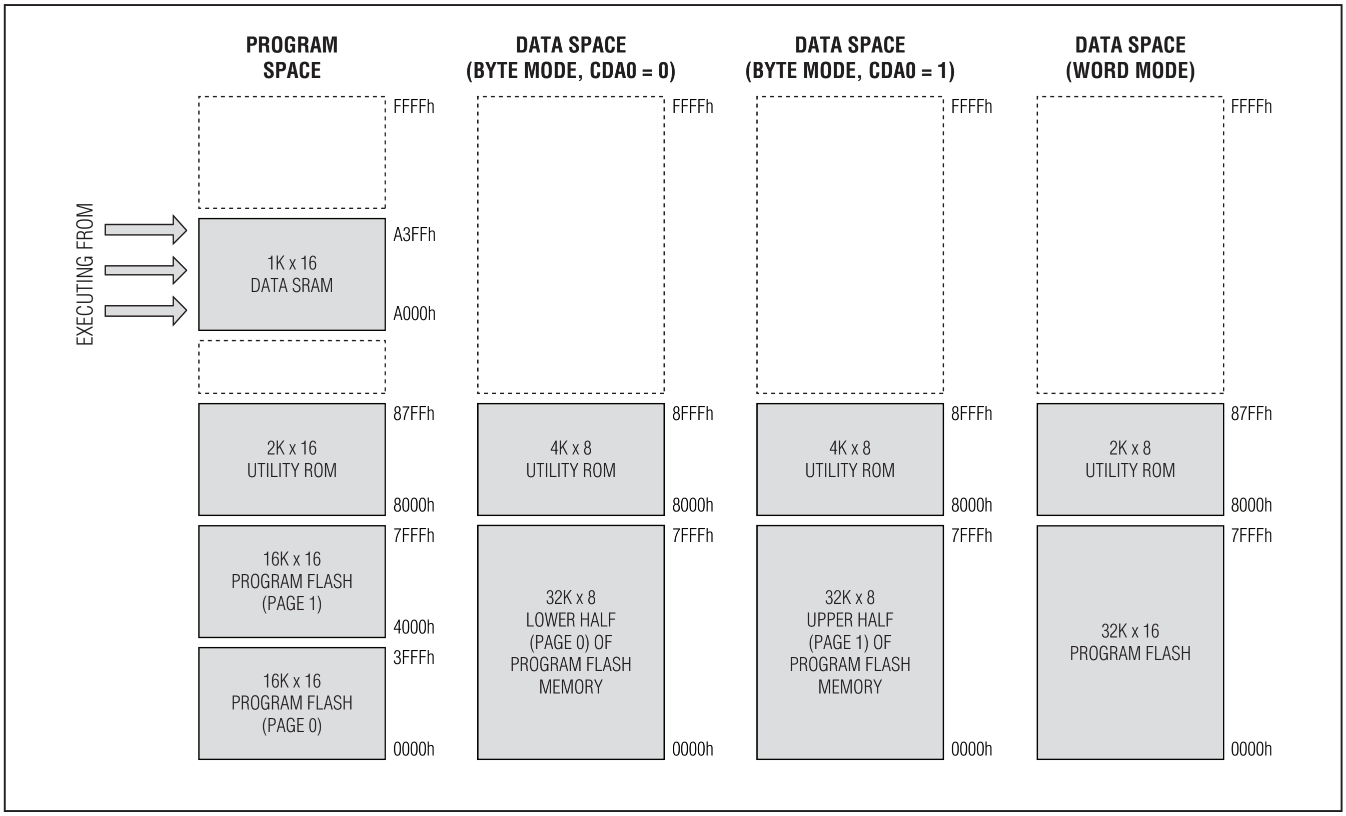

PROGRAM SPACE
DATA SPACE (BYTE MODE, CDA0 = 0)
DATA SPACE (BYTE MODE, CDA0 = 1)
DATA SPACE (WORD MODE)
EXECUTING FROM
FFFFh
A3FFh
1K x 16 DATA SRAM
A000h
87FFh
2K x 16 UTILITY ROM
8000h
7FFFh
16K x 16 PROGRAM FLASH (PAGE 1)
4000h
3FFFh
16K x 16 PROGRAM FLASH (PAGE 0)
0000h
8FFFh
4K x 8 UTILITY ROM
32K x 8 LOWER HALF (PAGE 0) OF PROGRAM FLASH MEMORY
32K x 8 UPPER HALF (PAGE 1) OF PROGRAM FLASH MEMORY
2K x 8 UTILITY ROM
32K x 16 PROGRAM FLASH

Figure 2-4. Memory Map When Executing from Data SRAM

2.6 Clock Generation

All functional modules in the MAXQ2010 are synchronized to a single system clock. This system clock can be generated from one of the following clock sources:

- External high-frequency clock
- Internal high-frequency oscillator using external crystal or resonator circuit
- External 32kHz clock
- Internal 32kHz oscillator using external crystal or resonator circuit
- 256x frequency-locked loop (FLL) using 32kHz clock as an input source

The MAXQ2010 does not support an external RC relaxation oscillator circuit, and the FLL takes the place of the ring oscillator found in some other devices. The 32kHz crystal oscillator could also be used directly by the LCD controller and the RTC, regardless of the currently selected system clock source.

The following registers and bits are used to control clock generation and selection. For more information, see the register descriptions in this guide and in the *MAXQ Family User's Guide*.

2.6.1 External High-Frequency Oscillator Circuit

The high-frequency oscillator operates as described in *Section 2.7: Clock Generation* of the *MAXQ Family User's Guid*e. If used, the external crystal or resonator circuit for this oscillator should be connected between the HFXIN and HFXOUT pins.

The high-frequency oscillator can be disabled by setting HFXD (PWCN.4) to 1; this is only allowed if the high-frequency oscillator is not currently being used as the clock source (FLLMD and FLLSL must both equal 1). In this configuration, an external clock can be used to directly drive HFXIN; refer to the IC data sheet for more details.

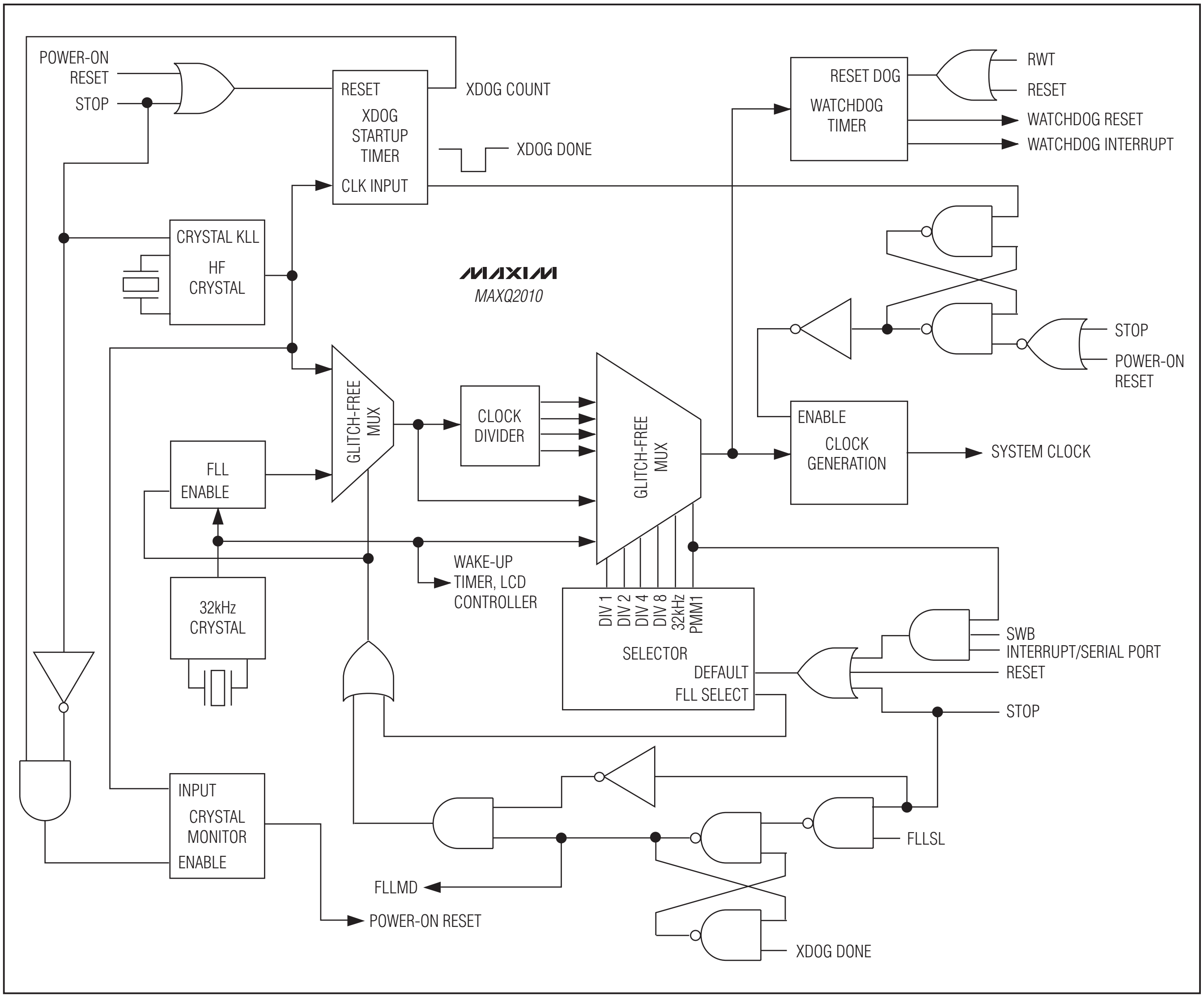


Figure 2-5. MAXQ2010 Clock Sources

2.6.2 External 32kHz Crystal Oscillator Circuit

The 32kHz oscillator operates as described in *Section 2.7: Clock Generation* of the *MAXQ Family User's Guid*e. It cannot be used directly as a system clock source, but instead works as an input to the FLL. It can also be used directly by the LCD controller and RTC modules, regardless of the current system clock source selection. This clock can be generated by an internal 32kHz crystal oscillator, using an external crystal connected between the 32KIN and 32KOUT pins.

Setting X32D (PWCN.2) to 1 disables the 32kHz clock source completely. This should only be done when all the following conditions are true:

- Either the high-frequency oscillator is being used as the system clock source, or if the FLL is being used as the system clock source, it is already enabled and locked.
- The 32kHz clock is not being used to drive either the RTC or the LCD controller.

If X32D = 0, the control bit X32KBYP (PWCN.5) can be used to switch between the internal oscillator and external 32kHz clock modes of operation.

Table 2-1. System Clock Generation and Control Registers

REGISTER	ADDRESS	BIT(S)	FUNCTION
CKCN	M8[0Eh]	[2:0]—PMME, CD[1:0]	000: System clock = high-frequency clock divided by 1. 001: System clock = high-frequency clock divided by 2. 010: System clock = high-frequency clock divided by 4. 011: System clock = high-frequency clock divided by 8. 1xx: System clock = high-frequency clock/256.
CKCN	M8[0Eh]	5—FLLMD	0: System clock is being provided by an external source. 1: System clock is being provided by the FLL.
CKCN	M8[0Eh]	6—FLLSL	0: Selects an external source for system clock generation. 1: Selects the FLL for system clock generation.
PWCN	M0[0Fh]	0—FLLEN	0: Disables the FLL (unless it is providing the system clock). 1: Enables the FLL and locks it to the 32kHz input.
PWCN	M0[0Fh]	1—FLOCK	0: Indicates the FLL is disabled or in the process of locking. 1: Indicates the FLL is locked to the 32kHz input.
PWCN	M0[0Fh]	2—X32D	0: 32kHz clock or oscillator operates normally (default). 1: Disables the 32kHz source completely (except in stop).
PWCN	M0[0Fh]	3—X32KRDY	0: Indicates the 32kHz oscillator is still warming up. 1: Indicates the 32kHz oscillator is ready for use.
PWCN	M0[0Fh]	4—HFXD	0: High-frequency oscillator operates normally (default). 1: Disables the high-frequency oscillator, allowing an external clock to be provided at HFXIN.
PWCN	M0[0Fh]	5—X32KBYP	0: 32kHz clock is provided by internal oscillator (default). 1: Disables the 32kHz oscillator, allowing an external clock to be provided at 32KIN.
PWCN	M0[0Fh]	[9:8]—X32KMD	00: 32kHz oscillator operates in noise immune mode. 01: 32kHz oscillator operates in quiet mode. 10: 32kHz oscillator operates in noise immune mode normally and in quiet mode during stop mode. **(Note: This setting should not be used on devices of rev B3 or earlier; refer to the device errata for more details.**) 11: 32kHz oscillator operates in quiet mode normally and in noise immune mode during stop mode.

2.6.3 Frequency-Locked Loop (FLL)

The MAXQ2010 contains an FLL circuit that provides an optional, low-cost method of generating a high-frequency system clock. The FLL uses the 32kHz clock as an input, multiplying its frequency by 256 to generate a clock output of approximately 8.4MHz (refer to the IC data sheet for details). This clock output can be used as a system clock source.

On power-on reset, the FLL is automatically enabled as the system clock source while the high-frequency oscillator warms up. Once the warmup count for the high-frequency oscillator has completed, the clock source switches to the high-frequency oscillator automatically. If no external crystal or resonator circuit is provided at HFXIN, the switchover never occurs, and the clock runs from the FLL indefinitely.

To select the FLL as the system clock source permanently, the FLLSL bit (CKCN.6) must be set to 1. Setting this bit immediately switches over the system clock source to the FLL. The FLLMD (CKCN.5) bit indicates the current system clock source. If the FLL is currently providing the system clock, FLLMD = 1; otherwise, FLLMD = 0.

Since the FLLSL bit is cleared by power-on reset only, if this bit is set before entering stop mode, the FLL is still used as the system clock source when stop mode is exited. In this case, a four-cycle warmup delay is required when exiting stop mode before execution resumes using the FLL as the system clock source.

When the system clock source is switched back from the FLL to the high-frequency oscillator by clearing FLLSL to zero, the FLL is still used as the system clock source until the warmup period has completed for the high-frequency oscillator. This is reflected by the value of the FLLMD bit, which remains at 1 until the warmup for the high-frequency oscillator has completed and the clock switches over, at which point FLLMD switches to 0.

2.7 Interrupts

In general, interrupt handling on the MAXQ2010 operates as described in the *MAXQ Family User's Guid*e. All interrupt sources have the same priority, and all interrupts cause program execution to branch to the location specified by the Interrupt Vector (IV) register, which defaults to 0000h.

Table 2-2 lists all possible interrupt sources for the MAXQ2010, along with their corresponding module interrupt enable bits, local interrupt enable bits, and interrupt flags.

- Each module interrupt enable bit, when cleared to 0, blocks interrupts originating in that module from being acknowledged. When the module interrupt enable bit is set to 1, interrupts from that module are acknowledged (unless all interrupts have been disabled globally).
- Each local interrupt enable bit, when cleared to 0, disables the corresponding interrupt. When the local interrupt-enable bit is set to 1, the interrupt is triggered whenever its interrupt flag is set to 1 by hardware or by software.
- Each interrupt flag bit, when set to 1, causes its corresponding interrupt to trigger. Interrupt flag bits are typically set by hardware and must be cleared by software (generally in the interrupt handler routine).

Note that for an interrupt to fire, the following five conditions must exist:

- Interrupts must be enabled globally by setting IGE (IC.0) to 1.
- The module interrupt enable bit for the interrupt source's module must be set to 1.
- The local interrupt enable bit for the specific interrupt source must be set to 1.
- The interrupt flag for the interrupt source must be set to 1. Typically, this is done by hardware when the condition that requires interrupt service occurs.
- The interrupt-in-service (INS) bit must be cleared to 0. This bit is set automatically upon vectoring to the interrupt handler (IV) address and cleared automatically upon exit (RETI/POPI), so the only reason to clear this bit manually (inside the interrupt handler routine) is to allow nested interrupt handling.

Table 2-2. Interrupt Sources and Control Bits

INTERRUPT	MODULE ENABLE BIT	LOCAL ENABLE BIT	INTERRUPT FLAG
Watchdog Interrupt	IMS (IMR.7)	EWDI (WDCN.6)	WDIF (WDCN.3)
External Interrupt 0 (P0.0)	IM0 (IMR.0)	EX0 (EIE0.0)	IE0 (EIF0.0)
External Interrupt 1 (P0.1)	IM0 (IMR.0)	EX1 (EIE0.1)	IE1 (EIF0.1)
External Interrupt 2 (P0.2)	IM0 (IMR.0)	EX2 (EIE0.2)	IE2 (EIF0.2)
External Interrupt 3 (P0.3)	IM0 (IMR.0)	EX3 (EIE0.3)	IE3 (EIF0.3)
External Interrupt 4 (P0.4)	IM0 (IMR.0)	EX4 (EIE0.4)	IE4 (EIF0.4)
External Interrupt 5 (P0.5)	IM0 (IMR.0)	EX5 (EIE0.5)	IE5 (EIF0.5)
External Interrupt 6 (P0.6)	IM0 (IMR.0)	EX6 (EIE0.6)	IE6 (EIF0.6)
External Interrupt 7 (P0.7)	IM0 (IMR.0)	EX7 (EIE0.7)	IE7 (EIF0.7)
RTC Time-of-Day Alarm	IM0 (IMR.0)	ADE (RCNT.1)	ALDF (RCNT.6)
RTC Subsecond Alarm	IM0 (IMR.0)	ASE (RCNT.2)	ALSF (RCNT.7)
External Interrupt 8 (P5.0)	IM1 (IMR.1)	EX8 (EIE1.0)	IE8 (EIF1.0)
External Interrupt 9 (P5.1)	IM1 (IMR.1)	EX9 (EIE1.1)	IE9 (EIF1.1)
External Interrupt 10 (P5.2)	IM1 (IMR.1)	EX10 (EIE1.2)	IE10 (EIF1.2)
External Interrupt 11 (P5.3)	IM1 (IMR.1)	EX11 (EIE1.3)	IE11 (EIF1.3)
External Interrupt 12 (P5.4)	IM1 (IMR.1)	EX12 (EIE1.4)	IE12 (EIF1.4)

Table 2-2. Interrupt Sources and Control Bits (continued)

INTERRUPT	MODULE ENABLE BIT	LOCAL ENABLE BIT	INTERRUPT FLAG
External Interrupt 13 (P5.5)	IM1 (IMR.1)	EX13 (EIE1.5)	IE13 (EIF1.5)
External Interrupt 14 (P5.6)	IM1 (IMR.1)	EX14 (EIE1.6)	IE14 (EIF1.6)
External Interrupt 15 (P6.0)	IM1 (IMR.1)	EX15 (EIE2.0)	IE15 (EIF2.0)
External Interrupt 16 (P6.1)	IM1 (IMR.1)	EX16 (EIE2.1)	IE16 (EIF2.1)
External Interrupt 17 (P6.2)	IM1 (IMR.1)	EX17 (EIE2.2)	IE17 (EIF2.2)
External Interrupt 18 (P6.3)	IM1 (IMR.1)	EX18 (EIE2.3)	IE18 (EIF2.3)
External Interrupt 19 (P6.4)	IM1 (IMR.1)	EX19 (EIE2.4)	IE19 (EIF2.4)
External Interrupt 20 (P6.5)	IM1 (IMR.1)	EX20 (EIE2.5)	IE20 (EIF2.5)
External Interrupt 21 (P6.6)	IM1 (IMR.1)	EX21 (EIE2.6)	IE21 (EIF2.6)
External Interrupt 22 (P6.7)	IM1 (IMR.1)	EX22 (EIE2.7)	IE22 (EIF2.7)
Supply Voltage Monitor Interrupt	IM1 (IMR.1)	SVMIE (SVM.2)	SVMI (SVM.3)
SPI Mode Fault Interrupt	IM1 (IMR.1)	ESPII (SPICF.7)	MODF (SPICN.3)
SPI Write Collision Interrupt	IM1 (IMR.1)	ESPII (SPICF.7)	WCOL (SPICN.4)
SPI Receive Overrun Interrupt	IM1 (IMR.1)	ESPII (SPICF.7)	ROVR (SPICN.5)
SPI Transfer Complete Interrupt	IM1 (IMR.1)	ESPII (SPICF.7)	SPIC (SPICN.6)
I^{2}C START Condition Interrupt	IM3 (IMR.3)	I2CSRI (I2CST.0)	I2CSRIE (I2CIE.0)
I^{2}C Transmit Complete Interrupt	IM3 (IMR.3)	I2CTXI (I2CST.1)	I2CTXIE (I2CIE.1)
I^{2}C Receive Ready Interrupt	IM3 (IMR.3)	I2CRXI (I2CST.2)	I2CRXIE (I2CIE.2)
I^{2}C Clock Stretch Interrupt	IM3 (IMR.3)	I2CSTRI (I2CST.3)	I2CSTRIE (I2CIE.3)
I^{2}C Timeout Interrupt	IM3 (IMR.3)	I2CTOI (I2CST.4)	I2CTOIE (I2CIE.4)
I^{2}C Slave Address Match Interrupt	IM3 (IMR.3)	I2CAMI (I2CST.5)	I2CAMIE (I2CIE.5)
I^{2}C Arbitration Loss Interrupt	IM3 (IMR.3)	I2CALI (I2CST.6)	I2CALIE (I2CIE.6)
I^{2}C NACK Interrupt	IM3 (IMR.3)	I2CNACKI (I2CST.7)	I2CNACKIE (I2CIE.7)
I^{2}C General Call Address Interrupt	IM3 (IMR.3)	I2CGCI (I2CST.8)	I2CGCIE (I2CIE.8)
I^{2}C Receiver Overrun Interrupt	IM3 (IMR.3)	I2CROI (I2CST.9)	I2CROIE (I2CIE.9)
I^{2}C STOP Condition Interrupt	IM3 (IMR.3)	I2CSPI (I2CST.11)	I2CSPIE (I2CIE.11)
Serial Port 0 Receive	IM3 (IMR.3)	ESI (SMD0.2)	RI (SCON0.0)
Serial Port 0 Transmit	IM3 (IMR.3)	ESI (SMD0.2)	TI (SCON0.1)
Serial Port 1 Receive	IM3 (IMR.3)	ESI (SMD1.2)	RI (SCON1.0)
Serial Port 1 Transmit	IM3 (IMR.3)	ESI (SMD1.2)	TI (SCON1.1)
ADC Data Available Interrupt	IM4 (IMR.4)	ADDAIE (ADCN.5)	ADDAI (ADST.5)
Type B Timer 0—External Trigger	IM4 (IMR.4)	EXFB (TB0CN.6)	ETB (TB0CN.1)
Type B Timer 0—Overflow	IM4 (IMR.4)	TFB (TB0CN.7)	ETB (TB0CN.1)
Type B Timer 1—External Trigger	IM4 (IMR.4)	EXFB (TB1CN.6)	ETB (TB1CN.1)
Type B Timer 1—Overflow	IM4 (IMR.4)	TFB (TB1CN.7)	ETB (TB1CN.1)
Type B Timer 2—External Trigger	IM4 (IMR.4)	EXFB (TB2CN.6)	ETB (TB2CN.1)
Type B Timer 2—Overflow	IM4 (IMR.4)	TFB (TB2CN.7)	ETB (TB2CN.1)

2.8 Reset Conditions

There are four possible reset sources for the MAXQ2010. While in the reset state, the enabled system clock oscillator continues running, but no code execution occurs. Once the reset condition has been removed or has completed, code execution resumes at address 8000h for all reset types.

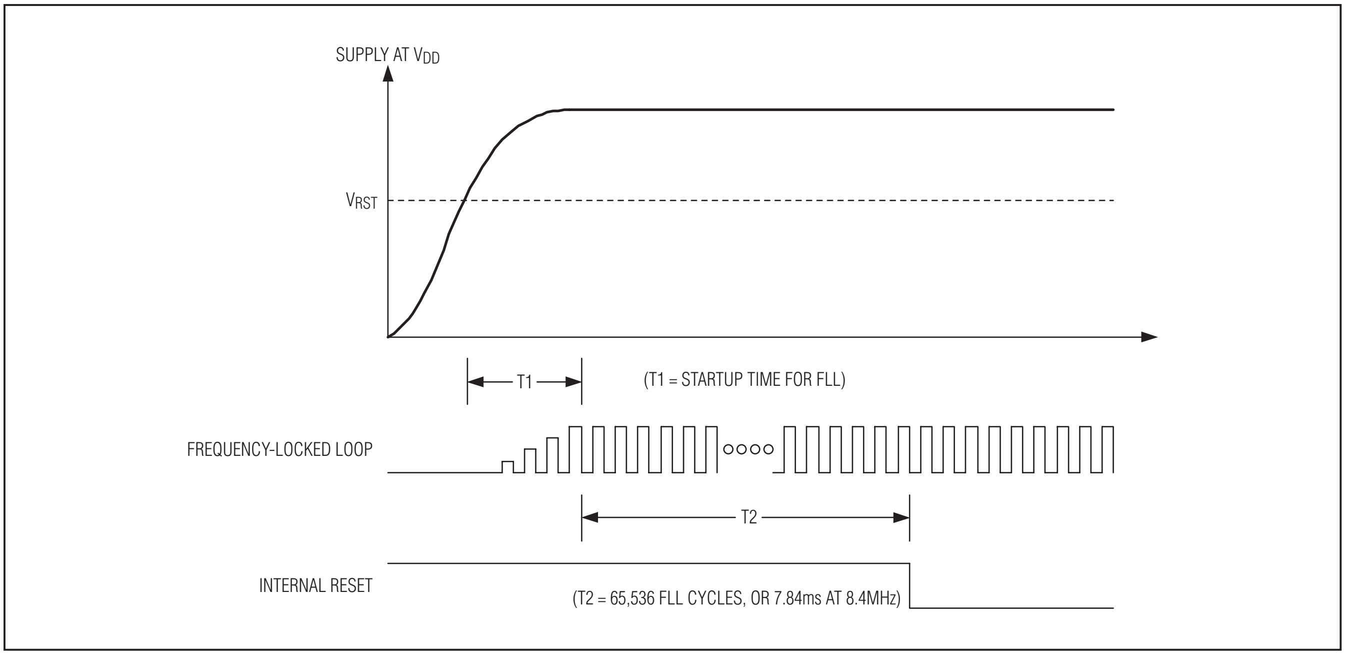

SUPPLY AT V_{DD}
V_{RST}
T1
(T1 = STARTUP TIME FOR FLL)
FREQUENCY-LOCKED LOOP
T2
INTERNAL RESET
(T2 = 65,536 FLL CYCLES, OR 7.84ms AT 8.4MHz)

Figure 2-6. Power-On Reset Timing

2.8.1 Power-On Reset

When power is first applied to the MAXQ2010, or when the supply voltage at V_{DD} drops below the V_{RST} level, the processor is held in a power-on reset state. See Figure 2-6. For the MAXQ2010 to exit power-on reset, the following two conditions must apply:

- The supply voltage at V_{DD} is above the power-on reset level V_{RST}.
- The FLL has completed 65,536 cycles (delay for power supply to stabilize).

2.8.2 Watchdog Timer Reset

The watchdog timer on the MAXQ2010 functions as described in the *MAXQ Family User's Guid*e.

2.8.3 External Reset

External reset through the $\overline{\text{RST}}$ input is a synchronous reset source. After the external reset low has been removed and sampled, execution resumes following a delay of four clock cycles, as shown in Figure 2-7.

2.9 Power-Management Features

The MAXQ2010 provides the following features to assist in power management:

- Divide-by-256 (PMM) mode to reduce current consumption.
- Switchback mode to exit PMM mode automatically when rapid processing is required.
- Ultra-low-power stop mode.
- Selective regulator and brownout detection disable during stop mode.
- Selectable noise immune mode or quiet mode for 32kHz oscillator operation.

Table 2-3 shows the system registers and bits used to control power-management features. For more information, see the register descriptions in this document and in the *MAXQ Family User's Guid*e.

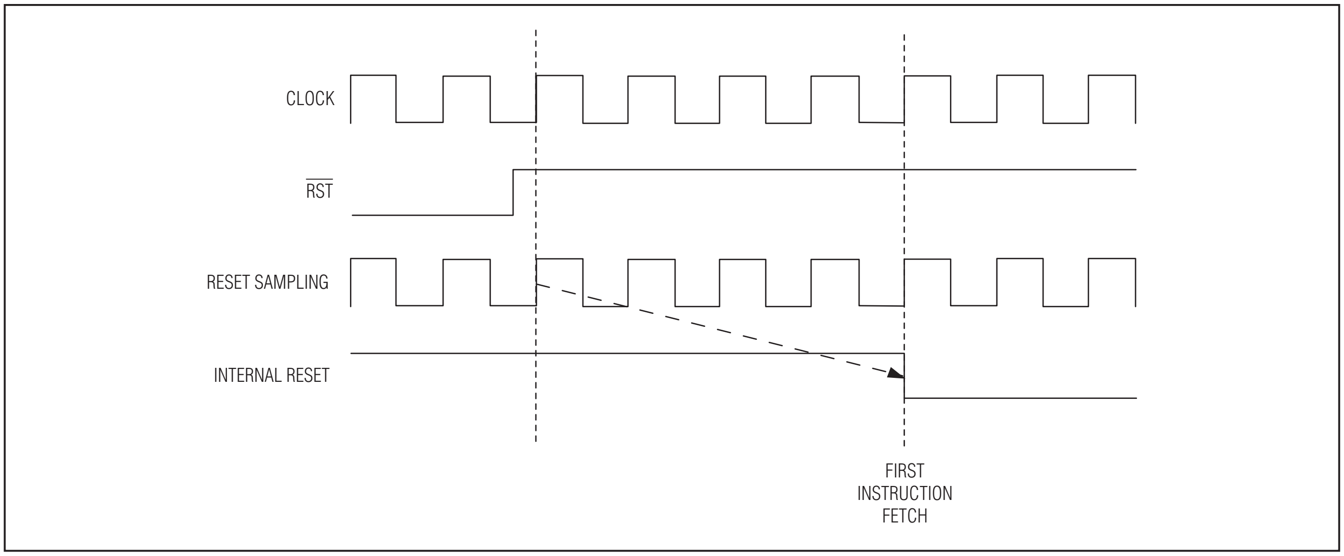


Figure 2-7. External Reset Timing

Table 2-3. System Power-Management Registers

REGISTER	ADDRESS	BIT	FUNCTION
CKCN	M8[0Eh]	[1:0]—CD[1:0]	00: System clock = high-frequency clock divided by 1. 01: System clock = high-frequency clock divided by 2. 10: System clock = high-frequency clock divided by 4. 11: System clock = high-frequency clock divided by 8.
CKCN	M8[0Eh]	2—PMME	0: System clock is determined by the settings of CD[1:0]. 1: System clock = high-frequency clock divided by 256.
CKCN	M8[0Eh]	3—SWB	When set to 1, enables automatic switchback from PMM (divide-by-256 mode) to normal clock-divide mode under certain conditions.
CKCN	M8[0Eh]	4—STOP	When set to 1, causes the processor to enter stop mode.
PWCN	M0[0Fh]	0—FLLEN	When set to 1, enables the FLL and causes it to lock to the 32kHz input.
PWCN	M0[0Fh]	1—FLOCK	0: FLL is disabled or warming up. 1: FLL is enabled and locked to the 32kHz input.
PWCN	M0[0Fh]	2—X32D	When set to 1, disables the 32kHz clock source.
PWCN	M0[0Fh]	3—X32KRDY	Read-only status bit; when 1, indicates that the 32kHz clock is ready for use.
PWCN	M0[0Fh]	4—HFXD	0: Enables the high-frequency oscillator. 1: Disables the high-frequency oscillator, allowing an external clock to be provided at HFXIN.
PWCN	M0[0Fh]	5—X32KBYP	0: Enables the internal 32kHz oscillator. 1: Disables the internal 32kHz oscillator, allowing an external clock to be provided at 32KIN.
PWCN	M0[0Fh]	6—REGEN	0: Internal regulator is shut down during stop mode. 1: Internal regulator remains powered on during stop mode.
PWCN	M0[0Fh]	7—BOD	0: Brownout detection remains enabled during stop mode. 1: Brownout detection is enabled during stop mode.

Table 2-3. System Power-Management Registers (continued)

REGISTER	ADDRESS	BIT	FUNCTION
PWCN	M0[0Fh]	[9:8]—X32KMD[1:0]	Selects operating mode of the 32kHz oscillator as follows. 00: Always operate in noise immune mode. 01: Always operate in quiet mode. 10: Operate in noise immune mode normally and in quiet mode during stop. Wait for 32kHz oscillator warmup upon stop mode exit. **(Note: This setting should not be used on devices of rev B3 or earlier; refer to the device errata for more details.)** 11: Operate in noise immune mode normally and in quiet mode during stop. Skip 32kHz oscillator warmup upon stop mode exit.
PWCN	M0[0Fh]	15—FREQMD	This bit is used to adjust the operating mode of the MAXQ2010 to provide optimal current consumption based on operating frequency. Typically, when running at a frequency of 3MHz or higher, this bit should be set to 0 to optimize current consumption. When running at a frequency under 3MHz, this bit should typically be set to 1. Refer to the "V_{DD} Supply Current vs. Clock Frequency" graph in the IC data sheet for more details.

2.9.1 Divide-by-256 Mode (PMM)

In this power-management mode, all operations continue as normal, but at a reduced clock rate (the high-frequency clock divided by 256). This power-management mode affects module clock rates as follows:

- Program execution occurs at the high-frequency clock rate divided by 256.
- The RTC module continues to operate using the 32kHz clock.
- The LCD module continues to operate using its originally selected clock, which is either the high-frequency clock divided by 512 or the 32kHz clock, as selected by the LCCS bit (LCRA.6).
- All other functional modules (timers, USARTs, SPI) operate at the high-frequency clock rate divided by 256.

This power-management mode is entered by setting the PMME bit (CKCN.2) to 1. When PMM mode is exited (either by clearing the PMME bit or as a result of a switchback trigger), system operation reverts to divide-by-1 mode.

2.9.2 Switchback Mode

As described in the *MAXQ Family User's Guide*, switchback mode is used to provide an automatic exit from power-management mode when a higher clock rate is required to respond to I/O, such as USART activity, SPI activity, or an external interrupt.

Switchback mode is enabled when the SWB (CKCN.3) bit is set to 1 and the PMME (CKCN.2) bit is set to 1 (the system is in the PMM mode). If switchback is enabled, the PMME bit is cleared (causing the system to exit power-management mode) when any of the following conditions occur:

- An external interrupt condition occurs on an external interrupt pin and the corresponding external interrupt is enabled.
- An active-low transition occurs on the RXD0 or RXD1 pin, and the corresponding USART is enabled to receive data.
- The SBUF0 or SBUF1 register is written to transmit a byte over the corresponding USART.
- The SPIB register is written to transmit a byte with the SPI interface enabled in master mode.
- The $\overline{\text{SSEL}}$ signal is asserted low with the SPI interface enabled in slave mode.
- A START condition occurs on the I^2C bus and the I^2C start interrupt is triggered.
- The supply voltage drops below the supply voltage monitor (SVM) threshold, and the SVM interrupt is triggered.
- A time-of-day alarm is generated by the RTC module.
- A subsecond alarm is generated by the RTC module.

- An ADC conversion is initiated by setting ADCONV to 1.
- Active debug mode is entered either by a breakpoint match or direct issuance of the debug command from background mode.

As described in the *MAXQ Family User's Guide*, if any of these conditions is true (a switchback source is active) and the SWB bit has been set, the PMME bit cannot be set to enter power-management mode.

2.9.3 Stop Mode

Stop mode disables all circuits within the MAXQ2010 except for the 32kHz crystal amplifier and any circuitry that is clocked directly by the 32kHz clock. All other on-chip clocks, timers, serial ports, and other peripherals are stopped, and no code execution occurs. Once in stop mode, the MAXQ2010 is in a near-static state, with power consumption determined largely by leakage currents.

The RTC always continues to run during stop mode if it was enabled upon stop mode entry. The LCD controller continues to run during stop mode if the following conditions are true:

- The 32kHz clock is selected as the LCD controller clock source (LCCS = 0).
- The LCD controller is in normal operation mode (OPM = 1).
- The LCD controller is enabled to operate during stop mode (SMO = 1).

Stop mode is invoked by setting the STOP bit to 1. The MAXQ2010 enters stop mode immediately when the STOP bit is set. Entering stop mode does not affect the setting of the clock control bits; this allows the system to return to its original operating frequency following stop mode removal.

The processor exits stop mode if any of the following conditions occur. In order to exit stop mode by means of an interrupt, the interrupt must be enabled globally, by module, and locally prior to entering stop mode.

- External reset (from the $\overline{\text{RST}}$ pin)
- Power-on/brownout reset
- External interrupt
- RTC time-of-day or subsecond alarm
- I^{2}C start interrupt
- Supply voltage monitor interrupt (SVMSTOP must be set to 1)

Note that exiting stop mode through external reset or power-on reset causes the processor to undergo a normal reset cycle, as opposed to resuming execution at the point at which it entered stop mode. Exiting stop mode by means of an interrupt or RTC alarm causes the processor to vector to the interrupt handler routine at IV. Following the completion of the interrupt handler, execution resumes at the instruction following the one that caused the entry into stop mode.

When the processor exits stop mode, program execution resumes after approximately four FLL oscillator cycles (refer to the IC data sheet for timing details). The clock source following stop mode exit is selected as follows:

- If FLLSL = 1, the processor continues running from the FLL indefinitely.
- If FLLSL = 0, the processor continues running from the FLL until the high-frequency clock source completes its 65,536-cycle warmup count, at which point it switches over to the high-frequency clock automatically.

Note: The MAXQ2010 powers down parts of the memory interface during stop mode to conserve power. Because of this, application code must always "re-prime" the active data pointer (DP[0], DP[1], or BP[OFFS]) following any exit from stop mode before using that data pointer to read from memory. This can be accomplished by:

- **Writing a new address (or rewriting the existing address) to the active data pointer.**
- **Writing to the DPC register to select a new data pointer (or reselect the active data pointer).**

ADDENDUM TO SECTION 3: PROGRAMMING

Refer to *Section 3: Programming* of the *MAXQ Family User's Guide* for examples of general program operations involving the MAXQ core. The MAXQ2010 contains the MAXQ20 (16-bit accumulator version) of the MAXQ core.

ADDENDUM TO SECTION 4: SYSTEM REGISTER DESCRIPTIONS

Refer to *Section 4: System Register Description*s of the *MAXQ Family User's Guide* for functional descriptions of the registers and bits in Table 4-1.

Table 4-1. System Register Map

CYCLES TO READ	CYCLES TO WRITE	REGISTER INDEX	M8	M9	M11	M12	M13	M14	M15
1	1	00h	AP	**A[0]**	**PFX[0]**	**IP**	—	—	—
1	1	01h	APC	**A[1]**	**PFX[1]**	—	**SP**	—	—
1	1	02h	—	**A[2]**	**PFX[2]**	—	**IV**	—	—
1	1	03h	—	**A[3]**	**PFX[3]**	—	—	OFFS	**DP[0]**
1	1	04h	PSF	**A[4]**	**PFX[4]**	—	—	**DPC**	—
1	1	05h	IC	**A[5]**	**PFX[5]**	—	—	**GR**	—
1	1	06h	IMR	**A[6]**	**PFX[6]**	—	**LC[0]**	GRL	—
1	1	07h	—	**A[7]**	**PFX[7]**	—	**LC[1]**	**BP**	**DP[1]**
1	2	08h	SC	**A[8]**	—	—	—	***GRS***	—
1	2	09h	—	**A[9]**	—	—	—	GRH	—
1	2	0Ah	—	**A[10]**	—	—	—	***GRXL***	—
1	2	0Bh	*IIR*	**A[11]**	—	—	—	***BP[OFFS]***	—
1	2	0Ch	—	**A[12]**	—	—	—	—	—
1	2	0Dh	—	**A[13]**	—	—	—	—	—
1	2	0Eh	CKCN	**A[14]**	—	—	—	—	—
1	2	0Fh	WDCN	**A[15]**	—	—	—	—	—

Note: *Register names that appear in italics indicate read-only registers. Register names that appear in bold indicate 16-bit registers. All other registers are 8 bits in width.*

Table 4-2. System Register Bit Functions

REG	BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AP									—	—	—	—	AP (4 bits)			
APC									CLR	IDS	—	—	—	MOD2	MOD1	MOD0
PSF									Z	S	—	GPF1	GPF0	OV	C	E
IC									—	—	—	—	—	—	INS	IGE
IMR									IMS	—	—	IM4	IM3	IM2	IM1	IM0
SC									TAP	—	—	CDA0	—	—	PWL	—
IIR									IIS	—	—	II4	II3	II2	II1	II0
CKCN									—	FLLSL	FLLMD	STOP	SWB	PMME	CD1	CD0
WDCN									POR	EWDI	WD1	WD0	WDIF	WTRF	EWT	RWT
A[0:15]	A[0:15] (16 bits)															
PFX	PFX (16 bits)															
IP	IP (16 bits)															
SP	—	—	—	—	—	—	—	—	—	—	—	—	SP (4 bits)			
IV	IV (16 bits)															
LC[0]	LC[0] (16 bits)															
LC[1]	LC[1] (16 bits)															

Table 4-2. System Register Bit Functions (continued)

REG	BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFS									OFFS (8 bits)							
DPC	—	—	—	—	—	—	—	—	—	—	—	WBS2	WBS1	WBS0	SDPS1	SDPS0
GR	GR (16 bits)															
GRL									GR.7	GR.6	GR.5	GR.4	GR.3	GR.2	GR.1	GR.0
BP	BP (16 bits)															
GRS	GR.7	GR.6	GR.5	GR.4	GR.3	GR.2	GR.1	GR.0	GR.15	GR.14	GR.13	GR.12	GR.11	GR.10	GR.9	GR.8
GRH									GR.15	GR.14	GR.13	GR.12	GR.11	GR.10	GR.9	GR.8
GRXL	GR.7	GR.7	GR.7	GR.7	GR.7	GR.7	GR.7	GR.7	GR.7	GR.6	GR.5	GR.4	GR.3	GR.2	GR.1	GR.0
BP[OFFS]	BP[OFFS] (16 bits)															
DP[0]	DP[0] (16 bits)															
DP[1]	DP[1] (16 bits)															

Table 4-3. System Register Reset Values

REG	BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AP									0	0	0	0	0	0	0	0
APC									0	0	0	0	0	0	0	0
PSF									1	0	0	0	0	0	0	0
IC									0	0	0	0	0	0	0	0
IMR									0	0	0	0	0	0	0	0
SC									1	0	0	0	0	0	*s*	0
IIR									0	0	0	0	0	0	0	0
CKCN									1	*s*	*s*	0	0	0	0	0
WDCN									*s*	*s*	0	0	0	*s*	*s*	0
A[0:15]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PFX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IP	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SP	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
IV	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LC[0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LC[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OFFS									0	0	0	0	0	0	0	0
DPC	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
GR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GRL									0	0	0	0	0	0	0	0
BP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GRS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GRH									0	0	0	0	0	0	0	0
GRXL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BP[OFFS]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DP[0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DP[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: *Bits marked as "s" have special behavior upon reset; see the register descriptions for details.*

4.1 System Register Descriptions

This section details the functionality of any system register contained in the MAXQ2010 that operates differently from its description in the *MAXQ Family User's Guide*. Addresses for all system and peripheral registers are given as "Mx[yy]," where x is the module number (from 0 to 15 decimal) and yy is the register index (from 00h to 1Fh hexadecimal). Fields in the bit definition tables are defined as follows:

- **Name:** Symbolic names of bits or bit fields in this register.
- **Reset:** The value of each bit in this register following a standard reset. If this field reads "unchanged," the given bit is unaffected by standard reset. If this field reads "s," the given bit does not have a fixed 0 or 1 reset value because its value is determined by another internal state or external condition.
- **POR:** If present this field defines the value of each bit in this register following a power-on reset (as opposed to a standard reset). Some bits are unaffected by standard resets and are set/cleared by POR only.
- **Access:** Bits can be read-only (r) or read/write (rw). Any special restrictions or conditions that could apply when reading or writing this bit are detailed in the bit description.

4.1.1 Processor Status Flags Register (PSF, M8[04h])

Bit #	7	6	5	4	3	2	1	0
Name	Z	S	—	GPF1	GPF0	OV	C	E
Reset	1	0	0	0	0	0	0	0
Access	r	r	r	rw	rw	rw	rw	rw

This register operates as described in the *MAXQ Family User's Guide*, with the exception that the overflow bit (OV) can be written by software.

4.1.2 Interrupt Mask Register (IMR, M8[06h])

Bit #	7	6	5	4	3	2	1	0
Name	IMS	—	—	IM4	IM3	IM2	IM1	IM0
Reset	0	0	0	0	0	0	0	0
Access	rw	r	r	rw	rw	rw	rw	rw

The first five bits in this register are interrupt mask bits for modules 0 to 4, one bit per module. The eighth bit, IMS, serves as a mask for any system module interrupt sources. Setting a mask bit allows the enabled interrupt sources for the associated module or system (with IMS) to generate interrupt requests. Clearing the mask bit effectively disables all interrupt sources associated with that module or, in the case of IMS, all system interrupt sources. The IMR register is intended to facilitate user-definable interrupt prioritization.

Bit 7: System Module Interrupt Mask (IMS)

Bits 5:6: Reserved

Bit 4: Module 4 Interrupt Mask (IM4)

Bit 3: Module 3 Interrupt Mask (IM3)

Bit 2: Module 2 Interrupt Mask (IM2)

Bit 1: Module 1 Interrupt Mask (IM1)

Bit 0: Module 0 Interrupt Mask (IM0)

4.1.3 System Control Register (SC, M8[08h])

Bit #	7	6	5	4	3	2	1	0
Name	TAP	—	—	CDA0	—	—	PWL	—
Reset	1	0	0	0	0	0	Unchanged	0
POR	1	0	0	0	0	0	1	0
Access	rw	r	r	r	r	r	rw	r

Bit 7: Test Access (Debug) Port Enable (TAP)

0 = Debug port functions are disabled, and P6.0 to P6.3 can be used as general-purpose I/O pins.

1 = Port pins P6.0 to P6.3 are enabled to act as debug port inputs and outputs.

Bits 6:5, 3:2, 0: Reserved

Bit 4: Code Data Access (CDA0). Setting this bit to 0 or 1 enables access to either the low or high page of program memory in data space when accessing data in byte mode, as shown in Figure 2-3 and Figure 2-4. When accessing data space in word mode, the setting of this bit has no effect.

Bit 1: Password Lock (PWL). This bit defaults to 1 on power-on reset only. When this bit is 1, it requires a 32-byte password to be matched with the password in the program space (words 10h to 1Fh) before allowing access to the ROM loader's utilities for read/write of program memory and debug functions. Clearing this bit to 0 disables the password protection to the ROM loader.

4.1.4 Interrupt Identification Register (IIR, M8[0Bh])

Bit #	7	6	5	4	3	2	1	0
Name	IIS	—	—	II4	II3	II2	II1	II0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

The first four bits in this register indicate interrupts pending in modules 0 to 4, one bit per module. The eighth bit, IIS, indicates a pending system interrupt (from the watchdog timer or other system function). The interrupt pending flags are set only for enabled interrupt sources waiting for service. The interrupt pending flag is cleared when the pending interrupt source(s) within that module are disabled or when the interrupt flag(s) are cleared by software.

Bit 7: Interrupt Pending Flag for System Modules (IIS)

Bits 6:5: Reserved

Bit 4: Interrupt Pending Flag for Module 4 (II4)

Bit 3: Interrupt Pending Flag for Module 3 (II3)

Bit 2: Interrupt Pending Flag for Module 2 (II2)

Bit 1: Interrupt Pending Flag for Module 1 (II1)

Bit 0: Interrupt Pending Flag for Module 0 (II0)

4.1.5 System Clock Control Register (CKCN, M8[0Eh])

Bit #	7	6	5	4	3	2	1	0
Name	—	FLLSL	FLLMD	STOP	SWB	PMME	CD1	CD0
Reset	1	Unchanged	0	0	0	0	0	0
POR	1	0	0	0	0	0	0	0
Access	r	rw	s	rw	rw	rw	rw*	rw*

**Unrestricted read access. This bit can only be modified when PMME = 0.*

The CKCN register bit settings determine the system clock source and clock divider as described in Table 4-4.

Bit 7: Reserved

Bit 6: Frequency-Locked Loop Select (FLLSL)

0 = Selects the high-frequency oscillator as the system clock source.

1 = Selects the FLL as the system clock source.

Bit 5: Frequency-Locked Loop Mode (FLLMD). This read-only status bit indicates the clock source that is currently being used.

0 = High-frequency oscillator is currently being used as the system clock source, because the FLL is not selected (FLLSL = 0).

1 = FLL is currently being used as the system clock source. This is either because it is selected as the clock source (FLLSL = 1), or because the high-frequency oscillator is in the process of warming up.

Bit 4: Stop-Mode Select (STOP). Setting this bit to 1 causes the processor to enter stop mode. This does not change the currently selected clock-divide ratio.

Bit 3: Switchback Enable (SWB). Setting this bit to 1 enables switchback mode. If power-management mode (divide by 256) is active and switchback is enabled, the PMME bit is cleared to 0 when any of the following conditions occurs.

- An external interrupt is generated based on an edge detect.
- Either serial port 0 or serial port 1 is enabled to receive data and detects a low condition on its data receive pin.
- Either serial port 0 or serial port 1 has a byte written to its buffer register by software.
- The SPI interface is enabled in master mode, and the SPIB register is written by software.
- The SPI interface is enabled is slave mode, and an external master drives the $\overline{\text{SSEL}}$ line low.
- A START condition occurs on the I^2C bus, causing an I^2C start interrupt to be generated.
- The power supply drops below the SVM threshold, causing an SVM interrupt to be generated.
- An ADC conversion is initiated by software by setting the ADCONV bit to 1.
- A time-of-day interrupt or subsecond alarm interrupt occurs from the RTC.
- Debug mode is entered through command entry or a breakpoint match (exits from PMM1 only).

Triggering a switchback condition only clears the PMME bit; the settings of CD0 and CD1 remain the same. When either power-management mode is active, the SWB bit cannot be set to 1 as long as any of the above conditions are true.

Bit 2: Power-Management Mode Enable (PMME)

Bits 1:0: Clock Divide 1:0 (CD[1:0]). These three bits control the divide ratio or enable power-management mode for the system clock as shown in Table 4-4. CD0 and CD1 can always be read, and they can be written as long as PMME = 0.

Setting the PMME bit to 1 activates PMM mode, causing the system clock to be divided by 256. While PMME is set to 1, CD0 and CD1 cannot be changed; their values determine the clock-divide ratio that is used when the processor exits power-management mode.

Table 4-4. System Clock Modes

FLLMD	SWB	PMME	CD1	CD0	SYSTEM CLOCK	SWITCHBACK
0	0	0	0	0	High-frequency clock/1	—
0	0	0	0	1	High-frequency clock/2	—
0	0	0	1	0	High-frequency clock/4	—
0	0	0	1	1	High-frequency clock/8	—
0	0	1	X	X	High-frequency clock/256	Not Active
0	1	1	X	X	High-frequency clock/256	Active
1	0	0	0	0	FLL clock/1	—
1	0	0	0	1	FLL clock/2	—
1	0	0	1	0	FLL clock/4	—
1	0	0	1	1	FLL clock/8	—
1	0	1	X	X	FLL clock/256	Not Active
1	1	1	X	X	FLL clock/256	Active

ADDENDUM TO SECTION 5: PERIPHERAL REGISTER MODULES

Table 5-1. Peripheral Register Map

CYCLES TO READ	CYCLES TO WRITE	REGISTER INDEX	M0	M1	M2	M3	M4	M5
1	1	00h	PO0	PO4	MCNT	**I2CBUF**	**TB0R**	—
1	1	01h	PO1	PO5	**MA**	**I2CST**	**TB0C**	—
1	1	02h	PO2	PO6	**MB**	**I2CIE**	**TB1R**	—
1	1	03h	PO3	**SPIB**	**MC2**	—	**TB1C**	—
1	1	04h	EIF0	EIF1	**MC1**	SCON0	**TB2R**	—
1	1	05h	EIE0	EIE1	**MC0**	SBUF0	**TB2C**	—
1	1	06h	—	EIF2	LCFG	SCON1	**ADST**	—
1	1	07h	—	EIE2	—	SBUF1	**ADADDR**	—
1	2	08h	*PI0*	*PI4*	***MC1R***	SMD0	**TB0CN**	—
1	2	09h	*PI1*	*PI5*	***MC0R***	**PR0**	**TB0V**	—
1	2	0Ah	*PI2*	*PI6*	**LCRA**	SMD1	**TB1CN**	—
1	2	0Bh	*PI3*	EIES1	LCD0	**PR1**	**TB1V**	—
1	2	0Ch	EIES0	EIES2	LCD1	I2CCN	**TB2CN**	—
1	2	0Dh	—	**SVM**	LCD2	**I2CCK**	**TB2V**	—
1	2	0Eh	—	—	LCD3	I2CTO	**ADCN**	—
1	2	0Fh	**PWCN**	—	LCD4	**I2CSLA**	**ADDATA**	—
2	2	10h	PD0	PD4	LCD5	—	—	—
2	2	11h	PD1	PD5	LCD6	—	—	—
2	2	12h	PD2	PD6	LCD7	—	—	—
2	2	13h	PD3	—	LCD8	—	—	—
2	2	14h	—	—	LCD9	—	—	—
2	2	15h	—	SPICN	LCD10	—	—	—
2	2	16h	—	SPICF	LCD11	—	—	—
2	2	17h	—	SPICK	LCD12	—	—	—
2	2	18h	RTRM	—	LCD13	—	—	—
2	2	19h	**RCNT**	—	LCD14	—	—	—
2	2	1Ah	RTSS	—	LCD15	—	—	—
2	2	1Bh	**RTSH**	—	LCD16	—	—	—
2	2	1Ch	**RTSL**	—	LCD17	—	—	—
2	2	1Dh	RSSA	—	LCD18	—	—	—
2	2	1Eh	RASH	—	LCD19	—	—	—
2	2	1Fh	**RASL**	—	LCD20	—	—	—

Note: *Register names that appear in italics indicate registers in which all bits are read-only. Register names that appear in bold indicate 16-bit registers. All other registers are 8 bits in width.*

Table 5-2. Peripheral Register Bit Functions

REG	BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PO0									PO0 (8 bits)							
PO1									PO1 (8 bits)							
PO2									PO2 (8 bits)							
PO3									PO3 (8 bits)							
EIF0									IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
EIE0									EX7	EX6	EX5	EX4	EX3	EX2	EX1	EX0
PI0									PI0 (8 bits)							
PI1									PI1 (8 bits)							
PI2									PI2 (8 bits)							
PI3									PI3 (8 bits)							
EIES0									IT7	IT6	IT5	IT4	IT3	IT2	IT1	IT0
PWCN	FREQMD	—	—	—	—	—	X32KMD (2 bits)		BOD	REGEN	X32K BYP	HFXD	X32K RDY	X32D	FLOCK	FLLEN
PD0									PD0 (8 bits)							
PD1									PD1 (8 bits)							
PD2									PD2 (8 bits)							
PD3									PD3 (8 bits)							
RTRM									TSGN	TRM (7 bits)						
RCNT	WE	—	ACS	—	—	—	FT	SQE	ALSF	ALDF	RDYE	RDY	BUSY	ASE	ADE	RTCE
RTSS									RTSS (8 bits)							
RTSH	RTSH (16 bits)															
RTSL	RTSL (16 bits)															
RSSA									RSSA (8 bits)							
RASH									—	—	—	—	RASH (4 bits)			
RASL									RASL (8 bits)							
PO4									PO4 (8 bits)							
PO5									—	PO5 (7 bits)						
PO6									PO6 (8 bits)							
SPIB	SPIB (16 bits)															
EIF1									—	IE14	IE13	IE12	IE11	IE10	IE9	IE8
EIE1									—	EX14	EX13	EX12	EX11	EX10	EX9	EX8
EIF2									IE22	IE21	IE20	IE19	IE18	IE17	IE16	IE15
EIE2									EX22	EX21	EX20	EX19	EX18	EX17	EX16	EX15
PI4									PI4 (8 bits)							
PI5									—	PI5 (7 bits)						
PI6									PI6 (8 bits)							
EIES1									—	IT14	IT13	IT12	IT11	IT10	IT9	IT8
EIES2									IT22	IT21	IT20	IT19	IT18	IT17	IT16	IT15
SVM	—	—	—	—	SVTH (4 bits)				—	—	—	SVM STOP	SVMI	SVMIE	SVM RDY	SVMEN
PD4									PD4 (8 bits)							
PD5									—	PD5 (7 bits)						
PD6									PD6 (8 bits)							

Table 5-2. Peripheral Register Bit Functions (continued)

REG	BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPICN									STBY	SPIC	ROVR	WCOL	MODF	MODFE	MSTM	SPIEN
SPICF									ESPII	—	—	—	—	CHR	CKPHA	CKPOL
SPICK									SPICK (8 bits)							
MCNT									OF	MCW	CLD	SQU	OPCS	MSUB	MMAC	SUS
MA	MA (16 bits)															
MB	MB (16 bits)															
MC2	MC2 (16 bits)															
MC1	MC1 (16 bits)															
MC0	MC0 (16 bits)															
LCFG									PCF4	PCF3	PCF2	PCF1	PCF0	SMO	OPM	DPE
MC1R	MC1R (16 bits)															
MC0R	MC0R (16 bits)															
LCRA	—	—	—	DUTY1	DUTY0	FRM3	FRM2	FRM1	FRM0	LCCS	LRIG	—	LRA3	LRA2	LRA1	LRA0
LCD[n]									LCD[n] (8 bits)							
I2CBUF	I2CBUF (16 bits)															
I2CST	I2C BUS	I2C BUSY	—	—	I2CSPI	I2CSCL	I2CROI	I2CGCI	I2C NACKI	I2CALI	I2CAMI	I2CTOI	I2C STRI	I2CRXI	I2CTXI	I2CSRI
I2CIE	—	—	—	—	I2C SPIE	—	I2C ROIE	I2C GCIE	I2C NACKIE	I2C ALIE	I2C AMIE	I2C TOIE	I2C STRIE	I2C RXIE	I2C TXIE	I2C SRIE
SCON0									SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI
SBUF0									SBUF0 (8 bits)							
SCON1									SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI
SBUF1									SBUF1 (8 bits)							
SMD0									—	—	—	—	—	ESI	SMOD	FEDE
PR0	PR0 (16 bits)															
SMD1									—	—	—	—	—	ESI	SMOD	FEDE
PR1	PR1 (16 bits)															
I2CCN	I2C RST	—	—	—	—	—	I2C STREN	I2C GCEN	I2C STOP	I2C START	I2C ACK	I2C STRS	—	I2C MODE	I2C MST	I2CEN
I2CCK	I2CCKH (8 bits)								I2CCKL (8 bits)							
I2CTO									I2CTO (8 bits)							
I2CSLA	—	—	—	—	—	—	I2CSLA (10 bits)									
TB0R	TB0R (16 bits)															
TB0C	TB0C (16 bits)															
TB1R	TB1R (16 bits)															
TB1C	TB1C (16 bits)															
TB2R	TB2R (16 bits)															
TB2C	TB2C (16 bits)															
ADST	—	—	—	—	ADDAT (4 bits)				REFOK	AD CONV	ADDAI	AD CFG	ADIDX (4 bits)			
ADADDR	—	—	—	—	SEQSTORE (4 bits)				—	SEQSTART (3 bits)			—	SEQEND (3 bits)		

Table 5-2. Peripheral Register Bit Functions (continued)

REG	BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TB0CN	C/TB	—	—	TBCS	TBCR	TBPS (3 bits)			TFB	EXFB	TBOE	DCEN	EXENB	TRB	ETB	CP/ RLB
TB0V	TB0V (16 bits)															
TB1CN	C/TB	—	—	TBCS	TBCR	TBPS (3 bits)			TFB	EXFB	TBOE	DCEN	EXENB	TRB	ETB	CP/ RLB
TB1V	TB1V (16 bits)															
TB2CN	C/TB	—	—	TBCS	TBCR	TBPS (3 bits)			TFB	EXFB	TBOE	DCEN	EXENB	TRB	ETB	CP/ RLB
TB2V	TB2V (16 bits)															
ADCN	—	—	—	—	ADINT (2 bits)		ADCLK (2 bits)		IREF EN	AD CONT	AD DAIE	AD PMO	ADACQ (4 bits)			
ADDATA	ADDATA (16 bits)															

Table 5-3. Peripheral Register Bit Reset Values

REG	BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PO0									1	1	1	1	1	1	1	1
PO1									1	1	1	1	1	1	1	1
PO2									1	1	1	1	1	1	1	1
PO3									1	1	1	1	1	1	1	1
EIF0									0	0	0	0	0	0	0	0
EIE0									0	0	0	0	0	0	0	0
PI0									*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*
PI1									*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*
PI2									*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*
PI3									*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*
EIES0									0	0	0	0	0	0	0	0
PWCN	0	0	0	0	0	0	0	0	0	0	0	0	*s*	*s*	0	0
PD0									0	0	0	0	0	0	0	0
PD1									0	0	0	0	0	0	0	0
PD2									0	0	0	0	0	0	0	0
PD3									0	0	0	0	0	0	0	0
RTRM									0	0	0	0	0	0	0	0
RCNT	0	0	0	0	0	0	0	0	0	0	0	0	*s*	0	0	*s*
RTSS									*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*
RTSH	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*
RTSL	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*
RSSA									0	0	0	0	0	0	0	0
RASH									0	0	0	0	0	0	0	0
RASL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PO4									1	1	1	1	1	1	1	1
PO5									1	1	1	1	1	1	1	1
PO6									1	1	1	1	1	1	1	1

Table 5-3. Peripheral Register Bit Reset Values (continued)

REG	BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPIB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EIF1									0	0	0	0	0	0	0	0
EIE1									0	0	0	0	0	0	0	0
EIF2									0	0	0	0	0	0	0	0
EIE2									0	0	0	0	0	0	0	0
PI4									*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*
PI5									0	*s*	*s*	*s*	*s*	*s*	*s*	*s*
PI6									*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*
EIES1									0	0	0	0	0	0	0	0
EIES2									0	0	0	0	0	0	0	0
SVM	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
PD4									0	0	0	0	0	0	0	0
PD5									0	0	0	0	0	0	0	0
PD6									0	0	0	0	0	0	0	0
SPICN									0	0	0	0	0	0	0	0
SPICF									0	0	0	0	0	0	0	0
SPICK									0	0	0	0	0	0	0	0
MCNT									0	0	0	0	0	0	0	0
MA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MC2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MC1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MC0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LCFG									0	0	0	0	0	0	0	0
MC1R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MC0R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LCRA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LCD[n]									0	0	0	0	0	0	0	0
I2CBUF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I2CST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I2CIE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SCON0									0	0	0	0	0	0	0	0
SBUF0									0	0	0	0	0	0	0	0
SCON1									0	0	0	0	0	0	0	0
SBUF1									0	0	0	0	0	0	0	0
SMD0									0	0	0	0	0	0	0	0
PR0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SMD1									0	0	0	0	0	0	0	0
PR1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I2CCN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I2CCK	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
I2CTO									0	0	0	0	0	0	0	0
I2CSLA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TB0R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-3. Peripheral Register Bit Reset Values (continued)

REG	BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TB0C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TB1R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TB1C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TB2R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TB2C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADADDR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TB0CN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TB0V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TB1CN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TB1V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TB2CN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TB2V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADCN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADDATA	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*	*s*

Note: *Bits marked as "s" have special behavior upon reset; see the register descriptions for details.*

ADDENDUM TO SECTION 6: GENERAL-PURPOSE I/O MODULE

The MAXQ2010 provides up to 55 port pins for general-purpose I/O that are grouped into logical ports P0 to P6. Each of these port pins has the following features:

- CMOS output drivers
- Schmitt trigger inputs
- Optional weak pullup to DVDD when operating in input mode

Many of the port pins on the MAXQ2010 are also multiplexed with special and alternate functions as listed below. All these functions are disabled by default with the exception of the debug port interface pins, which are enabled by default following any reset. The behavior of these functions breaks down into two overall categories.

- **Special functions** override the PD and PO settings for the port pin when they are enabled. Once the special function takes control, normal control of the port pin is lost until the special function completes its task or is disabled. Examples of special functions include serial port 0 transmit and LCD segment drive.
- **Alternate functions** operate in parallel with the PD and PO settings for the port pin, and generally consist of input-only functions such as external interrupts. When an alternate function is enabled for a port pin, the port pin's output state is still controlled in the usual manner.

Table 6-1. Port Pin Special and Alternate Functions

PORT PIN	FUNCTION TYPE	FUNCTION	ENABLED WHEN
P0.0	Special Analog	LCD Segment SEG0	DPE = 1, PCF0 = 1, and EX0 = 0
P0.0	Alternate	External Interrupt 0	(EIE0.0) EX0 = 1
P0.1	Special Analog	LCD Segment SEG1	DPE = 1, PCF0 = 1, and EX1 = 0
P0.1	Alternate	External Interrupt 1	(EIE0.1) EX1 = 1
P0.2	Special Analog	LCD Segment SEG2	DPE = 1, PCF0 = 1, and EX2 = 0
P0.2	Alternate	External Interrupt 2	(EIE0.2) EX2 = 1
P0.3	Special Analog	LCD Segment SEG3	DPE = 1, PCF0 = 1, and EX3 = 0
P0.3	Alternate	External Interrupt 3	(EIE0.3) EX3 = 1
P0.4	Special Analog	LCD Segment SEG4	DPE = 1, PCF0 = 1, and EX4 = 0
P0.4	Alternate	External Interrupt 4	(EIE0.4) EX4 = 1
P0.5	Special Analog	LCD Segment SEG5	DPE = 1, PCF0 = 1, and EX5 = 0
P0.5	Alternate	External Interrupt 5	(EIE0.5) EX5 = 1
P0.6	Special Analog	LCD Segment SEG6	DPE = 1, PCF0 = 1, and EX6 = 0
P0.6	Alternate	External Interrupt 6	(EIE0.6) EX6 = 1
P0.7	Special Analog	LCD Segment SEG7	DPE = 1, PCF0 = 1, and EX7 = 0
P0.7	Alternate	External Interrupt 7	(EIE0.7) EX7 = 1
P1.0	Special Analog	LCD Segment SEG8	DPE = 1 and PCF1 = 1
P1.1	Special Analog	LCD Segment SEG9	DPE = 1 and PCF1 = 1
P1.2	Special Analog	LCD Segment SEG10	DPE = 1 and PCF1 = 1
P1.3	Special Analog	LCD Segment SEG11	DPE = 1 and PCF1 = 1
P1.4	Special Analog	LCD Segment SEG12	DPE = 1 and PCF1 = 1
P1.5	Special Analog	LCD Segment SEG13	DPE = 1 and PCF1 = 1
P1.6	Special Analog	LCD Segment SEG14	DPE = 1 and PCF1 = 1
P1.7	Special Analog	LCD Segment SEG15	DPE = 1 and PCF1 = 1
P2.0	Special Analog	LCD Segment SEG16	DPE = 1 and PCF2 = 1
P2.1	Special Analog	LCD Segment SEG17	DPE = 1 and PCF2 = 1
P2.2	Special Analog	LCD Segment SEG18	DPE = 1 and PCF2 = 1

Table 6-1. Port Pin Special and Alternate Functions (continued)

PORT PIN	FUNCTION TYPE	FUNCTION	ENABLED WHEN
P2.3	Special Analog	LCD Segment SEG19	DPE = 1 and PCF2 = 1
P2.4	Special Analog	LCD Segment SEG20	DPE = 1 and PCF2 = 1
P2.5	Special Analog	LCD Segment SEG21	DPE = 1 and PCF2 = 1
P2.6	Special Analog	LCD Segment SEG22	DPE = 1 and PCF2 = 1
P2.7	Special Analog	LCD Segment SEG23	DPE = 1 and PCF2 = 1
P3.0	Special Analog	LCD Segment SEG24	DPE = 1 and PCF3 = 1
P3.1	Special Analog	LCD Segment SEG25	DPE = 1 and PCF3 = 1
P3.2	Special Analog	LCD Segment SEG26	DPE = 1 and PCF3 = 1
P3.3	Special Analog	LCD Segment SEG27	DPE = 1 and PCF3 = 1
P3.4	Special Analog	LCD Segment SEG28	DPE = 1 and PCF3 = 1
P3.5	Special Analog	LCD Segment SEG29	DPE = 1 and PCF3 = 1
P3.6	Special Analog	LCD Segment SEG30	DPE = 1 and PCF3 = 1
P3.7	Special Analog	LCD Segment SEG31	DPE = 1 and PCF3 = 1
P4.0	Special Analog	LCD Segment SEG32	DPE = 1 and PCF4 = 1
P4.1	Special Analog	LCD Segment SEG33	DPE = 1 and PCF4 = 1
P4.2	Special Analog	LCD Segment SEG34	DPE = 1 and PCF4 = 1
P4.3	Special Analog	LCD Segment SEG35	DPE = 1 and PCF4 = 1
P4.4	Special Analog	LCD Segment SEG36	DPE = 1 and PCF4 = 1
P4.5	Special Analog	LCD Segment SEG37	DPE = 1 and PCF4 = 1
P4.6	Special Analog	LCD Segment SEG38	DPE = 1 and PCF4 = 1
P4.7	Special Analog	LCD Segment SEG39	DPE = 1 and PCF4 = 1
P5.0	Alternate	External Interrupt 8	(EIE1.0) EX8 = 1
P5.0	Alternate	Type B Timer 0 Input B	EXENB = 1
P5.0	Special	Type B Timer 0 Output B (Compare Output)	TBCR:TBCS<>00b **(overrides Serial 0 Data)**
P5.0	Special	Serial Port 0 Data (Mode 0)	Mode 0: SBUF0 is written or REN is set to 1 (until serial transmission completes)
P5.0	Alternate	Serial Port 0 Receive (Modes 1/2/3)	Modes 1/2/3: REN = 1
P5.1	Alternate	External Interrupt 9	(EIE1.1) EX9 = 1
P5.1	Alternate	Type B Timer 0 Input A (Counter Input)	C/TB = 1
P5.1	Special	Type B Timer 0 Output A (Clock Output)	C/TB = 0 and TB0E = 1 **(overrides serial port 0 transmit/clock)**
P5.1	Special	Serial Port 0 Transmit/Clock	Mode 0: REN set to 1 (until serial transmission completes) Mode 1/2/3: SBUF0 is written (until serial transmission completes)
P5.2	Alternate	External Interrupt 10	(EIE1.2) EX10 = 1
P5.2	Special	RTC Square-Wave Output	RTCE = 1 and SQE = 1
P5.3	Alternate	External Interrupt 11	(EIE1.3) EX11 = 1
P5.3	Alternate	SPI Slave-Select Input ($\overline{\text{SSEL}}$)	SPIEN = 1 and MSTM = 0
P5.4	Alternate	External Interrupt 12	(EIE1.4) EX12 = 1
P5.4	Alternate	MOSI: SPI Slave Input (Slave Mode)	SPIEN = 1 and MSTM = 0
P5.4	Special	MOSI: SPI Master Output (Master Mode)	SPIEN = 1 and MSTM = 1
P5.5	Alternate	External Interrupt 13	(EIE1.5) EX13 = 1
P5.5	Alternate	SCLK: SPI Clock Input (Slave Mode)	SPIEN = 1 and MSTM = 0

Table 6-1. Port Pin Special and Alternate Functions (continued)

PORT PIN	FUNCTION TYPE	FUNCTION	ENABLED WHEN
P5.5	Special	SCLK: SPI Clock Output (Master Mode)	SPIEN = 1 and MSTM = 1
P5.6	Alternate	External Interrupt 14	(EIE1.6) EX14 = 1
P5.6	Special	MISO: SPI Slave Output (Slave Mode)	SPIEN = 1 and MSTM = 0
P5.6	Alternate	MISO: SPI Master Input (Master Mode)	SPIEN = 1 and MSTM = 1
P6.0	Alternate	JTAG Interface: TAP Clock (TCK)	(SC.7) TAP = 1
P6.0	Alternate	External Interrupt 15	(EIE2.0) EX15 = 1
P6.1	Alternate	JTAG Interface: TAP Data Input (TDI)	(SC.7) TAP = 1
P6.1	Alternate	External Interrupt 16	(EIE2.1) EX16 = 1
P6.2	Alternate	JTAG Interface: TAP Mode Select (TMS)	(SC.7) TAP = 1
P6.2	Alternate	External Interrupt 17	(EIE2.2) EX17 = 1
P6.3	Special	JTAG Interface: TAP Data Output (TDO)	(SC.7) TAP = 1
P6.3	Alternate	External Interrupt 18	(EIE2.3) EX18 = 1
P6.4	Alternate	External Interrupt 19	(EIE2.4) EX19 = 1
P6.4	Alternate	Type B Timer 1 Input B	EXENB = 1
P6.4	Special	Type B Timer 1 Output B (Compare Output)	TBCR:TBCS<>00b **(overrides serial port 1 data)**
P6.4	Special	Serial Port 1 Data (Mode 0)	Mode 0: SBUF1 is written or REN is set to 1 (until serial transmission completes)
P6.4	Alternate	Serial Port 1 Receive (Modes 1/2/3)	Modes 1/2/3: REN = 1
P6.5	Alternate	External Interrupt 20	(EIE2.5) EX20 = 1
P6.5	Alternate	Type B Timer 0 Input A (Counter Input)	C/TB = 1
P6.5	Special	Type B Timer 0 Output A (Clock Output)	C/TB = 0 and TB0E = 1 **(overrides serial port 1 transmit/clock)**
P6.5	Special	Serial Port 1 Transmit/Clock	Mode 0: REN set to 1 (until serial transmission completes) Mode 1/2/3: SBUF1 is written (until serial transmission completes)
P6.6	Alternate	External Interrupt 21	(EIE2.6) EX21 = 1
P6.6	Alternate	Type B Timer 2 Input B	EXENB = 1
P6.6	Special	Type B Timer 2 Output B (Compare Output)	TBCR:TBCS<>00b
P6.6	Special	SCL: I^{2}C Clock Line	I2CEN = 1 **(overrides timer B output B)**
P6.7	Alternate	External Interrupt 22	(EIE2.7) EX22 = 1
P6.7	Alternate	Type B Timer 0 Input A (Counter Input)	C/TB = 1
P6.7	Special	Type B Timer 0 Output A (Clock Output)	C/TB = 0 and TB0E = 1
P6.7	Special	SDA: I^{2}C Data Line	I2CEN = 1 **(overrides timer B output A)**

The port pins on the MAXQ2010 operate the same as standard MAXQ port pins, with input/output states defined in Table 6-2.

Table 6-2. Port Pin Input/Output States (in Standard Mode)

PDx.y	POx.y	PORT PIN MODE	PORT PIN (Px.y) STATE
0	0	Input	Three-State
0	1	Input	Weak pullup high
1	0	Output	Strong drive low
1	1	Output	Strong drive high

6.1 GPIO and External Interrupt Register Descriptions

The following peripheral registers are used to control the general-purpose I/O and external interrupt features specific to the MAXQ2010. Addresses of registers are given as “Mx[yy],” where x is the module number (from 0 to 15 decimal) and yy is the register index (from 00h to 1Fh hexadecimal). Fields in the bit definition tables are defined as follows:

- **Name:** Symbolic names of bits or bit fields in this register.
- **Reset:** The value of each bit in this register following a standard reset. If this field reads “unchanged,” the given bit is unaffected by standard reset. If this field reads “s,” the given bit does not have a fixed 0 or 1 reset value because its value is determined by another internal state or external condition.
- **POR:** If present this field defines the value of each bit in this register following a power-on reset (as opposed to a standard reset). Some bits are unaffected by standard resets and are set/cleared by POR only.
- **Access:** Bits can be read-only (r) or read/write (rw). Any special restrictions or conditions that could apply when reading or writing this bit are detailed in the bit description.

6.1.1 Port 0 Direction Register (PD0, M0[10h])

Bit #	7	6	5	4	3	2	1	0
Name	PD0							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Each of the bits in this register controls the input/output direction of a port pin (P0.0 to P0.7) as follows:

0 = The port pin is in input mode, either with a weak pullup (if PO = 1) or three-stated (if PO = 0).

1 = The port pin is in output mode, with the output level to drive given by PO.

6.1.2 Port 1 Direction Register (PD1, M0[11h])

Bit #	7	6	5	4	3	2	1	0
Name	PD1							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Each of the bits in this register controls the input/output direction of a port pin (P1.0 to P1.7) as follows:

0 = The port pin is in input mode, either with a weak pullup (if PO = 1) or three-stated (if PO = 0).

1 = The port pin is in output mode, with the output level to drive given by PO.

6.1.3 Port 2 Direction Register (PD2, M0[12h])

Bit #	7	6	5	4	3	2	1	0
Name	PD2							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Each of the bits in this register controls the input/output direction of a port pin (P2.0 to P2.7) as follows:

0 = The port pin is in input mode, either with a weak pullup (if PO = 1) or three-stated (if PO = 0).

1 = The port pin is in output mode, with the output level to drive given by PO.

6.1.4 Port 3 Direction Register (PD3, M0[13h])

Bit #	7	6	5	4	3	2	1	0
Name	PD3							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Each of the bits in this register controls the input/output direction of a port pin (P3.0 to P3.7) as follows:

0 = The port pin is in input mode, either with a weak pullup (if PO = 1) or three-stated (if PO = 0).

1 = The port pin is in output mode, with the output level to drive given by PO.

6.1.5 Port 4 Direction Register (PD4, M1[10h])

Bit #	7	6	5	4	3	2	1	0
Name	PD4							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Each of the bits in this register controls the input/output direction of a port pin (P4.0 to P4.7) as follows:

0 = The port pin is in input mode, either with a weak pullup (if PO = 1) or three-stated (if PO = 0).

1 = The port pin is in output mode, with the output level to drive given by PO.

6.1.6 Port 5 Direction Register (PD5, M1[11h])

Bit #	7	6	5	4	3	2	1	0
Name	—	PD5						
Reset	0	0	0	0	0	0	0	0
Access	r	rw	rw	rw	rw	rw	rw	rw

Each of the bits in this register controls the input/output direction of a port pin (P5.0 to P5.6) as follows:

0 = The port pin is in input mode, either with a weak pullup (if PO = 1) or three-stated (if PO = 0).

1 = The port pin is in output mode, with the output level to drive given by PO.

6.1.7 Port 6 Direction Register (PD6, M1[12h])

Bit #	7	6	5	4	3	2	1	0
Name	PD6							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Each of the bits in this register controls the input/output direction of a port pin (P6.0 to P6.7) as follows:

0 = The port pin is in input mode, either with a weak pullup (if PO = 1) or three-stated (if PO = 0).

1 = The port pin is in output mode, with the output level to drive given by PO.

6.1.8 Port 0 Output Register (PO0, M0[00h])

Bit #	7	6	5	4	3	2	1	0
Name	PO0							
Reset	1	1	1	1	1	1	1	1
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0: Port 0 Output. This register stores the data that is output on any of the pins of port 0 that have been defined as output pins. If the port pins are in input mode, this register controls the weak pullup for each pin. Changing the data direction of any pins for this port (through register PD0) does not affect the value in this register.

6.1.9 Port 1 Output Register (PO1, M0[01h])

Bit #	7	6	5	4	3	2	1	0
Name	PO1							
Reset	1	1	1	1	1	1	1	1
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0: Port 1 Output. This register stores the data that is output on any of the pins of port 1 that have been defined as output pins. If the port pins are in input mode, this register controls the weak pullup for each pin. Changing the data direction of any pins for this port (through register PD1) does not affect the value in this register.

6.1.10 Port 2 Output Register (PO2, M0[02h])

Bit #	7	6	5	4	3	2	1	0
Name	PO2							
Reset	1	1	1	1	1	1	1	1
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0: Port 2 Output. This register stores the data that is output on any of the pins of port 2 that have been defined as output pins. If the port pins are in input mode, this register controls the weak pullup for each pin. Changing the data direction of any pins for this port (through register PD2) does not affect the value in this register.

6.1.11 Port 3 Output Register (PO3, M0[03h])

Bit #	7	6	5	4	3	2	1	0
Name	PO3							
Reset	1	1	1	1	1	1	1	1
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0: Port 3 Output. This register stores the data that is output on any of the pins of port 3 that have been defined as output pins. If the port pins are in input mode, this register controls the weak pullup for each pin. Changing the data direction of any pins for this port (through register PD3) does not affect the value in this register.

6.1.12 Port 4 Output Register (PO4, M1[00h])

Bit #	7	6	5	4	3	2	1	0
Name	PO4							
Reset	1	1	1	1	1	1	1	1
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0: Port 4 Output. This register stores the data that is output on any of the pins of port 4 that have been defined as output pins. If the port pins are in input mode, this register controls the weak pullup for each pin. Changing the data direction of any pins for this port (through register PD4) does not affect the value in this register.

6.1.13 Port 5 Output Register (PO5, M1[01h])

Bit #	7	6	5	4	3	2	1	0
Name	—	PO5						
Reset	1	1	1	1	1	1	1	1
Access	r	rw	rw	rw	rw	rw	rw	rw

Bits 7:0: Port 5 Output. This register stores the data that is output on any of the pins of port 5 that have been defined as output pins. If the port pins are in input mode, this register controls the weak pullup for each pin. Changing the data direction of any pins for this port (through register PD5) does not affect the value in this register.

6.1.14 Port 6 Output Register (PO6, M1[02h])

Bit #	7	6	5	4	3	2	1	0
Name	PO6							
Reset	1	1	1	1	1	1	1	1
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0: Port 6 Output. This register stores the data that is output on any of the pins of port 6 that have been defined as output pins. If the port pins are in input mode, this register controls the weak pullup for each pin. Changing the data direction of any pins for this port (through register PD6) does not affect the value in this register.

6.1.15 Port 0 Input Register (PI0, M0[08h])

Bit #	7	6	5	4	3	2	1	0
Name	PI0							
Reset	s	s	s	s	s	s	s	s
Access	r	r	r	r	r	r	r	r

Each of the read-only bits in this register reflects the logic state present at the corresponding port pin.

6.1.16 Port 1 Input Register (PI1, M0[09h])

Bit #	7	6	5	4	3	2	1	0
Name	PI1							
Reset	s	s	s	s	s	s	s	s
Access	r	r	r	r	r	r	r	r

Each of the read-only bits in this register reflects the logic state present at the corresponding port pin.

6.1.17 Port 2 Input Register (PI2, M0[0Ah])

Bit #	7	6	5	4	3	2	1	0
Name	PI2							
Reset	s	s	s	s	s	s	s	s
Access	r	r	r	r	r	r	r	r

Each of the read-only bits in this register reflects the logic state present at the corresponding port pin.

6.1.18 Port 3 Input Register (PI3, M0[0Bh])

Register Name	**PI3**
Register Description	Port 3 Input Register
Register Address	M0[0Bh]

Bit #	7	6	5	4	3	2	1	0
Name	PI3							
Reset	s	s	s	s	s	s	s	s
Access	r	r	r	r	r	r	r	r

Each of the read-only bits in this register reflects the logic state present at the corresponding port pin.

6.1.19 Port 4 Input Register (PI4, M1[08h])

Bit #	7	6	5	4	3	2	1	0
Name	PI4							
Reset	s	s	s	s	s	s	s	s
Access	r	r	r	r	r	r	r	r

Each of the read-only bits in this register reflects the logic state present at the corresponding port pin.

6.1.20 Port 5 Input Register (PI5, M1[09h])

Bit #	7	6	5	4	3	2	1	0
Name	—	PI5						
Reset	0	s	s	s	s	s	s	s
Access	r	r	r	r	r	r	r	r

Each of the read-only bits in this register reflects the logic state present at the corresponding port pin.

6.1.21 Port 6 Input Register (PI6, M1[0Ah])

Bit #	7	6	5	4	3	2	1	0
Name	PI6							
Reset	s	s	s	s	s	s	s	s
Access	r	r	r	r	r	r	r	r

Each of the read-only bits in this register reflects the logic state present at the corresponding port pin.

6.1.22 External Interrupt Flag 0 Register (EIF0, M0[04h])

Bit #	7	6	5	4	3	2	1	0
Name	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Each bit in this register is set when a negative or positive edge (depending on the ITn bit setting) is detected on the corresponding interrupt pin. Once an external interrupt has been detected, the interrupt flag bit remains set until cleared by software or a reset. Setting any of these bits causes the corresponding interrupt to trigger if it is enabled to do so.

Bit 7: External Interrupt 7 Edge Detect (IE7)

Bit 6: External Interrupt 6 Edge Detect (IE6)

Bit 5: External Interrupt 5 Edge Detect (IE5)

Bit 4: External Interrupt 4 Edge Detect (IE4)

Bit 3: External Interrupt 3 Edge Detect (IE3)

Bit 2: External Interrupt 2 Edge Detect (IE2)

Bit 1: External Interrupt 1 Edge Detect (IE1)

Bit 0: External Interrupt 0 Edge Detect (IE0)

6.1.23 External Interrupt Flag 1 Register (EIF1, M1[04h])

Bit #	7	6	5	4	3	2	1	0
Name	—	IE14	IE13	IE12	IE11	IE10	IE9	IE8
Reset	0	0	0	0	0	0	0	0
Access	r	rw	rw	rw	rw	rw	rw	rw

Each bit in this register is set when a negative or positive edge (depending on the ITn bit setting) is detected on the corresponding interrupt pin. Once an external interrupt has been detected, the interrupt flag bit remains set until cleared by software or a reset. Setting any of these bits causes the corresponding interrupt to trigger if it is enabled to do so.

Bit 7: Reserved

Bit 6: External Interrupt 14 Edge Detect (IE14)

Bit 5: External Interrupt 13 Edge Detect (IE13)

Bit 4: External Interrupt 12 Edge Detect (IE12)

Bit 3: External Interrupt 11 Edge Detect (IE11)

Bit 2: External Interrupt 10 Edge Detect (IE10)

Bit 1: External Interrupt 9 Edge Detect (IE9)

Bit 0: External Interrupt 8 Edge Detect (IE8)

6.1.24 External Interrupt Flag 2 Register (EIF2, M1[06h])

Bit #	7	6	5	4	3	2	1	0
Name	IE22	IE21	IE20	IE19	IE18	IE17	IE16	IE15
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Each bit in this register is set when a negative or positive edge (depending on the ITn bit setting) is detected on the corresponding interrupt pin. Once an external interrupt has been detected, the interrupt flag bit remains set until cleared by software or a reset. Setting any of these bits causes the corresponding interrupt to trigger if it is enabled to do so.

Bit 7: External Interrupt 22 Edge Detect (IE22)

Bit 6: External Interrupt 21 Edge Detect (IE21)

Bit 5: External Interrupt 20 Edge Detect (IE20)

Bit 4: External Interrupt 19 Edge Detect (IE19)

Bit 3: External Interrupt 18 Edge Detect (IE18)

Bit 2: External Interrupt 17 Edge Detect (IE17)

Bit 1: External Interrupt 16 Edge Detect (IE16)

Bit 0: External Interrupt 15 Edge Detect (IE15)

6.1.25 External Interrupt Enable 0 Register (EIE0, M0[05h])

Bit #	7	6	5	4	3	2	1	0
Name	EX7	EX6	EX5	EX4	EX3	EX2	EX1	EX0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Each bit in this register controls the enable for one external interrupt. If the bit is set to 1, the interrupt is enabled (if it is not otherwise masked). If the bit is set to 0, its corresponding interrupt is disabled.

Bit 7: External Interrupt 7 Enable (EX7)

Bit 6: External Interrupt 6 Enable (EX6)

Bit 5: External Interrupt 5 Enable (EX5)

Bit 4: External Interrupt 4 Enable (EX4)

Bit 3: External Interrupt 3 Enable (EX3)

Bit 2: External Interrupt 2 Enable (EX2)

Bit 1: External Interrupt 1 Enable (EX1)

Bit 0: External Interrupt 0 Enable (EX0)

6.1.26 External Interrupt Enable 1 Register (EIE1, M1[05h])

Bit #	7	6	5	4	3	2	1	0
Name	—	EX14	EX13	EX12	EX11	EX10	EX9	EX8
Reset	0	0	0	0	0	0	0	0
Access	r	rw	rw	rw	rw	rw	rw	rw

Each bit in this register controls the enable for one external interrupt. If the bit is set to 1, the interrupt is enabled (if it is not otherwise masked). If the bit is set to 0, its corresponding interrupt is disabled.

Bit 7: Reserved

Bit 6: External Interrupt 14 Enable (EX14)

Bit 5: External Interrupt 13 Enable (EX13)

Bit 4: External Interrupt 12 Enable (EX12)

Bit 3: External Interrupt 11 Enable (EX11)

Bit 2: External Interrupt 10 Enable (EX10)

Bit 1: External Interrupt 9 Enable (EX9)

Bit 0: External Interrupt 8 Enable (EX8)

6.1.27 External Interrupt Enable 2 Register (EIE2, M1[07h])

Bit #	7	6	5	4	3	2	1	0
Name	EX22	EX21	EX20	EX19	EX18	EX17	EX16	EX15
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Each bit in this register controls the enable for one external interrupt. If the bit is set to 1, the interrupt is enabled (if it is not otherwise masked). If the bit is set to 0, its corresponding interrupt is disabled.

Bit 7: External Interrupt 22 Enable (EX22)

Bit 6: External Interrupt 21 Enable (EX21)

Bit 5: External Interrupt 20 Enable (EX20)

Bit 4: External Interrupt 19 Enable (EX19)

Bit 3: External Interrupt 18 Enable (EX18)

Bit 2: External Interrupt 17 Enable (EX17)

Bit 1: External Interrupt 16 Enable (EX16)

Bit 0: External Interrupt 15 Enable (EX15)

6.1.28 External Interrupt Edge Select 0 Register (EIES0, M0[0Ch])

Bit #	7	6	5	4	3	2	1	0
Name	IT7	IT6	IT5	IT4	IT3	IT2	IT1	IT0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Each bit in this register controls the edge select mode for an external interrupt as follows:

0 = The external interrupt triggers on a rising (positive) edge.

1 = The external interrupt triggers on a negative (falling) edge.

Bit 7: Edge Select for External Interrupt 7 (IT7)

Bit 6: Edge Select for External Interrupt 6 (IT6)

Bit 5: Edge Select for External Interrupt 5 (IT5)

Bit 4: Edge Select for External Interrupt 4 (IT4)

Bit 3: Edge Select for External Interrupt 3 (IT3)

Bit 2: Edge Select for External Interrupt 2 (IT2)

Bit 1: Edge Select for External Interrupt 1 (IT1)

Bit 0: Edge Select for External Interrupt 0 (IT0)

6.1.29 External Interrupt Edge Select 1 Register (EIES1, M1[0Bh])

Bit #	7	6	5	4	3	2	1	0
Name	—	IT14	IT13	IT12	IT11	IT10	IT9	IT8
Reset	0	0	0	0	0	0	0	0
Access	r	rw	rw	rw	rw	rw	rw	rw

Each bit in this register controls the edge select mode for an external interrupt as follows:

0 = The external interrupt triggers on a rising (positive) edge.

1 = The external interrupt triggers on a negative (falling) edge.

Bit 7: Reserved

Bit 6: Edge Select for External Interrupt 14 (IT14)

Bit 5: Edge Select for External Interrupt 13 (IT13)

Bit 4: Edge Select for External Interrupt 12 (IT12)

Bit 3: Edge Select for External Interrupt 11 (IT11)

Bit 2: Edge Select for External Interrupt 10 (IT10)

Bit 1: Edge Select for External Interrupt 9 (IT9)

Bit 0: Edge Select for External Interrupt 8 (IT8)

6.1.30 External Interrupt Edge Select 2 Register (EIES2, M1[0Ch])

Bit #	7	6	5	4	3	2	1	0
Name	IT22	IT21	IT20	IT19	IT18	IT17	IT16	IT15
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Each bit in this register controls the edge select mode for an external interrupt as follows:

0 = The external interrupt triggers on a rising (positive) edge.

1 = The external interrupt triggers on a negative (falling) edge.

Bit 7: Edge Select for External Interrupt 22 (IT22)

Bit 6: Edge Select for External Interrupt 21 (IT21)

Bit 5: Edge Select for External Interrupt 20 (IT20)

Bit 4: Edge Select for External Interrupt 19 (IT19)

Bit 3: Edge Select for External Interrupt 18 (IT18)

Bit 2: Edge Select for External Interrupt 17 (IT17)

Bit 1: Edge Select for External Interrupt 16 (IT16)

Bit 0: Edge Select for External Interrupt 15 (IT15)

6.2 GPIO and External Interrupt Code Examples

6.2.1 GPIO Example 1: Driving Outputs on Port 0

```
move PO0, #000h        ; Set all outputs low
move PD0, #0FFh        ; Set all P0 pins to output mode
```

6.2.2 GPIO Example 2: Receiving Inputs on Port 1

```
move PO1, #0FFh        ; Set weak pullups ON on all pins
move PD1, #000h        ; Set all P1 pins to input mode

nop                    ; Wait for external source to drive P1 pins

move Acc, PI1          ; Get input values from P1 (will return FF if
                       ; no other source drives the pins low)
```

6.2.3 External Interrupt Example: Handling Interrupt on INT10/P5.2

```
  move  PD5.2, #0             ; Set P5.2 to input mode
  move  PO5.2, #1             ; Enable weak pullup

  move  IV, #intHandler       ; Set interrupt vector
  move  IMR.1, #1             ; Enable interrupts for module 1
  move  EIE1.2, #1            ; Enable external interrupt 10
  move  EIES1.2, #1           ; Set INT10 to trigger on falling edge

  move  EIF1.2, #0            ; Force interrupt flag cleared
  nop
  nop
  nop
  nop
  move  IC.0, #1              ; Enable interrupts globally

  sjump $                     ; Wait for int to trigger (press SW3)

intHandler:
  move  EIF1.2, #0            ; Clear interrupt flag
  nop
  nop                         ; [SET BREAKPOINT HERE]
  nop
  reti
```

ADDENDUM TO SECTION 7: TIMER/COUNTER 0 MODULE

The MAXQ2010 does not provide this peripheral module.

ADDENDUM TO SECTION 8: TIMER/COUNTER 1 MODULE

The MAXQ2010 does not provide this peripheral module.

ADDENDUM TO SECTION 9: TIMER/COUNTER 2 MODULE

The MAXQ2010 does not provide this peripheral module.

ADDENDUM TO SECTION 10: SERIAL I/O MODULE

The MAXQ2010 provides two serial universal synchronous/asynchronous receiver-transmitter (USART) interfaces (serial 0 and 1) that operate as described in the *MAXQ Family User’s Guid*e.

10.1 Serial USART I/O Pins and Control Registers

Table 10-1. Serial USART Input and Output Pins

SERIAL USART FUNCTION	PIN	MULTIPLEXED WITH GPIO
RX0: Serial 0 Receive	68	P5.0
TX0: Serial 0 Transmit	67	P5.1
RX1: Serial 1 Receive	28	P6.4
TX1: Serial 1 Transmit	25	P6.5

Table 10-2. Serial USART Control Registers

REGISTER	ADDRESS	FUNCTION
SCON0	M3[04h]	Serial Port 0 Control Register. Serial port mode, receive enable, 9th bit control, and interrupt flags.
SBUF0	M3[05h]	Serial Port 0 Data Buffer. Input and output data buffer.
SMD0	M3[08h]	Serial Port 0 Mode Register. Controls baud rate, interrupt enable, and framing error detection.
PR0	M3[09h]	Serial Port 0 Phase Register. Contains counter reload value for baud-rate generation.
SCON1	M3[06h]	Serial Port 1 Control Register. Serial port mode, receive enable, 9th bit control, and interrupt flags.
SBUF1	M3[07h]	Serial Port 1 Data Buffer. Input and output data buffer.
SMD1	M3[0Ah]	Serial Port 1 Mode Register. Controls baud rate, interrupt enable, and framing error detection.
PR1	M3[0Bh]	Serial Port 1 Phase Register. Contains counter reload value for baud-rate generation.

10.2 Serial USART Code Examples

10.2.1 Serial USART Example: Echo Characters in 10-Bit Asynchronous Mode

```
  move SCON0.6, #1       ; Set to mode 1 (10-bit asynchronous)
  move SCON0.4, #1       ; Enable receiver
  move SMD0.1,  #1       ; Baud rate = 16 x baud clock
  move PR0, #007DDh      ; P = 2^21 * 9600 / 10.000MHz

  move SCON0.0, #0       ; Clear received character flag
  move SCON0.1, #0       ; Clear transmit character flag


  move Acc, #0Dh
  call TxChar0
  move Acc, #0Ah
  call TxChar0
  move Acc, #’>’
  call TxChar0
  move Acc, #’ ‘
  call TxChar0
```

```
mainLoop:
   call RxChar0
   call TxChar0
   jump mainLoop

;=============================================================================
;=
;= TxChar0 - Outputs a character to serial port 0.
;=
;= Inputs : Acc - Character to send.
;=

TxChar0:
   move  SBUF0, Acc       ; Send character
TxChar0_Loop:
   move  C, SCON0.1       ; Check transmit flag
   sjump NC, TxChar0_Loop ; Stall until last transmit has completed
   move  SCON0.1, #0      ; Clear the transmit flag
   ret

;=============================================================================
;=
;= RxChar0 - Receives a character from serial port 0.
;=
;= Outputs : Acc - Character received.
;=

RxChar0:
   move  C, SCON0.0       ; Wait for receive flag to be set to 1
   sjump NC, RxChar0
   move  Acc, SBUF0       ; Get received character
   move  SCON0.0, #0      ; Clear receive interrupt flag
   ret
```

ADDENDUM TO SECTION 11: SERIAL PERIPHERAL INTERFACE (SPI) MODULE

The MAXQ2010 provides a serial peripheral interface (SPI) module, which operates as described in the *MAXQ Family User's Guide*.

11.1 SPI Input/Output Pins and Control Registers

Table 11-1. SPI Input and Output Pins

SPI INTERFACE FUNCTION	PIN	MULTIPLEXED WITH GPIO
$\overline{\text{SSEL}}$: Slave Select	60	P5.3
SCLK: Slave Clock	58	P5.5
MOSI: Master Out-Slave In	59	P5.4
MISO: Master In-Slave Out	57	P5.6

Table 11-2. SPI Control Registers

REGISTER	ADDRESS	FUNCTION
SPICN	M1[15h]	SPI Control Register. Enable, master/slave-mode select, and status and interrupt flags.
SPICF	M1[16h]	SPI Configuration Register. Clock polarity/phase, character length, and interrupt enable.
SPICK	M1[17h]	SPI Clock Register. Master baud rate = 0.5 x Sysclk/(SPICK + 1).
SPIB	M1[03h]	SPI Data Buffer. Writes go to the SPI write buffer; reads come from the SPI read buffer.

11.2 SPI Code Examples

11.2.1 SPI Example 1: Transmitting Data in Master Mode

```
  move PD5.2, #1        ; Chip select for slave device
  move PO5.2, #1        ; Start high

  move SPICN, #03h      ; Enable SPI in master mode
  move SPICF, #00h      ; Sample data at clock rising edge, 8 bit character
  move SPICK, #63       ; SPI clock = sysclk/128

  move PO5.2, #0        ; Drive chip select low
  move SPIB,  #12h      ; Transmit byte
  call waitXfer
  move SPIB,  #34h      ; Transmit byte
  call waitXfer
  move PO5.2, #1        ; Release chip select

  ....

waitXfer:
  move C, SPICN.6       ; Wait for transfer to complete
  jump NC, waitXfer
  move SPICN.6, #0      ; Clear transfer flag
  ret
```

11.2.2 SPI Example 2: Receiving Data in Slave Mode

```
   move SPICN, #01h       ; Enable SPI in slave mode
   move SPICF, #00h       ; Sample data at clock rising edge, 8 bit character

   call getByte
   move A[0], GR
   call getByte
   move A[1], GR
   call getByte
   move A[2], GR
   call getByte
   move A[3], GR

   ...

getByte:
   move C, SPICN.6        ; Wait for transfer to complete
   jump NC, getByte
   move SPICN.6, #0       ; Clear transfer flag
   move GR, SPIB          ; Get character
   ret
```

ADDENDUM TO SECTION 12: HARDWARE MULTIPLIER MODULE

The MAXQ2010 provides a hardware multiplier module that provides the following features (detailed in the *MAXQ Family User's Guide*):

- Completes a 16-bit x 16-bit multiply-accumulate or multiply-subtract operation in a single cycle
- Includes 48-bit accumulator
- Supports seven different multiplication operations

  Unsigned 16-bit multiply

  Unsigned 16-bit multiply and accumulate

  Unsigned 16-bit multiply and subtract

  Signed 16-bit multiply

  Signed 16-bit multiply and negate

  Signed 16-bit multiply and accumulate

  Signed 16-bit multiply and subtract

12.1 Hardware Multiplier Control Registers

The associated registers for this module are listed in Table 12-1.

Table 12-1. Hardware Multiplier Control Registers

REGISTER	ADDRESS	FUNCTION
MCNT	M2[00h]	Multiplier Control Register. Controls the operation and mode selection for the multiplier.
MA	M2[01h]	Multiplier Operand A Register. Input register for the multiplier operations.
MB	M2[02h]	Multiplier Operand B Register. Input register for the multiplier operations.
MC2	M2[03h]	Multiplier Accumulate Register 2. Contains bits 32 to 47 of the accumulator.
MC1	M2[04h]	Multiplier Accumulate Register 1. Contains bits 16 to 31 of the accumulator.
MC0	M2[05h]	Multiplier Accumulate Register 0. Contains bits 0 to 15 of the accumulator.
MC1R	M2[08h]	Multiplier Read Register 1. Contains bits 16 to 31 of the last multiply operation result.
MC0R	M2[09h]	Multiplier Read Register 0. Contains bits 0 to 15 of the last multiply operation result.

12.2 Hardware Multiplier Code Examples

12.2.1 Hardware Multiplier Example: Multiply and Square/Accumulate

```
move  MCNT,#021h      ; Unsigned multiply, no accumulate
move  MA,  #00155h
move  MB,  #000AAh    ; MC[2:0] = 00_0000_E272h

move  MCNT,#012h      ; Square single operand and accumulate
move  MA,  #000FFh    ; MC[2:0] = 00_0001_E073h
move  MB,  #00099h    ; MC[2:0] = 00_0002_3BE4h
```

ADDENDUM TO SECTION 13: 1-Wire BUS MASTER

The MAXQ2010 does not provide this peripheral module.

ADDENDUM TO SECTION 14: REAL-TIME CLOCK MODULE

The MAXQ2010 provides a real-time clock (RTC) that operates as described in the *MAXQ Family User's Guide*. Specific functions provided by the MAXQ2010 are as follows:

- Time-of-day alarm
- Subsecond interval alarm (8-bit width to support up to one-second intervals)
- Trim compensation function (controlled by RTRM register)
- Square-wave output for frequency generation and testing, controlled by the SQE and FT bits

14.1 RTC Pins and Control Registers

The associated pins and registers for this module are listed in Table 14-1 and Table 14-2.

Table 14-1. RTC Output Pins

SPI INTERFACE FUNCTION	PIN	MULTIPLEXED WITH GPIO
SQW: Square-Wave Output	61	P5.2

Table 14-2. RTC Control Registers

REGISTER	ADDRESS	FUNCTION
RTRM	M0[18h]	RTC Trim Register. Contains the 7-bit signed trim calibration value for the RTC.
RCNT	M0[19h]	RTC Control Register. Sets modes and alarm enables for the clock.
RTSS	M0[1Ah]	RTC Subsecond Counter Register. Contains the 1/256 subsecond count.
RTSH	M0[1Bh]	RTC Second Counter High Register. Contains the high-order byte of the 32-bit second count.
RTSL	M0[1Ch]	RTC Second Counter Low Register. Contains the low-order byte of the 32-bit second count.
RSSA	M0[1Dh]	RTC Subsecond Alarm Register. Contains the subsecond alarm reload value.
RASH	M0[1Eh]	RTC Time-of-Day Alarm High Register. Contains the high-order 8 bits of the time-of-day alarm value.
RASL	M0[1Fh]	RTC Time-of-Day Alarm Low Register. Contains the low-order 16 bits of the time-of-day alarm value.

14.2 RTC Operation Overview

The binary RTC module keeps the time of day in absolute seconds with 1/256-second resolution. The 32-bit second counter can count up to approximately 136 years and be translated to calendar format by application software. A time-of-day alarm and independent subsecond alarm can cause an interrupt or wake the MAXQ2010 from stop mode. See Figure 14-1.

The independent subsecond alarm runs from the same RTC and allows the user application to support interrupts with a minimum interval of approximately 3.9ms with a maximum interval of one second. This creates an additional timer that can be used to measure long periods of time without performance degradation.

Traditionally, long time periods have been measured using multiple interrupts from shorter interrupt intervals. Each timer interrupt required servicing, with each accompanying interruption slowing system operation. By using the RTC subsecond timer as a long-period timer, only one interrupt is needed, eliminating the performance hit associated with using a shorter timer.

An internal crystal oscillator clocks the RTC using integrated 6pF load capacitors, and yields the best performance when mated with a 32.768kHz crystal rated for a 6pF load. No external load capacitors are required.

Higher accuracy can be obtained by supplying an external clock source to the RTC.

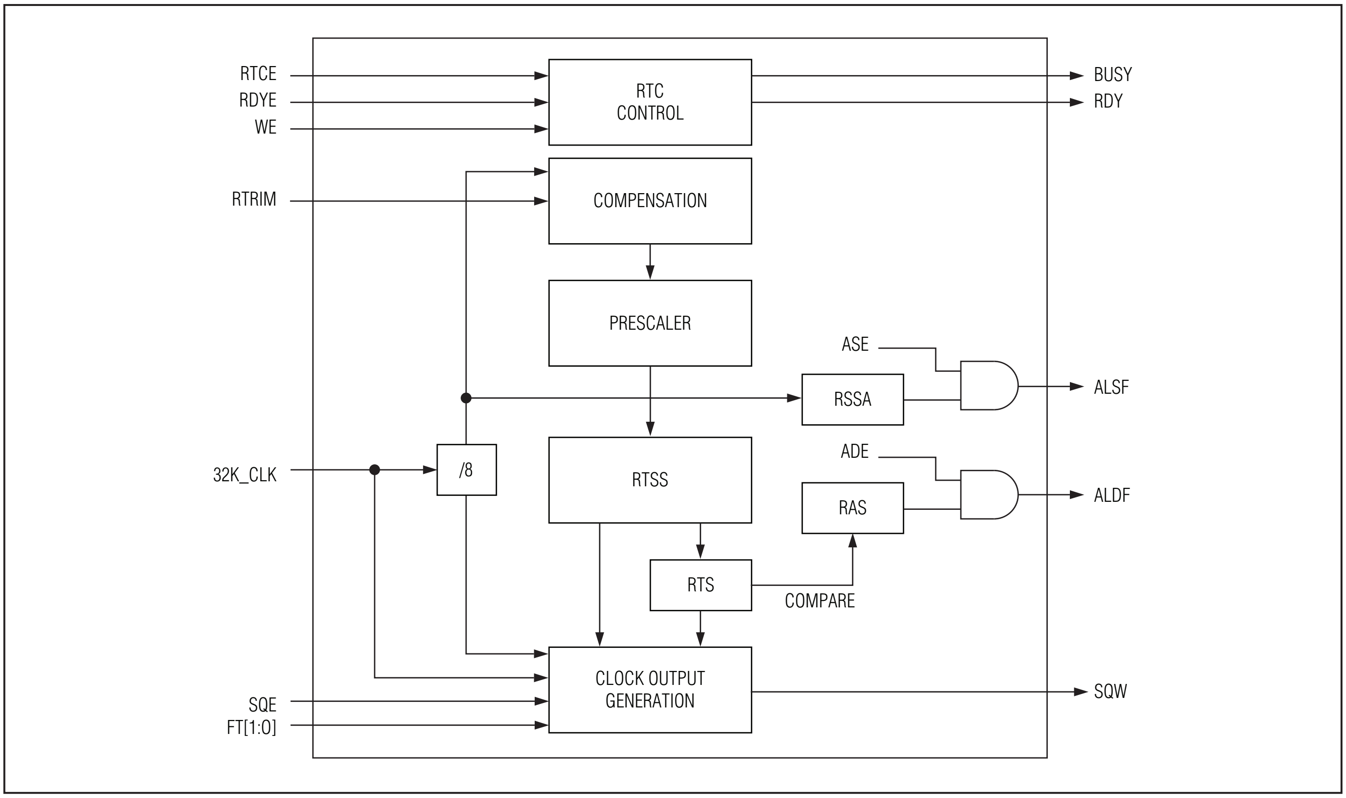


Figure 14-1. RTC Block Diagram

14.3 RTC Trim Operation

The uncompensated accuracy of the RTC is a function of the attached crystal oscillator (and its respective temperature drift characteristics within the end system). To accommodate those applications requiring high accuracy, a digital trim facility is made accessible to the user. The trim facility, instead of adjusting the internal capacitive load, allows extra clocks to be inserted, removed at the 4096Hz stage in the prescaler. Four trim bits (TRM[3:0]) are used to control the trim adjustment, and the sign bit (TSGN) designates whether pulses should be added (TSGN = 1) or subtracted (TSGN = 0) from the prescaler count. Every 10 seconds (at the second boundary), 4K clocks can be added or subtracted from the prescaler. See Figure 14-2.

The minimum adjustment TRM[3:0] = 01h would result in a time adjustment of 1/(4096x10) = 24.41ppm (244μs) in 10 seconds. The maximum adjustment would be achieved by programming TRM[3:0] = 0Fh. This would result in an adjustment of 366.2ppm (3.662ms). This range of adjustment should be satisfactory to cover the temperature drift characteristics of most 32kHz crystals over the industrial temperature range.

Normally, the prescaler counts 16 clocks of the 4K input to advance the RTSS subsecond count by 1. When TSGN = 1, the prescaler count limit is increased by the value as indicated in RTRM[3:0]. For example, if the RTRM = 1000 0011b, instead of counting 16 clocks of the 4K input, the prescaler counts 16 + 3 = 19 of 4K clocks before advancing the RTSS subsecond count by 1, effectively adding 732μs of delay and thus slowing down the clock. See Figure 14-3.

Similarly, when TSGN = 0, the prescaler count limit is decremented by the value specified in RTRM[3:0]. For example, if the RTRM = 0000 0011b, instead of counting 16 clocks of the 4K input, the prescaler counts 16 - 3 = 13 of the 4K clocks before advancing the RTSS subsecond count by 1, effectively advancing the RTC ahead by 732μs and hence speeding up the clock. See Figure 14-4.

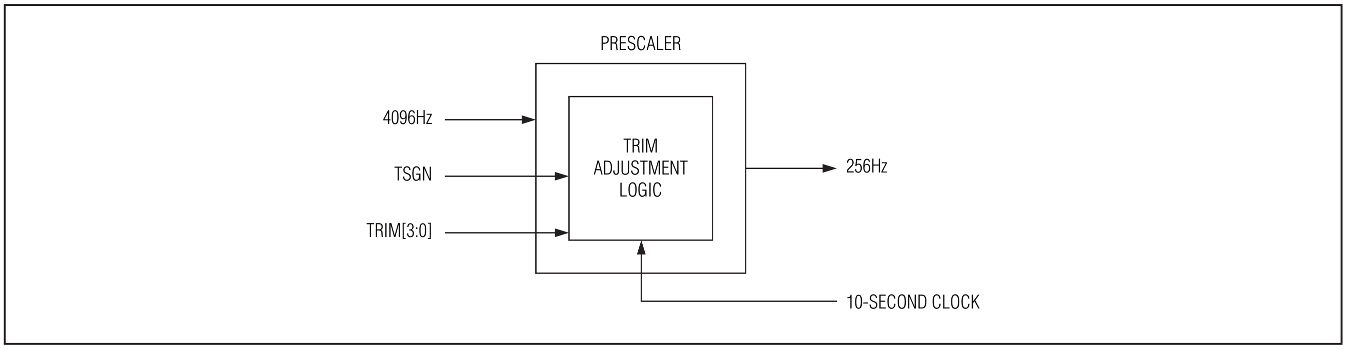


Figure 14-2. RTC Prescaler and Trim

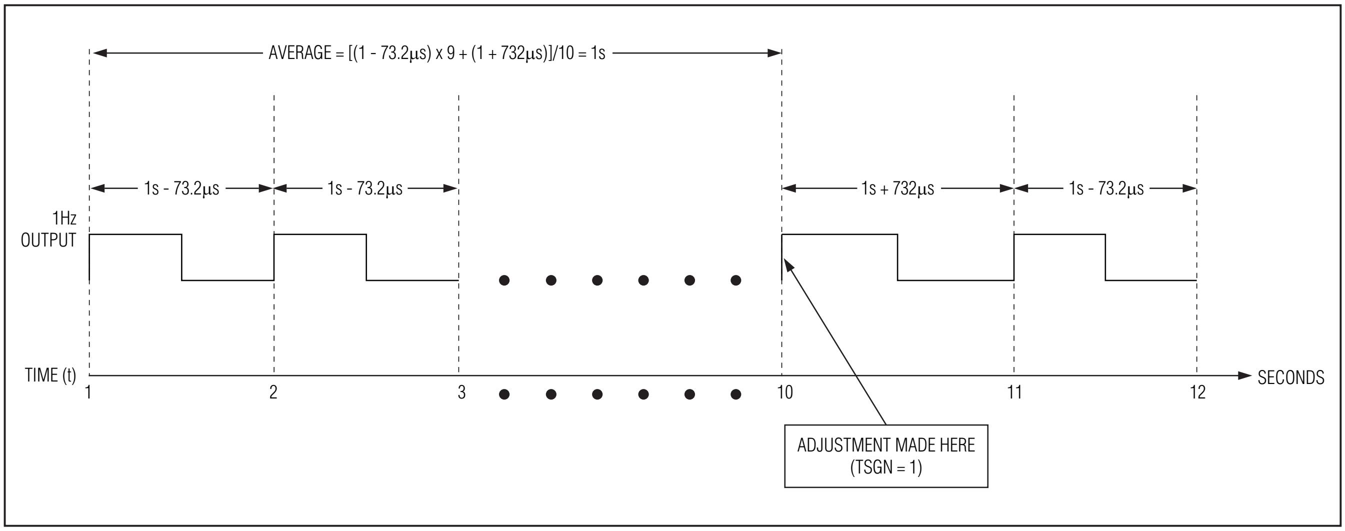


Figure 14-3. RTC Trim Adjustment (TSGN = 1)

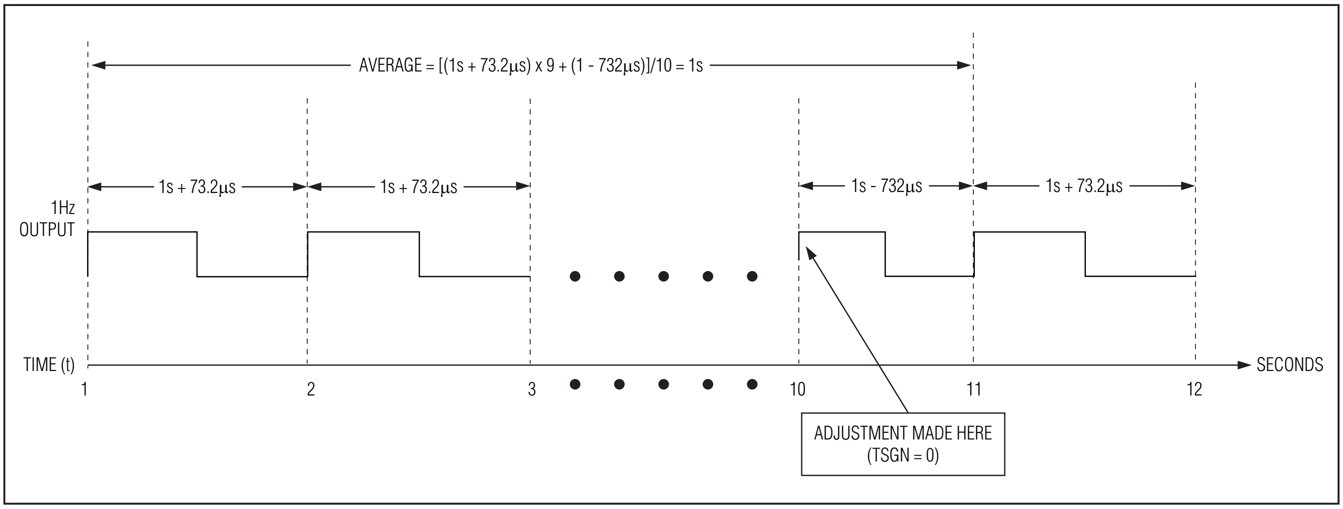


Figure 14-4. RTC Trim Adjustment (TSGN = 0)

14.4 RTC Register Descriptions

Addresses of registers are given as "Mx[yy]," where x is the module number (from 0 to 15 decimal) and yy is the register index (from 00h to 1Fh hexadecimal). Fields in the bit definition tables are defined as follows:

- **Name:** Symbolic names of bits or bit fields in this register.
- **Reset:** The value of each bit in this register following a standard reset. If this field reads "unchanged," the given bit is unaffected by standard reset. If this field reads "s," the given bit does not have a fixed 0 or 1 reset value because its value is determined by another internal state or external condition.
- **POR:** If present this field defines the value of each bit in this register following a power-on reset (as opposed to a standard reset). Some bits are unaffected by standard resets and are set/cleared by POR only.
- **Access:** Bits can be read-only (r) or read/write (rw). Any special restrictions or conditions that could apply when reading or writing this bit are detailed in the bit description.

14.4.1 RTC Trim Register (RTRM, M0[18h])

Bit #	7	6	5	4	3	2	1	0
Name	TSGN	—	—	—	TRM			
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Note 1: *This register is reset by power-on reset only.*

Note 2: *Write to this register is ignored when WE = 0 or RCNT.BUSY = 1.*

Bit 7: RTC Trim Sign Bit (TSGN). This register bit selects whether 32K clocks are inserted (TSGN = 1) or removed (TSGN = 0).

Bits 6:4: Reserved. Read returns zero.

Bits 3:0: RTC Trim Calibration Register (TRM[3:0]). These register bits provide a binary value between 00h to 0Fh, which is used for adjusting 32K clocks insertion/removal. At every 10-second interval, a number of 32K clocks equal to the RTRM[3:0] numeric value x 8 is inserted/removed from the RTC counter depending on the TSGN sign bit.

14.4.2 RTC Control Register (RCNT, M0[19h])

Bit #	15	14	13	12	11	10	9	8
Name	WE	—	ACS	—	—	—	FT	SQE
Reset	0	1	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	ALSF	ALDF	RDYE	RDY	BUSY	ASE	ADE	RTCE
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	r	rw	rw	rw

Note 1: *Bits 0 and 13 are reset by power-on reset only.*

Note 2: *Bits 14, 12:4, and 2:1 are reset by any reset source.*

Note 3: *Bit 3 is set to a 1 on system reset.*

Note 4: *Bit 0 is only writable when WE = 1 and BUSY = 0.*

Note 5: *Bit 13 is only writable when RTCE = 0.*

Note 6: *Bits 9:6 and 2:1 are only writable when BUSY = 0.*

Bit 15: RTC Write Enable (WE). This register bit serves as a protection mechanism against undesirable writes to the RTCE bit and RTRM register. This bit must be set to a 1 to give write access to the RTRM register and the RTCE bit; otherwise (when WE bit = 0) these protected bits are read-only.

Bit 14: Reserved. Read returns 1 after reset, or last written value (when BUSY = 0).

Bit 13: Alternate Clock Select (ACS). This bit enables the use of the system clock to drive the RTC in place of the 32kHz clock. When the alternate clock is selected (ACS = 1), the RTC input clock is driven by system clock/128. This bit is provided for those applications where a 32kHz clock may not be present, or when the RTC module is intended to be used as a timer based on the system clock. This bit can only be changed when RTCE = 0. When ACS = 1, the RTC is effectively halted any time that the system clock is disabled (e.g., stop mode).

Bits 12:10: Reserved. Read returns zero.

Bit 9: RTC Frequency Test (FT). This register bit selects the frequency output on the SQW pin if the square-wave output is enabled (SQE = 1). Setting FT = 1 selects the RTC input clock/8 output (512Hz for 32.768kHz applied to 32KIN), while FT = 0 selects the RTC input clock/4096 (1Hz for 32.768kHz applied to 32KIN). This bit has no function when the square-wave output is disabled (SQE = 0).

Bit 8: RTC Square-Wave Output Enable (SQE). Setting this bit to 1 enables the frequency specified by FT to be outputted to the SQW pin. When cleared to 0, the SQW pin is not driven by the RTC.

Bit 7: Alarm Subsecond Flag (ALSF). This bit is set when the subsecond timer has been reloaded by the RSSA register. Setting the ALSF causes an interrupt request to the CPU if the ASE is set and interrupt is allowed at the system level. This flag must be cleared by software once set. This alarm is qualified as wake-up to the stop and the switchback function if its interrupt has not been masked.

Bit 6: Alarm Time-of-Day Flag (ALDF). This bit is set when the contents of RTSH and RTSL counter registers match the 20-bit value in the RASH and RASL alarm registers. Setting the ALDF causes an interrupt request to the CPU if the ADE is set and interrupt is allowed at the system level. This flag must be cleared by software once set. This alarm is qualified as wake-up to the stop and the switchback function if its interrupt has not been masked.

Bit 5: RTC Ready Enable (RDYE). Setting this bit to 1 allows a system interrupt to be generated when RDY becomes active (if interrupts are enabled globally and modularly). Clearing this bit to 0 disables the RDY interrupt.

Bit 4: RTC Ready (RDY). This bit is set to 1 by hardware when the RTC count registers update. It can be cleared to 0 by software at any time. It is also cleared to 0 by hardware just prior to an update of the RTC count register. This bit can generate an interrupt if the RDYE bit is set to 1

Bit 3: RTC Busy (BUSY). This bit is set to 1 by hardware when any of the following conditions occur:

1) System reset.

2) Software writes to RTC count registers or trim register.

3) Software changes RTCE, ASE, or ADE.

For conditions 2 and 3, the write or change should not be considered complete until hardware clears the BUSY bit. This is an indication that a 32kHz synchronized version of the register bit(s) is in place.

Bit 2: Alarm Subsecond Enable (ASE). The ASE bit is the RTC's subsecond timer enable and must be set to logic 1 for the subsecond alarm to generate a system interrupt request. When the ASE is cleared to logic 0, the subsecond alarm is disabled and no interrupt is generated, even if the alarm is set.

Bit 1: Alarm Time-of-Day Enable (ADE). The ADE bit is the RTC's time-of-day alarm enable and must be set to logic 1 for the alarm to generate a system interrupt request. When the ADE is cleared to logic 0, the time-of-day alarm is disabled and no interrupt is generated, even if the alarm is set.

Bit 0: RTC Enable (RTCE). Setting this bit to logic 1 activates the clocking by allowing the divided clock to the ripple counters. Clearing this bit to logic 0 disables the clock.

14.4.3 Real-Time Subsecond Counter Register (RTSS, M0[1Ah])

Bit #	7	6	5	4	3	2	1	0
Name	RTSS							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Note 1: *This register is cleared by power-on reset only.*

Note 2: *Read accessible when RDY = 1.*

Note 3: *Write accessible when RTCE = 0 and BUSY = 0.*

Bits 7:0: RTC Subsecond Counter Register. This ripple counter represents 1/256-second resolution for the RTC and its content is incremented with each 256Hz clock tick derived from the 32.768kHz oscillator. When the counter rolls over, its output is used to drive the 32-bit second counter.

14.4.4 RTC Seconds Counter High Register (RTSH, M0[1Bh])

Bit #	15	14	13	12	11	10	9	8
Name	RTSH							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	RTSH							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Note 1: *This register is cleared by power-on reset only.*

Note 2: *Read accessible when RDY = 1.*

Note 3: *Write accessible when RTCE = 0 and BUSY = 0.*

Bits 15:0: RTC Second Counter High Register. This register contains the most significant bits for the 32-bit second counter. The RTC is a 48-bit ripple counter consisting of three cascaded counter registers: the 8-bit subsecond counter (RTSS), the 16-bit low-order seconds counter (RTSL), and the 16-bit high-order second counter (RTSH).

14.4.5 RTC Seconds Counter Low Register (RTSL, M0[1Ch])

Bit #	15	14	13	12	11	10	9	8
Name	RTSL							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	RTSL							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Note 1: *This register is cleared by power-on reset only.*

Note 2: *Read accessible when RDY = 1.*

Note 3: *Write accessible when RTCE = 0 and BUSY = 0.*

Bits 15:0: RTC Second Counter Low Register. This register contains the least significant bits for the 32-bit second counter. The RTC is a 48-bit ripple counter consisting of three cascaded counter registers: the 8-bit subsecond counter (RTSS), the 16-bit low-order seconds counter (RTSL), and the 16-bit high-order second counter (RTSH).

14.4.6 RTC Subsecond Alarm Register (RSSA, M0[1Dh])

Bit #	7	6	5	4	3	2	1	0
Name	RSSA							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Note: *Write accessible when BUSY = 0 and either ASE = 0 or RTCE = 0.*

Bits 7:0: RTC Subsecond Alarm Register. This register contains the reload value for the subsecond alarm. The ALSF bit is set when an autoreload occurs.

14.4.7 RTC Time-of-Day Alarm High Register (RASH, M0[1Eh])

Bit #	7	6	5	4	3	2	1	0
Name	RASH							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	Rw

Note: *Write accessible when BUSY = 0 and either ASE = 0 or RTCE = 0.*

Bits 7:0: RTC Time-of-Day Alarm High Register. This register contains the most significant bits for the 24-bit time-of-day alarm. The time-of-day alarm is formed by the RASH and the RASL registers, and only the lower 20 bits is meaningful for the alarm function. Each time the subsecond counter rolls over (once per second), the RTC compares the lowest 20 bits of the time-of-day alarm setting (RASH[3:0]:RASL[15:0]) with the 20 least significant bits of the seconds counter (RTSH[3:0]:RTSL[15:0]). The time-of-day alarm is triggered when these two 20-bit values match.

14.4.8 RTC Time-of-Day Alarm Low Register (RASL, M0[1Fh])

Bit #	15	14	13	12	11	10	9	8
Name	RASL							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	RASL							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Note: *Write accessible when BUSY = 0 and either ASE = 0 or RTCE = 0.*

Bits 15:0: RTC Time-of-Day Alarm Low Register. This register contains the least significant bits for the 20-bit time-of-day alarm. The time-of-day alarm is formed by the RASH and the RASL registers and only the lower 20 bits are meaningful for the alarm function. Each time the subsecond counter rolls over (once per second), the RTC compares the lowest 20 bits of the time-of-day alarm setting (RASH[3:0]:RASL[15:0]) with the 20 least significant bits of the seconds counter (RTSH[3:0]:RTSL[15:0]). The time-of-day alarm is triggered when these two 20-bit values match.

14.5 RTC Example: Starting and Setting the Clock

```
  move RCNT, #08000h     ;RTC write enable

  call wait_busy         ;Call subroutine to verify RCNT.3 (BUSY) is low
  move RTSS, #00h        ;Clear sub-second count
  move RTSL, #0000h      ;Clear low-order seconds counter
  move RTSH, #0000h      ;Clear high-order seconds counter

  move RCNT, #08001h     ;Start clock
```

ADDENDUM TO SECTION 15: TEST ACCESS PORT (TAP)

The JTAG/TAP port on the MAXQ2010 is multiplexed with port pins P6.0, P6.1, P6.2, and P6.3. These pins default to the JTAG/TAP function on reset, which means that the part is always ready for in-circuit debugging or in-circuit programming operations following any reset.

Once an application has been loaded and starts running, the JTAG/TAP port can still be used for in-circuit debugging operations. If in-circuit debugging functionality is not needed, the associated port pins can be reclaimed for application use by setting the TAP (SC.7) bit to 0. This disables the JTAG/TAP interface and allows the four pins to operate as normal port pins.

ADDENDUM TO SECTION 16: IN-CIRCUIT DEBUG MODE

The MAXQ2010 provides an in-circuit debugging interface through the debug port as described in the *MAXQ Family User's Guide*. This interface provides the following functions for use in debugging application software:

- Single-step (trace) execution
- Four program address breakpoints
- Two breakpoints configurable as data address or register address breakpoints
- Register read and write
- Program stack read
- Data memory read and write
- Optional password protection

The following sections provide specific notes on the operation of the MAXQ2010 in debugging mode.

16.1 Register Read and Write Commands

Any register location can be read or written using these commands, including reserved locations and those used for op code support. No protection is provided by the debugging interface, and avoiding side effects is the responsibility of the host system communicating with the MAXQ2010.

Writing to the IP register alters the address that execution resumes at once the debugging engine exits.

In general, reading a register through the debug interface returns the value that was in that register **before** the debugging engine was invoked. An exception to this rule is the SP register; reading the SP register through the debug interface actually returns the value (SP+1).

16.2 Data Memory Read Command

When invoking this command, ICDA should be set to the word address of the starting location to read from, and ICDD should be set to the number of words. The input address must be based on the utility ROM memory map, as shown in Figure 2-3.

Data memory words returned by this command are output LSB first.

16.3 Data Memory Write Command

When invoking this command, ICDA should be set to the word address of the location to write to, and ICDD should be set to the data word to write. The input address must be based on the utility ROM memory map, as shown in Figure 2-3.

16.4 Program Stack Read Command

When invoking this command, ICDA should be set to the address of the starting stack location (value of SP) to read from, and ICDD should be set to the number of words. The address given in ICDA is the highest value that is used, as words are popped off the stack and returned in descending order.

Stack words returned by this command are output LSB first.

16.5 Read Register Map Command

This command outputs all peripheral registers in the range M0[00h] to M4[0Fh], along with a fixed set of system registers. The following formatting rules apply to the returned data:

- System registers are output as 8 bits or 16 bits, least significant byte first.
- All peripheral registers are output as 16 bits, least significant byte first. The top byte of 8-bit registers are returned as 00h.
- Nonimplemented and reserved peripheral registers in the range M0[00h] to M4[0Fh] are represented as empty word values in Table 16-1. These values should be ignored.

- Registers SPIB, I2CBUF, and ADDATA are not read, and their values are returned as 0000h.

The first byte output by this command is the value **144** (090h), which represents the number of peripheral register words output. Table 16-1 lists the remaining 352 bytes output by this command.

Table 16-1. Output from DebugReadMap Command

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	PO0		PO1		PO2		PO3		EIF0		EIE0		—		—	
1x	PI0		PI1		PI2		PI3		EIES0		—		—		PWCN	
2x	PD0		PD1		PD2		PD3		—		—		—		—	
3x	RTRM		RCNT		RTSS		RTSH		RTSL		RSSA		RASH		RASL	
4x	PO4		PO5		PO6		0000h		EIF1		EIE1		EIF2		EIE2	
5x	PI4		PI5		PI6		EIES1		EIES2		SVM		—		—	
6x	PD4		PD5		PD6		—		—		SPICN		SPICF		SPICK	
7x	—		—		—		—		—		—		—		—	
8x	MCNT		MA		MB		MC2		MC1		MC0		LCFG		—	
9x	MC1R		MC0R		LCRA		LCD0		LCD1		LCD2		LCD3		LCD4	
Ax	LCD5		LCD6		LCD7		LCD8		LCD9		LCD10		LCD11		LCD12	
Bx	LCD13		LCD14		LCD15		LCD16		LCD17		LCD18		LCD19		LCD20	
Cx	0000h		I2CST		I2CIE		—		SCON0		SBUF0		SCON1		SBUF1	
Dx	SMD0		PR0		SMD1		PR1		I2CCN		I2CCK		I2CTO		I2CSLA	
Ex	—		—		—		—		—		—		—		—	
Fx	—		—		—		—		—		—		—		—	
10x	TB0R		TB0C		TB1R		TB1C		TB2R		TB2C		ADST		ADADDR	
11x	TB0CN		TB0V		TB1CN		TB1V		TB2CN		TB2V		ADCN		0000h	
12x	AP	APC	PSF	IC	IMR	SC	IIR	CKCN	WDCN	00	A[0]		A[1]		A[2]	
13x	A[3]		A[4]		A[5]		A[6]		A[7]		A[8]		A[9]		A[10]	
14x	A[11]		A[12]		A[13]		A[14]		A[15]		IP		SP+1		IV	
15x	LC[0]		LC[1]		OFFS		DPC		GR		BP		DP[0]		DP[1]	

ADDENDUM TO SECTION 17: IN-SYSTEM PROGRAMMING (JTAG)

The MAXQ2010 provides a JTAG-compatible debug port interface for in-system programming (bootloader) operations. In order to use this interface for in-system programming, the SPE bit must be set through the debug port. This is done while the device is held in reset, using the system programming instruction as described in the *MAXQ Family User's Guide*.

17.1 JTAG Bootloader Protocol

The only bootloader interface supported for the MAXQ2010 is the debug port. When using the debug port, the clock rate (TCK) must be kept below 1/8 the system clock rate.

All bootloader commands begin with a single command byte. The high four bits of this command byte define the command family (from 0 to 15), while the low four bits define the specific command within that family.

All commands (except for those in Family 0) follow this format:

BYTE 1	BYTE 2	BYTE 3	BYTE 4	(LENGTH) BYTES/WORDS
Command	Length	Param 1	Param 2	Data

After each command has completed, the loader outputs a "prompt" byte to indicate that it has finished the operation. The prompt byte is the single character ">" (byte value 03Eh).

Bootloader commands that fail for any reason set the bootloader status byte to an error code value describing the reason for the failure. This status byte can be read by means of the Get Status command (04h).

Table 17-1. Bootloader Status Codes

STATUS VALUE	FUNCTION
00	No Error. The last command completed successfully.
01	Family Not Supported. An attempt was made to use a command from a family the bootloader does not support.
02	Invalid Command. An attempt was made to use a nonexistent command within a supported command family.
03	No Password Match. An attempt was made to use a password-protected command without first matching a valid password. Or, the Match Password command was called with an incorrect password value.
04	Bad Parameter. The parameter (address or otherwise) passed to the command was out of range or otherwise invalid.
05	Verify Failed. The verification step failed on a Load/Verify or Verify command.
06	Unknown Register. An attempt was made to read from or write to a nonexistent register.
07	Word Mode Not Supported. An attempt was made to set word-mode access, but the bootloader supports byte-mode access only.
08	Master Erase Failed. The bootloader was unable to perform master erase.

All commands in Family 0 can be executed without first matching the password. All other commands (in families 1x through Fx) are password protected; the password must first be matched before these commands can be executed.

A special case exists when the program memory has not been initialized (following master erase). If the password (stored in word locations 0010h to 001Fh in program memory) is all 0000h words or all FFFFh words, the bootloader treats the password as having been matched and automatically unlocks the password bit. This allows access to password-protected commands following master erase (when no password has been set in program memory).

When providing addresses for code or data read or write to bootloader commands, all addresses run from 0000h to (memory size – 1).

17.2 Family 0 Commands (Not Password Protected)

Command 00h—No Operation

I/O	BYTE 1
Input	00h
Output	

Command 01h—Exit Loader

This command causes the bootloader command loop to exit. Upon exit, the MAXQ2010 clears the SPE bit and resets itself internally. Following the internal reset, execution jumps to the beginning of application code at address 0000h.

I/O	BYTE 1
Input	01h
Output	

Command 02h—Master Erase

This command erases (programs to FFFFh) all words in the program flash memory and writes all words in the data SRAM to zero. After this command completes, the password lock bit is automatically cleared, allowing access to all bootloader commands.

I/O	BYTE 1
Input	02h
Output	

Command 03h—Password Match

This command accepts a 32-byte password value, which is matched against the password in program memory (in byte mode) from addresses 0020h to 003Fh. If the value matches, the password lock is cleared.

I/O	BYTE 1	BYTES 2 TO 33	BYTE 34	BYTE 35
Input	03h	32-Byte Password Value	00h	00h
Output				03Eh

Command 04h—Get Status

The status code returned by this command is defined in Table 17-1. The flags byte contains the bit status flags as shown in Table 17-2.

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5
Input	04h	00h	00h	00h	00h
Output			Flags	Status Code	03Eh

Table 17-2. Bootloader Status Flags

FLAG BIT	FUNCTION
0	Password Lock 0 = The password is unlocked or had a default value; password-protected commands can be used. 1 = The password is locked. Password-protected commands cannot be used.
1	Word/Byte Mode 0 = The bootloader is currently in byte mode for memory reads/writes. 1 = The bootloader is currently in word mode for memory reads/writes. (Note: The MAXQ2010 supports byte mode only.)
2	Word/Byte Mode Supported 0 = The bootloader supports byte mode only. 1 = The bootloader supports word mode as well as byte mode. (Note: The MAXQ2010 supports byte mode only.)
3 to 8	Reserved

Command 05h—Get Supported Commands

The SupportL (LSB) and SupportH (MSB) bytes form a 16-bit value that indicates which command families this bootloader supports. If bit 0 is set to 1, it indicates that Family 0 is supported. If bit 1 is set to 1, it indicates that Family 1 is supported, and so on. For the MAXQ2010, the value returned is 403Fh, indicating that command families 0, 1, 2, 3, 4, 5 and E are supported.

The CodeLen and DataLen bytes return the fixed block lengths used by the Load/Dump/Verify Fixed Length commands for code and data space, respectively. Because fixed block load is not supported on the MAXQ2010, both these values are returned as 00h.

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Input	05h	00h	00h	00h	00h	00h	00h
Output			SupportL	SupportH	CodeLen	DataLen	03Eh

Command 06h—Get Code Size

This command returns SizeH:SizeL, which represents the size of available code memory in words minus 1. If this command is unsupported, the return value is 0000h, meaning "unknown amount of memory".

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5
Input	06h	00h	00h	00h	00h
Output			SizeL	SizeH	03Eh

Command 07h—Get Data Size

This command returns SizeH:SizeL, which represents the size of available data memory in words minus 1. If this command is unsupported, the return value is 0000h, meaning "unknown amount of memory".

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5
Input	07h	00h	00h	00h	00h
Output			SizeL	SizeH	03Eh

Command 08h—Get Loader Version

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5
Input	08h	00h	00h	00h	00h
Output			VersionL	VersionH	03Eh

Command 09h—Get Utility ROM Version

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5
Input	09h	00h	00h	00h	00h
Output			VersionL	VersionH	03Eh

Command 0Ah—Set Word/Byte-Mode Access

The mode byte should be 0 to set byte-access mode or 1 to set word-access mode. The current access mode is returned in the status flag byte by command 04h, as well as a flag to indicate whether word-access mode is supported by this particular bootloader. **Note: The MAXQ2010 supports byte mode only.**

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4
Input	0Ah	Mode	00h	00h
Output				03Eh

Command 0Dh—Get ID Information

For the MAXQ2010, the information returned by this command is a zero-terminated ROM banner string.

I/O	BYTE 1	(VARIABLE)	LAST BYTE
Input	0Dh	00h, 00h, 00h . . .	00h
Output		Device-Dependent Information	03Eh

17.3 Family 1 Commands: Load Variable Length (Password Protected)

Command 10h—Load Code Variable Length

This command programs (Length) bytes of data into the program flash starting at byte address (AddressH:AddressL), with the following restrictions.

- The low bit of the address is always forced to zero, since instructions in program flash must be word aligned.
- In byte mode, if an odd number of bytes is input, the final word written to the program flash has its most significant byte set to 00h by default.
- Memory locations in flash that have previously been loaded must be erased (using the Master Erase command, the Erase Code Fixed Length command, or the **flashErasePage** or **flashEraseAll** utility ROM routines) before they can be loaded with a different value.
- In keeping with standard MAXQ little-endian memory architecture, the least significant byte of each word is loaded first. For example, if one loads bytes (11h, 22h, 33h, 44h) starting at address 0000h, the first two words of program space are written to (2211h, 4433h).

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	(LENGTH) BYTES	BYTE LENGTH+5	BYTE LENGTH+6
Input	10h	Length	AddressL	AddressH	Data to load	00h	00h
Output							03Eh

Command 11h—Load Data Variable Length

This command writes (Length) bytes of data into the data SRAM starting at byte address (AddressH:AddressL).

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	(LENGTH) BYTES	BYTE LENGTH+5	BYTE LENGTH+6
Input	11h	Length	AddressL	AddressH	Data to load	00h	00h
Output							03Eh

17.4 Family 2 Commands: Dump Variable Length (Password Protected)

Command 20h—Dump Code Variable Length

This command has a slightly different format depending on the length of the dump requested. It returns the contents of the program flash memory: (Length) or (LengthH:LengthL) bytes starting at byte address (AddrH:AddrL).

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	(LENGTH) BYTES	BYTE LENGTH+7
Input (to dump < 256 bytes)	20h	1	AddrL	AddrH	Length	00h	00h...	00h
Output							Memory contents	03Eh

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7	(LENGTH) BYTES	BYTE LENGTH+8
Input (to dump 256+ bytes)	20h	2	AddrL	AddrH	LengthL	LengthH	00h	00h...	00h
Output								Memory contents	03Eh

Command 21h—Dump Data Variable Length

This command has a slightly different format depending on the length of the dump requested. It returns the contents of the data SRAM - (Length) or (LengthH:LengthL) bytes starting at byte address (AddrH:AddrL).

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	(LENGTH) BYTES	BYTE LENGTH+7
Input (to dump < 256 bytes)	21h	1	AddrL	AddrH	Length	00h	00h...	00h
Output							Memory contents	03Eh

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7	(LENGTH) BYTES	BYTE LENGTH+8
Input (to dump 256+ bytes)	21h	2	AddrL	AddrH	LengthL	LengthH	00h	00h...	00h
Output								Memory contents	03Eh

17.5 Family 3 Commands: CRC Variable Length (Password Protected)

Command 30h—CRC Code Variable Length

This command has a slightly different format depending on the length of the CRC requested. It returns the CRC-16 value (CrcH:CrcL) of the program flash - (Length) or (LengthH:LengthL) bytes/words starting at (AddrH:AddrL).

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7	BYTE 8	BYTE 9
Input (to CRC < 256 bytes)	30h	1	AddrL	AddrH	Length	00h	00h	00h	00h
Output							CrcH	CrcL	03Eh

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7	BYTE 8	BYTE 9	BYTE 10
Input (to CRC 256+ bytes)	30h	2	AddrL	AddrH	LengthL	LengthH	00h	00h	00h	00h
Output								CrcH	CrcL	03Eh

Command 31h—CRC Data Variable Length

This command has a slightly different format depending on the length of the CRC requested. It returns the CRC-16 value (CrcH:CrcL) of the data SRAM - (Length) or (LengthH:LengthL) bytes/words starting at (AddrH:AddrL).

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7	BYTE 8	BYTE 9
Input (to CRC < 256 bytes)	31h	1	AddrL	AddrH	Length	00h	00h	00h	00h
Output							CrcH	CrcL	03Eh

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7	BYTE 8	BYTE 9	BYTE 10
Input (to CRC 256+ bytes)	31h	2	AddrL	AddrH	LengthL	LengthH	00h	00h	00h	00h
Output								CrcH	CrcL	03Eh

17.6 Family 4 Commands: Verify Variable Length (Password Protected)

Command 40h—Verify Code Variable Length

This command operates in the same manner as the "Load Code Variable Length" command, except that instead of programming the input data into flash memory, it verifies that the input data matches the data already in code space. If the data does not match, the status code is set to reflect this failure.

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	(LENGTH) BYTES	BYTE LENGTH+5	BYTE LENGTH+6
Input	40h	Length	AddressL	AddressH	Data to verify	00h	00h
Output							03Eh

Command 41h—Verify Data Variable Length

This command operates in the same manner as the "Load Data Variable Length" command, except that instead of writing the input data into data SRAM, it verifies that the input data matches the data already in data space. If the data does not match, the status code is set to reflect this failure.

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	(LENGTH) BYTES	BYTE LENGTH+5	BYTE LENGTH+6
Input	41h	Length	AddressL	AddressH	Data to verify	00h	00h
Output							03Eh

17.7 Family 5 Commands: Load and Verify Variable Length (Password Protected)

Command 50h—Load and Verify Code Variable Length

This command combines the functionality of the "Load Code Variable Length" and "Verify Code Variable Length" commands.

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	(LENGTH) BYTES	BYTE LENGTH+5	BYTE LENGTH+6
Input	50h	Length	AddressL	AddressH	Data to load and verify	00h	00h
Output							03Eh

Command 51h—Load and Verify Data Variable Length

This command combines the functionality of the "Load Data Variable Length" and "Verify Data Variable Length" commands.

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4	(LENGTH) BYTES	BYTE LENGTH+5	BYTE LENGTH+6
Input	51h	Length	AddressL	AddressH	Data to load and verify	00h	00h
Output							03Eh

17.8 Family E Commands: Erase Fixed Length (Password Protected)

Command E0h—Erase Code Fixed Length

This command erases (programs to FFFFh) all words in a 512-word page of the program flash memory. The address given should be located in the 512-word page to be erased. For example, providing address 0000h (in byte mode) to this command erases the first 512-word page, address 0400h erases the second page, and so on. There are 64 flash pages total, from 0000h to 7C00h.

I/O	BYTE 1	BYTE 2	BYTE 3	BYTE 4
Input	E0h	0	AddressL	AddressH
Output				

ADDENDUM TO SECTION 18: MAXQ FAMILY INSTRUCTION SET SUMMARY

Refer to the *MAXQ Family User's Guide*. Table 18-1 from the *MAXQ Family User's Guide* is reproduced here.

Table 18-1. Instruction Set Summary

	MNEMONIC	DESCRIPTION	16-BIT INSTRUCTION WORD	STATUS BITS AFFECTED	AP INC/DEC	NOTES
LOGICAL OPERATIONS	AND src	Acc ← Acc AND src	f001 1010 ssss ssss	S, Z	Y	1
	OR src	Acc ← Acc OR src	f010 1010 ssss ssss	S, Z	Y	1
	XOR src	Acc ← Acc XOR src	f011 1010 ssss ssss	S, Z	Y	1
	CPL	Acc ← ~Acc	1000 1010 0001 1010	S, Z	Y	
	NEG	Acc ← ~Acc + 1	1000 1010 1001 1010	S, Z	Y	
	SLA	Shift Acc left arithmetically	1000 1010 0010 1010	C, S, Z	Y	
	SLA2	Shift Acc left arithmetically twice	1000 1010 0011 1010	C, S, Z	Y	
	SLA4	Shift Acc left arithmetically four times	1000 1010 0110 1010	C, S, Z	Y	
	RL	Rotate Acc left (w/o C)	1000 1010 0100 1010	S	Y	
	RLC	Rotate Acc left (through C)	1000 1010 0101 1010	C, S, Z	Y	
	SRA	Shift Acc right arithmetically	1000 1010 1111 1010	C, Z	Y	
	SRA2	Shift Acc right arithmetically twice	1000 1010 1110 1010	C, Z	Y	
	SRA4	Shift Acc right arithmetically four times	1000 1010 1011 1010	C, Z	Y	
	SR	Shift Acc right (0 → msbit)	1000 1010 1010 1010	C, S, Z	Y	
	RR	Rotate Acc right (w/o C)	1000 1010 1100 1010	S	Y	
	RRC	Rotate Acc right (though C)	1000 1010 1101 1010	C, S, Z	Y	
BIT OPERATIONS	MOVE C, Acc.<b>	C ← Acc.<b>	1110 1010 bbbb 1010	C		
	MOVE C, #0	C ← 0	1101 1010 0000 1010	C		
	MOVE C, #1	C ← 1	1101 1010 0001 1010	C		
	CPL C	C ← ~C	1101 1010 0010 1010	C		
	MOVE Acc.<b>, C	Acc.<b> ← C	1111 1010 bbbb 1010	S, Z		
	AND Acc.<b>	C ← C AND Acc.<b>	1001 1010 bbbb 1010	C		
	OR Acc.<b>	C ← C OR Acc.<b>	1010 1010 bbbb 1010	C		
	XOR Acc.<b>	C ← C XOR Acc.<b>	1011 1010 bbbb 1010	C		
	MOVE dst.<b>, #1	dst.<b> ← 1	1ddd dddd 1bbb 0111	C, S, E, Z		2
	MOVE dst.<b>, #0	dst.<b> ← 0	1ddd dddd 0bbb 0111	C, S, E, Z		2
	MOVE C, src.<b>	C ← src.<b>	fbbb 0111 ssss ssss	C		
MATH	ADD src	Acc ← Acc + src	f100 1010 ssss ssss	C, S, Z, OV	Y	1
	ADDC src	Acc ← Acc + (src + C)	f110 1010 ssss ssss	C, S, Z, OV	Y	1
	SUB src	Acc ← Acc – src	f101 1010 ssss ssss	C, S, Z, OV	Y	1
	SUBB src	Acc ← Acc – (src + C)	f111 1010 ssss ssss	C, S, Z, OV	Y	1

Table 18-1. Instruction Set Summary (continued)

	MNEMONIC	DESCRIPTION	16-BIT INSTRUCTION WORD	STATUS BITS AFFECTED	AP INC/DEC	NOTES
BRANCHING	{L/S}JUMP src	IP ← IP + src or src	f000 1100 ssss ssss			6
	{L/S}JUMP C, src	If C=1, IP ← (IP + src) or src	f010 1100 ssss ssss			6
	{L/S}JUMP NC, src	If C=0, IP ← (IP + src) or src	f110 1100 ssss ssss			6
	{L/S}JUMP Z, src	If Z=1, IP ← (IP + src) or src	f001 1100 ssss ssss			6
	{L/S}JUMP NZ, src	If Z=0, IP ← (IP + src) or src	f101 1100 ssss ssss			6
	{L/S}JUMP E, src	If E=1, IP ← (IP + src) or src	0011 1100 ssss ssss			6
	{L/S}JUMP NE, src	If E=0, IP ← (IP + src) or src	0111 1100 ssss ssss			6
	{L/S}JUMP S, src	If S=1, IP ← (IP + src) or src	f100 1100 ssss ssss			6
	{L/S}DJNZ LC[n], src	If --LC[n] <> 0, IP← (IP + src) or src	f10n 1101 ssss ssss			6
	{L/S}CALL src	@++SP ← IP+1; IP ← (IP+src) or src	f011 1101 ssss ssss			6, 7
	RET	IP ← @SP--	1000 1100 0000 1101			
	RET C	If C=1, IP ← @SP--	1010 1100 0000 1101			
	RET NC	If C=0, IP ← @SP--	1110 1100 0000 1101			
	RET Z	If Z=1, IP ← @SP--	1001 1100 0000 1101			
	RET NZ	If Z=0, IP ← @SP--	1101 1100 0000 1101			
	RET S	If S=1, IP ← @SP--	1100 1100 0000 1101			
	RETI	IP ← @SP-- ; INS← 0	1000 1100 1000 1101			
	RETI C	If C=1, IP ← @SP-- ; INS← 0	1010 1100 1000 1101			
	RETI NC	If C=0, IP ← @SP-- ; INS← 0	1110 1100 1000 1101			
	RETI Z	If Z=1, IP ← @SP-- ; INS← 0	1001 1100 1000 1101			
	RETI NZ	If Z=0, IP ← @SP-- ; INS← 0	1101 1100 1000 1101			
	RETI S	If S=1, IP ← @SP-- ; INS← 0	1100 1100 1000 1101			
DATA TRANSFER	XCH (MAXQ20 only)	Swap Acc bytes	1000 1010 1000 1010	S	Y	
	XCHN	Swap nibbles in each Acc byte	1000 1010 0111 1010	S	Y	
	MOVE dst, src	dst ← src	fddd dddd ssss ssss	C, S, Z, E	(Note 8)	7, 8
	PUSH src	@++SP ← src	f000 1101 ssss ssss			7
	POP dst	dst ← @SP--	1ddd dddd 0000 1101	C, S, Z, E		7
	POPI dst	dst ← @SP-- ; INS ← 0	1ddd dddd 1000 1101	C, S, Z, E		7
	CMP src	E ← (Acc = src)	f111 1000 ssss ssss	E		
	NOP	No operation	1101 1010 0011 1010			

Note 1: The active accumulator (Acc) is not allowed as the src in operations where it is the implicit destination.

Note 2: Only module 8 and modules 0 to 5 (when implemented for a given product) are supported by these single-cycle bit operations. Potentially affects C or E if PSF register is the destination. Potentially affects S and/or Z if AP or APC is the destination.

Note 3: The terms Acc and A[AP] can be used interchangeably to denote the active accumulator.

Note 4: Any index represented by <b> or found inside [ ] brackets is considered variable, but required.

Note 5: The active accumulator (Acc) is not allowed as the dst if A[AP] is specified as the src.

Note 6: The '{L/S}' prefix is optional.

Note 7: Instructions that attempt to simultaneously push/pop the stack (e.g. PUSH @SP--, PUSH @SPI--, POP @++SP, POPI @++SP) or modify SP in a conflicting manner (e.g., MOVE SP, @SP--) are invalid.

Note 8: Special cases: If 'MOVE APC, Acc' sets the APC.CLR bit, AP is cleared, overriding any autoinc/dec/modulo operation specified for AP. If 'MOVE AP, Acc' causes an autoinc/dec/modulo operation on AP, this overrides the specified data transfer (i.e., Acc is not transferred to AP).

SECTION 19: ANALOG-TO-DIGITAL CONVERTER (SPECIFIC TO MAXQ2010)

The MAXQ2010 provides a 12-bit, successive approximation analog-to-digital converter (ADC) with an integrated analog multiplexer. The ADC can perform either single-ended conversions from one of eight channels or differential conversions from one of four channel pairs. The voltage reference used for each conversion can be selected from an internal precision bandgap reference, an external reference, or the AVDD analog power supply.

19.1 Analog-to-Digital Converter Features

- 12-bit single-ended conversion with up to eight analog channel inputs
- 12-bit differential conversion with up to four analog channel pair inputs (each differential pair takes the place of two single-ended channel inputs)
- Autoscan feature performs up to eight conversions in sequence automatically without CPU intervention
- Conversion sequence can be performed once (single conversion mode) or repeatedly (continuous conversion mode)
- Up to 16 sample words can be stored in a dedicated data buffer until the processor is ready to retrieve them
- Selectable clock divider runs the ADC from a divide by 1, divide by 2, divide by 4, or divide by 8 of the system clock
- Sample acquisition time can optionally be extended on a per-conversion basis
- Data results can be left-aligned or right-aligned on a per-conversion basis
- Converter reference is switchable among AVDD, internal reference, and external reference
- Optional power management mode shuts the ADC off between conversions to save power
- Configurable data available interrupt signals the CPU following each conversion, each sequence, or after every 12 or 16 samples

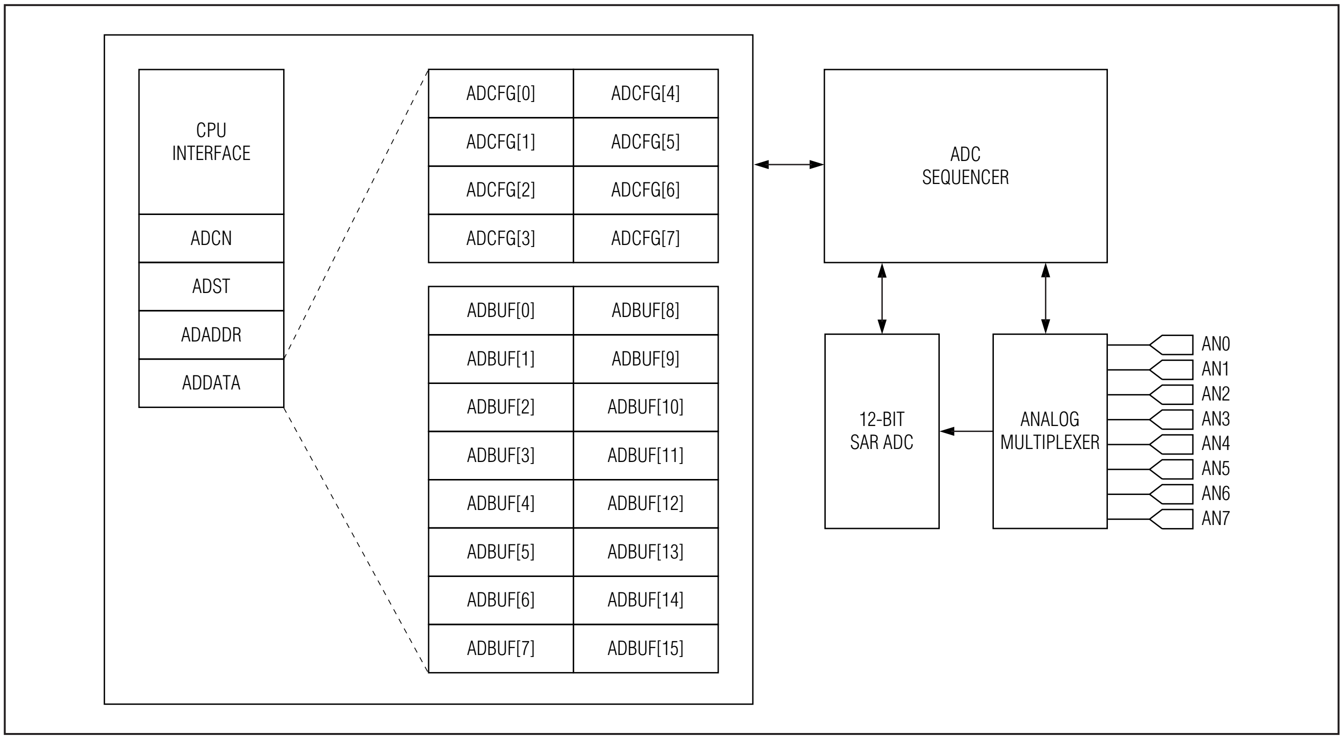


Figure 19-1. ADC Block Diagram

19.2 Analog-to-Digital Pins and Control Registers

Tables 19-1 and 19-2 list the pins and control registers dedicated to the ADC. Note that all ADC pins are dedicated, so none of them is multiplexed with GPIO port pins. Addresses for all registers are given as “Mx[yy],” where x is the module number (from 0 to 15 decimal) and yy is the register index (from 00h to 1Fh hexadecimal). Fields in the bit definition tables are defined as follows:

- **Name:** Symbolic names of bits or bit fields in this register.
- **Reset:** The value of each bit in this register following a standard reset. If this field reads “unchanged,” the given bit is unaffected by standard reset. If this field reads “s,” the given bit does not have a fixed 0 or 1 reset value because its value is determined by another internal state or external condition.
- **POR:** If present this field defines the value of each bit in this register following a power-on reset (as opposed to a standard reset). Some bits are unaffected by standard resets and are set/cleared by POR only.
- **Access:** Bits can be read-only (r) or read/write (rw). Any special restrictions or conditions that could apply when reading or writing this bit are detailed in the bit description.

Table 19-1. ADC Input and Power-Supply Pins

PIN	NAME	ADC INTERFACE FUNCTION
82	AVDD	Analog Supply Voltage
79	AGND	Analog Ground
70	AVREF	External ADC Voltage Reference
78	AN0	Single-Ended Analog Input Channel 0
77	AN1	Single-Ended Analog Input Channel 1
76	AN2	Single-Ended Analog Input Channel 2
75	AN3	Single-Ended Analog Input Channel 3
74	AN4	Single-Ended Analog Input Channel 4
73	AN5	Single-Ended Analog Input Channel 5
72	AN6	Single-Ended Analog Input Channel 6
71	AN7	Single-Ended Analog Input Channel 7
78, 77	(AN0, AN1)	Differential Input Channel 0
76, 75	(AN2, AN3)	Differential Input Channel 1
74, 73	(AN4, AN5)	Differential Input Channel 2
72, 71	(AN6, AN7)	Differential Input Channel 3

Table 19-2. ADC Control Registers

REGISTER	ADDRESS	FUNCTION
ADST	M4[06h]	ADC Status Register. Contains the ADCFG and ADBUF register index selection bits, conversion start bit, and other status bits for the ADC.
ADADDR	M4[07h]	ADC Address Register. Defines the first and last ADCFG registers used in a conversion sequence as well as the first ADBUF register written in a conversion sequence.
ADCN	M4[0Eh]	ADC Control Register. Controls sample acquisition extend, power-management mode, single/continuous sequence conversion, interrupt modes, and clock division for the ADC.
ADDATA	M4[0Fh]	ADC Data Register. Acts as a read/write access point to registers ADCFG[0] to ADCFG[7] and ADBUF[0] to ADBUF[15].
ADCFG[0]	ADDATA[10h]	ADC Sequence Configuration Register 0
ADCFG[1]	ADDATA[11h]	ADC Sequence Configuration Register 1
ADCFG[2]	ADDATA[12h]	ADC Sequence Configuration Register 2
ADCFG[3]	ADDATA[13h]	ADC Sequence Configuration Register 3

Table 19-2. ADC Control Registers (continued)

REGISTER	ADDRESS	FUNCTION
ADCFG[4]	ADDATA[14h]	ADC Sequence Configuration Register 4
ADCFG[5]	ADDATA[15h]	ADC Sequence Configuration Register 5
ADCFG[6]	ADDATA[16h]	ADC Sequence Configuration Register 6
ADCFG[7]	ADDATA[17h]	ADC Sequence Configuration Register 7
ADBUF[0]	ADDATA[00h]	ADC Sample Buffer Register 0. Read-only register containing ADC conversion result.
ADBUF[1]	ADDATA[01h]	ADC Sample Buffer Register 1. Read-only register containing ADC conversion result.
ADBUF[2]	ADDATA[02h]	ADC Sample Buffer Register 2. Read-only register containing ADC conversion result.
ADBUF[3]	ADDATA[03h]	ADC Sample Buffer Register 3. Read-only register containing ADC conversion result.
ADBUF[4]	ADDATA[04h]	ADC Sample Buffer Register 4. Read-only register containing ADC conversion result.
ADBUF[5]	ADDATA[05h]	ADC Sample Buffer Register 5. Read-only register containing ADC conversion result.
ADBUF[6]	ADDATA[06h]	ADC Sample Buffer Register 6. Read-only register containing ADC conversion result.
ADBUF[7]	ADDATA[07h]	ADC Sample Buffer Register 7. Read-only register containing ADC conversion result.
ADBUF[8]	ADDATA[08h]	ADC Sample Buffer Register 8. Read-only register containing ADC conversion result.
ADBUF[9]	ADDATA[09h]	ADC Sample Buffer Register 9. Read-only register containing ADC conversion result.
ADBUF[10]	ADDATA[0Ah]	ADC Sample Buffer Register 10. Read-only register containing ADC conversion result.
ADBUF[11]	ADDATA[0Bh]	ADC Sample Buffer Register 11. Read-only register containing ADC conversion result.
ADBUF[12]	ADDATA[0Ch]	ADC Sample Buffer Register 12. Read-only register containing ADC conversion result.
ADBUF[13]	ADDATA[0Dh]	ADC Sample Buffer Register 13. Read-only register containing ADC conversion result.
ADBUF[14]	ADDATA[0Eh]	ADC Sample Buffer Register 14. Read-only register containing ADC conversion result.
ADBUF[15]	ADDATA[0Fh]	ADC Sample Buffer Register 15. Read-only register containing ADC conversion result.

***Note:** Address ADDATA[xxh] refers to reading/writing the ADDATA register with ADCFG:ADIDX[3:0] (ADST[4:0]) set to xxh. For example, ADBUF[7] is read by setting ADST[4:0] to 00111b and reading the ADDATA register.*

The following peripheral registers are used to control the analog-to-digital converter functions.

19.2.1 Analog-to-Digital Converter Status Register (ADST, M4[06h])

Bit #	15	14	13	12	11	10	9	8
Name	—	—	—	—	ADDAT[3:0]			
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

Bit #	7	6	5	4	3	2	1	0
Name	REFOK	ADCONV	ADDAI	ADCFG	ADIDX[3:0]			
Reset	0	0	0	0	0	0	0	0
Access	r	rw*	rw	rw	rw	rw	rw	rw

**ADCONV cannot be written when PMME = 1 and SWB = 0.*

Bits 15:12: Reserved

Bits 11:8: ADC Data Available Address Bits (ADDAT[3:0]). These read-only status bits indicate the ADBUF data buffer location that was last written by the ADC hardware. During a conversion sequence, these bits are updated by hardware as each conversion is completed and written to the data buffer.

Bit 7: Internal Reference OK (REFOK). This read-only status bit indicates whether the internal reference is ready for use by the ADC.

0 = The internal reference is either disabled (IREFEN = 0) or is still warming up.

1 = The internal reference is ready for use.

Bit 6: ADC Start Conversion (ADCONV). Writing this bit to 1 starts the ADC conversion sequence. In single-conversion mode, this bit is cleared automatically by hardware at the end of the conversion sequence. In continuous-conversion mode, this bit remains set (and conversion continues) until it is reset to 0 by software. Setting this bit to 0 causes the current conversion sequence to stop. If the ADC is in the middle of a conversion, it stops after that conversion has completed. If the ADC is in the middle of an extended acquisition time period, it stops immediately.

Entering stop, PMM mode causes the current conversion to stop and the ADCONV bit to clear to 0. This bit cannot be written to 1 (to start a conversion) in PMM mode unless switchback is enabled (SWB = 1).

Bit 5: ADC Data Available Interrupt Flag (ADDAI). This bit is set to 1 by hardware when the conditions defined by ADINT[1:0] (ADCN[11:10]) are met. Setting this bit triggers an interrupt if ADDAIE = 1 and the interrupt is not otherwise masked. This bit is cleared by hardware automatically when an ADC conversion is started (ADCONV is written to 1); it can also be cleared to 0 by software.

Bit 4: ADC Conversion Configuration Register Select (ADCFG); Bits 3:0: ADC Configuration/Data Buffer Register Index (ADIDX[3:0]). These register bits select the ADC configuration or ADC data buffer register that is accessed when ADDATA is read or written. Note that the ADC data buffer registers are read-only.

Reading from or writing to ADDATA causes the value ADIDX[3:0] to autoincrement, but does not affect the value of ADCFG, even if the ADIDX value rolls over from 1111b to 0000b. For example, setting ADCFG to 1 and ADIDX[3:0] to 1101b selects ADCFG[5] for read/write access. Reading ADDATA successively then returns the values ADCFG[5], ADCFG[6], ADCFG[7], ADCFG[0], ADCFG[1], and so on.

ADCFG	ADIDX3	ADIDX2	ADIDX1	ADIDX0	READING ADDATA	WRITING ADDATA
0	0	0	0	0	Reads ADBUF[0]	No effect
0	0	0	0	1	Reads ADBUF[1]	No effect
0	0	0	1	0	Reads ADBUF[2]	No effect
0	0	0	1	1	Reads ADBUF[3]	No effect
0	0	1	0	0	Reads ADBUF[4]	No effect
0	0	1	0	1	Reads ADBUF[5]	No effect
0	0	1	1	0	Reads ADBUF[6]	No effect
0	0	1	1	1	Reads ADBUF[7]	No effect
0	1	0	0	0	Reads ADBUF[8]	No effect
0	1	0	0	1	Reads ADBUF[9]	No effect
0	1	0	1	0	Reads ADBUF[10]	No effect
0	1	0	1	1	Reads ADBUF[11]	No effect
0	1	1	0	0	Reads ADBUF[12]	No effect
0	1	1	0	1	Reads ADBUF[13]	No effect
0	1	1	1	0	Reads ADBUF[14]	No effect
0	1	1	1	1	Reads ADBUF[15]	No effect
1	X	0	0	0	Reads ADCFG[0]	Writes to ADCFG[0]
1	X	0	0	1	Reads ADCFG[1]	Writes to ADCFG[1]
1	X	0	1	0	Reads ADCFG[2]	Writes to ADCFG[2]
1	X	0	1	1	Reads ADCFG[3]	Writes to ADCFG[3]
1	X	1	0	0	Reads ADCFG[4]	Writes to ADCFG[4]
1	X	1	0	1	Reads ADCFG[5]	Writes to ADCFG[5]
1	X	1	1	0	Reads ADCFG[6]	Writes to ADCFG[6]
1	X	1	1	1	Reads ADCFG[7]	Writes to ADCFG[7]

19.2.2 ADC Conversion Sequence Address Register (ADADDR, M4[07h])

Bit #	15	14	13	12	11	10	9	8
Name	—	—	—	—	SEQSTORE[3:0]			
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	rw*	rw*	rw*	rw*

Bit #	7	6	5	4	3	2	1	0
Name	—	SEQSTART[2:0]			—	SEQEND[2:0]		
Reset	0	0	0	0	0	0	0	0
Access	r	rw*	rw*	rw*	r	rw*	rw*	rw*

**Can only be written when ADCONV = 0.*

Bits 15:12, 7, 3: Reserved

Bits 11:8: ADC Sequence Sample Storage Address (SEQSTORE[3:0]). These bits contain the index of the first ADBUF register (inclusive) that is used to store samples from the ADC conversion sequence.

Bits 6:4: ADC Sequence Start Address (SEQSTART[2:0]). These bits contain the index of the first ADCFG register (inclusive) that is used to define the ADC conversion sequence.

Bits 2:0: ADC Sequence End Address (SEQEND[2:0]). These bits contain the index of the last ADCFG register (inclusive) that is used to define the ADC conversion sequence.

19.2.3 ADC Control Register (ADCN, M4[0Eh])

Bit #	15	14	13	12	11	10	9	8
Name	—	—	—	—	ADINT1	ADINT0	ADCLK1	ADCLK0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	rw*	rw*	rw*	rw*

Bit #	7	6	5	4	3	2	1	0
Name	IREFEN	ADCONT	ADDAIE	ADPMO	ADACQ[3:0]			
Reset	0	0	0	0	0	0	0	0
Access	rw*	rw*	rw*	rw*	rw*	rw*	rw*	rw*

**Can only be written when ADCONV = 0.*

Bits 15:12: Reserved

Bits 11:10: ADC Data Available Interrupt Interval (ADINT[1:0]). These bits select the condition for generating an ADC data available interrupt (setting ADDAI to 1).

ADINT1	ADINT0	SET ADDAI to 1 . . .
0	0	After each ADC conversion (every sample)
0	1	After the conversion sequence has completed (works for single-conversion mode only)
1	0	Every 12 ADC samples
1	1	Every 16 ADC samples

Bits 9:8: ADC Clock Divider (ADCLK[1:0]). These bits control the generation of the ADC clock from the system clock as follows:

ADCLK1	ADCLK0	ADC CLOCK
0	0	System Clock/1 (default setting)
0	1	System Clock/2
1	0	System Clock/4
1	1	System Clock/8

However, since there is an upper limit on the sample rate of the ADC (approximately 300ksps; refer to the IC data sheet), not all clock division settings could be valid, depending on the system clock frequency. A single ADC conversion requires 16 ADC clocks, which allows us to calculate possible ADC sample rates as shown in Table 19-3.

Bit 7: ADC Internal Reference Enable (IREFEN). This bit controls the ADC internal reference.

0 = The internal reference is disabled. The ADREF bit in each configuration register (ADCFG) selects between AVDD and the external reference.

1 = The internal reference is enabled. Once REFOK = 1, the ADREF bit in each configuration register (ADCFG) selects between AVDD and the internal reference.

Bit 6: ADC Continuous Sequence Mode (ADCONT). This bit selects single- or continuous-sequence mode.

0 = Single-conversion sequence mode. In this mode, setting ADCONV = 1 starts a single-conversion sequence, with starting and ending configuration registers as defined in the ADADDR register. Once the conversion sequence completes, ADCONV automatically clears to 0, and the ADC powers down (if PMO = 0).

1 = Continuous-conversion sequence mode. In this mode, setting ADCONV = 1 also starts a conversion sequence, but once the sequence has completed, it simply repeats again. To stop the conversions, ADCONV must explicitly be cleared to 0 by software.

Bit 5: ADC Data Available Interrupt Enable (ADDAIE). This bit controls the ADC data available interrupt.

0 = The ADC interrupt is disabled.

1 = An interrupt is triggered (if not otherwise masked) when ADDAI = 1.

Bit 4: ADC Power-Management Override (PMO). This bit controls power management for the ADC.

0 = The ADC automatically powers up at the beginning of a conversion sequence and powers down when the sequence has finished (or when ADCONV is set to 0). This adds a delay of approximately 20 ADC clocks to the conversion sequence time.

1 = ADC power management is disabled. After setting PMO to 1, the software should wait long enough for the ADC to power up before initiating a conversion (refer to the IC data sheet for timing). Once the ADC has powered up, it remains powered on as long as PMO is set to 1, unless stop mode is entered.

Table 19-3. ADC Sample Rates Using a 10MHz Crystal

ADCLK[1:0]	SAMPLE RATE AT 10MHz (CLOCK/1) (ksps)	SAMPLE RATE AT 5MHz (CLOCK/2) (ksps)	SAMPLE RATE AT 2.5MHz (CLOCK/4) (ksps)	SAMPLE RATE AT 1.25MHz (CLOCK/8) (ksps)	SAMPLE RATE AT 8.4MHz (FLL) (ksps)
00	625 (invalid)	312.5	156.25	78.13	525 (invalid)
01	312.5	156.25	78.13	39	262.5 (invalid)
10	156.25	78.13	39	19.5	131
11	78.13	39	19.5	9.76	65.6

Bits 3:0: ADC Sample Acquisition Time Extend (ADACQ[3:0]). These bits set the extended sample acquisition time period. For a given conversion, sample acquisition time is extended if the ADACQEN bit is set in the conversion configuration (ADCFG) register. If this bit is set, the acquisition time is extended by:

16 x (ADACQ[3:0] + 1) x ADC clock period

19.2.4 ADC Data Register (ADDATA, M4[0Fh])

Bit #	15	14	13	12	11	10	9	8
Name	ADDATA							
Reset	s	s	s	s	s	s	s	s
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	ADDATA							
Reset	s	s	s	s	s	s	s	s
Access	rw	rw	rw	rw	rw	rw	rw	rw

Note: *The effect of read or write operation depends on ADIDX and ADCFG bit settings.*

This register is an access point for the eight ADC configuration registers (ADCFG[0] to ADCFG[7]) and 16 ADC data buffer registers (ADBUF[0] to ADBUF[15]). Reading or writing ADDATA actually reads or writes the selected register, as determined by ADST[4:0].

19.2.5 ADC Data Buffer Registers (ADBUF[0] to ADBUF[15], ADDATA[00h] to ADDATA[0Fh])

Bit #	15	14	13	12	11	10	9	8
Name	ADBUF							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	ADBUF							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

The 16 ADC data buffer registers ADBUF[0] to ADBUF[15] serve as temporary holding storage for ADC conversion samples until the samples can be read by the processor. They can be read or written at any time, whether or not an ADC conversion is in progress.

As each ADC conversion completes, the resulting sample is written to one of the ADBUF registers, starting with the index given by SEQSTORE (ADADDR[11:8]) and incrementing from there with each new sample written. The index of the most recent ADBUF register written to by the ADC controller is always available in the ADDAT bit field (ADST[11:8]).

The ADC samples are written into the ADBUF registers in either left-aligned or right-aligned format as selected by the ADALGN bit for that conversion configuration register. Once a sample is written into an ADBUF register, the data remains there until it is erased by software or until it is overwritten by another ADC sample, 16 conversions later. When continuous-conversion mode is used, it is the responsibility of the user software to monitor the data available interrupt and read ADC samples from the ADBUF registers before they are overwritten by subsequent samples.

19.2.6 ADC Conversion Configuration Registers (ADCFG[0] to ADCFG[7], ADDATA[10h] to ADDATA[17h])

Bit #	15	14	13	12	11	10	9	8
Name	—	—	—	—	—	—	—	—
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

Bit #	7	6	5	4	3	2	1	0
Name	—	ADREF	ADACQEN	ADALGN	ADDIFF	ADCH2	ADCH1	ADCH0
Reset	0	0	0	0	0	0	0	0
Access	r	rw*	rw*	rw*	rw*	rw*	rw*	rw*

**Can only be written when ADCONV = 0.*

The eight conversion configuration registers ADCFG[0] to ADCFG[7] provide settings for each individual conversion in an ADC conversion sequence. As the ADC autoscans, it reads each configuration register in the sequence in turn and performs a conversion using the settings from that register. The starting and ending configuration registers (inclusive) in the sequence are given by the SEQSTART and SEQEND bit fields in the ADADDR register.

The number of configuration registers selected by ADADDR also determines the number of conversions performed in the sequence (one conversion per register selected). The ADCFG registers cannot be written to while a conversion sequence is in progress (ADCONV = 1).

Bits 15:7: Reserved

Bit 6: ADC Reference Select (ADREF). This bit determines (in conjunction with IREFEN) which reference is used for this ADC conversion.

0 = AVDD (default) is used as the reference for this conversion.

1 = If IREFEN = 1, the internal reference is used for this conversion; otherwise, the external reference is used.

Bit 5: ADC Sample Acquisition Extension Enable. This bit determines whether the acquisition time for this conversion is extended by the number of ADC clock cycles given by ADACQ[3:0].

Bit 4: ADC Data Alignment Select (ADALGN). This bit determines how the ADC sample for this conversion is stored in the ADBUF register.

0 = The ADC data is stored right-adjusted in bits [11:0] of the ADBUF register. For single-ended conversions, bits [15:12] are filled with zeros, while for differential conversions bits [15:12] are sign extended from bit 11.

1 = The ADC data is stored left-adjusted in bits [15:4] of the ADBUF register with bits [3:0] zero padded.

Bit 3: ADC Differential Mode Select (ADDIFF); Bits 2:0: ADC Channel Select (ADCH[2:0]). These three bits control which channel or pair of channels is used for a given conversion, and whether the conversion is performed in single or differential mode. In differential mode, since there are only four channel pairs, bit ADCH[2] is ignored.

ADDIFF	ADCH2	ADCH1	ADCH0	ADC CONVERSION TYPE
0	0	0	0	Single conversion: AN0
0	0	0	1	Single conversion: AN1
0	0	1	0	Single conversion: AN2
0	0	1	1	Single conversion: AN3
0	1	0	0	Single conversion: AN4
0	1	0	1	Single conversion: AN5
0	1	1	0	Single conversion: AN6
0	1	1	1	Single conversion: AN7

ADDIFF	ADCH2	ADCH1	ADCH0	ADC CONVERSION TYPE
1	X	0	0	Differential conversion: (AN0–AN1)
1	X	0	1	Differential conversion: (AN2–AN3)
1	X	1	0	Differential conversion: (AN4–AN5)
1	X	1	1	Differential conversion: (AN6–AN7)

19.3 Analog-to-Digital Converter Code Examples

19.3.1 ADC Example 1: Single Conversion

```
  move ADCN,   #0300h   ; Set ADC clock to sysclk/8 (78ksps at 10MHz)

  move ADST,   #0010h   ; Points ADDATA to config register 0
  move ADDATA, #06h     ; Single-ended conversion on channel AN6, AVDD ref
  move ADST.6, #1       ; Start conversion

waitConvert:
  move C, ADST.6
  jump C, waitConvert   ; Conversion has completed when ADST.6 clears to 0

  move ADST,   #0000h   ; Points ADDATA to data register 0
  move Acc,    ADDATA   ; Get conversion result
```

19.3.2 ADC Example 2: Continuous Conversion

```
  move ADCN,   #0F00h   ; Set ADC clock to sysclk/8 (78ksps at 10MHz),
                        ;   also set Data Available interrupt to trigger
                        ;   following every 16 samples
  move ADCN.6, #1       ; Enable continuous conversion mode

  move ADST,   #0010h   ; Points ADDATA to config register 0
  move ADDATA, #06h     ; ACFG[0]: Single-ended conversion on AN6, AVDD ref
  move ADDATA, #07h     ; ACFG[1]: Single-ended conversion on AN7, AVDD ref
  move ADADDR, #0001h   ; Sequence runs from ACFG[0] to ACFG[1] inclusive

  move ADST.6, #1       ; Start conversion (continuous)
waitConvert:
  move C, ADST.5
  jump NC, waitConvert  ; Wait for 16 samples to be captured (ADDAI=1)

  move ADST.6, #0       ; Stop conversion
  move ADST.5, #0       ; Clear data available flag

  move ADST,   #0000h   ; Points ADDATA to data register 0
  move A[0],   ADDATA   ; Get conversion data
  move A[1],   ADDATA
  move A[2],   ADDATA
  move A[3],   ADDATA
  move A[4],   ADDATA
  move A[5],   ADDATA
  move A[6],   ADDATA
  move A[7],   ADDATA
  move A[8],   ADDATA
  move A[9],   ADDATA
  move A[10],  ADDATA
  move A[11],  ADDATA
  move A[12],  ADDATA
  move A[13],  ADDATA
  move A[14],  ADDATA
  move A[15],  ADDATA
```

SECTION 20: LCD CONTROLLER (SPECIFIC TO MAXQ2010)

20.1 LCD Controller Overview

The MAXQ2010 provides an on-board LCD controller module that can generate segment and common signals for an LCD based on display memory content. Once the LCD controller settings and display memory have been initialized, the LCD segment and common signals are generated automatically at the selected display frequency. No additional processor overhead is required while the LCD controller is running.

The LCD controller provides the following features and modes:

- Automatic LCD segment and common drive signal generation
- Four types of display modes supported:

  Static

  1/2 duty multiplexed with 1/2 bias voltages

  1/3 duty multiplexed with 1/3 bias voltages

  1/4 duty multiplexed with 1/3 bias voltages
- Up to 43 segment (SEG0 to SEG42) outputs and four common (COM0 to COM3) outputs
- Unused segment outputs SEG0 to SEG39 can be used as general-purpose port pins
- 21 bytes (168 bits) of display memory
- Unused display memory can be used for general-purpose storage
- Flexible LCD clock source, selectable from 32kHz or (high-frequency clock source/512)
- Adjustable frame frequency
- Internal voltage-divider resistors eliminate requirement for external components
- Internal adjustable resistor allows contrast adjustment without external components
- Capability to use external resistors to adjust drive voltages and current capacity

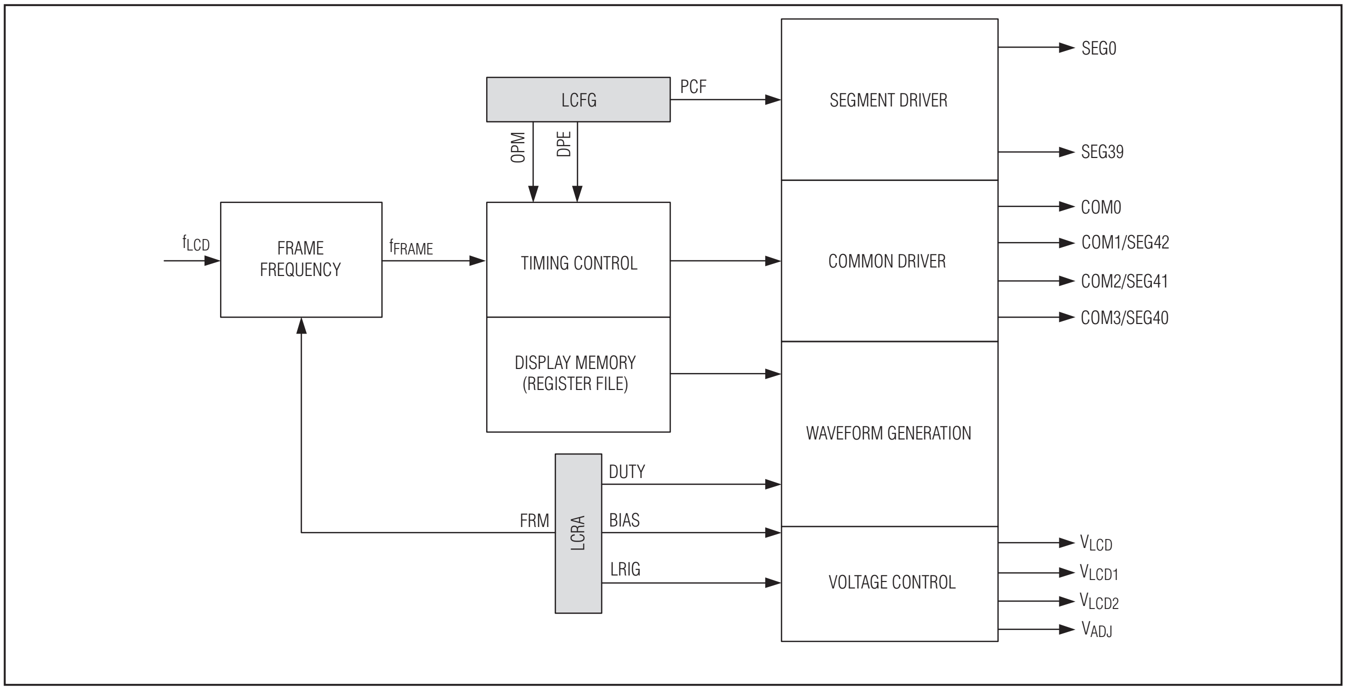


Figure 20-1. LCD Controller Block Diagram

20.2 LCD Controller Register Descriptions

The following peripheral registers are used to control the LCD display controller. Addresses for all registers are given as "Mx[yy]," where x is the module number (from 0 to 15 decimal) and yy is the register index (from 00h to 1Fh hexadecimal). Fields in the bit definition tables are defined as follows:

- **Name:** Symbolic names of bits or bit fields in this register.
- **Reset:** The value of each bit in this register following a standard reset. If this field reads "unchanged," the given bit is unaffected by standard reset. If this field reads "s," the given bit does not have a fixed 0 or 1 reset value because its value is determined by another internal state or external condition.
- **POR:** If present this field defines the value of each bit in this register following a power-on reset (as opposed to a standard reset). Some bits are unaffected by standard resets and are set/cleared by POR only.
- **Access:** Bits can be read-only (r) or read/write (rw). Any special restrictions or conditions that could apply when reading or writing this bit are detailed in the bit description.

20.2.1 LCD Configuration Register (LCFG, M2[06h])

Bit #	7	6	5	4	3	2	1	0
Name	PCF4	PCF3	PCF2	PCF1	PCF0	SMO	OPM	DPE
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit 7: Segment Pin Configuration for Port 4 (PCF4). This bit determines whether the pins on port 4 operate in GPIO mode or LCD segment driver mode.

0 = Port 4 pins operate as GPIO.

1 = Port 4 pins operate as segment drivers SEG32 to SEG39.

Bit 6: Segment Pin Configuration for Port 3 (PCF3). This bit determines whether the pins on port 3 operate in GPIO mode or LCD segment driver mode.

0 = Port 3 pins operate as GPIO.

1 = Port 3 pins operate as segment drivers SEG24 to SEG31.

Bit 5: Segment Pin Configuration for Port 2 (PCF2). This bit determines whether the pins on port 2 operate in GPIO mode or LCD segment driver mode.

0 = Port 2 pins operate as GPIO.

1 = Port 2 pins operate as segment drivers SEG16 to SEG23.

Bit 4: Segment Pin Configuration for Port 1 (PCF1). This bit determines whether the pins on port 1 operate in GPIO mode or LCD segment driver mode.

0 = Port 1 pins operate as GPIO.

1 = Port 1 pins operate as segment drivers SEG8 to SEG15.

Bit 3: Segment Pin Configuration for Port 0 (PCF0). This bit determines whether the pins on port 0 operate in GPIO mode or LCD segment driver mode. Note that if an external interrupt is enabled on a port 0 pin, it operates as GPIO, regardless of the setting of this bit.

0 = Port 0 pins operate as GPIO.

1 = Port 0 pins operate as segment drivers SEG0 to SEG7.

Bit 2: Stop Mode Operation (SMO). This bit determines whether the LCD controller continues operating in stop mode. Note that running the LCD controller in stop mode requires that the 32kHz clock is selected as the LCD clock source (LCCS = 0).

0 = The LCD controller goes into suspended mode automatically during stop.

1 = The LCD controller continues running normally during stop mode.

Bit 1: Operation Mode (OPM). This bit determines whether the LCD controller is operating (driving SEG and COM lines) or suspended (with its clock gated off).

0 = The LCD controller is suspended.

1 = The LCD controller is in normal operating mode.

Bit 0: Display Enable (DPE). When the LCD controller is in normal operating mode, this bit controls whether the display register data is used to drive the LCD. This bit has no meaning when LCD operation is suspended (OPM = 0).

0 = Disables the LCD display. SEG and COM waveforms are driven to turn all segments off.

1 = Drives the LCD display normally.

20.2.2 LCD Contrast Adjust Register (LCRA, M2[0Ah])

Bit #	15	14	13	12	11	10	9	8
Name	—	—	—	DUTY1	DUTY0	FRM3	FRM2	FRM1
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	FRM0	LCCS	LRIG	—	LRA3	LRA2	LRA1	LRA0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

This register can only be written to when the LCD controller is in suspended mode (OPM = 0).

Bits 15:13, 4: Reserved

Bits 12:11: LCD Duty-Cycle Select (DUTY[1:0]). These bits select the LCD display duty cycle and corresponding bias generation mode as follows:

DUTY1	DUTY0	DUTY CYCLE	BIAS MODE
0	0	Static	Static
0	1	1/2	1/2
1	0	1/3	1/3
1	1	1/4	1/3

Bits 10:7: LCD Frame Frequency (FRM[3:0]). These bits select the LCD frame frequency as follows:

For 1/3 bias mode: $f_{FRAME} = f_{LCD}/((FRM[3:0]) + 1) \times 96)$

For all other modes: $f_{FRAME} = f_{LCD}/((FRM[3:0]) + 1) \times 64)$

Bit 6: LCD Clock Select (LCCS). This bit selects the source clock (f_{LCD}) used for LCD segment and common timing generation.

0 = f_{LCD} = 32kHz clock

1 = f_{LCD} = high-frequency oscillator/512

Note: Because the high-frequency clock is halted when stop mode is invoked, LCD operation from the divided high-frequency clock (LCCS = 1) during stop mode is not possible. The user is advised to suspend LCD operation before entering stop mode or use the 32kHz for frame frequency generation if LCD operation is required during stop mode.

Bit 5: LCD Resistor Internally Grounded (LRIG)

0 = R_{ADJ} is disconnected from ground internally.

1 = R_{ADJ} is connected to ground internally.

Bits 3:0: LCD Register Adjust (LRA[3:0]). These bits control the resistance of the internal LCD resistor R_{ADJ}. The approximate resistance can be determined as:

$$R_{ADJ} = LCRA[3:0] \text{ x } 5.33k\Omega$$

The following registers contain display memory for the LCD controller.

20.2.3 LCD Display Register 0 (LCD0, M2[0Bh])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.4 LCD Display Register 1 (LCD1, M2[0Ch])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.5 LCD Display Register 2 (LCD2, M2[0Dh])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.6 LCD Display Register 3 (LCD3, M2[0Eh])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.7 LCD Display Register 4 (LCD4, M2[0Fh])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.8 LCD Display Register 5 (LCD5, M2[10h])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.9 LCD Display Register 6 (LCD6, M2[11h])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.10 LCD Display Register 7 (LCD7, M2[12h])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.11 LCD Display Register 8 (LCD8, M2[13h])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.12 LCD Display Register 9 (LCD9, M2[14h])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.13 LCD Display Register 10 (LCD10, M2[15h])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.14 LCD Display Register 11 (LCD11, M2[16h])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.15 LCD Display Register 12 (LCD12, M2[17h])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.16 LCD Display Register 13 (LCD13, M2[18h])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.17 LCD Display Register 14 (LCD14, M2[19h])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.18 LCD Display Register 15 (LCD15, M2[1Ah])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.19 LCD Display Register 16 (LCD16, M2[1Bh])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.20 LCD Display Register 17 (LCD17, M2[1Ch])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.21 LCD Display Register 18 (LCD18, M2[1Dh])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.22 LCD Display Register 19 (LCD19, M2[1Eh])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.2.23 LCD Display Register 20 (LCD20, M2[1Fh])

Bit #	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

20.3 LCD Controller Operation Modes

The LCD controller defaults to suspended mode (OPM = 0, DPE = x) on power-up. In this mode, the LCD controller is completely shut down to conserve power. Any pins that are configured for LCD operation (as well as any dedicated LCD segment/common pins) are driven by a weak pullup to V_{DDIO}.

Setting the OPM bit to 1 places the LCD controller into normal operating mode. In this mode, the LCD segment and common drivers generate waveforms to display the contents of display register memory if the DPE bit is set to 1. If DPE is set to 0, the LCD controller generates waveforms to turn all segments off.

20.4 LCD Drive Voltages

The LCD controller provides internal voltage-divider resistors to generate the voltage bias levels needed for the LCD display control. The top voltage level, V_{LCD}, must be provided by an external supply. As shown in Figure 20-2, no external connections (other than the power supply to V_{LCD}) are needed for static and 1/3 bias modes. For 1/2 bias mode, V_{LCD1} and V_{LCD2} must be shunted together externally.

20.5 Selecting the LCD Display Mode

The DUTY1 and DUTY0 bits select one of four possible display modes for the LCD controller as shown in Table 20-1. The display mode required for a given application depends on the number of segments needed and the multiplexing and voltage bias requirements for a given LCD display.

20.6 Segment Pin Configuration

The PCF[4:0] bits in the LCFG register are used to switch five banks of eight pins (port 0, port 1, port 2, port 3, and port 4) between LCD segment display mode and general-purpose port pin mode. Because all the PCF bits default to 0 on reset, all pins that share LCD segment and port pin capability act as port pins by default. To enable these pins to be used for LCD segment display, the PCF bits must be set appropriately, and the LCD controller must be in normal operational mode.

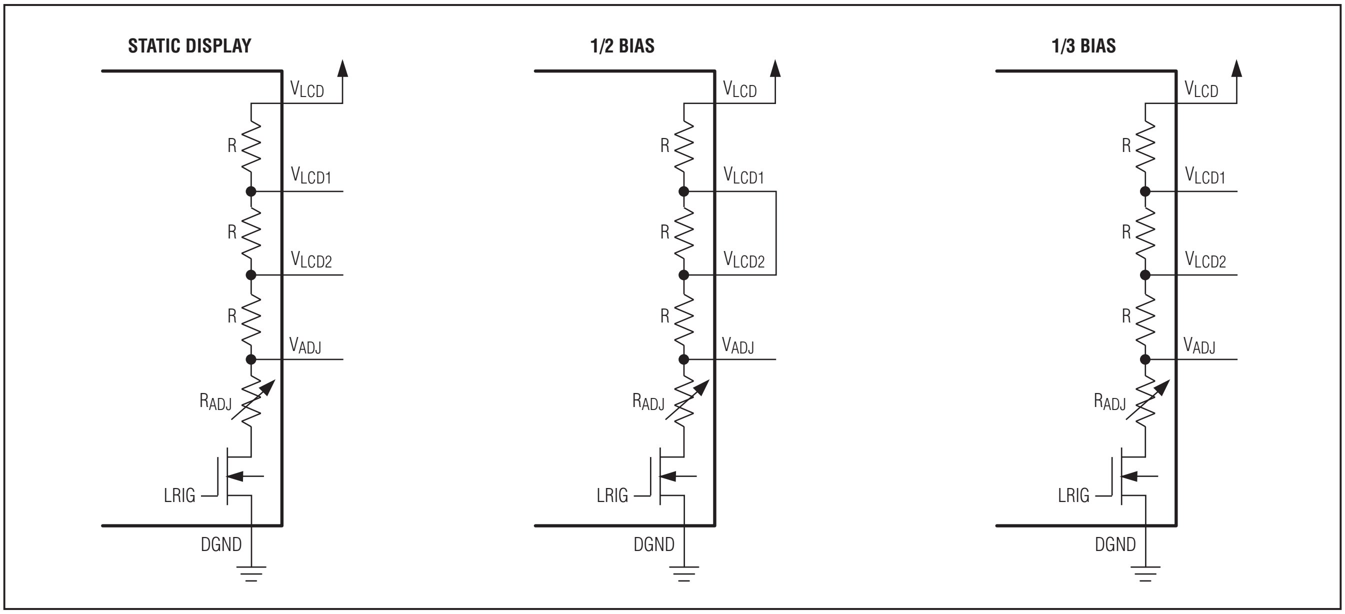


Figure 20-2. LCD Drive Voltage Generation

Table 20-1. LCD Display Modes

DUTY[1:0]	DUTY CYCLE	BIAS	DISPLAY SEGMENT DRIVE CAPACITY	COMMONS	V_{LCD2} VOLTAGE*	V_{LCD1} VOLTAGE*
00	Static	—	43	COM0	—	—
01	1/2	1/2	84	COM0 COM1	V_{ADJ} + (1/2 x V_{LCD} - V_{ADJ})	V_{ADJ} + (1/2 x V_{LCD} - V_{ADJ})
10	1/3	1/3	123	COM0 COM1 COM2	V_{ADJ} + (1/3 x V_{LCD} - V_{ADJ})	V_{ADJ} + (2/3 x V_{LCD} - V_{ADJ})
11	1/4	1/3	160	COM0 COM1 COM2 COM3	V_{ADJ} + (1/3 x V_{LCD} - V_{ADJ})	V_{ADJ} + (2/3 x V_{LCD} - V_{ADJ})

**For 1/2 bias mode, this assumes an external shunt in place between V_{LCD1} and V_{LCD2}.*

20.7 LCD Internal Adjustable Contrast Resistor

For an LCD segment to be in the off state, the V_{RMS} voltage between its COM and SEG signals must remain below the threshold voltage for that particular LCD display. As the V_{RMS} voltage difference increases, the LCD segment remains off until the threshold voltage is reached, at which point it turns on. As the V_{RMS} difference continues to increase, the contrast of the LCD segment increases as well (the segment becomes darker).

In order to adjust the visible contrast level for all LCD segments, the internal adjustable resistor R_{ADJ} can be varied between approximately 0 and 80kΩ by setting the bits LRA[3:0] (LCRA[3:0]). Changing this value causes the difference between V_{LCD}, V_{LCD1}, V_{LCD2}, and V_{ADJ} to increase or decrease evenly for all four drive voltages.

For the internal resistor R_{ADJ} to be used in this manner, the LRIG bit must be set to 1 to connect R_{ADJ} to ground internally. If an external adjustable resistor is used for the contrast adjustment function, LRIG should be set to 0 and the external resistor R_{EXT} should be connected between V_{ADJ} and ground as shown in Figure 20-3.

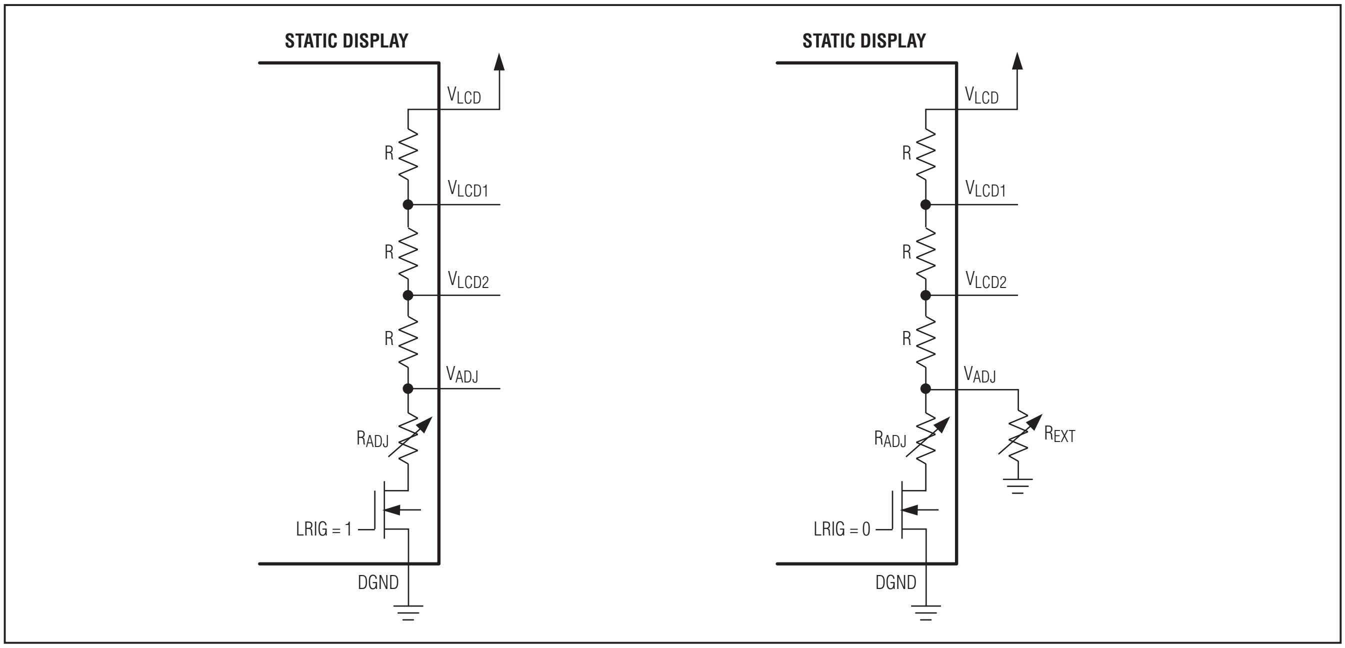


Figure 20-3. LCD Internal and External Display Contrast Adjustment

20.8 LCD Frame Frequency

The LCD controller clock frequency (f_{LCD}) can be sourced from either the 32kHz clock or the high-frequency clock divided by 128 as specified by the LCCS bit. If stop mode is entered and the 32kHz clock is being used, LCD display operation continues (if DPE = 1, OPM = 1, and SMO = 1). If stop mode is entered and the high-frequency clock divided by 128 is being used, the LCD controller is suspended automatically until stop mode is exited and the high-frequency clock completes its warmup period.

The bits FRM[3:0] control the generation of the LCD frame frequency from the selected LCD clock source (32kHz or HFClk/128). The selected frame frequency (f_{FRAME}) is defined as follows:

For 1/3 duty: $f_{FRAME} = f_{LCD}/((FRM[3:0]) + 1) \times 96)$

For static, 1/2 and 1/4 duty: $f_{FRAME} = f_{LCD}/((FRM[3:0]) + 1) \times 64)$

For best LCD drive performance, the frame frequency should typically be between 30Hz and 128Hz. Table 20-2 shows example frequencies that can be generated from typical clock frequencies.

Table 20-2. Approximate LCD Frame Frequencies (Hz)

FRM[3:0]	f_{LCD} = 32,768Hz/2		f_{LCD} = 19,531Hz (10MHz/512)	
	1, 1/2, 1/4	1/3	1, 1/2, 1/4	1/3
0	256	171	305	204
1	128	85	153	102
2	85	57	102	68
3	64	43	76	51
4	51	34	61	41
5	43	29	51	34
6	37	25	44	29
7	32	22	38	26

Table 20-2. Approximate LCD Frame Frequencies (Hz) (continued)

FRM[3:0]	f_{LCD} = 32,768Hz/2		f_{LCD} = 19,531Hz (10MHz/512)	
	1, 1/2, 1/4	1/3	1, 1/2, 1/4	1/3
8	29	19	34	23
9	26	17	31	21
10	24	16	28	19
11	22	14	26	17
12	20	13	24	16
13	19	12	22	15
14	17	12	21	14
15	16	11	19	13

20.9 LCD Display Memory

The LCD0 to LCD20 registers provide 21 bytes of display memory. Depending on the duty cycle and package type, not all display memory can be used by the LCD controller. Any display memory that is unused by the LCD controller can be used for application purposes if desired. The following tables show the display memory mapping for different duty cycle and package configurations.

Table 20-3. LCD Display Memory Map (Static)

REGISTER	BIT 7 COM0	BIT 6 COM0	BIT 5 COM0	BIT 4 COM0	BIT 3 COM0	BIT 2 COM0	BIT 1 COM0	BIT 0 COM0
LCD0	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
LCD1	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
LCD2	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
LCD3	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
LCD4	SEG39	SEG38	SEG37	SEG36	SEG35	SEG34	SEG33	SEG32
LCD5						SEG42	SEG41	SEG40
LCD6								
LCD7								
LCD8								
LCD9								
LCD10								
LCD11								
LCD12								
LCD13								
LCD14								
LCD15								
LCD16								
LCD17								
LCD18								
LCD19								
LCD20								

Table 20-4. LCD Display Memory Map (1/2 Duty)

REGISTER	BIT 7 COM1	BIT 6 COM0	BIT 5 COM1	BIT 4 COM0	BIT 3 COM1	BIT 2 COM0	BIT 1 COM1	BIT 0 COM0
LCD0	SEG3	SEG3	SEG2	SEG2	SEG1	SEG1	SEG0	SEG0
LCD1	SEG7	SEG7	SEG6	SEG6	SEG5	SEG5	SEG4	SEG4
LCD2	SEG11	SEG11	SEG10	SEG10	SEG9	SEG9	SEG8	SEG8
LCD3	SEG15	SEG15	SEG14	SEG14	SEG13	SEG13	SEG12	SEG12
LCD4	SEG19	SEG19	SEG18	SEG18	SEG17	SEG17	SEG16	SEG16
LCD5	SEG23	SEG23	SEG22	SEG22	SEG21	SEG21	SEG20	SEG20
LCD6	SEG27	SEG27	SEG26	SEG26	SEG25	SEG25	SEG24	SEG24
LCD7	SEG31	SEG31	SEG30	SEG30	SEG29	SEG29	SEG28	SEG28
LCD8	SEG35	SEG35	SEG34	SEG34	SEG33	SEG33	SEG32	SEG32
LCD9	SEG39	SEG39	SEG38	SEG38	SEG37	SEG37	SEG36	SEG36
LCD10					SEG41	SEG41	SEG40	SEG40
LCD11								
LCD12								
LCD13								
LCD14								
LCD15								
LCD16								
LCD17								
LCD18								
LCD19								
LCD20								

Table 20-5. LCD Display Memory Map (1/3 Duty)

REGISTER	BIT 7	BIT 6 COM2	BIT 5 COM1	BIT 4 COM0	BIT 3	BIT 2 COM2	BIT 1 COM1	BIT 0 COM0
LCD0		SEG1	SEG1	SEG1		SEG0	SEG0	SEG0
LCD1		SEG3	SEG3	SEG3		SEG2	SEG2	SEG2
LCD2		SEG5	SEG5	SEG5		SEG4	SEG4	SEG4
LCD3		SEG7	SEG7	SEG7		SEG6	SEG6	SEG6
LCD4		SEG9	SEG9	SEG9		SEG8	SEG8	SEG8
LCD5		SEG11	SEG11	SEG11		SEG10	SEG10	SEG10
LCD6		SEG13	SEG13	SEG13		SEG12	SEG12	SEG12
LCD7		SEG15	SEG15	SEG15		SEG14	SEG14	SEG14
LCD8		SEG17	SEG17	SEG17		SEG16	SEG16	SEG16
LCD9		SEG19	SEG19	SEG19		SEG18	SEG18	SEG18
LCD10		SEG21	SEG21	SEG21		SEG20	SEG20	SEG20
LCD11		SEG23	SEG23	SEG23		SEG22	SEG22	SEG22
LCD12		SEG25	SEG25	SEG25		SEG24	SEG24	SEG24
LCD13		SEG27	SEG27	SEG27		SEG26	SEG26	SEG26
LCD14		SEG29	SEG29	SEG29		SEG28	SEG28	SEG28
LCD15		SEG31	SEG31	SEG31		SEG30	SEG30	SEG30
LCD16		SEG33	SEG33	SEG33		SEG32	SEG32	SEG32

Table 20-5. LCD Display Memory Map (1/3 Duty) (continued)

REGISTER	BIT 7	BIT 6 COM2	BIT 5 COM1	BIT 4 COM0	BIT 3	BIT 2 COM2	BIT 1 COM1	BIT 0 COM0
LCD17		SEG35	SEG35	SEG35		SEG34	SEG34	SEG34
LCD18		SEG37	SEG37	SEG37		SEG36	SEG36	SEG36
LCD19		SEG39	SEG39	SEG39		SEG38	SEG38	SEG38
LCD20						SEG40	SEG40	SEG40

Table 20-6. LCD Display Memory Map (1/4 Duty)

REGISTER	BIT 7 COM3	BIT 6 COM2	BIT 5 COM1	BIT 4 COM0	BIT 3 COM3	BIT 2 COM2	BIT 1 COM1	BIT 0 COM0
LCD0	SEG1	SEG1	SEG1	SEG1	SEG0	SEG0	SEG0	SEG0
LCD1	SEG3	SEG3	SEG3	SEG3	SEG2	SEG2	SEG2	SEG2
LCD2	SEG5	SEG5	SEG5	SEG5	SEG4	SEG4	SEG4	SEG4
LCD3	SEG7	SEG7	SEG7	SEG7	SEG6	SEG6	SEG6	SEG6
LCD4	SEG9	SEG9	SEG9	SEG9	SEG8	SEG8	SEG8	SEG8
LCD5	SEG11	SEG11	SEG11	SEG11	SEG10	SEG10	SEG10	SEG10
LCD6	SEG13	SEG13	SEG13	SEG13	SEG12	SEG12	SEG12	SEG12
LCD7	SEG15	SEG15	SEG15	SEG15	SEG14	SEG14	SEG14	SEG14
LCD8	SEG17	SEG17	SEG17	SEG17	SEG16	SEG16	SEG16	SEG16
LCD9	SEG19	SEG19	SEG19	SEG19	SEG18	SEG18	SEG18	SEG18
LCD10	SEG21	SEG21	SEG21	SEG21	SEG20	SEG20	SEG20	SEG20
LCD11	SEG23	SEG23	SEG23	SEG23	SEG22	SEG22	SEG22	SEG22
LCD12	SEG25	SEG25	SEG25	SEG25	SEG24	SEG24	SEG24	SEG24
LCD13	SEG27	SEG27	SEG27	SEG27	SEG26	SEG26	SEG26	SEG26
LCD14	SEG29	SEG29	SEG29	SEG29	SEG28	SEG28	SEG28	SEG28
LCD15	SEG31	SEG31	SEG31	SEG31	SEG30	SEG30	SEG30	SEG30
LCD16	SEG33	SEG33	SEG33	SEG33	SEG32	SEG32	SEG32	SEG32
LCD17	SEG35	SEG35	SEG35	SEG35	SEG34	SEG34	SEG34	SEG34
LCD18	SEG37	SEG37	SEG37	SEG37	SEG36	SEG36	SEG36	SEG36
LCD19	SEG39	SEG39	SEG39	SEG39	SEG38	SEG38	SEG38	SEG38
LCD20								

20.10 Display Waveform Generation

Once the operational modes and display memory registers on the LCD controller have been properly initialized, the controller generates the segment and common drive waveforms needed to display the enabled segments on the attached LCD display.

In static mode, each segment pin is connected to a single LCD segment. There is only one common, or backplane, signal that is driven by COM0. All on segments are driven for the entire frame period.

In x2 multiplexed mode, also known as 1/2 duty cycle mode, each segment pin can drive up to two LCD segments. There are two common backplane signals, driven by COM0 and COM1. Each on segment is only driven for half the frame period.

In x3 multiplexed mode, also known as 1/3 duty cycle mode, each segment pin can drive up to three LCD segments. There are three common backplane signals, driven by COM0, COM1, and COM2. Each on segment is only driven for 1/3 the frame period.

In x4 multiplexed mode, also known as 1/4 duty cycle mode, each segment pin can drive up to four LCD segments. There are four common backplane signals, driven by COM0, COM1, COM2, and COM3. Each on segment is only driven for 1/4 the frame period.

In each of the following examples, the LCD controller is driving a "2" to the display. This is a conventional 7-segment display, and the "2" consists of on segments a, b, d, e, g, and DP as shown in Figure 20-4. The voltage waveforms shown assume that V_{ADJ} equals ground (R_{ADJ} is set to 0Ω). The portions of the memory maps used apply to either package type.

20.11 LCD Controller Static Drive Example

In this example, SEG0 to SEG7 are used to drive the LCD segments. The segments and common signals are connected as shown in Figure 20-5.

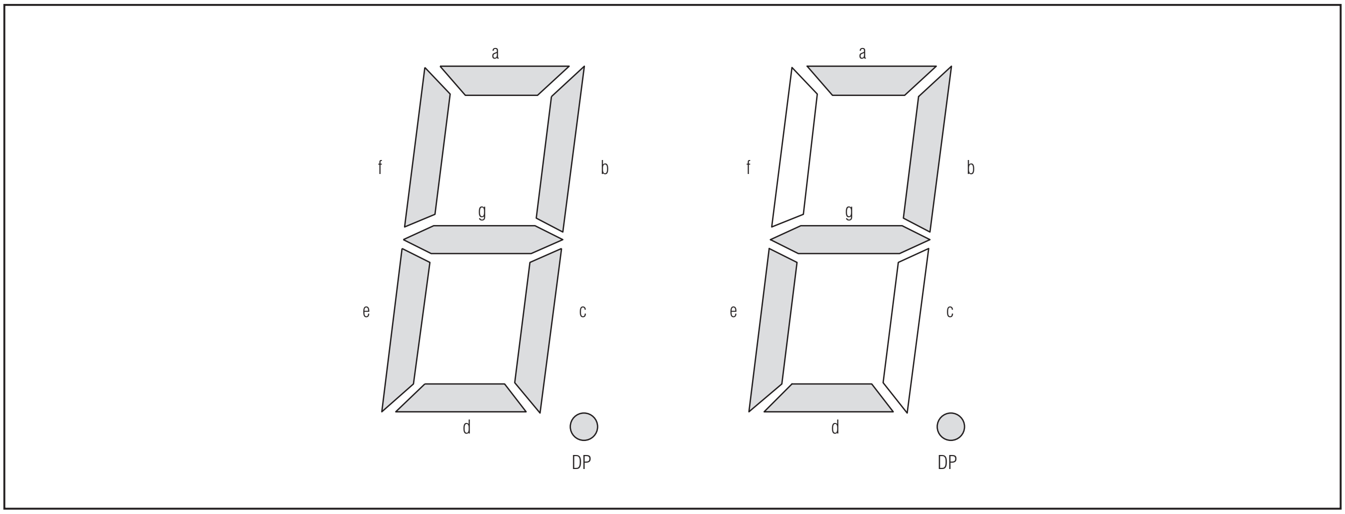


Figure 20-4. Sample 7-Segment LCD Display

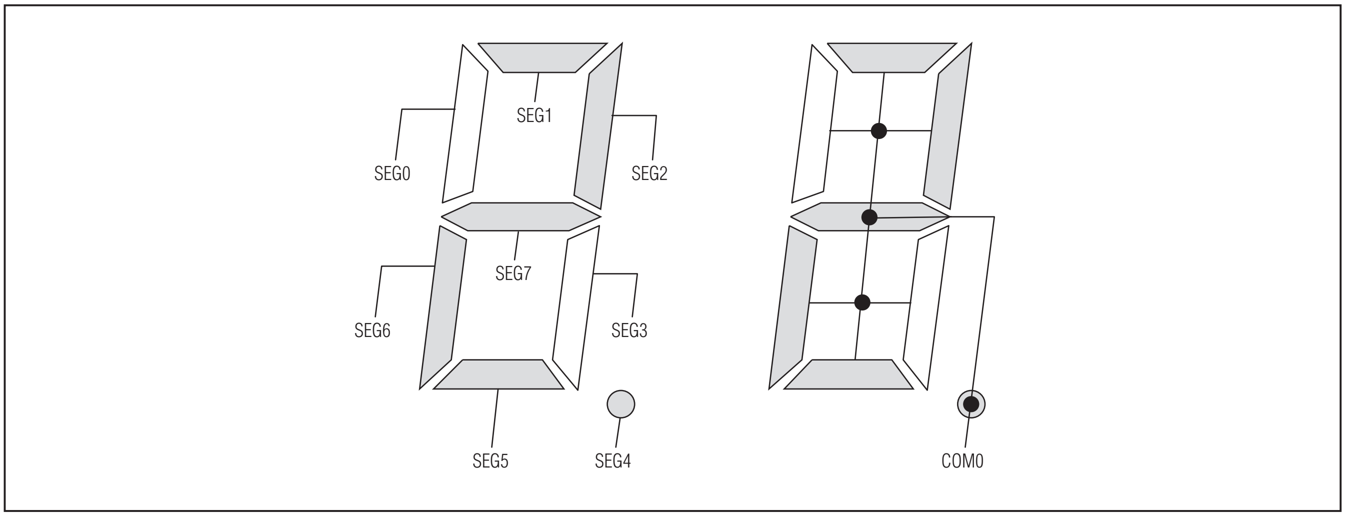


Figure 20-5. Static Drive Example Display Connection

Table 20-7. Static Drive Example Common Signal Selection

	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
COM0	On	On	On	On	Off	On	On	Off
COM1								
COM2								
COM3								

According to the static memory map table, a value of 0F6h should be written to the LCD0 register as shown in Table 20-8.

Table 20-8. Static Drive Example Register Content

	BIT 7 COM0	BIT 6 COM0	BIT 5 COM0	BIT 4 COM0	BIT 3 COM0	BIT 2 COM0	BIT 1 COM0	BIT 0 COM0
LCD0	1	1	1	1	0	1	1	0
LCD1								
LCD2								
LCD3								

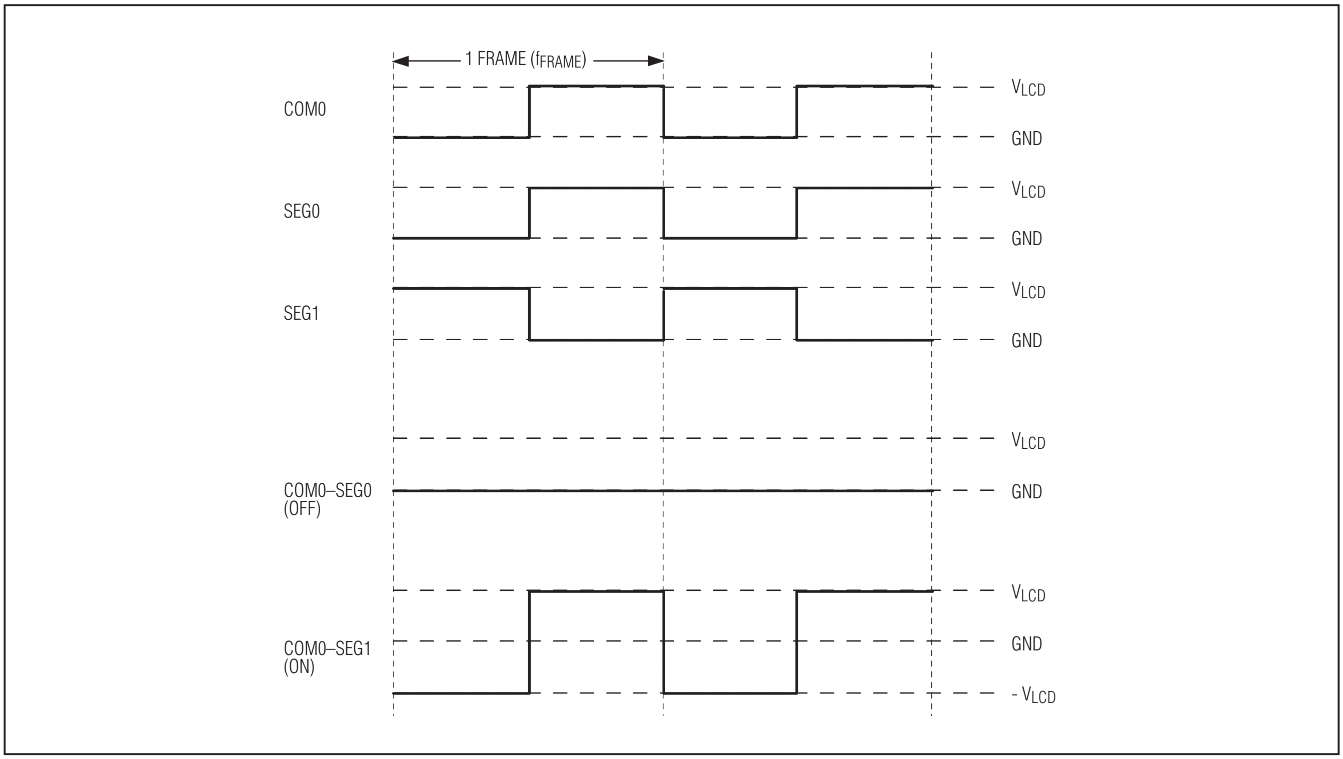


Figure 20-6. Static Drive Example Waveform Timing

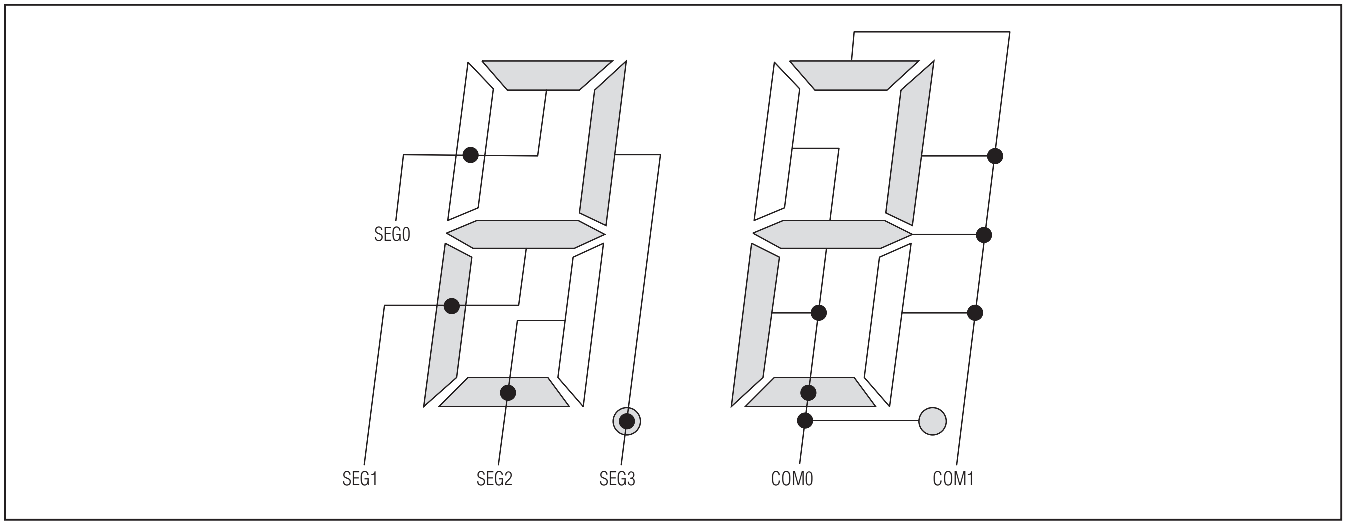


Figure 20-7. 1/2 Duty Drive Example Display Connection

Table 20-9. 1/2 Duty Drive Example Common Signal Selection

	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
COM0					On	On	On	Off
COM1					On	Off	On	On
COM2								
COM3								

Table 20-10. 1/2 Duty Drive Example Register Content

	BIT 7 COM1	BIT 6 COM0	BIT 5 COM1	BIT 4 COM0	BIT 3 COM1	BIT 2 COM0	BIT 1 COM1	BIT 0 COM0
LCD0	1	1	0	1	1	1	1	0
LCD1								
LCD2								
LCD3								

20.12 LCD Controller 1/2 Duty Cycle Drive Example

In this example, SEG0 to SEG3 are used to drive the LCD segments. The segments and common signals are connected as shown in Figure 20-7.

According to the 1/2 duty drive memory map table, a value of 0DEh should be written to the LCD0 register as shown in Table 20-10.

20.13 LCD Controller 1/3 Duty Cycle Drive Example

In this example, SEG0 to SEG2 are used to drive the LCD segments. The segments and common signals are connected as shown in Figure 20-9 .

According to the 1/3 duty drive memory map table, LCD0 should be set to 071h and LCD1 should be set to 05h as shown in Table 20-12.

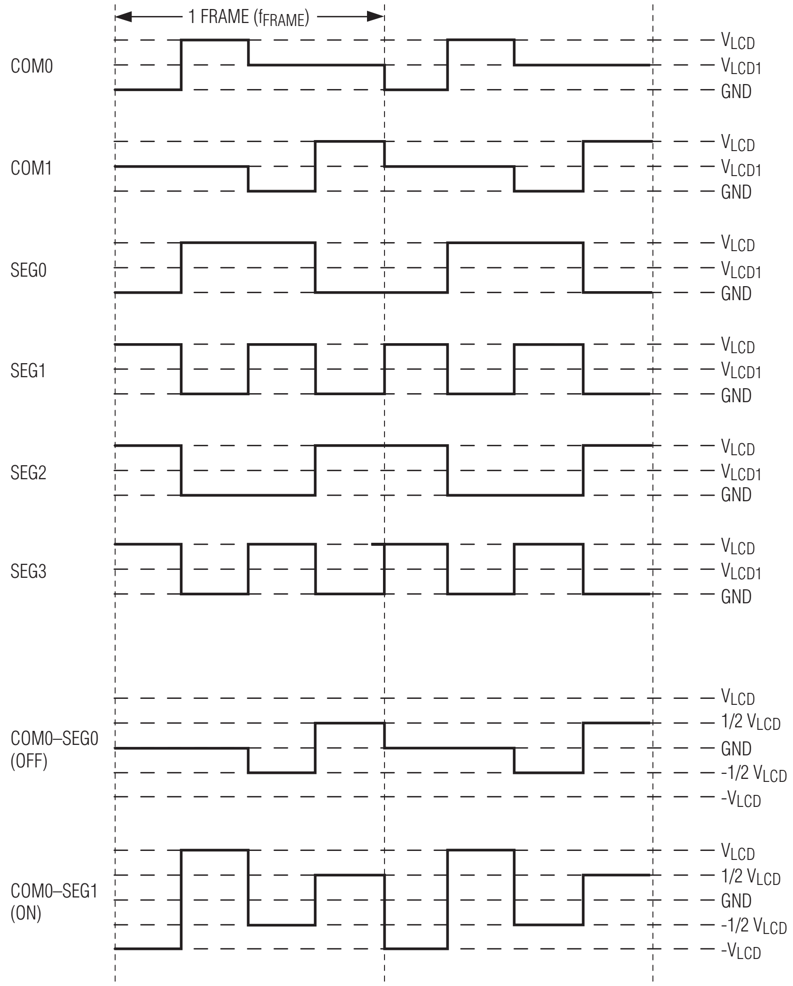


Figure 20-8. 1/2 Duty Drive Example Waveform Timing

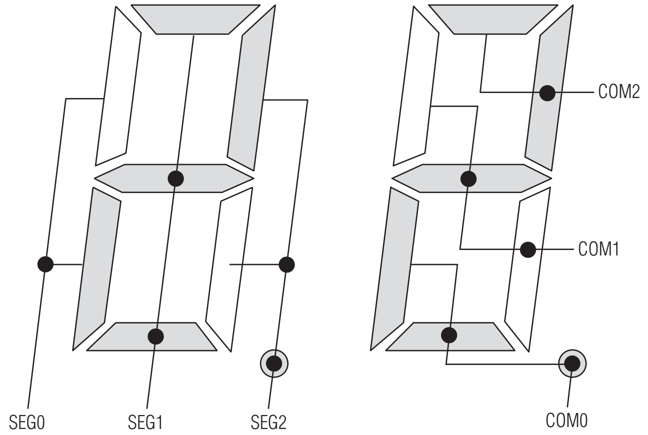


Figure 20-9. 1/3 Drive Example Display Connection

Table 20-11. 1/3 Duty Drive Example Common Signal Selection

	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
COM0						On	On	On
COM1						Off	On	Off
COM2						On	On	(Don't Care)
COM3								

Table 20-12. 1/3 Duty Drive Example Register Content

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		COM2	COM1	COM0		COM2	COM1	COM0
LCD0	0	1	1	1	0	0	0	1
LCD1	0	0	0	0	0	1	0	1
LCD2								
LCD3								

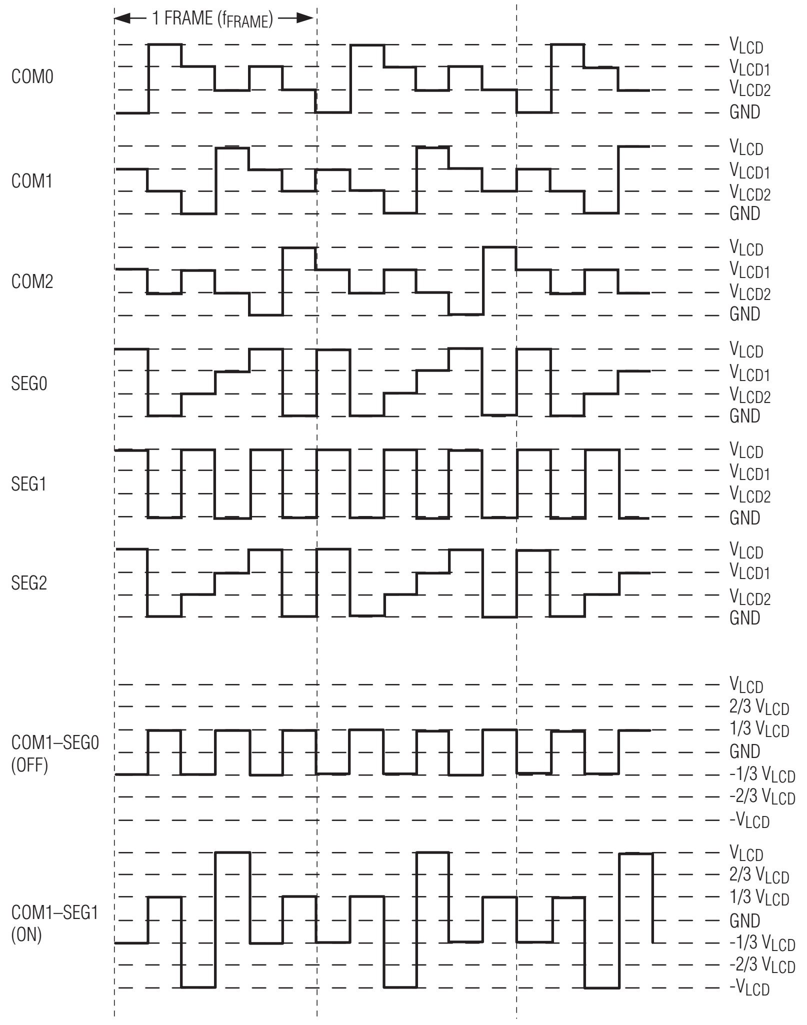


Figure 20-10. 1/3 Duty Drive Example Waveform Timing

20.14 LCD Controller 1/4 Duty Cycle Drive Example

In this example, SEG0 and SEG1 are used to drive the LCD segments. The segments and common signals are connected as shown in Figure 20-11.

According to the 1/4 duty drive memory map table, LCD0 should be set to 0D7h as shown in Table 20-14.

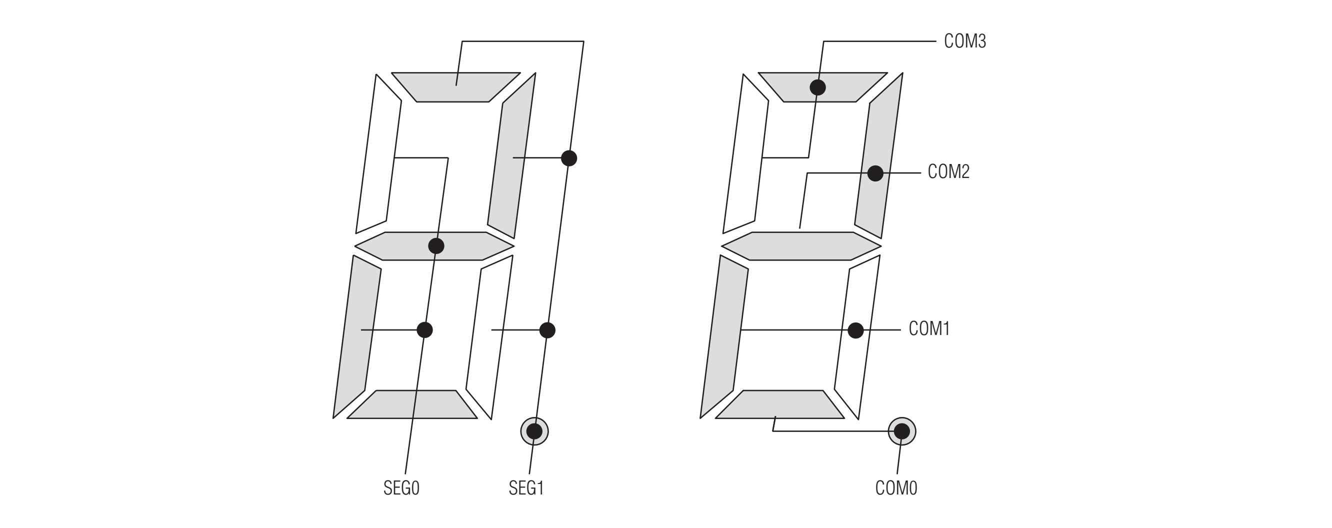


Figure 20-11. 1/4 Duty Drive Example Display Connection

Table 20-13. 1/4 Duty Drive Example Common Signal Selection

	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
COM0							On	Off
COM1							Off	On
COM2							On	On
COM3							On	On

Table 20-14. 1/4 Duty Drive Example Register Content

	BIT 7 COM3	BIT 6 COM2	BIT 5 COM1	BIT 4 COM0	BIT 3 COM3	BIT 2 COM2	BIT 1 COM1	BIT 0 COM0
LCD0	1	1	0	1	0	1	1	1
LCD1								
LCD2								
LCD3								

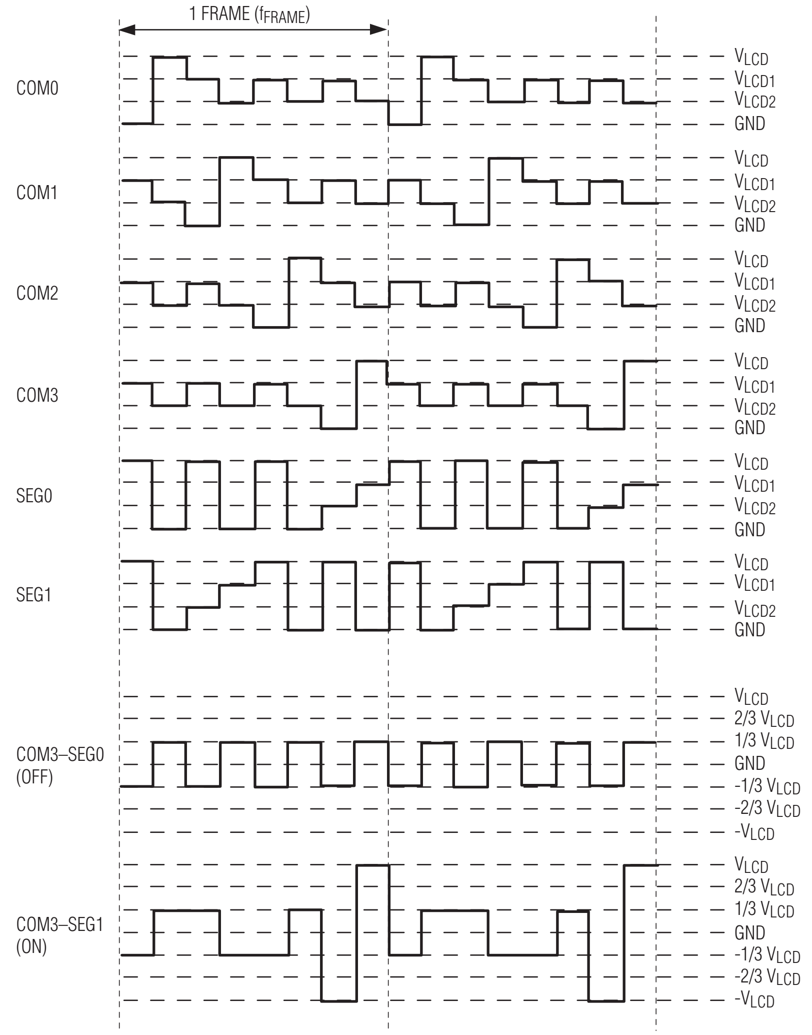


Figure 20-12. 1/4 Duty Drive Example Waveform Timing

20.15 LCD Controller Example: Initializing the LCD Controller

```
move LCRA, #03E0h
; LCRA.FRM = 7 Set up frame frequency.
; LCRA.LCCS = 1 Set clock source to HFClk / 128.
; LCRA.DUTY = 0 Set up static duty cycle.
; LCRA.LRA = 0 Set R-adj to 0.
; LCRA.LRIGC = 1 Ground Radj resistor internally.

move LCFG, #0F3h
; LCFG.PCF = 0x0F Set up all segments as outputs.
; LCFG.OPM = 1 Set to normal operation mode.
; LCFG.DPE = 1 Enable display.
```

SECTION 21: TIMER/COUNTER B MODULE (SPECIFIC TO MAXQ2010)

The MAXQ2010 provides three Type B timer/counter modules that operate as described in this section. Table 21-1 and Table 21-2 list the associated pins and registers for these timer/counter modules.

Table 21-1. Type B Timer/Counter Input and Output Pins

TIMER/COUNTER FUNCTION	PIN	MULTIPLEXED WITH GPIO
TB0A: Timer 0 I/O Pin A	67	P5.1
TB0B: Timer 0 I/O Pin B	68	P5.0
TB1A: Timer 1 I/O Pin A	25	P6.5
TB1B: Timer 1 I/O Pin B	28	P6.4
TB2A: Timer 2 I/O Pin A	23	P6.7
TB2B: Timer 2 I/O Pin B	24	P6.6

Table 21-2. Type B Timer/Counter Control Registers

REGISTER	ADDRESS	FUNCTION
TB0R	M4[00h]	Type B Timer/Counter 0 Capture/Reload Register. Holds the reload value in autoreload mode; used to store capture values in capture mode.
TB0C	M4[01h]	Type B Timer/Counter 0 Compare Register. This register is used for comparison against the TB0V value when compare mode is enabled.
TB1R	M4[02h]	Type B Timer/Counter 1 Capture/Reload Register. Holds the reload value in autoreload mode; used to store capture values in capture mode.
TB1C	M4[03h]	Type B Timer/Counter 1 Compare Register. This register is used for comparison against the TB1V value when compare mode is enabled.
TB2R	M4[04h]	Type B Timer/Counter 2 Capture/Reload Register. Holds the reload value in autoreload mode; used to store capture values in capture mode.
TB2C	M4[05h]	Type B Timer/Counter 2 Compare Register. This register is used for comparison against the TB2V value when compare mode is enabled.
TB0CN	M4[08h]	Type B Timer/Counter 0 Control Register. Contains the control, mode, and interrupt bits.
TB0V	M4[09h]	Type B Timer/Counter 0 Value Register. Contains the current timer/counter value.
TB1CN	M4[0Ah]	Type B Timer/Counter 1 Control Register. Contains the control, mode, and interrupt bits.
TB1V	M4[0Bh]	Type B Timer/Counter 1 Value Register. Contains the current timer/counter value.
TB2CN	M4[0Ch]	Type B Timer/Counter 2 Control Register. Contains the control, mode, and interrupt bits.
TB2V	M4[0Dh]	Type B Timer/Counter 2 Value Register. Contains the current timer/counter value.

21.1 Timer/Counter B Register Descriptions

The following peripheral registers are used to control the LCD display controller. Addresses for all registers are given as "Mx[yy]," where x is the module number (from 0 to 15 decimal) and yy is the register index (from 00h to 1Fh hexadecimal). Fields in the bit definition tables are defined as follows:

- **Name:** Symbolic names of bits or bit fields in this register.
- **Reset:** The value of each bit in this register following a standard reset. If this field reads "unchanged," the given bit is unaffected by standard reset. If this field reads "s," the given bit does not have a fixed 0 or 1 reset value because its value is determined by another internal state or external condition.
- **POR:** If present this field defines the value of each bit in this register following a power-on reset (as opposed to a standard reset). Some bits are unaffected by standard resets and are set/cleared by POR only.
- **Access:** Bits can be read-only (r) or read/write (rw). Any special restrictions or conditions that could apply when reading or writing this bit are detailed in the bit description.

21.1.1 Timer B Timer/Counter 0/1/2 Capture/Reload Register (TB0R, TB1R, TB2R; M4[00h], M4[02h], M4[04h])

Bit #	15	14	13	12	11	10	9	8
Name	TBnR							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	TBnR							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0: Timer B Capture/Reload Register. This register is used to capture the TBnV value when Timer B is configured in capture mode. This register is also used as the 16-bit reload value when Timer B is configured in autoreload mode.

21.1.2 Timer B Timer/Counter 0/1/2 Compare Register (TB0C, TB1C, TB2C; M4[01h], M4[03h], M4[05h])

Bit #	15	14	13	12	11	10	9	8
Name	TBnC							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	TBnC							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0: Timer B Compare Register. This register is used for comparison vs. the TBnV value when Timer B is operated in compare mode.

21.1.3 Timer B Timer/Counter 0/1/2 Control Register (TB0CN, TB1CN, TB2CN; M4[08h], M4[0Ah], M4[0Ch])

Bit #	15	14	13	12	11	10	9	8
Name	C/TB	—	—	TBCS	TBCR	TBPS2	TBPS1	TBPS0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	TFB	EXFB	TBOE	DCEN	EXENB	TRB	ETB	CP/RLB
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15: Counter/Timer Select (C/TB). This bit determines whether Timer B functions as a timer or counter. Setting this bit to 1 causes Timer B to count negative transitions on the TBA pin. Clearing this bit to 0 causes Timer B to function as a timer. The speed of Timer B is determined by the TBPS[2:0] bits of TBnCN.

Bits 14:13: Reserved. Read returns zero.

Bits 12:11: TBB Pin Output Reset/Set Mode Bits (TBCS, TBCR). These mode bits define whether the PWM mode output function is enabled on the TBB pin, the initial output starting state, and what compare mode output function is in effect. Note that the TBB pin still has certain input functionality when the PWM output function is enabled.

Bits 10:8: Timer B Clock Prescaler Bits (TBPS[2:0]). These bits select the clock prescaler applied to the system clock input to Timer B. The TBPS[2:0] bits should be configured by the user when the timer is stopped (TRB = 0). While hardware does not prevent changing the TBPS[2:0] bits when the timer is running, the resultant behavior is indeterministic.

$$\text{Timer B Clock} = \text{System Clock}/2^{(2 \times \text{TBPS}[2:0])}$$

TBPS[2:0]	TIMER B INPUT CLOCK
000	Sysclk/1
001	Sysclk/4
010	Sysclk/16
011	Sysclk/64
100	Sysclk/256
101	Sysclk/1024
11x	Sysclk/1

Bit 7: Timer B Overflow Flag (TFB). This bit is set when Timer B overflows from TBnR or the count is equal to 0000h in down-count mode. It must be cleared by software.

Bit 6: External Timer B Trigger Flag (EXFB). When configured as a Timer (C/TB = 0), a negative transition on the TBB pin causes this flag to be set if (CP/RLB = EXENB = 1) or (CP/RLB = DCEN = 0 and EXENB = 1) or (CP/RLB = 0 and EXENB = 1 and TBCS:TBCR<>00b). When configured in any of these ways, this flag can be set independent of the state of the TRB bit (e.g., EXFB can still be set on detection of a negative edge when TRB = 0).

When CP/RLB = 0 and DCEN = 1 and TBCS:TBCR = 00b, EXFB toggles whenever Timer B underflows or overflows. Overflow/underflow condition is the same as described in TFB bit description. In this mode, EXFB can be used as the 17th timer bit and does not cause an interrupt. If set by a negative transition, this flag must be cleared by software. Setting this bit to 1 forces a timer interrupt if enabled.

Bit 5: Timer B Output Enable (TBOE). Setting this bit to 1 enables the clock output function on the TBA pin if C/TB = 0. Timer B rollovers do not cause interrupts. Clearing this bit to 0 allows the TBA pin to function as either a standard port pin or a counter input for Timer B.

Bit 4: Down-Count Enable (DCEN). This bit, in conjunction with the TBB pin, controls the direction that Timer B counts in 16-bit autoreload mode. Clearing this bit to 0 causes Timer B to count up only. Setting this bit to 1 enables the up/down-counting mode (i.e., it causes Timer B to count up if the TBB pin is 1 and to count down if the TBB pin is 0). When Timer B PWM-output mode functionality is enabled along with up/down counting (DCEN = 1), the up/down count control of Timer B is controlled internally based upon the count in relation to the register settings. In the compare modes, the DCEN bit controls whether the timer counts up and resets (DCEN = 0), or counts up and down (DCEN = 1).

Bit 3: Timer B External Enable (EXENB). Setting this bit to 1 enables the capture/reload function on the TBB pin for a negative transition (in up-counting mode). A reload results in TBnV being reset to 0000h. Clearing this bit to 0 causes Timer B to ignore all external events on TBB pin. When operating in autoreload mode (CP/RLB = 0) with the PWM output functionality enabled, enabling the TBB input function (EXENB = 1) allows PWM output negative transitions to set the EXFB flag, however no reload occurs as a result of the external negative-edge detection.

Bit 2: Timer B Run Control (TRB). This bit enables Timer B operation when set to 1. Clearing this bit to 0 halts Timer B operation and preserves the current count in TBnV.

Bit 1: Enable Timer B Interrupt (ETB). Setting this bit to 1 enables the interrupt from the Timer B TFB and EXFB flags in TBnCN. In Timer B clock-output mode (TBOE = 1), the timer overflow flag (TFB) is still set on an overflow, however, the TBOE = 1 condition prevents this flag from causing an interrupt when ETB = 1.

Bit 0: Capture/Reload Select (CP/RLB). This bit determines whether the capture or reload function is used for Timer B. Timer B functions in an autoreload mode following each overflow/underflow. See the TFB bit description for overflow/underflow condition. Setting this bit to 1 causes a Timer B capture to occur when a falling edge is detected on TBB if EXENB is 1. Clearing this bit to 0 causes an autoreload to occur when Timer B overflow or a falling edge is detected on TBB if EXENB is 1. It is not intended that the Timer B compare functionality should be used when operating in capture mode.

21.1.4 Timer B Timer/Counter 0/1/2 Value Register (TB0V, TB1V, TB2V; M4[09h], M4[0Bh], M4[0Dh])

Bit #	15	14	13	12	11	10	9	8
Name	TBnV							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	TBnV							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0: Timer B Value Register. This register is used to load and read the 16-bit Timer B value.

21.2 Timer/Counter B Operation

Timer/Counter B is a 16-bit programmable device that supports clock input prescaling and set/reset/toggle PWM/output control functionality not found on other MAXQ timer implementations. A new register, TBnC, supports certain PWM/output control functions in some implementations. Another distinguishing characteristic of Timer/Counter B is that its count ranges from 0000h to the value stored in the 16-bit capture/reload register (TBnR) whereas in other implementations (e.g., Timer 1), the count ranges from the value in the reload register to FFFFh.

The possible Timer B operating modes and related control bits are shown in Table 21-3. A complete description of each mode is contained in the subsequent sections. In all modes of operation, the timer is enabled by setting the Timer B run control bit (TRB) of the Timer B control register (TBnCN.2) to 1. If this bit is cleared to 0 (reset default condition), no timer activity is possible. When Timer B is operated as a timer (i.e., it counts scaled system clocks), the three bits TBPS2, TBPS1, and TBPS0 in the timer control register (TBnCN[10:8]) determine the factor by which the active system clock is divided (prescaled) before being counted by the timer. Other relevant control bits are described in the following mode descriptions. A complete listing of the Timer B registers and bits with their effects on timer operation are given in *Section 21.1: Timer/Counter B Register Descriptions*.

Table 21-3. Timer/Counter B Mode Summary

TIMER B OPERATIONAL MODE	TBnCN REGISTER BIT SETTINGS						
	TBCS:TBCR	TBOE	DCEN	EXENB	C/TB	CP/RLB	OPTIONAL CONTROL
Autoreload	00	0	0	0	X	0	—
Autoreload Using TBB Pin	00	0	0	1	X	0	—
Capture Using TBB Pin	00	0	0	1	X	1	—
Up/Down Count Using TBB Pin	00	0	1	0	X	0	—
Up-Count PWM/Output Control	<>00	X	0	X	X	0	—
Up/Down PWM/Output Control	<>00	X	1	X	X	0	—
—	—	0	X	X	1	X	Input Clock = TBA Pin
Clock Output on TBB Pin	—	1	X	X	0	0	—

21.2.1 Timer B 16-Bit Timer/Counter Mode with Autoreload

The 16-bit autoreload mode of Timer B is established by clearing the CP/RLB bit of the control register (TBnCN.0) to 0. In this mode, the timer performs a simple 16-bit timer or counter function that is reset to 0000h when a match between the Timer B count value register (TBnV) and the Timer B capture/reload register (TBnR) occurs. A functional diagram of autoreload mode is illustrated in Figure 21-1. If the C/TB bit of the timer's control register (TBnCN.15) is a logic 0, the timer's input clock is a prescaled system clock. When C/TB is a logic 1, pulses on the TBA pin are counted. As in all modes, counting or timing is enabled or disabled with the Timer B run control bit TRB (TBnCN.2)

When enabled (i.e., TRB = 1) in this mode, Timer B begins counting up from the current value contained in the TBnV register. The TBnV register contents can be read or written any time by software and contains the current value of the timer. When the value in the TBnV register reaches the value contained in capture/reload register TBnR, the TBF flag is set to 1. This flag can generate an interrupt if enabled. In addition, upon this match, the timer reloads the TBnV register with 0000h, and continues timing (or counting) up from that value. The reload value contained in the TBnR register is preloaded by software. This register cannot be used for the capture function while also performing autoreload, so these modes are mutually exclusive.

While in autoreload mode, Timer B can also be forced to reload the TBnV register with 0000h using the TBB pin. If the control bit EXENB (TBnCN.3) is set to 1, a 1-to-0 transition on the TBB pin causes a reload. If the EXENB bit is cleared to 0, the TBB pin is ignored.

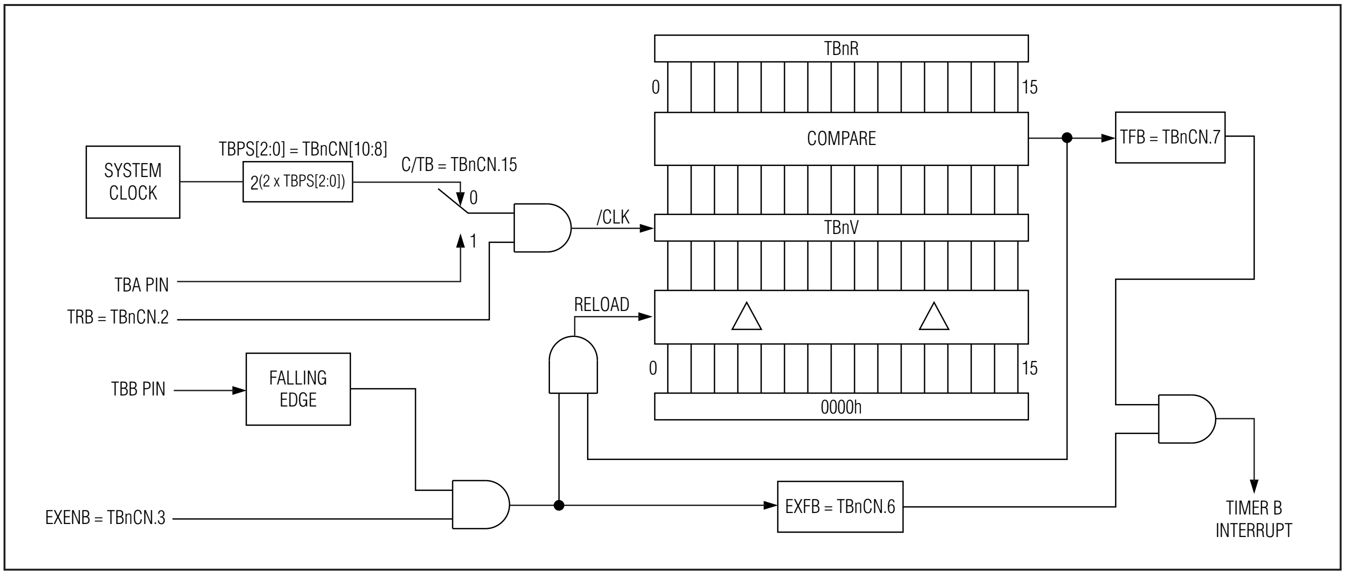


Figure 21-1. Timer B Autoreload Mode Block Diagram

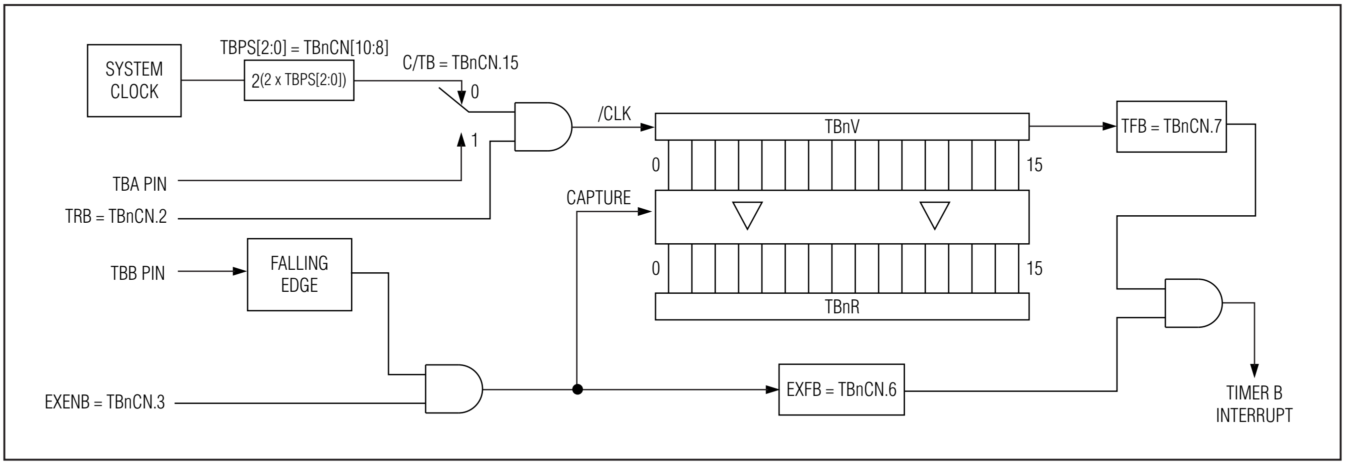


Figure 21-2. Timer B 16-Bit Capture Mode Block Diagram

21.2.2 Timer B 16-Bit Capture Mode

The 16-bit capture mode of Timer B is configured by setting the CP/RLB bit of the control register (TBnCN.0) to 1. A functional diagram of this mode is shown in Figure 21-2. When the timer is enabled in this mode, it begins counting up from the value contained in the TBnV register until reaching an overflow state, i.e., FFFFh → 0000h; at which point it sets the TBF flag (TBnCN.7) and continues counting upward. When the TBF flag is set, it can generate an interrupt if enabled. This count cycle is repeated without processor intervention as long as the timer is enabled. As the counting proceeds, the value in the TBnV register is captured in the capture/reload register (TBnR) if and when a high-to-low transition occurs on the TBB pin and the EXENB bit of the control register (TBnCN.3) is set to 1. The EXFB flag (TBnCN.6) is also set when the capture occurs, and this flag can generate an interrupt if enabled. If the EXENB bit is cleared to 0, transitions on the TBB pin do not cause a capture event.

21.2.3 Timer B 16-Bit Up/Down Count with Autoreload Mode

The 16-bit up/down-count autoreload mode is enabled by clearing the capture/reload bit CP/RLB of the control register (TBnCN.0) to 0 and setting the down-count enable bit, DCEN (TBnCN.4), to 1. This mode is illustrated in Figure 21-3. When DCEN is set to 1, Timer B counts up from 0000h or down from the value contained in the TBnR register as controlled by the state of the TBB pin. When the TBB pin is 1, the timer counts up, and counts down when the pin is 0. When DCEN is 0, Timer B can only count up.

When counting up and a match occurs between the value in the TBnV register and value in the TBnR register, the value of 0000h is loaded into the TBnV register. When counting down and the value in the TBnV register reaches 0000h, the value in the TBnR register is loaded into the TBnV register and downward counting continues.

Note that in this mode of operation, the overflow/underflow output of the timer is provided to an edge-detection circuit as well as to the TBF bit of the control register (TBnCN.7). This edge-detection circuit toggles the EXFB bit (TBnCN.6) on every overflow or underflow. Therefore, the EXFB bit behaves as a 17th bit of the counter, and can be used as such.

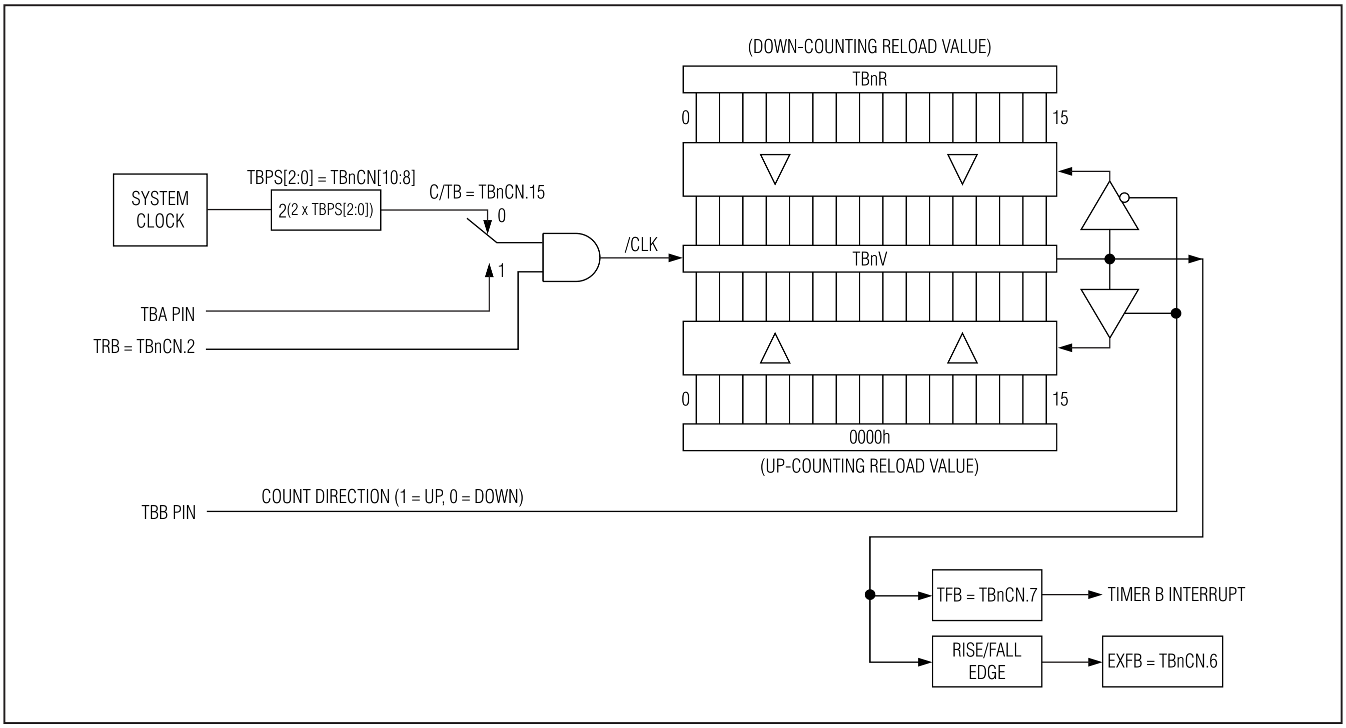


Figure 21-3. Timer B 16-Bit Up/Down Count with Autoreload Mode Block Diagram

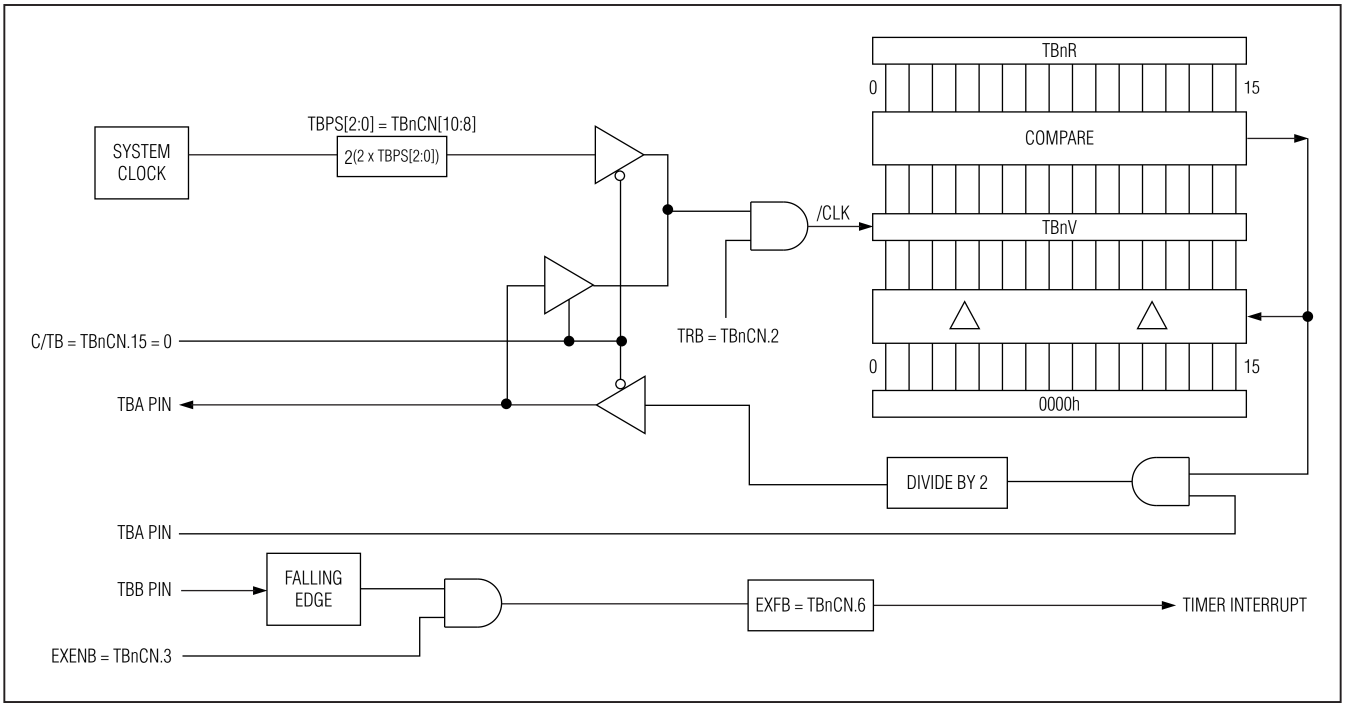


Figure 21-4. Timer B Clock Output Mode Block Diagram

21.2.4 Timer B Clock Output Mode

Timer B can be configured to drive a clock output on the TBA pin as shown in Figure 21-4. For the timer to operate in this mode, the capture/reload select bit (CP/RLB = TBnCN.0) and the counter/timer select bit (C/TB = TBnCN.15) must be cleared to 0, and the Timer B output-enable bit (TBOE = TBnCN.5)) must be set to 1. In this mode, the DCEN bit has no effect. The clock signal output is a 50% duty-cycle square wave with a frequency given by the equation:

Clock Output Frequency = System Clock/(2 x TBnR)

Therefore, for a system clock of 1MHz and a TBnR register value of 0005h (arbitrary example), the clock output frequency is 100kHz.

21.2.5 Timer B PWM/Output Control Functionality

The PWM/output control function is enabled whenever either of the TCBS or TBCR bits (TBnCN[12:11]) is set to 1. Table 21-4 shows how these bits determine the specific operation.

Table 21-4. Timer B PWM/Output Control Function

TBCS:TBCR	FUNCTION	INITIAL STATE (IF TRB = 0)
00	None (Disabled)	No change
01	Reset on TBnC Match, Set on 0000h	Low
10	Set on TBnC Match, Reset on TBnR Match	High
11	Toggle on TBnC Match	No change

When the timer is not running (i.e., TRB = 0), the initial output state of the TBB pin is established as low or high, respectively, if the reset function (TBCR = 1,TBCS = 0) or set function (TBCR = 0, TBCS = 1) is configured. Invoking the toggle function does not change the already defined starting state for TBB, thus a fixed high or low starting state can be defined for the toggle mode by first passing through the set or reset mode.

When the PWM/output control function is configured to the reset mode (TBCS = 0, TBCR = 1), clearing the TBnC register to 0000h or writing its value to something greater than the counting range (i.e., greater than the value in the TBnR register) effectively disables the reset operation, and produces a single pin set operation when an overflow occurs. When the PWM/output control function is configured to the set mode (TBCS = 1, TBCR = 0), making TBnC = TBnR or writing TBnC's value to something greater than the counting range disables the set operation, and produces a single reset on TBnR match. When the PWM/output control function is configured to toggle, loading the TBnC register with a value outside the counting range, to 0000h, or to the value in TBnR disables the toggle function.

21.2.6 16-Bit Up Count PWM/Output Control Mode

Figure 21-5 shows a functional diagram of the up count with PWM/output control mode, and Figure 21-6 shows example TBB pin output waveforms. The set and reset modes provide similar functionality. They support up to 16-bit resolution PWM with the ability to change the frequency using the TBnR reload value. The toggle mode allows a 50% duty-cycle waveform to be created (when the TBnC register remains fixed with Timer B running). With the TBnC register value outside of the count range, the set and reset functions allow a timed clear or set of the TBB pin without need of polling or interrupting the CPU for manual port pin control.

Up-count set, reset PWM duty cycle is calculated as follows (where period = TBnR + 1 Timer B clocks):

Set mode = (TBnR - TBnC)/(TBnR + 1)

Reset mode = TBnC/(TBnR + 1)

The period for the 50% duty-cycle signal generated in toggle mode is:

Toggle mode = 2 x (TBnR + 1) Timer B Clock Periods

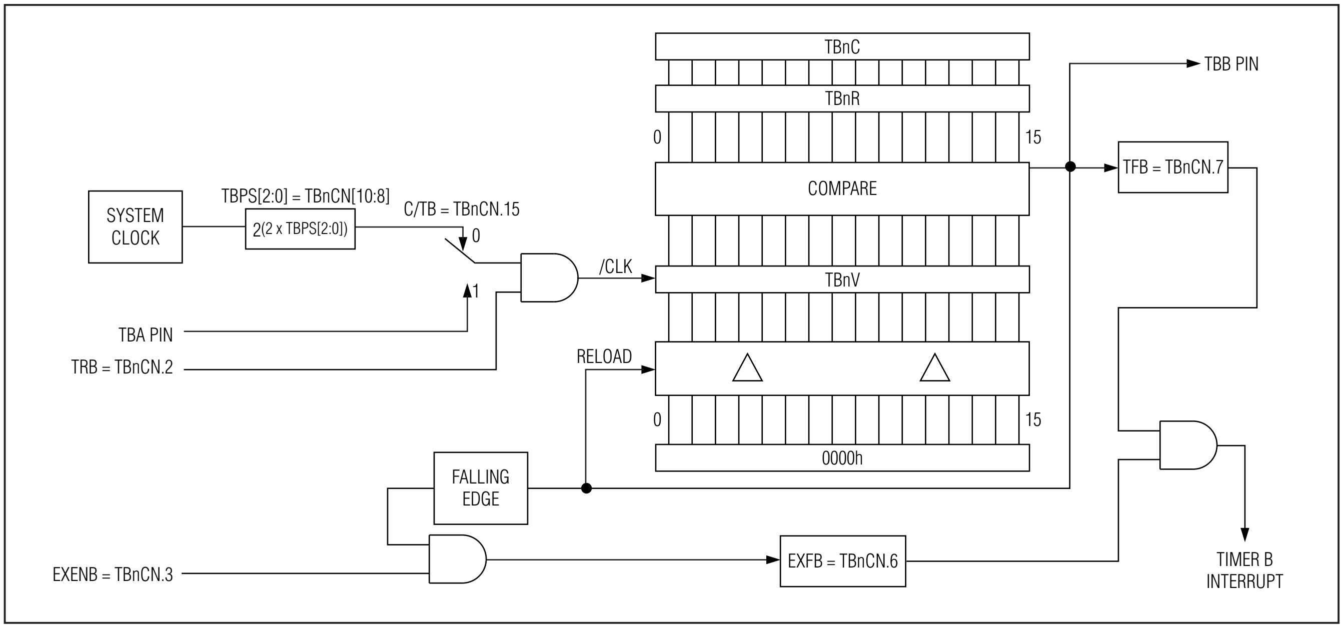


Figure 21-5. Up-Count PWM/Output Control Mode Block Diagram

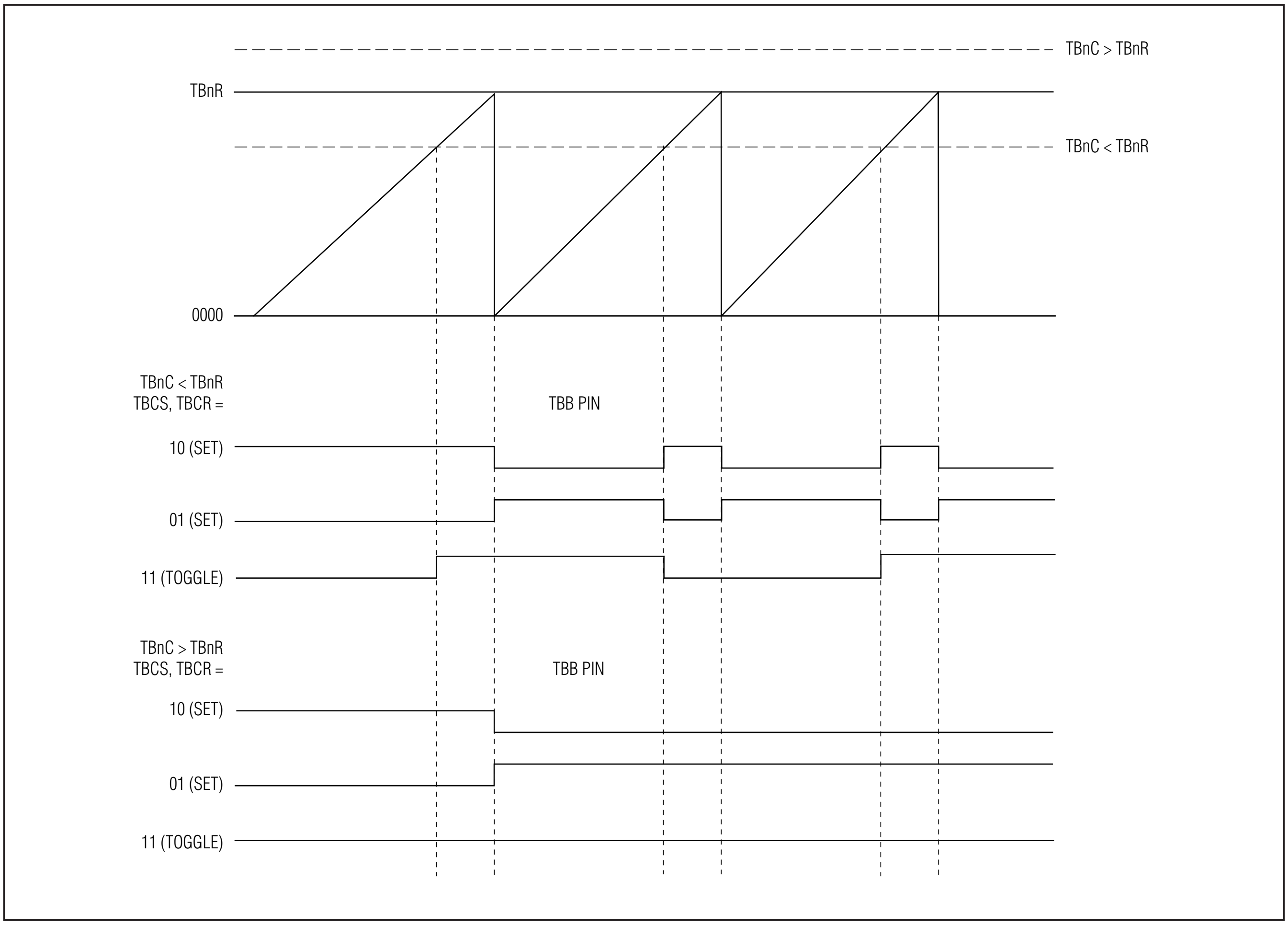


Figure 21-6. Timer B PWM/Output Control Mode Waveform (Count Up)

21.2.7 16-Bit Up/Down Count PWM/Output Control Mode

Figure 21-7 shows a functional diagram of the up/down-count PWM/output control mode. When the Timer B PWM/output control functionality is enabled at the same time as the up/down-count autoreload mode, the TBB pin no longer controls the direction of counting. Instead, the up/down counting is controlled by logic inside the timer and is determined by the value of the TBnV count value register. When the timer is counting upward and reaches the value in the TBnR register, it reverses its direction of counting in the next cycle. When the timer is down counting and reaches 0000h, it reverses direction and begins counting up. This behavior and the results of the TBCS and TBCR bit setting is shown in Figure 21-8.

The up/down-count PWM duty cycle is calculated as follows (where period = 2 x TBnR Timer B clocks):

Set mode = (TBnR + TBnC)/(2 x TBnR)

Reset mode = TBnC/(2 x TBnR)

Toggle mode = TBnC/TBnR or (TBnR - TBnC)/TBnR

The set and reset up/down-count PWM/output control modes effectively allow 17-bit resolution since set allows duty-cycle variation ≥ 50% with 50% of the period always being high, and reset allows duty-cycle variation ≤ 50% with 50% of the period always being low. The toggle mode provides a center-aligned 16-bit PWM with twice the period of the pure up-counting autoreload mode.

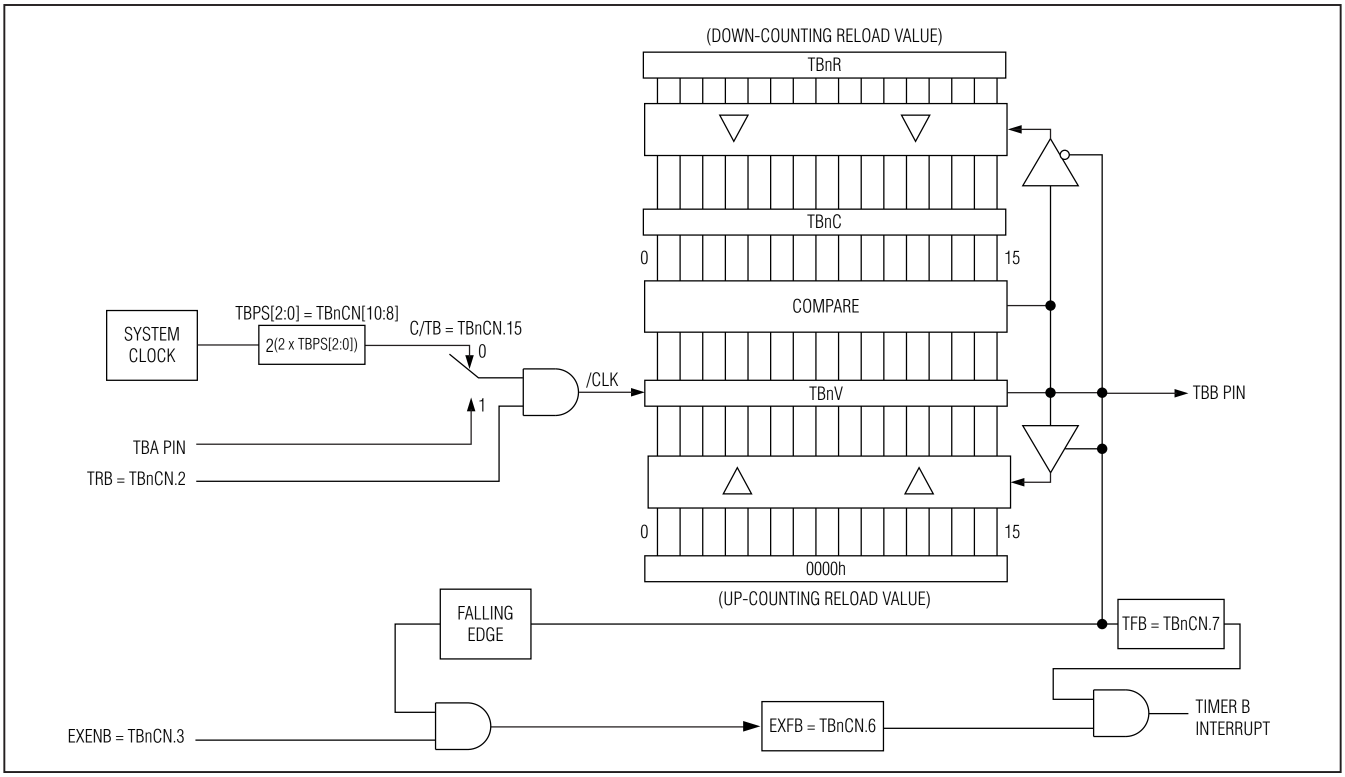


Figure 21-7. Timer B Up/Down-Count PWM/Output Control Mode Block Diagram

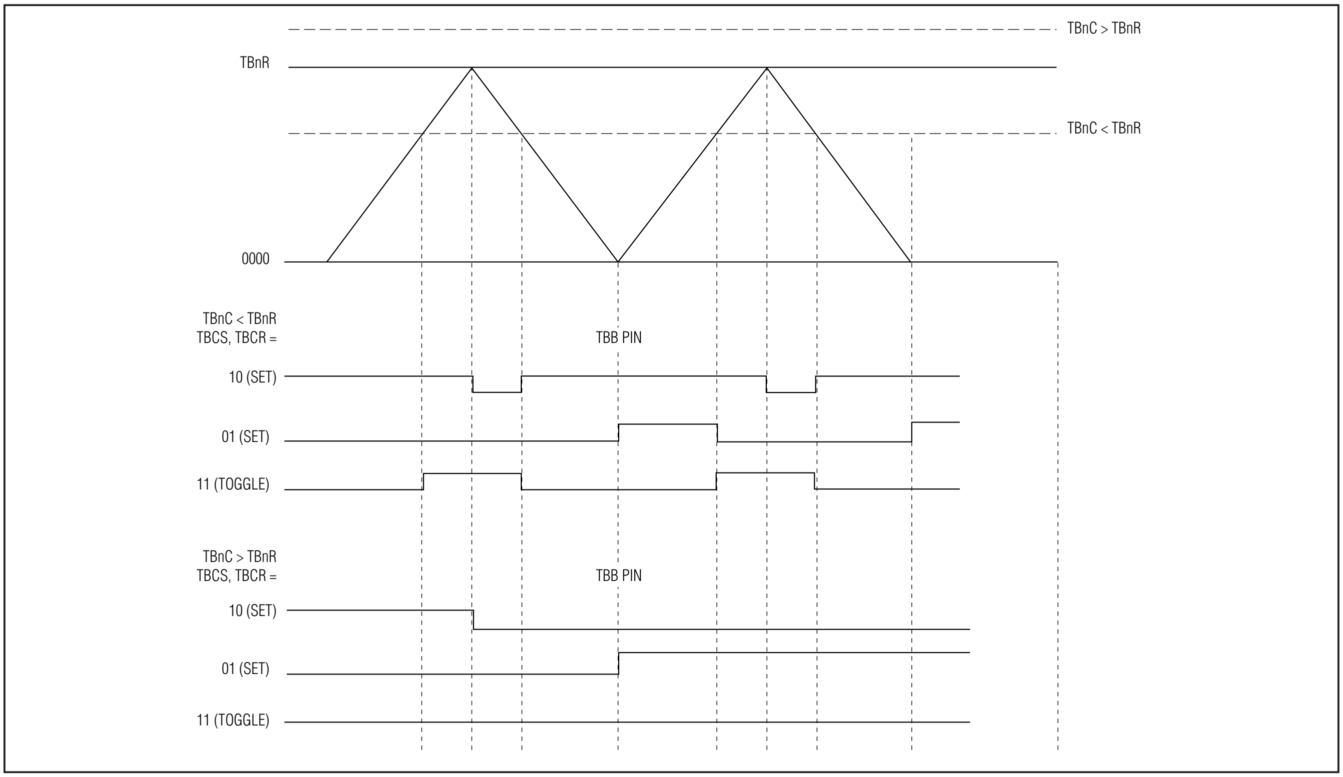


Figure 21-8. Timer B PWM/Output Control Mode Waveform (Up/Down Count)

21.2.8 EXENB Control During PWM/Output Control Mode

The TBB input function (EXENB = 1) and the PWM/output control function (TBCS:TBCR<>00b) can be enabled at the same time. However, the input function changes slightly when this is done. In this configuration, the detection of a falling edge on the TBB pin results in setting of the EXFB interrupt flag, but does **not** force an autoreload.

21.3 Timer B Examples

21.3.1 Timer B Example: Reloading Timer Mode

```
  move  PD5.2, #1       ; Set P5.2 to output mode
  move  PO5.2, #1       ; Drive high (LED off)

  move  TB0CN, #0000010000000000b ; Timer mode, reload mode, clock/256
  move  TB0R,  #8000h   ; Reload every 8000h timer cycles
  move  TB0CN.2, #1     ; Start timer

  ;; Reload time is 1/sysclk * 256 * 8000h == 838ms

timerLoop:
  move  C, TB0CN.7      ; Check reload flag
  jump  NC, timerLoop

  move  TB0CN.7, #0     ; Clear reload flag
  move  Acc, PO5
  xor   #0100b          ; Toggle bit 2
  move  PO5, Acc
  jump  timerLoop
```

SECTION 22 : I2C BUS INTERFACE (SPECIFIC TO MAXQ2010)

The MAXQ2010 provides an inter-IC (I2C) communications module that includes master and slave modes. The associated pins and registers for this interface are listed in Table 22-1 and Table 22-2.

Table 22-1. I2C Input and Output Pins

I2C INTERFACE FUNCTION	PIN	MULTIPLEXED WITH GPIO
SCL: Clock	24	P6.6
SDA: Data	23	P6.7

Table 22-2. I2C Interface Control Registers

REGISTER	ADDRESS	FUNCTION
I2CBUF	M3[00h]	I2C Data Buffer Register. Interface register for the input and output I2C buffers.
I2CST	M3[01h]	I2C Status Register. Contains the interrupt and status flags for the I2C interface.
I2CIE	M3[02h]	I2C Interrupt Enable Register. Contains the interrupt enable bits and control bits for general call address matching and clock stretching.
I2CCN	M3[0Ch]	I2C Control Register. Contains the control and configuration bits for the I2C interface.
I2CCK	M3[0Dh]	I2C Clock Control Register. Defines the SCL high and low clock periods for I2C master mode.
I2CTO	M3[0Eh]	I2C Timeout Register. Enables/disables the master mode timeout when the bus is busy or SCL is held low by another device longer than the maximum allowed time, and defines the timeout period.
I2CSLA	M3[0Fh]	I2C Slave Address Register. Defines the 7-bit address that is recognized by the slave portion of the I2C interface.

22.1 I2C Register Descriptions

The following peripheral registers are used to control the integrated I2C peripheral on the MAXQ2010. Addresses for all registers are given as "Mx[yy]," where x is the module number (from 0 to 15 decimal) and yy is the register index (from 00h to 1Fh hexadecimal). Fields in the bit definition tables are defined as follows:

- **Name:** Symbolic names of bits or bit fields in this register.
- **Reset:** The value of each bit in this register following a standard reset. If this field reads "unchanged," the given bit is unaffected by standard reset. If this field reads "s," the given bit does not have a fixed 0 or 1 reset value because its value is determined by another internal state or external condition.
- **POR:** If present this field defines the value of each bit in this register following a power-on reset (as opposed to a standard reset). Some bits are unaffected by standard resets and are set/cleared by POR only.
- **Access:** Bits can be read-only (r) or read/write (rw). Any special restrictions or conditions that could apply when reading or writing this bit are detailed in the bit description.

22.1.1 I2C Data Buffer Register (I2CBUF, M3[00h])

Bit #	7	6	5	4	3	2	1	0
Name	I2CBUF							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Note: *ADCONV cannot be written when PMME = 1 and SWB = 0.*

22.1.1.1 I2C Data Read and Write

Data for I2C transfer is read and written to this location. The I2C transmit and receive buffers are internally stored separately; however, both are accessed through this buffer.

22.1.1.2 I2C Address Transmission

When transmitting an I2C address, the address should be loaded into I2CBUF[6:0]. I2CBUF[7] is ignored and is not part of the I2C address.

22.1.2 I2C Status Register (I2CST, M3[01h])

Bit #	15	14	13	12	11	10	9	8
Name	I2CBUS	I2CBUSY	—	—	I2CSPI	I2CSCL	I2CROI	I2CGCI
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	rw	r	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	I2CNACKI	I2CALI	I2CAMI	I2CTOI	I2CSTRI	I2CRXI	I2CTXI	I2CSRI
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15: I2C Bus Busy (I2CBUS). This bit is set to 1 when a START/repeated START condition is detected and cleared to 0 when the STOP condition is detected. This bit is reset to 0 on all forms of reset and when I2CEN = 0. This bit is controlled by hardware and is read-only.

Bit 14: I2C Busy (I2CBUSY). This bit is used to indicate the current status of the I2C module. The I2CBUSY is set to 1 when the I2C controller is actively participating in a transaction or when it does not have control of the bus. This bit is controlled by hardware and is read-only.

Bits 13:12: Reserved. Read returns 0.

Bit 11: I2C STOP Interrupt Flag (I2CSPI). This bit is set to 1 when a STOP condition (P) is detected. This bit must be cleared to 0 by software once set. Setting this bit to 1 by software causes an interrupt if enabled.

Bit 10: I2C SCL Status (I2CSCL). This bit reflects the logic state of SCL signal. This bit is set to 1 when SCL is at a logic-high (1), and cleared to 0 when SCL is at a logic-low (0). This bit is controlled by hardware and is read-only.

Bit 9: I2C Receiver Overrun Flag (I2CROI). This bit indicates a receive overrun when set to 1. This bit is set to 1 if the receiver has already received two bytes since the last CPU read. This bit is cleared to 0 by software reading the I2CBUF. Setting this bit to 1 by software causes an interrupt if enabled. Writing 0 to this bit does not clear the interrupt.

Bit 8: I2C General Call Interrupt Flag (I2CGCI). This bit is set to 1 when the general call is enabled (I2CGCEN = 1) and the general call address is received. This bit must be cleared to 0 by software once set. Setting this bit to 1 by software causes an interrupt if enabled.

Bit 7: I2C NACK Interrupt Flag (I2CNACKI). This bit is set to 1 if the I2C transmitter receives a NACK from the receiver. Setting this bit to 1 by hardware causes an interrupt if enabled. This bit must be cleared to 0 by software once set. This bit is set by hardware only.

Bit 6: I2C Arbitration Loss Flag (I2CALI). This bit is set to 1 when the I2C is configured as a master and loses in the arbitration. When the master loses arbitration, the I2CMST bit is cleared to 0. Setting this bit to 1 by hardware causes an interrupt if enabled. This bit must be cleared to 0 by software once set. This bit is set by hardware only.

Bit 5: I2C Slave Address Match Interrupt Flag (I2CAMI). This bit is set to 1 when the I2C controller receives an address that matches the contents in its slave address register (I2CSLA) during the address stage. This bit must be cleared to 0 by software once set. Setting this bit to 1 by software causes an interrupt if enabled.

Bit 4: I2C Timeout Interrupt Flag (I2CTOI). This bit is set to 1 if either the I2C controller cannot generate a START condition or the I2C SCL low time has expired the timeout value specified in the I2CTO register. This happens when the I2C controller is operating in master mode and some other device on the bus is using the bus or holding SCL low for an extended period of time. This bit must be cleared to 0 by software once set. Setting this bit to 1 by software causes an interrupt if enabled.

Bit 3: I2C Clock Stretch Interrupt Flag (I2CSTRI). This bit indicates that the I2C controller is operating with clock stretching enabled and is holding the SCL clock signal low. The I2C controller releases SCL after this bit has been cleared to 0. Setting this bit to 1 by hardware causes an interrupt if enabled. This bit must be cleared to 0 by software once set. This bit is set by hardware only.

Bit 2: I2C Receive Ready Interrupt Flag (I2CRXI). This bit indicates that a data byte has been received in the I2C buffer. This bit must be cleared by software once set. Setting this bit to 1 by hardware causes an interrupt if enabled. This bit is set by hardware only.

Bit 1: I2C Transmit Complete Interrupt Flag (I2CTXI). This bit indicates that an address or a data byte has been successfully shifted out and the I2C controller has received an acknowledgment from the receiver (NACK or ACK). This bit must be cleared by software once set. Setting this bit to 1 by software causes an interrupt if enabled.

Bit 0: I2C START Interrupt Flag (I2CSRI). This bit is set to 1 when a START condition (S or Sr) is detected. This bit must be cleared to 0 by software once set. Setting this bit to 1 by software causes an interrupt if enabled.

22.1.3 Interrupt Enable Register (I2CIE, M3[02h])

Bit #	15	14	13	12	11	10	9	8
Name	—	—	—	—	I2CSPIE	—	I2CROIE	I2CGCIE
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	r	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	I2CNACKIE	I2CALIE	I2CAMIE	I2CTOIE	I2CSTRIE	I2CRXIE	I2CTXIE	I2CSRIE
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:12, 10: Reserved. Read returns 0.

Bit 11: I2C STOP Interrupt Enable (I2CSPIE). Setting this bit to 1 causes an interrupt to the CPU when a STOP condition is detected (I2CSPI = 1). Clearing this bit to 0 disables the STOP detection interrupt from generating.

Bit 9: I2C Receiver Overrun Interrupt Enable (I2CROIE). Setting this bit to 1 causes an interrupt to the CPU when a receiver overrun condition is detected (I2ROI = 1). Clearing this bit to 0 disables receiver overrun detection interrupt from generating.

Bit 8: I2C General Call Interrupt Enable (I2CGCIE). Setting this bit to 1 generates an I2CGCI (general call interrupt) to the CPU when general call is enabled (I2CGCEN = 1). Clearing this bit to 0 disables general call interrupt from generating.

Bit 7: I2C NACK Interrupt Enable (I2CNACKIE). Setting this bit to 1 causes an interrupt to the CPU when a NACK is detected (I2CNACKI = 1). Clearing this bit to 0 disables NACK detection interrupt from generating.

Bit 6: I2C Arbitration Loss Enable (I2CALIE). Setting this bit to 1 causes an interrupt to the CPU when the I2C master loses in an arbitration (I2CALI = 1). Clearing this bit to 0 disables arbitration loss interrupt from generating.

Bit 5: I2C Slave Address Match Interrupt Enable (I2CAMIE). Setting this bit to 1 causes an interrupt to the CPU when the I2C controller detects an address that matches the I2CSLA value (I2CAMI = 1). Clearing this bit to 0 disables address match interrupt from generating.

Bit 4: I2C Timeout Interrupt Enable (I2CTOIE). Setting this bit to 1 causes an interrupt to the CPU when a timeout condition is detected (I2CTOI = 1). Clearing this bit to 0 disables timeout interrupt from generating. **Bit 3: I2C Clock**

Stretch Interrupt Enable (I2CSTRIE). Setting this bit to 1 generates an interrupt to the CPU when the clock stretch interrupt flag is set (I2CSTRI = 1). Clearing this bit disables the clock stretch interrupt from generating.

Bit 2: I^{2}C Receive Ready Interrupt Enable (I2CRXIE). Setting this bit to 1 causes an interrupt to the CPU when the receive interrupt flag is set (I2CRXI = 1). Clearing this bit to 0 disables the receive interrupt from generating.

Bit 1: I^{2}C Transmit Complete Interrupt Enable (I2CTXIE). Setting this bit to 1 causes an interrupt to the CPU when the transmit interrupt flag is set (I2CTXI = 1). Clearing this bit to 0 disables the transmit interrupt from generating.

Bit 0: I^{2}C START Interrupt Enable (I2CSRIE). Setting this bit to 1 causes an interrupt to the CPU when a START condition is detected (I2CSRI = 1). Clearing this bit to 0 disables the START detection interrupt from generating.

22.1.4 I^{2}C Control Register (I2CCN, M3[0Ch])

Bit #	15	14	13	12	11	10	9	8
Name	I2CRST	—	—	—	—	—	I2CSTREN	I2CGCEN
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	I2CSTOP	I2CSTART	I2CACK	I2CSTRS	—	I2CMODE	I2CMST	I2CEN
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Note 1: *I2CSTART and I2CSTOP are mutually exclusive and reset to 0 when I2CMST = 0 or I2CEN = 0.*

Note 2: *I2CRST is reset to 0 when I2CEN = 0.*

Note 3: *Writes to I2CMST, I2CMODE, and I2CEN are ignored when I2CBUSY = 1.*

Note 4: *If I2CRST = 1, I2CEN can be written when I2CBUSY = 1.*

Note 5: *Write to I2CACK is ignored if I2RST = 1.*

Bit 15: I^{2}C Reset (I2CRST). Setting this bit to 1 aborts the current transaction and resets the I^{2}C controller. This bit is set to 1 by software and is only cleared to 0 by hardware after the reset or when I2CEN = 0.

Bits 14:10, 3: Reserved. Read returns 0.

Bit 9: I^{2}C Clock Stretch Enable (I2CSTREN). Setting this bit to 1 stretches the clock (holds SCL low) at the end of the clock cycle specified in I2CSTRS. Clearing this bit disables clock stretching.

Bit 8: I^{2}C General Call Enable (I2CGCEN). Setting this bit to 1 enables the I^{2}C to respond to a general call address (address = 0000 0000). Clearing this bit to 0 prevents the I^{2}C from responding to the general call address.

Bit 7: I^{2}C STOP Enable (I2CSTOP). Setting this bit to 1 generates a STOP condition. This bit automatically is self-cleared to 0 after the STOP condition has been generated.

In master mode, setting this bit can also start the timeout timer if enabled. If the timeout timer expires before the STOP condition can be generated, a timeout interrupt is generated to the CPU if enabled. The I2CSTOP bit is also cleared to 0 by the timeout event.

Note that this bit has no effect when the I^{2}C is operating in slave mode (I2CMST = 0), and is reset to 0 when I2CMST = 0 or I2CEN = 0. Setting the I2CSTOP bit to 1 while I2CSTART = 1 is an invalid operation and is ignored, leaving I2CSTOP bit cleared to 0.

Bit 6: I^{2}C START (I2CSTART). Setting this bit automatically generates a START condition when the bus is free or generates a repeated START condition during a transfer where the I^{2}C module is operating as the master. This bit is automatically self-cleared to 0 after the START condition has been generated. If the I^{2}C START interrupt is enabled, a START condition generates an interrupt to the CPU.

In master mode, setting this bit can also start the timeout timer if enabled. If the timeout timer expires before the START condition can be generated, a timeout interrupt is generated to the CPU if enabled. The I2CSTART bit is also cleared to 0 by the timeout event.

Note that this bit has no effect when the I^{2}C is operating in slave mode (I2CMST = 0) and is reset to 0 when I2CMST = 0 or I2CEN = 0. Also, the I2CSTART and I2CSTOP are mutually exclusive. If both bits are set at the same time, it is considered as an invalid operation and the I^{2}C controller ignores the request and resets both bits to 0. Setting the I2CSTART bit to 1 while I2CSTOP = 1 is an invalid operation and is ignored, leaving the I2CSTART bit cleared to 0.

Bit 5: I^{2}C Data Acknowledge Bit (I2CACK). This bit selects the acknowledge bit returned by the I^{2}C controller while acting as a receiver. Setting this bit to 1 generates a NACK (leaving SDA high). Clearing the I2CACK bit to 0 generates an ACK (pulling SDA low) during the acknowledgement cycle. This bit retains its value unless changed by software or hardware. When an I^{2}C abort is in progress (I2CRST = 1), this bit is set to 1 by hardware, and software writes to this bit are ignored when I2CRST = 1.

Bit 4: I^{2}C Clock Stretch Select (I2CSTRS). Setting this bit to 1 enables clock stretching after the falling edge of the 8th clock cycle. Clearing this bit to 0 enables clock stretching after the falling edge of the 9th clock cycle. This bit has no effect when clock stretching is disabled (I2CSTREN = 0).

Bit 2: I^{2}C Transfer Mode (I2CMODE). The transfer mode bit selects the direction of data transfer with respect to the master. When the I2CMODE bit is set to 1, the master is operating in receiver mode (reading from slave). When the I2CMODE bit is cleared to 0, the master is operating in transmitter mode (writing to slave).

Note that software writing to this bit is prohibited in slave mode. When operating in master mode, software configures this bit to the desired direction of data transfer. When operating in slave mode, the direction of data transfer is determined by the RW bit received during the address stage, and this bit reflects the actual R//W bit value in the current transfer and is set by hardware. Software writing to this bit in slave mode is ignored.

Bit 1: I^{2}C Master-Mode Enable (I2CMST). The I2CMST bit functions as a master-mode enable bit for the I^{2}C module. When the I2CMST bit is set to 1, the I^{2}C operates as a master. When the I2CMST is cleared to 0, the I^{2}C module operates in slave mode. This bit is automatically cleared whenever the I^{2}C controller receives a slave address match (I2CAMI = 1), loses arbitration (I2CALI = 1), or receives a general call address (I2CGCI = 1, I2CGCEN = 1).

Bit 0: I^{2}C Enable (I2CEN). This bit enables the I^{2}C function. When set to 1, the I^{2}C communication unit is enabled. When cleared to 0, the I^{2}C function is disabled.

22.1.5 Clock Control Register (I2CCK, M3[0Dh])

Bit #	15	14	13	12	11	10	9	8
Name	I2CCKH							
Reset	0	0	0	0	0	0	1	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	I2CCKL							
Reset	0	0	0	0	0	1	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Note 1: *Write to this register is ignored when I2CBUSY = 0.*

Note 2: *This register has no function in slave mode.*

Bits 15:8: I^{2}C Clock High (I2CCKH[7:0]). These bits define the I^{2}C SCL high period in number of system clock, with bit 7 as the most significant bit. The duration of SCL high time is calculated using the following equation:

I^{2}C High Time Period = System Clock x (I2CCKH[7:0] + 1)

When operating in master mode, I2CCKH must be set to a minimum value of 2 to ensure proper operation. Any value less than 2 is set to 2.

Bits 7:0: I^{2}C Clock Low (I2CCKL[7:0]). These bits define the I^{2}C SCL low period in number of system clock, with bit 7 as the most significant bit. The duration of SCL low time is calculated using the following equation:

I^{2}C Low Time Period = System Clock x (I2CCKL[7:0] + 1)

When operating in master mode, I2CCKL must be set to a minimum value of 4 to ensure proper operation. Any value less than 4 is set to 4.

22.1.6 I^{2}C Timeout Register (I2CTO, M3[0Eh])

Bit #	7	6	5	4	3	2	1	0
Name	I2CTO							
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0: I^{2}C Timeout Register. This register is used only in master mode. This register determines the number of the I^{2}C bit period (SCL high + SCL low) the I^{2}C master waits for SCL to go high. The timeout timer resets to 0 and starts to count after the I2CSTART bit is set, or every time the SCL goes low. When cleared to 00h, the timeout function is disabled and the I^{2}C waits for SCL to go high indefinitely during a transmission. When set to any other values, the I^{2}C waits until the timeout expires and sets the I2CTOI flag.

I^{2}C Timeout = I^{2}C Bit Rate x (I2CTO[7:0] + 1)

Note that these bits have no effect when the I^{2}C module is operating in slave mode (I2CMST = 0). When operating in slave mode, SCL is controlled by an external master.

22.1.7 I^{2}C Slave Address Register (I2CSLA, M3[0Fh])

Bit #	7	6	5	4	3	2	1	0
Name	—	I2CSLA						
Reset	0	0	0	0	0	0	0	0
Access	r	rw	rw	rw	rw	rw	rw	rw

Bit 7: Reserved. Read returns zero.

Bits 6:0: I^{2}C Slave Address Register. These address bits contain the address of the I^{2}C device. When a match to this address is detected, the I^{2}C controller automatically acknowledges the transmitter with the I2CACK bit value if the I^{2}C module is enabled (I2CEN = 1). The I2CAMI flag is set to 1 and the I2CMST bit is cleared to 0. An interrupt is generated to the CPU if enabled.

22.2 I2C Code Examples

22.2.1 I2C Example 1: Master Mode Transmit

```
; I2C configured as master, transmit to slave address 08h

; Setup for Master Mode Transmit

  move I2CCN, #003h      ; I2CEN = 1, I2CMST = 1
  call wait_busy         ; Polling routine to wait for I2CBUSY to clear

  move I2CCN, #043h      ; I2CEN = 1, I2CMST = 1, I2CMODE = 0, I2CSTART = 1
  call wait_start        ; Polling routine to wait for I2CSTART to clear
  call wait_busy         ; Polling routine to wait for I2CBUSY to clear

  move I2CIE.1, #01h     ; Enable Transmit Complete Interrupt
  move I2CBUF, #008h     ; Slave address set to 08h
  call wait_tx_complete  ; Wait for transmit interrupt


;; Verify ACK from slave

  move ACC, I2CST        ; Move I2C Status Register to accumulator
  and  #080h             ; Check for NACK bit set in status register
  cmp  #000h
  jump ne, FAIL          ; If NACK bit set, handle retransmission, else continue

  move I2CBUF, #0aah     ; Byte to transmit
  call wait_tx_complete  ; Wait for transmit interrupt
```

22.2.2 I2C Example 2: Master Mode Receive

```
; I2C configured as master, receive from slave address 08h:

; Setup for Master Mode Receive

  move I2CCN, #047h      ; I2CEN = 1, I2CMST = 1, I2CMODE = 1, I2CSTART = 1
  call wait_start        ; Polling routine to wait for I2CSTART to clear
  call wait_busy         ; Polling routine to wait for I2CBUSY to clear

  move I2CIE.2, #01h     ; Enable Receive Ready Interrupt
  move I2CBUF, #008h     ; Slave address set to 08h
  call wait_tx_complete  ; Wait for transmit interrupt
  call wait_rxbuf        ; Wait for receive interrupt

;; Byte received in I2CBUF, clear I2C interrupt flag and wait for next interrupt
```

22.2.3 I^{2}C Example 3: Slave Mode Receive

```
; I2C configured as slave with address 1ah

; Setup for Slave Mode Receive

  move  I2CSLA, #01ah    ; I2C Slave Address = 01ah
  move  I2CCN, #0001h    ; I2CEN = 1, I2CMST = 0, I2CMODE = 0, I2CSTART = 0
  call  wait_start       ; Polling routine to wait for I2CSTART to be set,
                         ; indicating a received START

;; Check for address match

  move  ACC, I2CST
  and   #0020h           ; Check for Address Match flag set
  cmp   #0020h
  jump  ne, no_match     ; If address match bit not set, not for us, else:
  move  I2CIE.2, #01h    ; Enable Receive Ready Interrupt
  call  wait_rxbuf       ; Wait for a receive interrupt

;; Byte received in I2CBUF, clear I2C interrupt flag and wait for next interrupt
```

22.2.4 I^{2}C Example 4: Slave Mode Receive

```
; I2C configured as slave with address 1ah

; Setup for Slave Mode Receive

  move I2CSLA, #01ah     ; I2C Slave Address = 01ah
  move I2CCN, #0001h     ; I2CEN = 1, I2CMST = 0, I2CMODE = 0, I2CSTART = 0
  call wait_start        ; Polling routine to wait for I2CSTART to be set,
                         ; indicating a received START

;; Check for address match


  move ACC, I2CST
  and  #0020h            ; Check for Address Match flag set
  cmp  #0020h
  jump ne, no_match      ; Not an address match

  move ACC, I2CCN        ; Check transfer mode is set
  and  #004h
  cmp  #004h
  jump ne, not_sl_xmit   ; If transfer mode is low, not a slave transmit, else:

  move I2CBUF, #0aah     ; Data byte to be transmitted
  call wait_xmit         ; Poll for transmit done

;; Verify ACK received from master

  move ACC, I2CST
  and  #080h
  cmp  #000h
  jump ne, FAIL          ; If NACK bit set, handle retransmission, else continue
```

SECTION 23: SUPPLY VOLTAGE MONITOR AND POWER CONTROL (SPECIFIC TO MAXQ2010)

The MAXQ2010 provides a number of features to allow monitoring and control of its on-board power supplies. The supply voltage monitor register (SVM) monitors the DVDD power supply and can alert the processor through an interrupt if DVDD falls below a programmable threshold. Other control bits allow the processor to reduce power consumption and configure various aspects of clock operation.

The MAXQ2010 provides the following power monitoring and control features:

- SVM compares DVDD against a programmable threshold from approximately 2.7V to 3.5V.
- Optional SVM register interrupt can be triggered when DVDD drops below the programmed threshold.
- SVM interrupt can be used to trigger switchback or exit from stop mode.
- Internal 1.8V regulator can optionally be disabled in stop mode to conserve power.
- Brownout detection can optionally be disabled in stop mode to conserve power.
- Selectable quiet mode and noise immune mode for the 32kHz internal oscillator.

23.1 SVM and Power Control Register Descriptions

The following peripheral registers are used to control the supply voltage monitoring and power control functions. Addresses for all registers are given as "Mx[yy]," where x is the module number (from 0 to 15 decimal) and yy is the register index (from 00h to 1Fh hexadecimal). Fields in the bit definition tables are defined as follows:

- **Name:** Symbolic names of bits or bit fields in this register.
- **Reset:** The value of each bit in this register following a standard reset. If this field reads "unchanged," the given bit is unaffected by standard reset. If this field reads "s," the given bit does not have a fixed 0 or 1 reset value because its value is determined by another internal state or external condition.
- **POR:** If present this field defines the value of each bit in this register following a power-on reset (as opposed to a standard reset). Some bits are unaffected by standard resets and are set/cleared by POR only.
- **Access:** Bits can be read-only (r) or read/write (rw). Any special restrictions or conditions that could apply when reading or writing this bit are detailed in the bit description.

23.1.1 Power Control Register (PWCN, M0[0Fh])

Bit #	15	14	13	12	11	10	9	8
Name	FREQMD	—	—	—	—	—	32KMD1	32KMD0
Reset	0	0	0	0	0	0	0	0
Access	rw	r	r	r	r	r	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	BOD	REGEN	32KBYP	HFXD	32KRDY	X32D	FLOCK	FLLEN
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw[1]	rw[2]	r	rw[1]	r	rw

Note 1: *X32D and 32KBYP can only be written when the RTC is not running (RTCE = 0).*

Note 2: *HFXD can only be written to 1 when both FLLSL = 1 and FLLMD = 1 (running from FLL).*

Bit 15: Operating Frequency Mode (FREQMD). This bit is used to adjust the operating mode of the MAXQ2010 to provide optimal current consumption based on operating frequency. Typically, when running at a frequency of 3MHz or higher, this bit should be set to 0 to optimize current consumption. When running at a frequency under 3MHz, this bit should typically be set to 1. Refer to the "V_{DD} Supply Current vs. Clock Frequency" graph in the IC data sheet for more details.

Bits 14:10: Reserved

Bits 9:8: 32kHz Oscillator Mode (32KMD[1:0]). These two bits determine the 32kHz oscillator operating mode as follows:

- 32KMD[1:0] = **00b**: Always operate in noise immune mode.
- 32KMD[1:0] = **01b**: Always operate in quiet mode.
- 32KMD[1:0] = **10b**: Operate in noise immune mode normally and switch to quiet mode on stop mode entry. Wait for the 32kHz oscillator to warm up before exiting stop mode. **(Note: This setting should not be used on devices of rev B3 or earlier; refer to device errata for more details.)**
- 32KMD[1:0] = **11b**: Operate in noise immune mode normally and switch to quiet mode on stop mode entry. When exiting stop mode, do not wait for the 32kHz oscillator to warm up.

Changing the value of these bits when the 32kHz clock is enabled (X32D = 0) resets the 32KRDY bit to 0 if the new setting requires the 32kHz oscillator to warm up.

Bit 7: Brownout-Detection Disable (BOD). This bit controls whether DVDD brownout detection is enabled during stop mode only. (Brownout detection is always enabled outside of stop mode.) Note that this setting is independent from the SVM settings.

0 = Brownout detection is enabled during stop mode.

1 = If REGEN = 0, brownout detection is disabled during stop mode to conserve power. If REGEN = 1, brownout detection is still enabled during stop mode, regardless of the BOD bit setting.

Bit 6: Regulator Enable (REGEN). This bit controls whether the internal 1.8V regulator is enabled during stop mode only. (The internal regulator is always enabled outside of stop mode.)

0 = The internal regulator is disabled during stop mode to conserve power.

1 = The internal regulator is enabled during stop mode.

Bit 5: 32kHz Bypass Enable (32KBYP). Setting this bit to 1 disables the 32kHz crystal oscillator, allowing a 32kHz external clock to be provided at 32KIN. Clearing this bit to 0 enables the 32kHz crystal oscillator, allowing it to run normally. This bit has no effect when X32D = 1.

Bit 4: High-Frequency Crystal Oscillator Disable (HFXD). Setting this bit to 1 disables the high-frequency crystal oscillator on the MAXQ2010. However, a high-frequency external clock can still be provided at HFXIN. Clearing this bit to 0 enables the high-frequency crystal oscillator.

Bit 3: 32kHz Clock Ready (32KRDY). This read-only bit indicates the current status of the 32kHz clock.

0 = The 32kHz clock is either disabled (X32D = 1) or is in the process of warming up.

1 = The 32kHz clock is ready for use.

Note: Software should check that 32KRDY = 1 before enabling any peripherals (RTC, LCD) that use the 32kHz clock as a time base. Failure to do so can compromise timing accuracy.

Bit 2: 32kHz Clock Disable (X32D). Setting this bit to 1 completely disables the 32kHz clock. When X32D = 1, the 32kHz crystal oscillator is shut down, and no external 32kHz clock input is accepted. Clearing this bit to 0 allows the 32kHz clock to operate normally, with its source (crystal oscillator or external clock) selected by the 32KBYP bit.

Note: If X32D is set to 1 and the RTC is in use, ACS must be set to 1 for proper operation. Similarly, if X32D is set to 1 and the LCD is in use, LCCS must be set to 1 for proper operation. However, even with these settings, if X32D = 1, both the LCD and the RTC are automatically suspended upon entry to stop mode.

Bit 1: FLL Locked (FLOCK). This read-only bit indicates the current status of the FLL.

0 = The FLL is disabled or is still in the process of warming up.

1 = The FLL has locked to the 32kHz input and is ready for use.

Bit 0: Frequency-Lock Loop Enable (FLLEN). Setting this bit to 1 enables the FLL (if it is not already running) and causes it to lock to the 32kHz input. Clearing this bit to 0 disables the FLL unless it is currently providing the system clock.

23.1.2 Supply Voltage Monitor Register (SVM, M1[0Dh])

Bit #	15	14	13	12	11	10	9	8
Name	—	—	—	—	SVTH3	SVTH2	SVTH1	SVTH0
Reset	0	0	0	0	0	1	1	1
Access	r	r	r	r	rw*	rw*	rw*	rw*

Bit #	7	6	5	4	3	2	1	0
Name	—	—	—	SVMSTOP	SVMI	SVMIE	SVMRDY	SVMEN
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	rw	rw	rw	r	rw

**Bits 15:12, 7:5: Reserved*

**SVTH[3:0] can only be written when the SVM is not running (SVMEN = 0).*

Bits 15:12, 7:5: Reserved

Bits 11:8: Supply Voltage Monitor Threshold (SVTH[3:0]). These bits select the programmable DVDD threshold setting for the SVM. The level can be adjusted from approximately 2.7V to 3.5V in steps of 0.1V, and is given by:

SVM threshold = 2.0V + (SVTH[3:0] x 0.1V)

Note: The minimum allowed value for DVDD is approximately 2.7V (refer to the IC data sheet). Therefore, the SVM threshold must be set to 2.7V or higher for the SVM to function in a useful manner. If the SVM threshold is set lower than this, the MAXQ2010 resets (typically at 2.6V) before the SVM is triggered.

The default setting (SVTH[3:0] = 0111b) is 2.7V. Note that these bits can only be changed when SVMEN = 0.

Bit 4: Supply Voltage Monitor Stop-Mode Enable (SVMSTOP). This bit controls the operation of the SVM in stop mode.

0 = The SVM is disabled during stop mode.

1 = The SVM is enabled during stop mode (if SVMEN = 1).

Bit 3: Supply Voltage Monitor Interrupt Flag (SVMI). This bit is set to 1 by hardware when the DVDD supply is below the threshold voltage set by SVTH[3:0]. If SVMIE = 1, setting this bit to 1 by either hardware or software triggers an interrupt. This bit must be cleared by software, but if DVDD is still below the threshold, the bit immediately is set by hardware again.

Bit 2: Supply Voltage Monitor Interrupt Enable (SVMIE). Setting this bit to 1 allows an interrupt to be generated (if not otherwise masked) when SVMI is set to 1. Clearing this bit to 0 disables the SVM interrupt.

Bit 1: Supply Voltage Monitor Ready (SVMRDY). This read-only status bit indicates whether the SVM is ready for use.

0 = The SVM is disabled (SVMEN = 0), stop mode was entered with SVMSTOP = 0, or the SVM is in the process of warming up.

1 = The SVM is enabled and ready for use.

Bit 0: Supply Voltage Monitor Enable (SVMEN). Setting this bit to 1 enables the SVM and begins monitoring DVDD against the programmed (SVTH[3:0]) threshold. Clearing this bit to 0 disables the SVM.

SECTION 24: UTILITY ROM (SPECIFIC TO MAXQ2010)

24.1 Overview

The MAXQ2010 utility ROM includes routines that provide the following functions to application software:

- In-application programming routines for flash memory (program, erase, mass erase)
- Single word/byte copy and buffer copy routines for use with lookup tables
- Entry into stop mode

To provide backwards compatibility among different versions of the utility ROM, a function address table is included that contains the entry points for all user-callable functions. With this table, user code can determine the entry point for a given function as follows:

1. Read the location of the function address table from address 0800Dh in the utility ROM.
2. The entry points for each function listed below are contained in the function address table, one word per function, in the order given by their function numbers.

For example, the entry point for the **UROM_flashEraseAll** function can be determined by the following procedure.

1. functionTable = dataMemory[0800Dh]
2. flashWriteEntry = dataMemory[functionTable + 2]

It is also possible to call utility ROM functions directly, using the entry points given above. Standard include files are provided for this purpose with the MAXQ development toolset; also see Appendix 1. This method calls functions more quickly, but the application may need to be recompiled in order to run properly with a different version of the utility ROM.

Table 24-1. Functions for MAXQ2010 Utility ROM Version 1.00

INDEX	FUNCTION NAME	ENTRY POINT	SUMMARY
0	UROM_flashWrite	83CEh	Programs a single word of flash memory.
1	UROM_flashErasePage	83F1h	Erases (programs to FFFFh) a 512-word sector of flash memory.
2	UROM_flashEraseAll	8407h	Erases (programs to FFFFh) all flash memory.
3	UROM_moveDP0	8416h	Reads a byte/word at DP[0].
4	UROM_moveDP0inc	8419h	Reads a byte/word at DP[0], then increments DP[0].
5	UROM_moveDP0dec	841Ch	Reads a byte/word at DP[0], then decrements DP[0].
6	UROM_moveDP1	841Fh	Reads a byte/word at DP[1].
7	UROM_moveDP1inc	8422h	Reads a byte/word at DP[1], then increments DP[0].
8	UROM_moveDP1dec	8425h	Reads a byte/word at DP[1], then decrements DP[0].
9	UROM_moveBP	8428h	Reads a byte/word at BP[OFFS].
10	UROM_moveBPinc	842Bh	Reads a byte/word at BP[OFFS], then increments OFFS.
11	UROM_moveBPdec	842Eh	Reads a byte/word at BP[OFFS], then decrements OFFS.
12	UROM_copyBuffer	8431h	Copies LC[0] values (up to 255) from DP[0] to BP[OFFS].
13	UROM_stopMode	8437h	Enters stop mode.

24.2 In-Application Programming Functions

24.2.1 UROM_flashWrite

Function:	UROM_flashWrite
Summary:	Programs a single word of flash memory.
Inputs:	A[0]: Word address in program flash memory to write to. A[1]: Word value to write to flash memory.
Outputs:	Carry: Set on error and cleared on success.
Destroys:	PSF, LC[1]

Notes:

- This function uses one stack level to save and restore values.
- If the watchdog reset function is active, it should be disabled before calling this function.
- If the flash location has already been programmed to a non-FFFF value, this function returns with an error (Carry set). In order to reprogram a flash location, it must first be erased by calling **UROM_flashErasePage** or **UROM_flashEraseAll**.

24.2.2 UROM_flashErasePage

Function:	UROM_flashErasePage
Summary:	Erases (programs to 0FFFFh) a 512-word page of flash memory.
Inputs:	A[0]: Word address located in the page to be erased. (The page number is the high 9 bits of A[0].)
Outputs:	Carry: Set on error and cleared on success.
Destroys:	LC[1], A[0]

Notes:

- If the watchdog reset function is active, it should be disabled before calling this function.
- When calling this function from flash, care should be taken that the return address is not in the page which is being erased.

24.2.3 UROM_flashEraseAll

Function:	UROM_flashEraseAll
Summary:	Erases (programs to 0FFFFh) all locations in flash memory.
Inputs:	None.
Outputs:	Carry: Set on error and cleared on success.
Destroys:	LC[1], A[0]

Notes:

- If the watchdog reset function is active, it should be disabled before calling this function.
- This function can only be called by code running from the RAM. Attempting to call this function while running from the flash results in an error.

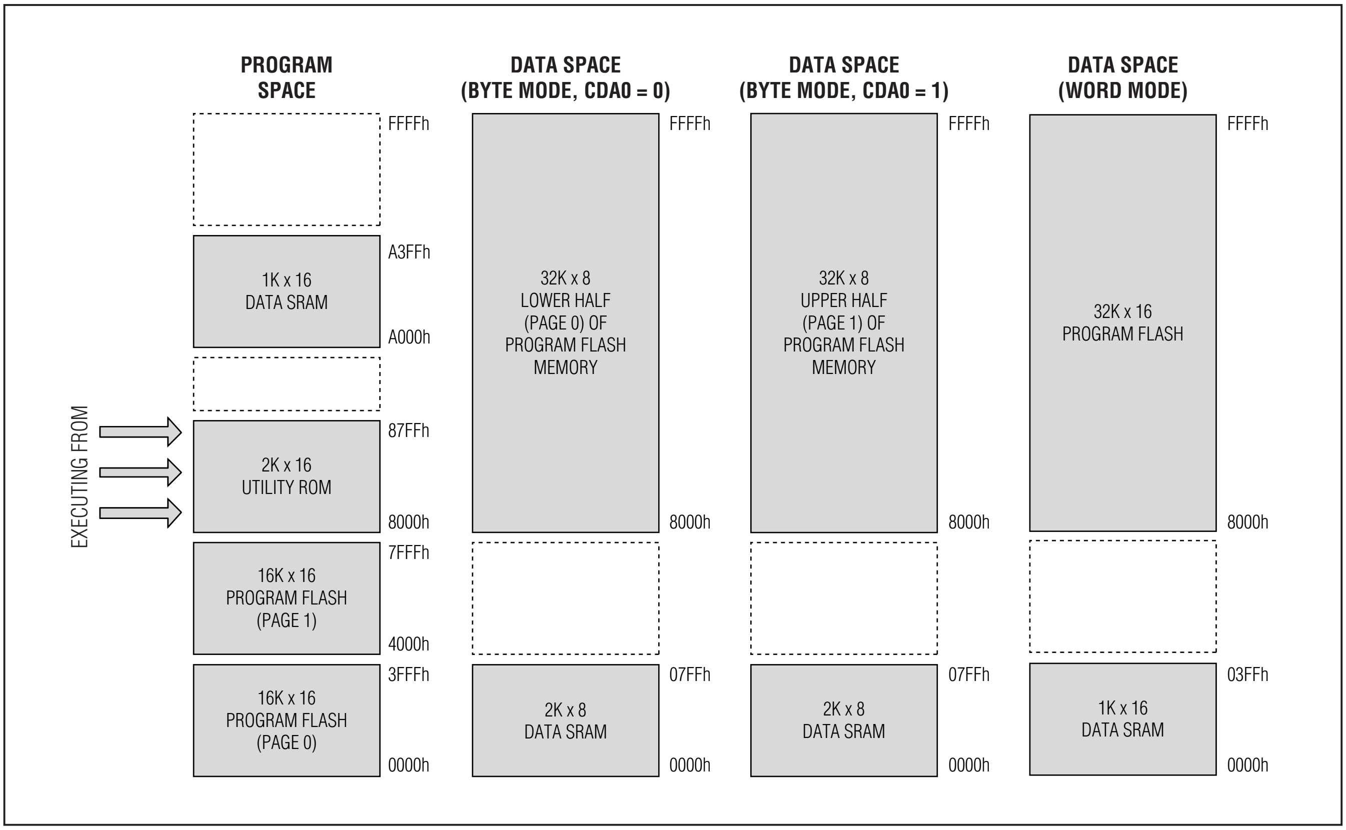

PROGRAM SPACE
DATA SPACE (BYTE MODE, CDA0 = 0)
DATA SPACE (BYTE MODE, CDA0 = 1)
DATA SPACE (WORD MODE)
FFFFh
A3FFh
1K x 16 DATA SRAM
A000h
87FFh
2K x 16 UTILITY ROM
8000h
7FFFh
16K x 16 PROGRAM FLASH (PAGE 1)
4000h
3FFFh
16K x 16 PROGRAM FLASH (PAGE 0)
0000h
EXECUTING FROM
32K x 8 LOWER HALF (PAGE 0) OF PROGRAM FLASH MEMORY
32K x 8 UPPER HALF (PAGE 1) OF PROGRAM FLASH MEMORY
32K x 16 PROGRAM FLASH
07FFh
2K x 8 DATA SRAM
03FFh
1K x 16 DATA SRAM

Figure 24-1. Memory Map When Executing from Utility ROM

24.3 Data Transfer Functions

The following utility ROM functions are used to transfer data from one memory segment to another. They fall into two categories:

- The **UROM_moveDP0<inc/dec>**, **UROM_moveDP1<inc/dec>**, and **UROM_moveBP<inc/dec**> functions are used to read data that exists in the currently executing segment of program memory. For example, code running from the program flash can use these functions to read data (such as constant string or array data or lookup tables) that is also stored in the program flash. These functions can also be used by code running from the RAM to read data stored in the RAM.
- The UROM_copyBuffer function is used to copy a block of data from the program flash or data RAM to a specified location in the data RAM. This is often used to copy code into the data RAM before executing it.

Since these functions are executed from the utility ROM, addresses must be specified correctly to point to memory spaces as seen when executing from the Utility ROM. This is illustrated in Figure 24-1.

For example, data located at word address 0100h in the program flash must be accessed at word address 8100h (or byte address 8200h) when using any of the functions listed in the following sections.

24.3.1 UROM_moveDP0

Function:	UROM_moveDP0
Summary:	Reads the byte/word value pointed to by DP[0].
Inputs:	DP[0]: Address to read from.
Outputs:	GR: Data byte/word read.
Destroys:	None.

Notes:

- Before calling this function, DPC should be set appropriately to configure DP[0] for byte or word mode.
- The address passed to this function should be based on the data memory mapping for the utility ROM, as shown in Figure 24-1. When a byte mode address is used, CDA0 must be set appropriately to access either the upper or lower half of program flash/ROM memory.
- This function automatically refreshes the data pointer before reading the byte/word value.

24.3.2 UROM_moveDP0inc

Function:	UROM_moveDP0inc
Summary:	Reads the byte/word value pointed to by DP[0], then increments DP[0].
Inputs:	DP[0]: Address to read from.
Outputs:	GR: Data byte/word read. DP[0] is incremented.
Destroys:	None.

Notes:

- Before calling this function, DPC should be set appropriately to configure DP[0] for byte or word mode.
- The address passed to this function should be based on the data memory mapping for the utility ROM, as shown in Figure 24-1. When a byte mode address is used, CDA0 must be set appropriately to access either the upper or lower half of program flash/ROM memory.
- This function automatically refreshes the data pointer before reading the byte/word value.

24.3.3 UROM_moveDP0dec

Function:	UROM_moveDP0dec
Summary:	Reads the byte/word value pointed to by DP[0], then decrements DP[0].
Inputs:	DP[0]: Address to read from.
Outputs:	GR: Data byte/word read. DP[0] is decremented.
Destroys:	None.

Notes:

- Before calling this function, DPC should be set appropriately to configure DP[0] for byte or word mode.
- The address passed to this function should be based on the data memory mapping for the utility ROM, as shown in Figure 24-1. When a byte mode address is used, CDA0 must be set appropriately to access either the upper or lower half of program flash/ROM memory.
- This function automatically refreshes the data pointer before reading the byte/word value.

24.3.4 UROM_moveDP1

Function:	UROM_moveDP1
Summary:	Reads the byte/word value pointed to by DP[1].
Inputs:	DP[1]: Address to read from.
Outputs:	GR: Data byte/word read.
Destroys:	None.

Notes:

- Before calling this function, DPC should be set appropriately to configure DP[1] for byte or word mode.
- The address passed to this function should be based on the data memory mapping for the utility ROM, as shown in Figure 24-1. When a byte mode address is used, CDA0 must be set appropriately to access either the upper or lower half of program flash/ROM memory.
- This function automatically refreshes the data pointer before reading the byte/word value.

24.3.5 UROM_moveDP1inc

Function:	UROM_moveDP1inc
Summary:	Reads the byte/word value pointed to by DP[1], then increments DP[1].
Inputs:	DP[1]: Address to read from.
Outputs:	GR: Data byte/word read. DP[1] is incremented.
Destroys:	None.

Notes:

- Before calling this function, DPC should be set appropriately to configure DP[1] for byte or word mode.
- The address passed to this function should be based on the data memory mapping for the utility ROM, as shown in Figure 24-1. When a byte mode address is used, CDA0 must be set appropriately to access either the upper or lower half of program flash/ROM memory.
- This function automatically refreshes the data pointer before reading the byte/word value.

24.3.6 UROM_moveDP1dec

Function:	UROM_moveDP1dec
Summary:	Reads the byte/word value pointed to by DP[1], then decrements DP[1].
Inputs:	DP[1]: Address to read from.
Outputs:	GR: Data byte/word read. DP[1] is decremented.
Destroys:	None.

Notes:

- Before calling this function, DPC should be set appropriately to configure DP[1] for byte or word mode.
- The address passed to this function should be based on the data memory mapping for the utility ROM, as shown in Figure 24-1. When a byte mode address is used, CDA0 must be set appropriately to access either the upper or lower half of program flash/ROM memory.
- This function automatically refreshes the data pointer before reading the byte/word value.

24.3.7 UROM_moveBP

Function:	UROM_moveBP
Summary:	Reads the byte/word value pointed to by BP[OFFS].
Inputs:	BP[OFFS]: Address to read from.
Outputs:	GR: Data byte/word read.
Destroys:	None.

Notes:

- Before calling this function, DPC should be set appropriately to configure BP[OFFS] for byte or word mode.
- The address passed to this function should be based on the data memory mapping for the utility ROM, as shown in Figure 24-1. When a byte mode address is used, CDA0 must be set appropriately to access either the upper or lower half of program flash/ROM memory.
- This function automatically refreshes the data pointer before reading the byte/word value.

24.3.8 UROM_moveBPinc

Function:	UROM_moveBPinc
Summary:	Reads the byte/word value pointed to by BP[OFFS], then increments OFFS.
Inputs:	DP[1]: Address to read from.
Outputs:	GR: Data byte/word read. OFFS is incremented.
Destroys:	None.

Notes:

- Before calling this function, DPC should be set appropriately to configure BP[OFFS] for byte or word mode.
- The address passed to this function should be based on the data memory mapping for the utility ROM, as shown in Figure 24-1. When a byte mode address is used, CDA0 must be set appropriately to access either the upper or lower half of program flash/ROM memory.
- This function automatically refreshed the data pointer before reading the byte/word value.

24.3.9 UROM_moveBPdec

Function:	UROM_moveBPdec
Summary:	Reads the byte/word value pointed to by BP[OFFS], then decrements OFFS.
Inputs:	DP[1]: Address to read from.
Outputs:	GR: Data byte/word read. OFFS is decremented.
Destroys:	None.

Notes:

- Before calling this function, DPC should be set appropriately to configure BP[OFFS] for byte or word mode.
- The address passed to this function should be based on the data memory mapping for the utility ROM, as shown in Figure 24-1. When a byte mode address is used, CDA0 must be set appropriately to access either the upper or lower half of program flash/ROM memory.
- This function automatically refreshes the data pointer before reading the byte/word value.

24.3.10 UROM_copyBuffer

Function:	UROM_copyBuffer
Summary:	Copies LC[0] bytes/words (up to 256) from DP[0] to BP[OFFS].
Inputs:	DP[0]: Address to copy from. BP[OFFS]: Address to copy to. LC[0]: Number of bytes or words to copy.
Outputs:	OFFS is incremented by LC[0]. DP[0] is incremented by LC[0].
Destroys:	LC[0]

Notes:

- This function can be used to copy from program flash to data RAM, or from one part of data RAM to another. It cannot be used to copy data into flash memory, however; the **UROM_writeFlash** function should be used for this purpose.
- Before calling this function, DPC should be set appropriately to configure DP[0] and BP[OFFS] for byte or word mode. Both DP[0] and BP[OFFS] should be configured to the same mode (byte or word) for correct buffer copying.
- The addresses passed to this function should be based on the data memory mapping for the utility ROM, as shown in Figure 24-1. When a byte mode address is used, CDA0 must be set appropriately to access either the upper or lower half of program flash/ROM memory.
- This function automatically refreshes the data pointers before reading the byte/word values.

24.4 Utility ROM Examples

24.4.1 Utility ROM Example 1: Reading Constant Word Data from Flash

```
   move    DPC, #1Ch                ; Set all pointers to word mode

   move    DP[0], #(table + 8000h)  ; Point to address of data as viewed
                                    ;  in the Utility ROM memory map

   lcall   #UROM_moveDP0inc
   move    A[0], GR                 ; A[0] = 1111h
   lcall   #UROM_moveDP0inc
   move    A[1], GR                 ; A[1] = 2222h
   lcall   #UROM_moveDP0inc
   move    A[2], GR                 ; A[2] = 3333h
   lcall   #UROM_moveDP0inc
   move    A[3], GR                 ; A[3] = 4444h

   sjump $

org 0100h

table:
   dw 1111h, 2222h, 3333h, 4444h
```

24.4.2 Utility ROM Example 2: Reading Constant Byte Data from Flash (Indirect Function Call)

```
   move  DPC, #1Ch                   ; Set all pointers to word mode
   move  DP[0], #800Dh               ; Fetch location of function table from Utility ROM
   move  BP, @DP[0]                  ; Set base pointer to function table location
   move  Offs, #INDX_moveDP0inc      ; Set offset to moveDP0inc entry in table
   move  A[7], @BP[Offs]             ; Get address of moveDP0inc function

   move  DPC, #00h                   ; Set all pointers to byte mode

   move  DP[0], #((table * 2) + 8000h); Point to address of data as viewed
                                     ;   in the Utility ROM memory map
                                     ;   and convert to byte mode pointer
   lcall A[7]                        ; moveDP0inc
   move  A[0], GR                    ; A[0] = 34h
   lcall A[7]                        ; moveDP0inc
   move  A[1], GR                    ; A[1] = 12h
   lcall A[7]                        ; moveDP0inc
   move  A[2], GR                    ; A[2] = 78h
   lcall A[7]                        ; moveDP0inc
   move  A[3], GR                    ; A[3] = 56h

   sjump $

org 0100h

table:
   dw    1234h, 5678h
```

APPENDIX 1: SAMPLE MAXQ2010 DEVICE INCLUDE FILE FOR MAX-IDE

```
#define PO0       M0[00h]     ; Port 0 Output
#define PO1       M0[01h]     ; Port 1 Output
#define PO2       M0[02h]     ; Port 2 Output
#define PO3       M0[03h]     ; Port 3 Output
#define EIF0      M0[04h]     ; External Interrupt Flag 0
#define EIE0      M0[05h]     ; External Interrupt Enable 0
;                 M0[06h]
;                 M0[07h]
#define PI0       M0[08h]     ; Port 0 Input
#define PI1       M0[09h]     ; Port 1 Input
#define PI2       M0[0Ah]     ; Port 2 Input
#define PI3       M0[0Bh]     ; Port 3 Input
#define EIES0     M0[0Ch]     ; External Interrupt Edge Select 0
;                 M0[0Dh]
;                 M0[0Eh]
#define PWCN      M0[0Fh]     ; Power Control

#define PD0       M0[10h]     ; Port 0 Direction
#define PD1       M0[11h]     ; Port 1 Direction
#define PD2       M0[12h]     ; Port 2 Direction
#define PD3       M0[13h]     ; Port 3 Direction
;                 M0[14h]
;                 M0[15h]
;                 M0[16h]
;                 M0[17h]
#define RTRM      M0[18h]     ; Real Time Clock Trim
#define RCNT      M0[19h]     ; Real Time Clock Control
#define RTSS      M0[1Ah]     ; Real Time Clock Subsecond Counter
#define RTSH      M0[1Bh]     ; Real Time Clock Second Counter High
#define RTSL      M0[1Ch]     ; Real Time Clock Second Counter Low
#define RSSA      M0[1Dh]     ; Real Time Clock Subsecond Alarm
#define RASH      M0[1Eh]     ; Real Time Clock Time-of-Day Alarm High
#define RASL      M0[1Fh]     ; Real Time Clock Time-of-Day Alarm Low

#define PO4       M1[00h]     ; Port 4 Output
#define PO5       M1[01h]     ; Port 5 Output
#define PO6       M1[02h]     ; Port 6 Output
#define SPIB      M1[03h]     ; SPI Data Buffer
#define EIF1      M1[04h]     ; External Interrupt Flag 1
#define EIE1      M1[05h]     ; External Interrupt Enable 1
#define EIF2      M1[06h]     ; External Interrupt Flag 2
#define EIE2      M1[07h]     ; External Interrupt Enable 2
#define PI4       M1[08h]     ; Port 4 Input
#define PI5       M1[09h]     ; Port 5 Input
#define PI6       M1[0Ah]     ; Port 6 Input
```

```
#define EIES1       M1[0Bh]      ; External Interrupt Edge Select 1
#define EIES2       M1[0Ch]      ; External Interrupt Edge Select 2
#define SVM         M1[0Dh]      ; Supply Voltage Monitor
;                   M1[0Eh]
;                   M1[0Fh]
#define PD4         M1[10h]      ; Port 4 Direction
#define PD5         M1[11h]      ; Port 5 Direction
#define PD6         M1[12h]      ; Port 6 Direction
;                   M1[13h]
;                   M1[14h]
#define SPICN       M1[15h]      ; SPI Control Register
#define SPICF       M1[16h]      ; SPI Configuration Register
#define SPICK       M1[17h]      ; SPI Clock Register

#define MCNT        M2[00h]      ; Multiplier Control Register
#define MA          M2[01h]      ; Multiplier Operand A
#define MB          M2[02h]      ; Multiplier Operand B
#define MC2         M2[03h]      ; Multiplier Accumulate Register 2
#define MC1         M2[04h]      ; Multiplier Accumulate Register 1
#define MC0         M2[05h]      ; Multiplier Accumulate Register 0
#define LCFG        M2[06h]      ; LCD Configuration Register
;                   M2[07h]
#define MC1R        M2[08h]      ; Multiplier Read Register 1
#define MC0R        M2[09h]      ; Multiplier Read Register 0
#define LCRA        M2[0Ah]      ; LCD Adjustment Register
#define LCD0        M2[0Bh]      ; LCD Display Memory 0
#define LCD1        M2[0Ch]      ; LCD Display Memory 1
#define LCD2        M2[0Dh]      ; LCD Display Memory 2
#define LCD3        M2[0Eh]      ; LCD Display Memory 3
#define LCD4        M2[0Fh]      ; LCD Display Memory 4

#define LCD5        M2[10h]      ; LCD Display Memory 5
#define LCD6        M2[11h]      ; LCD Display Memory 6
#define LCD7        M2[12h]      ; LCD Display Memory 7
#define LCD8        M2[13h]      ; LCD Display Memory 8
#define LCD9        M2[14h]      ; LCD Display Memory 9
#define LCD10       M2[15h]      ; LCD Display Memory 10
#define LCD11       M2[16h]      ; LCD Display Memory 11
#define LCD12       M2[17h]      ; LCD Display Memory 12
#define LCD13       M2[18h]      ; LCD Display Memory 13
#define LCD14       M2[19h]      ; LCD Display Memory 14
#define LCD15       M2[1Ah]      ; LCD Display Memory 15
#define LCD16       M2[1Bh]      ; LCD Display Memory 16
#define LCD17       M2[1Ch]      ; LCD Display Memory 17
#define LCD18       M2[1Dh]      ; LCD Display Memory 18
#define LCD19       M2[1Eh]      ; LCD Display Memory 19
#define LCD20       M2[1Fh]      ; LCD Display Memory 20
```

```
#define I2CBUF     M3[00h]      ; I2C Data Buffer
#define I2CST      M3[01h]      ; I2C Status
#define I2CIE      M3[02h]      ; I2C Interrupt Enable
;                  M3[03h]
#define SCON0      M3[04h]      ; Serial Port 0 Control
#define SBUF0      M3[05h]      ; Serial Port 0 Buffer
#define SCON1      M3[06h]      ; Serial Port 1 Control
#define SBUF1      M3[07h]      ; Serial Port 1 Buffer
#define SMD0       M3[08h]      ; Serial Port 0 Mode
#define PR0        M3[09h]      ; Serial Port 0 Phase Register
#define SMD1       M3[0Ah]      ; Serial Port 1 Mode
#define PR1        M3[0Bh]      ; Serial Port 1 Phase Register
#define I2CCN      M3[0Ch]      ; I2C Control
#define I2CCK      M3[0Dh]      ; I2C Clock Control
#define I2CTO      M3[0Eh]      ; I2C Timeout
#define I2CSLA     M3[0Fh]      ; I2C Slave Address
#define TB0R       M4[00h]      ; Type B Timer 0 Capture/Reload Value
#define TB0C       M4[01h]      ; Type B Timer 0 Compare Value
#define TB1R       M4[02h]      ; Type B Timer 1 Capture/Reload Value
#define TB1C       M4[03h]      ; Type B Timer 1 Compare Value
#define TB2R       M4[04h]      ; Type B Timer 2 Capture/Reload Value
#define TB2C       M4[05h]      ; Type B Timer 2 Compare Value
#define ADST       M4[06h]      ; ADC Status Register
#define ADADDR     M4[07h]      ; ADC Address Register
#define TB0CN      M4[08h]      ; Type B Timer 0 Control
#define TB0V       M4[09h]      ; Type B Timer 0 Value
#define TB1CN      M4[0Ah]      ; Type B Timer 1 Control
#define TB1V       M4[0Bh]      ; Type B Timer 1 Value
#define TB2CN      M4[0Ch]      ; Type B Timer 2 Control
#define TB2V       M4[0Dh]      ; Type B Timer 2 Value
#define ADCN       M4[0Eh]      ; ADC Control Register
#define ADDATA     M4[0Fh]      ; ADC Data Register

#define NUL        M6[7]        ; Bit bucket

;=====================================================================
;=
;= Utility ROM Function Indexes
;=
;=====================================================================

INDX_flashWrite       equ   0
INDX_flashErasePage   equ   1
INDX_flashEraseAll    equ   2
INDX_moveDP0          equ   3
INDX_moveDP0inc       equ   4
INDX_moveDP0dec       equ   5
INDX_moveDP1          equ   6
```

```
INDX_moveDP1inc       equ   7
INDX_moveDP1dec       equ   8
INDX_moveBP           equ   9
INDX_moveBPinc        equ   10
INDX_moveBPdec        equ   11
INDX_copyBuffer       equ   12
INDX_stopMode         equ   13

;======================================================================
;=
;= Utility ROM Entry Points (MAXQ2010 Loader 1.00 08-09-2006)
;=
;======================================================================

UROM_flashWrite       equ   083CEh
UROM_flashErasePage   equ   083F1h
UROM_flashEraseAll    equ   08407h
UROM_moveDP0          equ   08416h
UROM_moveDP0inc       equ   08419h
UROM_moveDP0dec       equ   0841Ch
UROM_moveDP1          equ   0841Fh
UROM_moveDP1inc       equ   08422h
UROM_moveDP1dec       equ   08425h
UROM_moveBP           equ   08428h
UROM_moveBPinc        equ   0842Bh
UROM_moveBPdec        equ   0842Eh
UROM_copyBuffer       equ   08431h
```

REVISION HISTORY

REVISION NUMBER	REVISION DATE	SECTION NUMBER	DESCRIPTION	PAGES CHANGED
0	6/09	—	Initial release.	—

Maxim cannot assume responsibility for use of any circuitry other than circuitry entirely embodied in a Maxim product. No circuit patent licenses are implied. Maxim reserves the right to change the circuitry and specifications without notice at any time.