

MAXQ7667 USER'S GUIDE

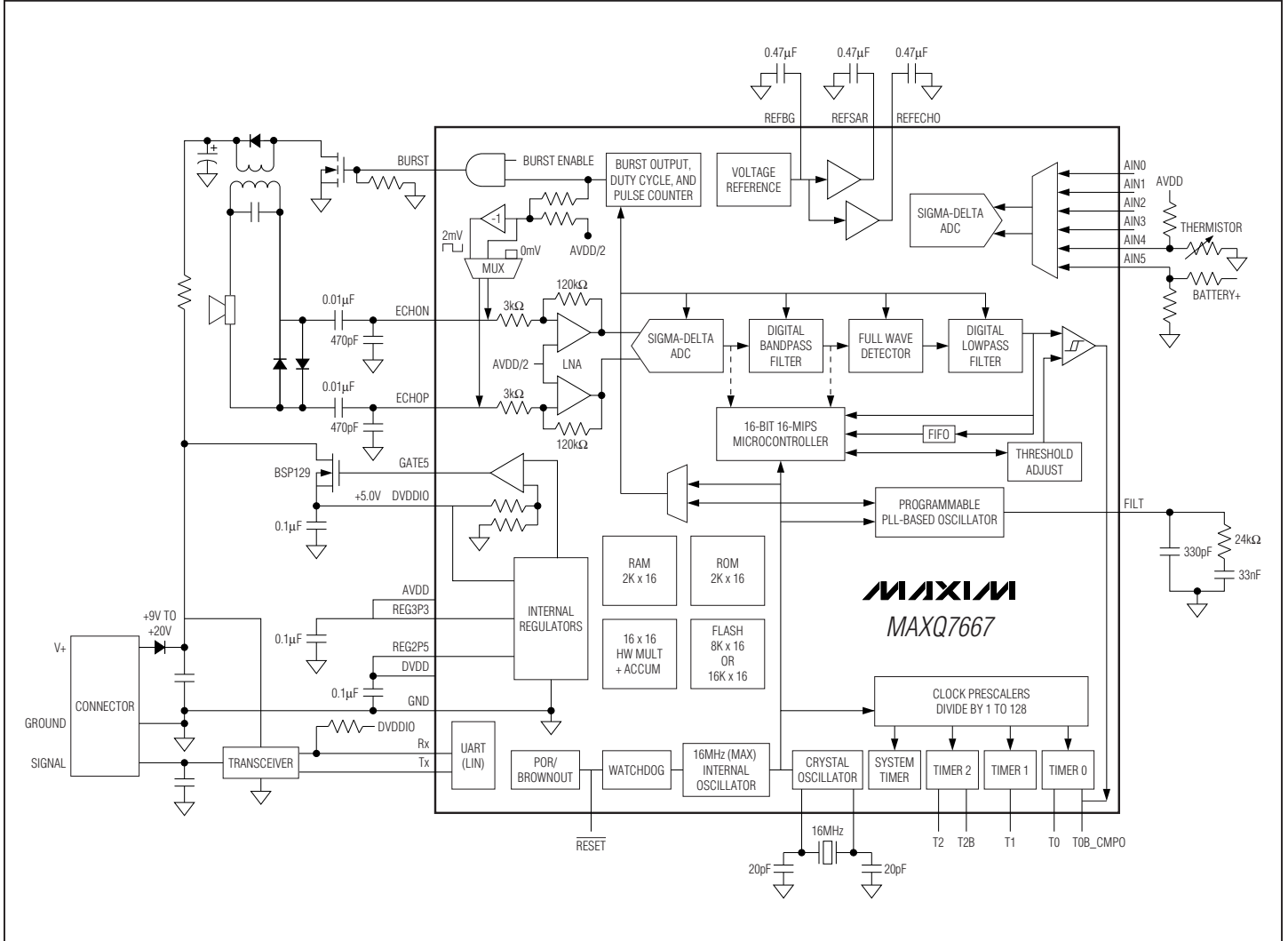


TABLE OF CONTENTS

SECTION 1: Overview	1-1
SECTION 2: Architecture	2-1
SECTION 3: Programming	3-1
SECTION 4: Register Descriptions	4-1
SECTION 5: General-Purpose I/O Module	5-1
SECTION 6: Type 2 Timer/Counter Module	6-1
SECTION 7: Schedule Timer	7-1
SECTION 8: UART and LIN	8-1
SECTION 9: Enhanced Serial Peripheral Interface (SPI) Module	9-1
SECTION 10: Hardware Multiplier Module	10-1
SECTION 11: Test Access Port (TAP)	11-1
SECTION 12: In-Circuit Debug Mode	12-1
SECTION 13: In-System Programming/Bootloader	13-1
SECTION 14: SAR ADC Module	14-1
SECTION 15: Oscillator/Clock Generation	15-1
SECTION 16: Power-Supply/Supervisory Monitoring	16-1
SECTION 17: Ultrasonic Distance Measurement Module— Burst Transmission and Echo Reception	17-1
SECTION 18: Utility ROM	18-1
SECTION 19: Instruction Set Summary	19-1

SECTION 1: OVERVIEW

The MAXQ7667 is a smart data-acquisition system based on the MAXQ[®] microcontroller (μ C) with integrated peripheral functions for ultrasonic, time-of-flight, distance measurement. The MAXQ processor is a high-performance reduced instruction set computing (RISC) core μ C designed for efficient peripheral multitasking applications. The MAXQ core contains a 16-bit Harvard Architecture RISC core that executes instructions in a single clock cycle from instruction fetch to cycle completion. The MAXQ core works without the aid of an instruction prefetch pipeline. This streamlines the entire instruction fetch, decode, and execute task, which dramatically improves code density, memory access benchmarks, and multitasking interrupt-based latency. The MAXQ architecture was designed specifically for analog I/O and peripheral multitasking applications.

As a member of the MAXQ family of 16-bit RISC μ Cs, the MAXQ7667 is ideal for low-cost, low-power embedded applications such as

- Automotive Parking
- Vehicle Security
- Industrial Processing
- Automation
- Handheld Devices

The flexible, modular architecture design used in these μ Cs allows development of targeted products for specific applications with minimal effort.

The MAXQ7667 includes the following general-purpose peripherals: maximum 16MHz (factory default is 13.5MHz) RC oscillator, crystal oscillator support, watchdog timer, schedule timer, three general-purpose timer/counters, two 8-bit GPIO ports, SPI[™] port, JTAG port, LIN-capable UART, 12-bit ADC with five input channels, and a voltage reference. These modules are useful for communication, diagnostics, and miscellaneous support.

Peripherals dedicated to ultrasonic measurement include a burst signal generator and echo signal processing for transducer frequencies from 25kHz to 100kHz. When triggered, the BURST output supplies the specified number of transducer excitation cycles at the specified frequency and duty cycle. Echo signals are received and digitized by a low noise amplifier (LNA) and 16-bit sigma-delta ADC that together provide a variable gain ranging from 38dB to 60dB. Following the ADC is a digital bandpass filter, demodulator, and digital lowpass filter with a 16-bit output. Supporting both the burst generator and echo reception is a programmable, PLL frequency synthesizer that supplies both the burst frequency and the clock for the digital filters. Using the PLL for both burst transmission and echo reception ensures that the bandpass filter always tracks the transducer excitation. Both the digital filtering and the clock synthesis are done without CPU intervention. All the CPU power is available for other tasks.

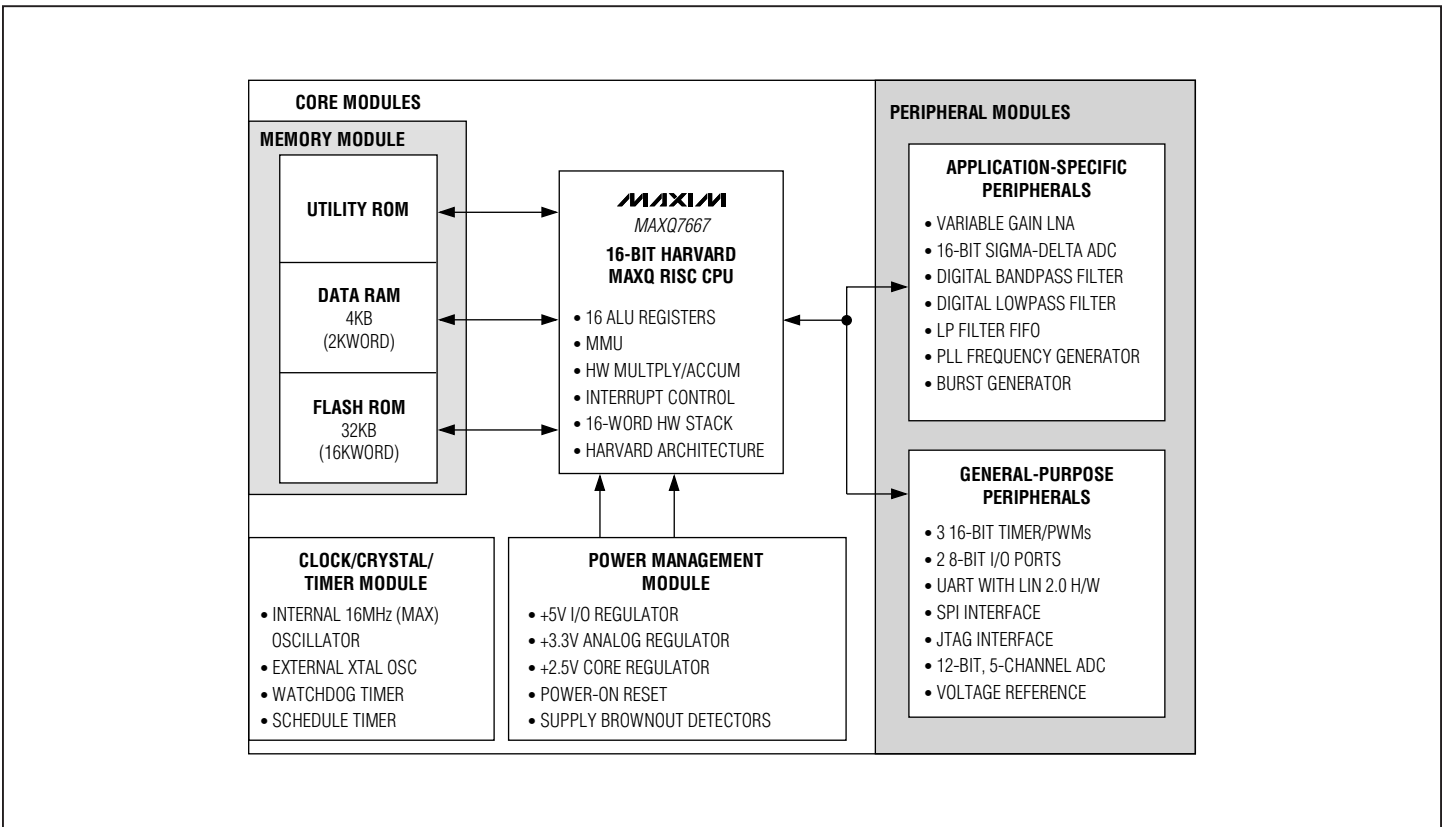
MAXQ7667 Features

- 16-Bit Single-Cycle RISC Core Processor
- Simultaneous Instruction/Data Harvard Memory Architecture
- 32KB Flash Memory
- 4KB Data RAM
- Utility ROM
- Burst-Pulse Generation Using Fractional N Frequency Synthesis
- Low-Noise, Variable Gain, Echo Receive Amplifier
 - 16-Bit Sigma-Delta ADC for Digitizing the Echo
 - Digital Bandpass Filter that Tracks the Burst Frequency
 - Digital Demodulator and Lowpass Filter to Create an Echo Envelope with 16-Bit Resolution
 - FIFO Stores Up to 8 Readings from the Lowpass Filter
 - Digital Comparator Trips When Output of Lowpass Filter Reaches the Programmed Value
- 12-Bit, 5-Channel SAR ADC
- 2.5V Reference

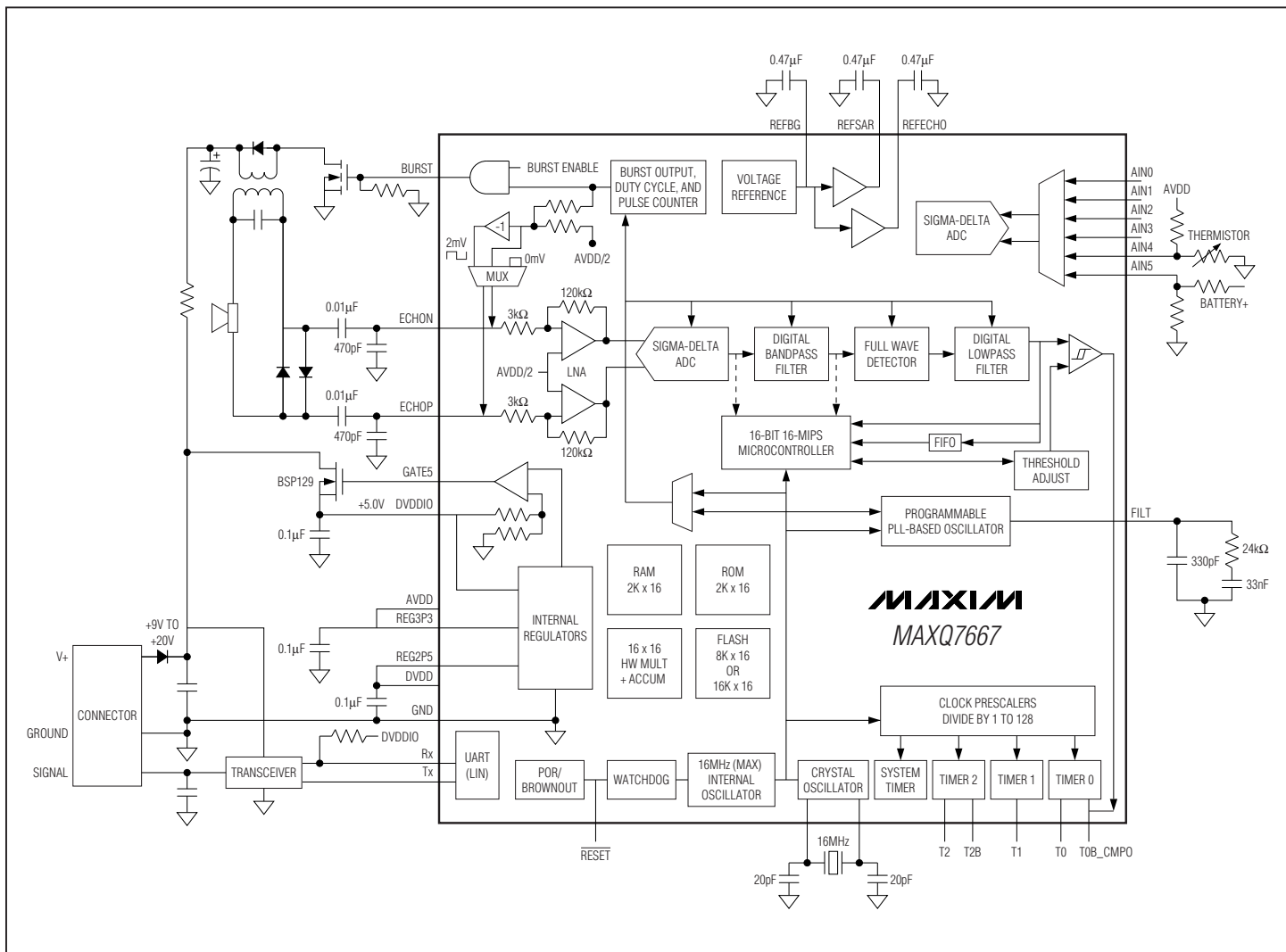
MAXQ is a registered trademark of Maxim Integrated Products, Inc.
SPI is a trademark of Motorola, Inc.

- One Schedule Timer
- Three General-Purpose Timers
- LIN-Compatible UART
- SPI Port
- JTAG Port
- Watchdog Timer
- Voltage Monitors

MAXQ7667 Module Functions



Typical Application Circuit



SECTION 2: ARCHITECTURE

This section contains the following information:

2.1 Overview	2-3
2.1.1 References	2-3
2.1.2 Instruction Set	2-4
2.1.3 Harvard Memory Architecture	2-5
2.1.4 Register Space	2-5
2.2 Architecture	2-6
2.2.1 Instruction Decoding	2-7
2.2.2 Register Space	2-8
2.2.3 Memory Organization	2-10
2.2.3.1 Program Memory	2-10
2.2.3.2 Utility ROM	2-11
2.2.3.3 Data Memory	2-11
2.2.3.4 Stack Memory	2-12
2.2.3.5 Pseudo-Von Neumann Memory Mapping	2-13
2.2.3.6 Pseudo-Von Neumann Memory Access	2-14
2.2.3.7 Data Alignment	2-14
2.2.3.8 Memory Management Unit	2-15
2.2.4 Interrupts	2-18
2.2.4.1 Servicing Interrupts	2-18
2.2.4.2 Interrupt System Operation	2-18
2.2.4.3 Synchronous vs. Asynchronous Interrupt Sources	2-19
2.2.4.4 Interrupt Prioritization by Software	2-19
2.2.4.5 Interrupt Exception Window	2-19
2.2.4.6 MAXQ7667 Interrupt Sources	2-19

LIST OF FIGURES

Figure 2-1. MAXQ7667 Block Diagram	2-4
Figure 2-2. MAXQ7667 Transport-Triggered Architecture	2-6
Figure 2-3. Instruction Word Format	2-7
Figure 2-4. Pseudo-Von Neumann Memory Map (MAXQ7667 Default)	2-13
Figure 2-5. Word Access Mode in MAXQ7667	2-16
Figure 2-6. Byte Access Mode in MAXQ7667	2-17
Figure 2-7. MAXQ7667 Interrupt Source Hierarchy Example	2-20

LIST OF TABLES

Table 2-1. Register-to-Register Transfer Operations	2-8
Table 2-2. MAXQ7667 Register Modules	2-9
Table 2-3. MAXQ7667 Interrupt Sources and Control Bits	2-21

SECTION 2: ARCHITECTURE

2.1 Overview

The MAXQ7667 is a low-power, high-performance, 16-bit RISC microcontroller based on the MAXQ architecture. It includes support for integrated, in-system-programmable flash memory and a wide range of peripherals supporting ultrasonic measurement, schedule timer, general-purpose timer/counters, GPIO, SPI, JTAG port, LIN-capable UART, 12-bit SAR ADC with five input channels, and a voltage reference.

The MAXQ7667 key features include:

- 16-Bit Single-Cycle RISC Core Processor
- Simultaneous instruction/Data Harvard Memory Architecture
- 32KB Flash Memory
- 4KB Data RAM
- Utility ROM
- Burst Generation and Echo Reception for Ultrasonic Measurement
- 12-Bit, 5-Channel SAR ADC
- One Schedule Timer
- Three General-Purpose Timers
- LIN-Compatible UART
- SPI Port
- JTAG Port
- Watchdog Timer
- Voltage Monitors

2.1.1 References

The online MAXQ7667 QuickView page contains information and data sheet links for all parts in the MAXQ7667 family. Errata sheets for the MAXQ products 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 home page at www.maxim-ic.com/MAXQ. For general questions and discussion of the MAXQ platform, visit our discussion board at <http://discuss.dalsemi.com>. Technical Support is also available at www.maxim-ic.com/support.

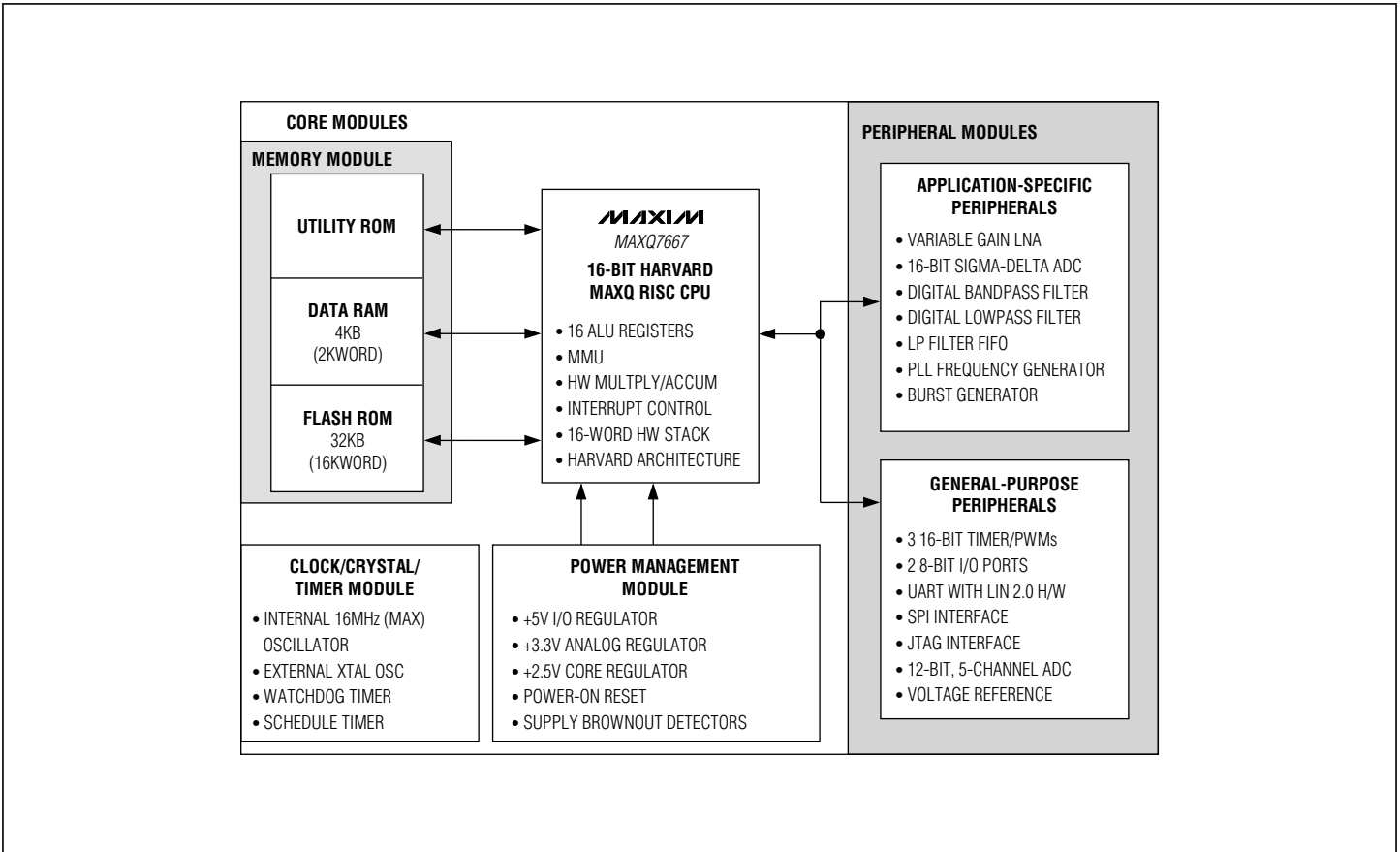


Figure 2-1. MAXQ7667 Block Diagram

2.1.2 Instruction Set

As part of the MAXQ family, the MAXQ7667 uses the standard 16-bit MAXQ20 instruction set, with all instructions a fixed 16 bits in length. A register-based, transport-triggered architecture allows all instructions to be coded as simple transfer operations. All instructions reduce to either writing an immediate value to a destination register or memory location or moving data between registers and/or memory locations.

This simple top-level instruction decoding allows all instructions to be executed in a single cycle. Since all CPU operations are performed on registers only, any new functionality can be added by simply adding new register modules. The simple instruction set also provides maximum flexibility for code optimization by a compiler.

2.1.3 Harvard Memory Architecture

As part of the MAXQ family, the MAXQ7667 core architecture is based on the MAXQ20 design, which implements a 16-bit internal data-bus and ALU. Program memory, data memory, and register space on the MAXQ7667 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 may be either 8 or 16 bits in width. Program memory is 16 bits in width to accommodate the standard MAXQ 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 MAXQ7667 includes a flexible memory management unit (MMU), which allows code to be executed from either the program flash, the utility ROM, or the internal data SRAM. Any of these three memory spaces may also be accessed in data space at any time, with the single restriction that whichever physical memory area is currently being used as program space cannot be read from in data space.

2.1.4 Register Space

Since all functions in the MAXQ family are accessed through registers, common functionality is provided through a common register set. Many of these registers provide the equivalent of higher level op codes by directly accessing the arithmetic logic unit (ALU), the loop counter registers, and the data pointer registers. Others, such as the interrupt registers, provide common control and configuration functions that are equivalent across all MAXQ microcontrollers.

The common register set, also known as the System Registers, includes the following:

- ALU access and control registers, including working accumulator registers and the processor status flags
- Two Data Pointers and a Frame Pointer for data memory access
- Autodecrementing Loop Counters for fast, compact looping
- Instruction Pointer and other branching control access points
- Stack Pointer and an access point to the 16-bit-wide dedicated hardware stack
- Interrupt vector, identification, and masking registers

The MAXQ7667 peripheral register space (modules 0 to 5) contains registers that access the following peripherals:

- Two general-purpose, 8-bit, I/O ports (P0, P1)
- LIN-compatible UART
- Serial peripheral interface (SPI)
- Hardware multiplier
- JTAG debug engine
- Three programmable Type 2 timer/counters
- Analog module
- Schedule timer
- Burst generator
- Echo receiver path

2.2 Architecture

The MAXQ7667 architecture is designed to be modular and expandable. Top-level instruction decoding is extremely simple and based on transfers to and from registers. The registers are organized into functional modules, which are in turn divided into the system register and peripheral register groups. Figure 2-2 illustrates the modular architecture.

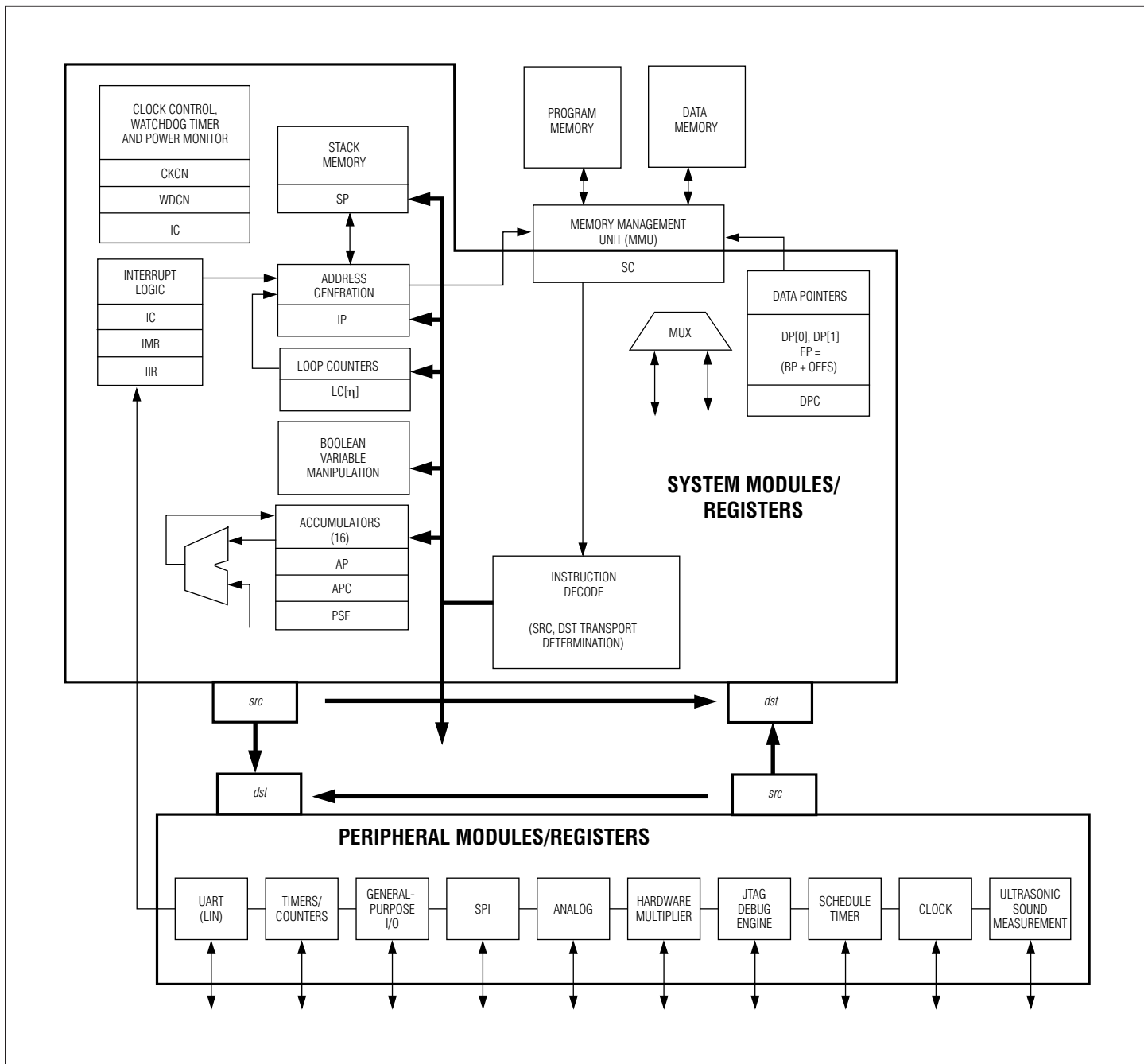


Figure 2-2. MAXQ7667 Transport-Triggered Architecture

Memory access from the MAXQ7667 is based on a Harvard architecture with separate address spaces for program and data memory. The simple instruction set and transport-triggered architecture allow the MAXQ7667 to run in a nonpipelined execution mode where each instruction can be fetched from memory, decoded, and executed in a single clock cycle. Data memory is accessed through one of three data pointer registers. Two of these data pointers, DP[0] and DP[1], are stand-alone 16-bit pointers. The third data pointer, FP, is composed of a 16-bit base pointer (BP) and an 8-bit offset register (OFFS). All three pointers support postincrement/decrement functionality for read operations and preincrement/decrement for write operations. For the Frame Pointer (FP = BP[OFFS]), the increment/decrement operation is executed on the OFFS register and does not affect the base pointer (BP). Stack functionality is provided by dedicated memory with a 16-bit width and depth of 16. An on-chip memory management unit (MMU) is accessible through system registers to allow logical remapping of physical program and data spaces, and thus facilitates in-system programming and fast access to data tables, arrays, and constants physically located in program memory.

2.2.1 Instruction Decoding

Every MAXQ7667 instruction is encoded as a single 16-bit word according to the format in Figure 2-3.

FORMAT	DESTINATION						SOURCE							
f	d	d	d	d	d	d	s	s	s	s	s	s	s	s

Figure 2-3. Instruction Word Format

Bit 15 (f) indicates the format for the source field of the instruction as follows:

- If f equals 0, the instruction is an immediate source instruction, and the source field represents an immediate 8-bit value.
- If f equals 1, the instruction is a register source instruction, and the source field represents the register that the source value will be read from.

Bits 0 to 7 (ssssssss) represent the source for the transfer. Depending on the value of the format field, this can either be an immediate value or a source register. If this field represents a register, the lower four bits contain the module specifier and the upper four bits contain the register index in that module.

Bits 8 to 14 (ddddddd) represent the destination for the transfer. This value always represents a destination register, with the lower four bits containing the module specifier and the upper three bits containing the register subindex within that module.

Since the source field is 8 bits wide and 4 bits are required to specify the module, any one of 16 registers in that module may be specified as a source. However, the destination field has one less bit, which means that only eight registers in a module can be specified as a destination in a single-cycle instruction.

While the asymmetry between source and destination fields of the op code may initially be considered a limitation, this space can be used effectively. Firstly, since read-only registers will never be specified as destinations, they can be placed in the second eight locations in a module to give single-cycle read access. Secondly, there are often critical control or configuration bits associated with system and certain peripheral modules where limited write access is beneficial (e.g., watchdog-timer enable and reset bits). By placing such bits in one of the upper 24 registers of a module, this write protection is added in a way that is virtually transparent to the assembly source code. Anytime that it is necessary to directly select one of the upper 24 registers as a destination, the prefix register PFX is used to supply the extra destination bits. This prefix register write is inserted automatically by the assembler and requires one additional execution cycle.

The MAXQ7667 architecture is transport-triggered. This means that writing to or reading from certain register locations will also cause side effects to occur. These side effects form the basis for the higher level op codes defined by the assembler, such as ADDC, OR, JUMP, and so on. While these op codes are actually implemented as MOVE instructions between certain register locations, the encoding is handled by the assembler and need not be a concern to the programmer. The registers defined in the System Register and Peripheral Register maps operate as described in the documentation; the unused "empty" locations are the ones used for these special cases.

The MAXQ7667 instruction set is designed to be highly orthogonal. All arithmetic and logical operations that use two registers can use any register along with the accumulator. Data can be transferred between any two registers in a single instruction.

2.2.2 Register Space

The MAXQ7667 architecture provides a total of 16 register modules. Each of these modules contains 32 registers. Of these possible 16 register modules, only 13 are used on the MAXQ7667—seven for system registers and six for peripheral registers. The first eight registers in each module may be read from or written to in a single cycle; the second eight registers may be read from in a single cycle and written to in two cycles (by using the prefix register PFX); the last 16 registers may be read or written in two cycles (always requiring use of the prefix register PFX).

Registers may be either 8 or 16 bits in length. Within a register, any number of bits can be implemented; bits not implemented are fixed at zero. Data transfers between registers of different sizes are handled as shown in Table 2-1.

- If the source and destination registers are both 8 bits wide, data is transferred bit to bit accordingly.
- If the source register is 8 bits wide and the destination register is 16 bits wide, the data from the source register is transferred into the lower 8 bits of the destination register. The upper 8 bits of the destination register are set to the current value of the prefix register; this value is normally zero, but it can be set to a different value by the previous instruction if needed. The prefix register reverts back to zero after one cycle, so this must be done by the instruction immediately before the one that will be using the value.
- If the source register is 16 bits wide and the destination register is 8 bits wide, the lower 8 bits of the source are transferred to the destination register.
- If both registers are 16 bits wide, data is copied bit to bit.

Table 2-1. Register-to-Register Transfer Operations

SOURCE REGISTER SIZE (BITS)	DESTINATION REGISTER SIZE (BITS)	PREFIX SET?	DESTINATION SET TO VALUE	
			HIGH 8 BITS	LOW 8 BITS
8	8	—		Source [7:0]
8	16	No	00h	Source [7:0]
8	16	Yes	Prefix [7:0]	Source [7:0]
16	8	—		Source [7:0]
16	16	No	Source [15:8]	Source [7:0]

The above rules apply to all data movements between defined registers. Data transfer to/from undefined register locations has the following behavior:

- If the destination is an undefined register, the MOVE is a dummy operation but may trigger an underlying operation according to the source register (e.g., @DP[n]--).
- If the destination is a defined register and the source is undefined, the source data for the transfer will depend upon the source module width. If the source is from a module containing 8-bit or 8-bit and 16-bit source registers, the source data will be equal to the prefix data concatenated with 00h. If the source is from a module containing only 16-bit source registers, 0000h source data is used for the transfer.

The 16 available register modules are broken up into two different groups. The low six modules (specifiers 0h through 5h) are known as the Peripheral Register modules, while the high 10 modules (specifiers 6h to Fh) are known as the System Register modules. These groupings are descriptive only, as there is no difference between accessing the two register groups from a programming perspective.

The System Registers define basic functionality that remains the same across all products based on the MAXQ architecture. This includes all register locations that are used to implement higher-level op codes as well as the following common system features.

- ALU (MAXQ20: 16 bits) and associated status flags (zero, equals, carry, sign, overflow)
- Eight working accumulator registers (MAXQ20: 16-bit width), along with associated control registers
- Instruction pointer
- Registers for interrupt control, handling, and identification
- Autodecrementing loop counters for fast, compact looping
- Two data pointer registers and a frame pointer for data memory access

The MAXQ7667 peripheral register space (modules 0 to 5) contains registers that access the following peripherals:

- Two general-purpose, 8-bit, I/O ports (P0, P1)
- External interrupts (up to 16)
- Three programmable Type 2 timer/counters
- LIN-compatible UART interface
- SPI
- Analog module (SAR ADC and ultrasonic measurement)
- Hardware multiplier
- JTAG debug engine
- Schedule timer

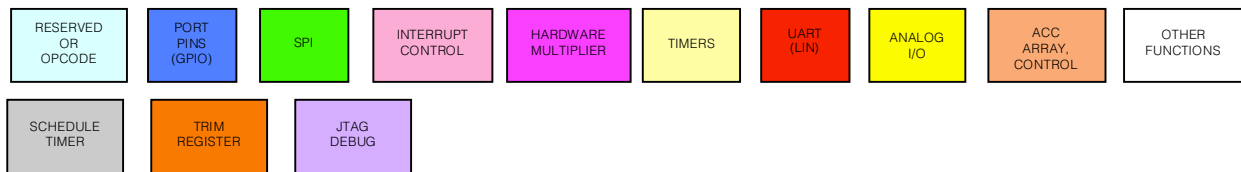
The lower 8 bits of all registers in modules 0 to 5 (as well as the AP module M8) are bit addressable.

Table 2-2. MAXQ7667 Register Modules

REGISTER INDEX	MODULE NAME (BASE SPECIFIER)												
	M0	M1	M2	M3	M4	M5	M8	M9	M11	M12	M13	M14	M15
00h	PO0	MCNT	T2CNA0	T2CNA2		BPH	AP	A[0]	PFX[0]	IP			
01h	PO1	MA	T2H0	T2H2		BTRN	APC	A[1]	PFX[1]		SP		
02h		MB	T2RH0	T2RH2		SARC		A[2]	PFX[2]		IV		
03h	EIF0	MC2	T2CH0	T2CH2		RCVC		A[3]	PFX[3]			OFFS	DP0
04h	EIF1	MC1	T2CNA1			PLLF	PSF	A[4]	PFX[4]			DPC	
05h		MCO	T2H1	CNT1		AIE	IC	A[5]	PFX[5]			GR	
06h		SPIB	T2RH1	SCON		CMPC	IMR	A[6]	PFX[6]		LC0	GRL	
07h		SPICN	T2CH1	SBUF		CMPT		A[7]	PFX[7]		LC1	BP	DP1
08h	PI0	SPICF	T2CNB0	T2CNB2		ASR	SC	A[8]				GRS	
09h	PI1	SPICK	T2V0	T2V2		SARD		A[9]				GRH	
0Ah			T2R0	T2R2		LPFC		A[10]				GRXL	
0Bh	EIE0		T2C0	T2C2		OSCC	IIR	A[11]				FP	
0Ch	EIE1	MC1R	T2CNB1	FSTAT		BPFI		A[12]					
0Dh		MC0R	T2V1	ERRR		BPFO		A[13]					
0Eh		SCNT	T2R1	CHKSUM		LPFD	CKCN	A[14]					
0Fh		STIM	T2C1	ISVEC		LPFF	WDCN	A[15]					
10h	PD0	SALM	T2CFG0	T2CFG2		APE							
11h	PD1	FPCTL	T2CFG1	STA0									
12h				SMD		FGAIN							
13h	EIES0			FCON		B1COEF							
14h	EIES1			CNT0		B2COEF							
15h				CNT2		B3COEF							
16h				IDFB		A2A							

Table 2-2. MAXQ7667 Register Modules (continued)

REGISTER INDEX	MODULE NAME (BASE SPECIFIER)												
	M0	M1	M2	M3	M4	M5	M8	M9	M11	M12	M13	M14	M15
17h		RCTRM†		SADDR		A2B							
18h	PS0		ICDT0	SADEN									
19h	PS1		ICDT1	BT		A2D							
1Ah			ICDC	TMR									
1Bh	PR0		ICDF			A3A							
1Ch	PR1	ID0	ICDB			A3B							
1Dh		ID1	ICDA										
1Eh			ICDD			A3D							
1Fh													



†The RCTRM register is a read/write register, but on power-up flash restores the factory-trimmed voltage. (Contact Maxim for write capability.)

2.2.3 Memory Organization

Beyond the internal register space, memory on the MAXQ7667 microcontroller is organized according to a Harvard architecture, with a separate address space and bus for program memory and data memory. Stack memory is also separate and is accessed through a dedicated register set.

To provide additional memory map flexibility, an MMU allows data memory space to be mapped into a predefined program memory segment, thus affording the possibility of code execution from data memory. Additionally, program memory space can be made accessible as data space, allowing access to constant data stored in program memory. 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.

2.2.3.1 Program Memory

The MAXQ7667 contains up to 16K x 16 (32KB) 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 and can be used for lookup tables and similar functions. 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 by calling the in-application functions provided by the utility ROM. The utility ROM provides routines to carry out the necessary operations (erase, write) on flash memory.

Program memory begins at address 0000h and is contiguous through the internal program memory. The actual size of the on-chip program memory available for user application is product dependent. Given a 16-bit program address bus, the maximum program space is 64KWords. Since the codewords are 16 bits, the program memory is therefore a 64K x 16 linear space.

Program memory is accessed directly by the program fetching unit and is addressed by the Instruction Pointer register. From an implementation perspective, system interrupts and branching instructions simply change the contents of the Instruction Pointer and force the op code fetch from a new program location. The Instruction Pointer is direct read/write accessible by the user software; write access to the Instruction Pointer will force program flow to the new address on the next cycle following the write. The contents of the Instruction Pointer will be incremented by 1 automatically after each fetch operation. The Instruction Pointer defaults to 8000h, which is the start-

ing address of the utility ROM. The default IP setting of 8000h is assigned to allow initial in-system programming to be accomplished with utility ROM code assistance. The utility ROM code interrogates a specific register bit in order to decide whether to execute in-system programming or jump immediately to user code starting at 0000h. The user code reset vector should always be stored in the lowest bytes of the program memory.

2.2.3.2 Utility ROM

A utility ROM (4K x 16) is normally placed in the upper 32KWord program memory space starting at address 8000h. This utility ROM potentially provides the following system utility functions:

- Reset vector
- Bootstrap function for system initialization
- In-application programming
- In-circuit debug

Following each reset, the processor automatically starts execution at address 8000h in the utility ROM, allowing ROM code to perform any necessary system support functions.

After a reset, the MAXQ7667 instruction pointer jumps to the ROM bootloader (0x8000). At this point the password location gets checked for a valid entry. If the password space (0x0010 to 0x001F) in flash is populated by all 0s or 1s (implying that no password has been set), the PWL bit (in SC register) is set to 0, allowing access to all the bootloader functions. Otherwise, the PWL bit gets set to 1, preventing access to the password-protected family of commands (more on this later) and eventually the user must provide the password to clear PWL to access all the bootloader functions.

The processor then looks for a request from the JTAG port. The JTAG port is established as the programming port before the MAXQ7667 is released from reset. While the MAXQ7667 is in reset, the SPE bit is set to 1 via the JTAG/TAP port. If the request is valid (i.e., SPE = 1, PSS = 00), the processor establishes communication between the ROM bootloader and the JTAG port. Otherwise, the UART is monitored for an autobaud character: 0x0D (carriage return). If the autobaud character is detected, the UART is established as the bootloader communication port and the MAXQ7667 responds with 0x3E. 0x3E is the acknowledgement that a loader command has been completed. After this, some or all of the bootloader functions are accessible through the UART, depending on password settings.

The processor jumps to the flash program space 0x0000 and starts executing application code when there is no JTAG request and a valid password is found (PWL = 1). The code execution also jumps to 0x0000 when the autobaud routine does not receive the 0x0D character within the 5-second built-in wait.

It is still possible to load a new program through the UART or the JTAG, after the MAXQ7667 begins executing code in the flash program space. To load code through the JTAG would merely require resetting the device and holding the device in reset while the SPE bit is set to 1 through the JTAG/TAP port, once the reset is released the device executes in the bootloader (SPE = 1, PSS = 00). To load new code through the UART would require the application code to call the UARTloader function in the utility ROM, which eventually passes control to the bootloader (more on this later).

If the MAXQ7667's password-protection feature is being used, it is important to note that setting the PWL (password lock) bit to 0 makes the MAXQ7667 vulnerable to attacks. It is recommended that after a communication link is established between the host and the MAXQ7667, the Password Match command (03h) be executed to access the password-protected family of commands.

2.2.3.3 Data Memory

The MAXQ7667 contains 2K x 16 (4096 bytes) of on-chip data SRAM that 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, as long as the DVDD (CORE) supply stays within the acceptable range.

On-chip data memory begins at address x0000h and is contiguous through the internal data memory. Data memory is accessed via indirect register addressing through a Data Pointer (@DP[n]) or Frame Pointer (@BP[OFFS]). The Data Pointer is used as one of the operands in a MOVE instruction. If the Data Pointer is used as source, the core performs a Load operation that reads data from the data memory location addressed by the Data Pointer. If the Data Pointer is used as destination, the core executes a Store operation that writes data to the data memory location addressed by the Data Pointer. The Data Pointer can be directly accessed by the user software.

The core incorporates two 16-bit Data Pointers (DP[0] and DP[1]) to support data memory accessing. All Data Pointers support indirect addressing mode and indirect addressing with autoincrement or autodecrement. Data Pointers DP[0] and DP[1] can be used as post increment/decrement source pointers by a MOVE instruction or pre increment/decrement destination pointers by a MOVE instruc-

tion. Using Data Pointer indirectly with "+" will automatically increase the content of the active Data Pointer by 1 immediately following the execution of read data transfer (@DP[n]++) or immediately preceding the execution of a write operation (@++DP[n]). Using Data Pointer indirectly with "--" will decrease the content of the active Data Pointer by 1 immediately following the execution of read data transfer (@DP[n]--) or immediately preceding the execution of a write operation (@--DP[n]).

The Frame Pointer (BP[OFFS]) is formed by 16-bit unsigned addition of Frame Pointer Base Register (BP) and Frame Pointer Offset Register (OFFS). Frame Pointer can be used as a post increment/decrement source pointer by a MOVE instruction or as a pre increment/decrement destination pointer. Using Frame Pointer indirectly with "+" (@BP[++OFFS] for a write or @BP[OFFS++] for a read) will automatically increase the content of the Frame Pointer Offset by 1 immediately before or after the execution of data transfer depending upon whether it is used as a destination or source pointer respectively. Using Frame Pointer indirectly with "--" (@BP[--OFFS] for a write or @BP[OFFS--] for a read) will decrease the content of the Frame Pointer Offset by 1 immediately before/after execution of data transfer depending upon whether it is used as a destination or source pointer respectively. Note that the increment/decrement function affects the content of the OFFS register only, while the contents of the BP register remain unaffected by the borrow/carry out from the OFFS register.

A data memory cycle contains only one system clock period to support fast internal execution. This allows read or write operations on SRAM to be completed in one clock cycle. Data memory mapping and access control are handled by the MMU. Read/write access to the data memory can be in word or in byte.

When using the in-circuit debugging features of the MAXQ7667, the top 32 bytes (bytes 0x7D0 to 0x7FF) 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 will not be used, the entire SRAM is available for application use.

2.2.3.4 Stack Memory

The MAXQ7667 provides a 16 x 16 hardware stack to support subroutine calls and system interrupts. A 16-bit wide on-chip stack is provided by the MAXQ7667 for storage of program return addresses and general-purpose use. The stack is used automatically by the processor when the CALL, RET, and RETI instructions are executed and when an interrupt is serviced; it can also be used explicitly to store and retrieve data by using the @SP- - source, @++SP destination, or the PUSH, POP, and POPI instructions. The POPI instruction acts identically to the POP instruction except that it additionally clears the INS bit.

The width of the stack is 16 bits to accommodate the instruction pointer size. The stack depth is 16 for the MAXQ7667. As the stack pointer register SP is used to hold the index of the top of the stack, the maximum size of the stack allowed is defined by the number of bits defined in the SP register (e.g., 4 bits for stack depth of 16).

On reset, the stack pointer SP initializes to the top of the stack (e.g. 0Fh for a 16-word stack). The CALL, PUSH, and interrupt vectoring operations increment SP and then store a value at @SP. The RET, RETI, POP, and POPI operations retrieve the value at @SP and then decrement SP.

As with the other RAM-based modules, the stack memory is initialized to indeterminate values upon reset or power-up. Stack memory is dedicated for stack operations only and cannot be accessed through program or data address spaces.

When using the in-circuit debugging features of the MAXQ7667, 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 will not be used, the entire stack is available for application use.

2.2.3.5 Pseudo-Von Neumann Memory Mapping

The MAXQ7667 supports a pseudo-Von Neumann memory structure that can merge program and data into a linear memory map. This is accomplished by mapping the data memory into the program space or mapping program memory segment into the data space. In all MAXQ processors the program memory ranges from x0000h to x7FFFh is the normal user code segment, followed by the utility ROM segment. The uppermost part of the 64KWord memory is the logical area for data memory when accessed as a code segment.

The program memory is logically divided into four program pages, in all MAXQ processors:

- P0 contains the lower 16KWords (available in MAXQ7667),
- P1 contains the second 16KWords (not available in MAXQ7667),
- P2 contains the third 16KWords (not available in MAXQ7667), and
- P3 contains the fourth 16KWords (not available in MAXQ7667).

The MAXQ7667 only has 16K of P0 space and hence the focus will be on P0.

The logical mapping of physical program memory page(s) into data space depends upon two factors: physical memory currently in use for program execution; and word/byte data memory access selection. If execution is from the utility ROM, physical program memory page (P0) can logically be mapped to the upper half of data memory space. If logical data memory is used for execution, physical program memory page can logically be mapped to the lower half of data memory space.

Figure 2-4 summarizes the default memory maps for this memory structure. The primary difference lies in the reset default settings for the data pointer Word/Byte Mode Select (WBSn) bits. The WBSn bits of the MAXQ7667 default to word access mode (WBSn = 1).

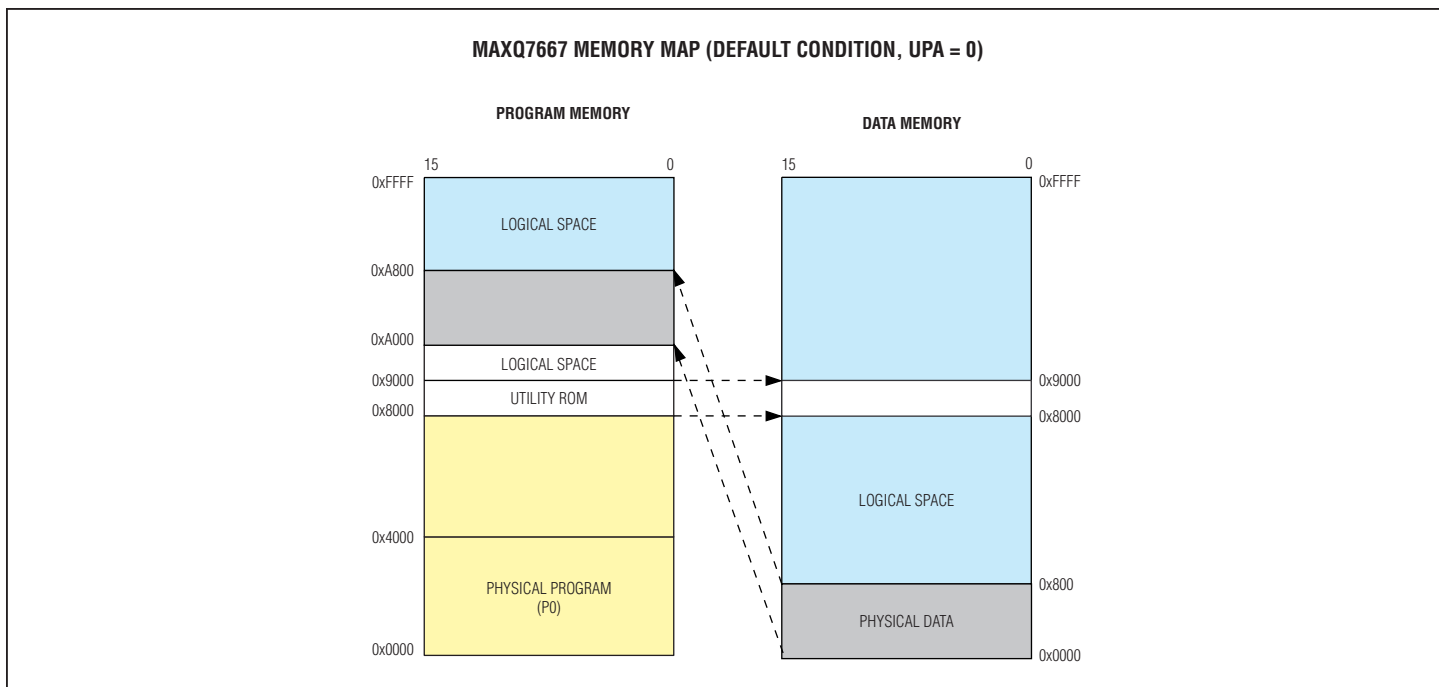


Figure 2-4. Pseudo-Von Neumann Memory Map (MAXQ7667 Default)

2.2.3.6 Pseudo-Von Neumann Memory Access

The pseudo-Von Neumann memory mapping is straightforward if there is no memory overlapping among the program, utility ROM, and data memory segments. When accessing the program memory as data, the CDA bit can be used to select the program pages as needed. Full data memory access to the physical program memory pages is based on the assumption that the maximum physical data memory is in the range of 16K x 16. The other restriction for accessing the pseudo-Von Neumann map is that when program execution is in a particular memory segment, the same memory segment cannot be simultaneously accessed as data.

When executing from the lower 16K program space (P0):

- The physical data memory is available for accessing as a code segment with offset at xA000h if the UPA bit is 0.
- Load and Store operations addressed to physical data memory are executed as normal.
- The utility ROM can be read as data, starting at x8000h of the data space.

When executing from the utility ROM:

- The lower 16K program space (P0) functions as normal program memory.
- The physical data memory is available for accessing as a code segment with offset at xA000h.
- Load and Store operations addressed to physical data memory are executed as normal.
- The 16K program space, P0, can be accessed as data, either in byte mode or word mode, with offset at 8000h.

When executing from the data memory:

- Program flows freely between the lower 16K user code (P0) and the utility ROM segment.
- The utility ROM can be accessed as data with offset at x8000h.
- The 16K program space, P0, can be accessed as data, either in byte mode or word mode, with offset at 0000h.

2.2.3.7 Data Alignment

To support merged program and data memory operation while maintaining efficiency on memory space usage, the data memory must be able to support both byte-wide and word-wide accessing. Data is aligned in data memory as word, but the effective data address is resolved to bytes. This data alignment allows direct program fetching in its native word size while maintaining accessibility at the byte level. It is important to realize that this accessibility requires strict word alignment. All executable words must align to an even address in byte mode. Care must be taken when updating the code segment in the unified data memory space as misalignment of words will likely result in loss of program execution control. Worst yet, this situation may not be detected if the watchdog timer is also disabled.

Data memory is organized as two byte-wide memory banks with common word address decode but two 8-bit data buses. The data memory will always be read as a complete word, independent of operation, whether program fetch or data access. The program decoder always uses the full 16-bit word, whereas the data access can utilize a word or an individual byte.

In byte mode, data pointer hardware reads out the word containing the selected byte using the effective data word address pointer (the least significant bit of the byte data pointer is not initially used). Then, the least significant data pointer bit functions as the byte select that is used to place the target byte to the data path. For write access, data pointer hardware addresses a particular word using the effective data word address while the least significant bit selects the corresponding data bank for write, leaving the contents of the another memory bank unaffected.

2.2.3.8 Memory Management Unit

Memory allocation and accessing control for program and data memory can be managed by the memory management unit (MMU). A single memory management unit option is discussed in this user's guide, however the memory management unit implementation for any given product depends upon the type and amount of memory addressable by the device. Users should consult the individual product data sheet(s) and/or user's guide supplement(s) for detailed information.

Although supporting less than the maximum addressable program and data memory segments, the MMU implementation presented provides a high degree of programming and access control flexibility. It supports the following:

- User program memory up to 32K x 16 (up to 64K x 16 with inclusion of UPA bit).
- Utility ROM up to 8K x 16.
- Data memory SRAM up to 16K x 16.
- In-system and in-application programming of embedded EEPROM, flash, or SRAM memories.
- Access to any of the three memory areas (SRAM, code memory, utility ROM) using the data memory pointers.
- Execution from any of the three memory areas (SRAM, code memory, factory written and tested utility-ROM routines).

Given these capabilities, the following rules apply to the memory map:

- A particular memory segment cannot be simultaneously accessed as both program and data.
- The offset address is xA000h when logically mapping data memory into the program space.
- The offset for logically mapping the utility ROM into the data memory space is x8000h.
- Program memory:
 - The lower half of the program memory (P0 and P1) is always accessible, starting at x0000h. (**Note:** P1 is not available in the MAXQ7667.)
 - The upper half of the program memory (P2 and P3) must be activated by setting the UPA bit to 1 when accessing for code execution, starting at x8000h. (**Note:** P2 and P3 are not available in the MAXQ7667.)
 - Setting the UPA bit to 1 disallows access to the utility ROM and logical data memory as program.
 - Physical program memory pages (P0, P1, P2, P3) are logically mapped into data space based upon the memory segment currently being used for execution, selection of byte/word access mode, and CDA1:0 bit settings (described in the *Pseudo-Von Neumann Memory Map* and *Pseudo-Von Neumann Memory Access* sections). (**Note:** This does not apply to the MAXQ7667 because it has only P0.)
- Data memory
 - Access can be either word or byte.
 - All 16 data pointer address bits are significant in either access mode (word or byte).

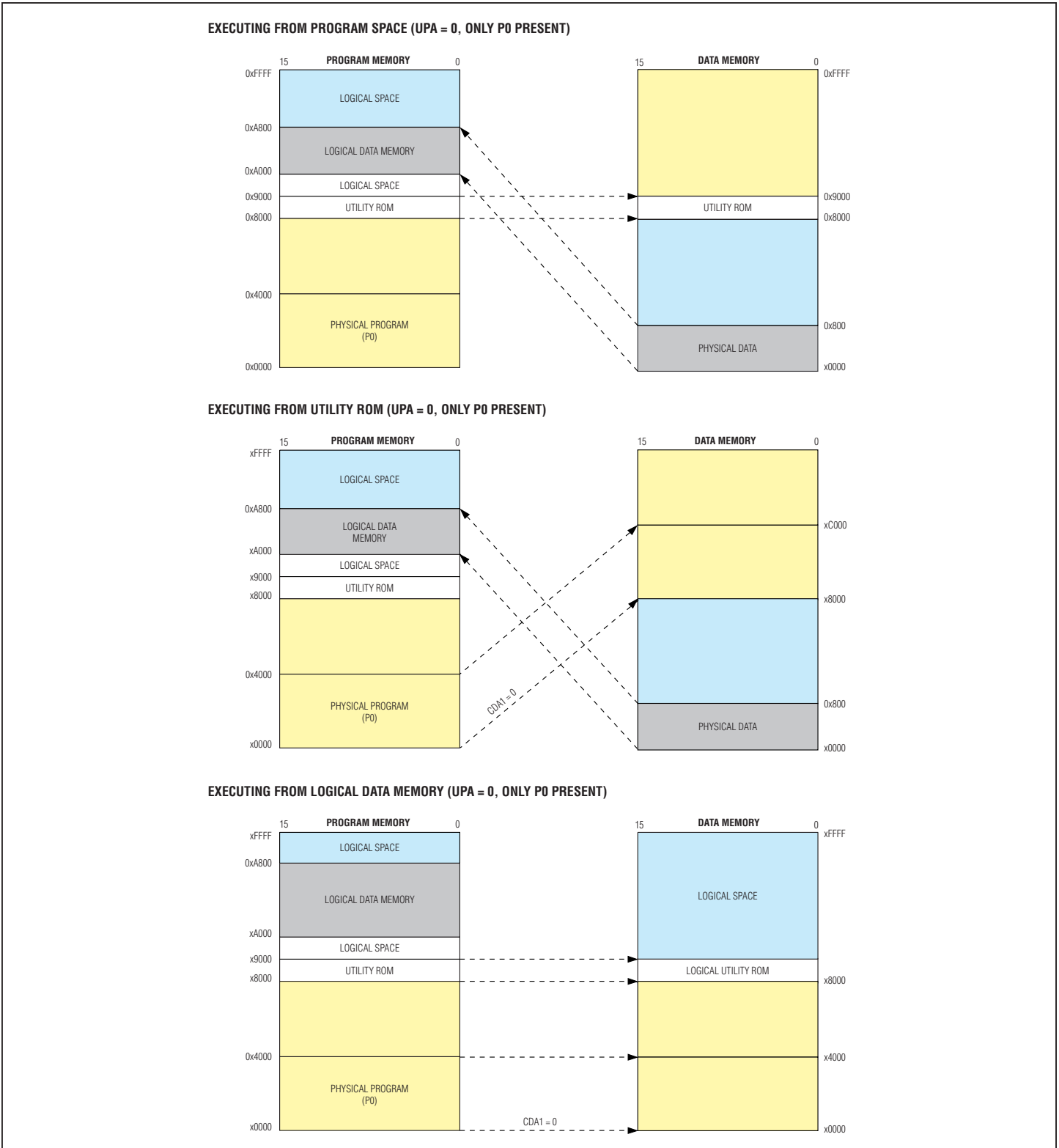


Figure 2-5. Word Access Mode in MAXQ7667

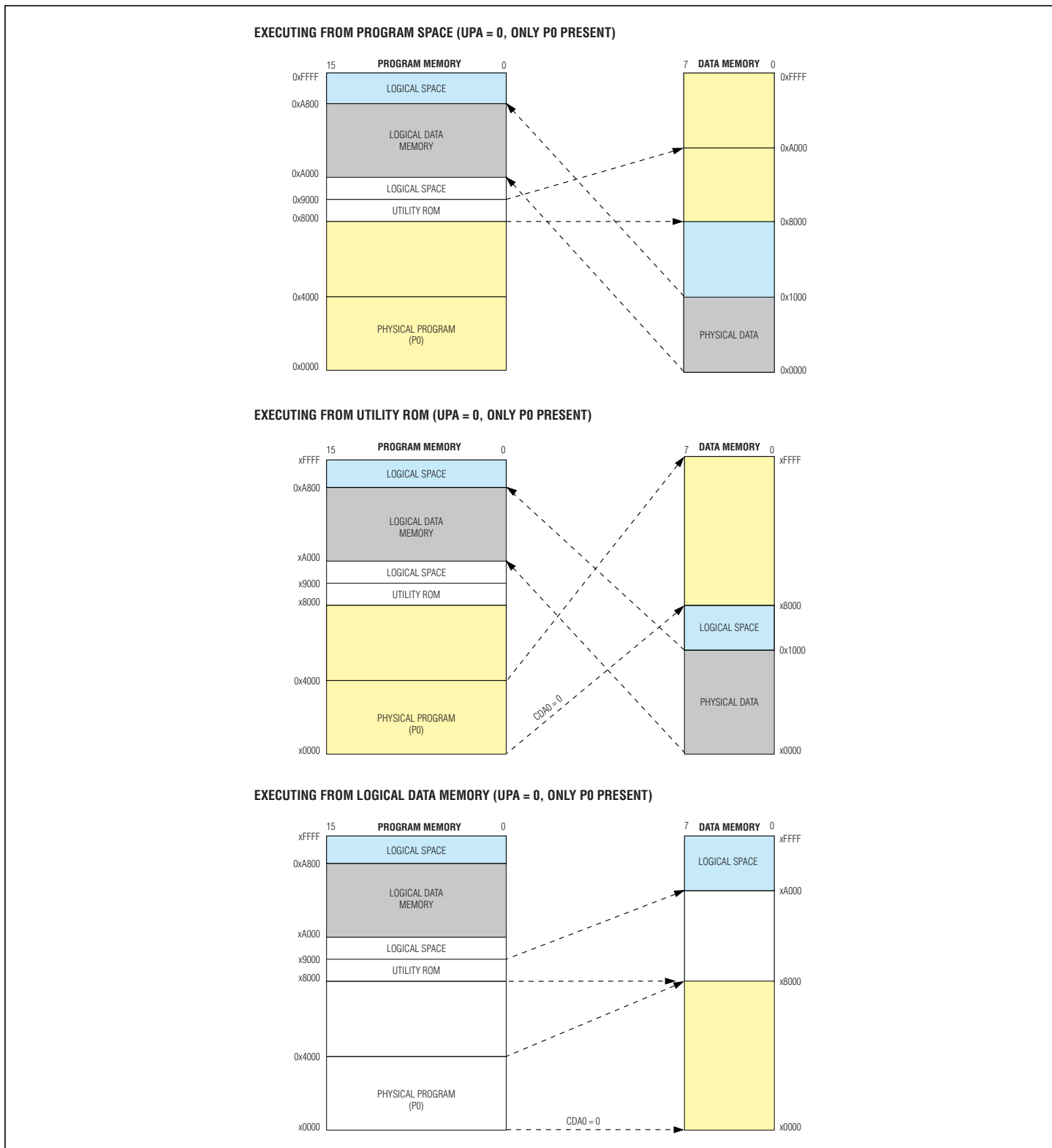


Figure 2-6. Byte Access Mode in MAXQ7667

2.2.4 Interrupts

The MAXQ7667 provides a single, programmable interrupt vector (IV) that can be used to handle internal and external interrupts. Interrupts can be generated from system level sources (e.g., watchdog timer) or by sources associated with the peripheral modules included in the specific MAXQ7667 microcontroller. Only one interrupt can be handled at a time, and all interrupts naturally have the same priority. A programmable interrupt mask register allows software-controlled prioritization and nesting of high-priority interrupts.

2.2.4.1 Servicing Interrupts

For the MAXQ7667 to service an interrupt, interrupts must be enabled globally, modularly, and locally. The Interrupt Global Enable (IGE) bit located in the Interrupt Control (IC) register acts as a global interrupt mask. This bit defaults to 0, and it must be set to 1 before any interrupt takes place.

The local interrupt-enable bit for a particular source is in one of the peripheral registers associated with that peripheral module, or in a system register for any system interrupt source. Between the global and local enables are intermediate per-module and system interrupt mask bits. These mask bits reside in the Interrupt Mask system register. By implementing intermediate per-module masking capability in a single register, interrupt sources spanning multiple modules can be selectively enabled/disabled in a single instruction. This promotes a simple, fast, and user-definable interrupt prioritization scheme. The interrupt source-enable hierarchy is illustrated in Figure 2-7.

When an interrupt condition occurs, its individual flag is set, even if the interrupt source is disabled at the local, module, or global level. Interrupt flags must be cleared within the user interrupt routine to avoid repeated interrupts from the same source.

Since all interrupts vector to the address contained in the Interrupt Vector (IV) register, the Interrupt Identification Register (IIR) may be used by the interrupt service routine to determine the module source of an interrupt. The IIR contains a bit flag for each peripheral module and one flag associated with all system interrupts; if the bit for a module is set, then an interrupt is pending that was initiated by that module. If a module is capable of generating interrupts for different reasons, then peripheral register bits inside the module provide a means to differentiate among interrupt sources.

The Interrupt Vector (IV) register provides the location of the interrupt service routine. It may be set to any location within program memory. The IV register defaults to 0000h on reset or power-up, so if it is not changed to a different address, the user program must determine whether a jump to 0000h came from a reset or interrupt source. Note that the password starts at 0x0010, thus leaving 16 words of programming space between the interrupt vector and the password, if 0000h is used as the IV value. (See Sections 12 and 13 for details on password handling.)

2.2.4.2 Interrupt System Operation

The interrupt handler hardware responds to any interrupt event when it is enabled. An interrupt event occurs when an interrupt flag is set. All interrupt requests are sampled at the rising edge of the clock and can be serviced by the processor one clock cycle later, assuming the request does not hit the interrupt exception window. The one-cycle stall between detection and acknowledgement/servicing is due to the fact that the current instruction may also be accessing the stack. For this reason, the CPU must allow the current instruction to complete before pushing the stack and vectoring to IV. If an interrupt exception window is generated by the currently executing instruction, the following instruction must be executed, so the interrupt service routine will be delayed an additional cycle.

Interrupt operation in the MAXQ7667 CPU is essentially a state machine generated long CALL instruction. When the interrupt handler services an interrupt, it temporarily takes control of the CPU to perform the following sequence of actions:

- 1) The next instruction fetch from program memory is cancelled.
- 2) The return address is pushed on to the stack.
- 3) The INS bit is set to 1 to prevent recursive interrupt calls.
- 4) The instruction pointer is set to the location of the interrupt service routine (contained in the Interrupt Vector register).
- 5) The CPU begins executing the interrupt service routine.

Once the interrupt service routine completes, it should use the RETI instruction to return to the main program. Execution of RETI involves the following sequence of actions:

- 1) The return address is popped off the stack.
- 2) The INS bit is cleared to 0 to re-enable interrupt handling.
- 3) The instruction pointer is set to the return address that was popped off the stack.
- 4) The CPU continues execution of the main program.

Pending interrupt requests will not interrupt an RETI instruction; a new interrupt will be serviced after first being acknowledged in the execution cycle which follows the RETI instruction and then after the standard one stall cycle of interrupt latency. This means there will be at least two cycles between back-to-back interrupts.

2.2.4.3 Synchronous vs. Asynchronous Interrupt Sources

Interrupt sources can be classified as either asynchronous or synchronous. All internal interrupts are synchronous interrupts. An internal interrupt is directly routed to the interrupt handler that can be recognized in one cycle. All external interrupts are asynchronous interrupts by nature. When the device is not in stop mode, asynchronous interrupt sources are passed through a 3-clock sampling/glitch filter circuit before being routed to the interrupt handler. The sampling/glitch filter circuit is running on the undivided source clock (i.e., before PMME, CD[1:0]-controlled clock divide) such that the number of system clocks required to recognize an asynchronous interrupt request depends upon the system clock divide ratio:

- if the system clock divide ratio is 1, the interrupt request is recognized after 3 system clock
- if the system clock divide ratio is 2, the interrupt request is recognized after 2 system clock (unavailable in MAXQ7667)
- if the system clock divide ratio is 4 or greater, the interrupt request is recognized after 1 system clock (unavailable in MAXQ7667)

An interrupt request with a pulse width less than three undivided clock cycles is not recognized. Note that the granularity of interrupt source is at module level. Synchronous interrupts and sampled asynchronous interrupts assigned to the same module product a single interrupt to the interrupt handler.

External interrupts, when enabled, can be used as switchback sources from power management mode. There is no latency associated with the switchback because the circuit is being clocked by an undivided clock source versus the divide-by-256 system clock. For the same reason, there is no latency for other switchback sources that do not qualify as interrupt sources.

2.2.4.4 Interrupt Prioritization by Software

All interrupt sources of the MAXQ7667 microcontroller naturally have the same priority. However, when CPU operation vectors to the programmed Interrupt Vector address, the order in which potential interrupt sources are interrogated is left entirely up to the user, as this often depends upon the system design and application requirements. The Interrupt Mask system register provides the ability to knowingly block interrupts from modules considered to be of lesser priority and manually re-enable the interrupt servicing by the CPU (by setting INS = 0). Using this procedure, a given interrupt service routine can continue executing, only to be interrupted by higher priority interrupts. An example demonstrating this software prioritization is provided in *Section 3.8: Handling Interrupts*.

2.2.4.5 Interrupt Exception Window

An interrupt exception window is a noninterruptable execution cycle. During this cycle, the interrupt handler does not respond to any interrupt requests. All interrupts that would normally be serviced during an interrupt exception window are delayed until the next execution cycle.

Interrupt exception windows are used when two or more instructions must be executed consecutively without any delays in between. Currently, there is a single condition in the MAXQ7667 microcontroller that causes an interrupt exception window: activation of the prefix (PFX) register.

When the prefix register is activated by writing a value to it, it retains that value only for the next clock cycle. For the prefix value to be used properly by the next instruction, the instruction that sets the prefix value and the instruction that uses it must always be executed back to back. Therefore, writing to the PFX register causes an interrupt exception window on the next cycle. If an interrupt occurs during an interrupt exception window, an additional latency of one cycle in the interrupt handling will be caused as the interrupt will not be serviced until the next cycle.

2.2.4.6 MAXQ7667 Interrupt Sources

Table 2-2 lists all possible interrupt sources for the MAXQ7667, along with their corresponding module interrupt enable bits, local interrupt enable bits, and interrupt flags.

- Each module interrupt enable bit, when cleared to 0, will block 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 the interrupts have been disabled globally).
- Each local interrupt enable bit, when cleared to 0, will disable the corresponding interrupt. When the local interrupt enable bit is set to 1, the interrupt will be triggered whenever the interrupt flag is set to 1 (either by software or hardware).
- All interrupt flag bits cause the corresponding interrupt to trigger when the bit is set to 1. These 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:

- 1) Interrupts must be enabled globally by setting IGE (IC.0) to 1.
- 2) The module interrupt enable bit for that interrupt source's module must be set to 1.

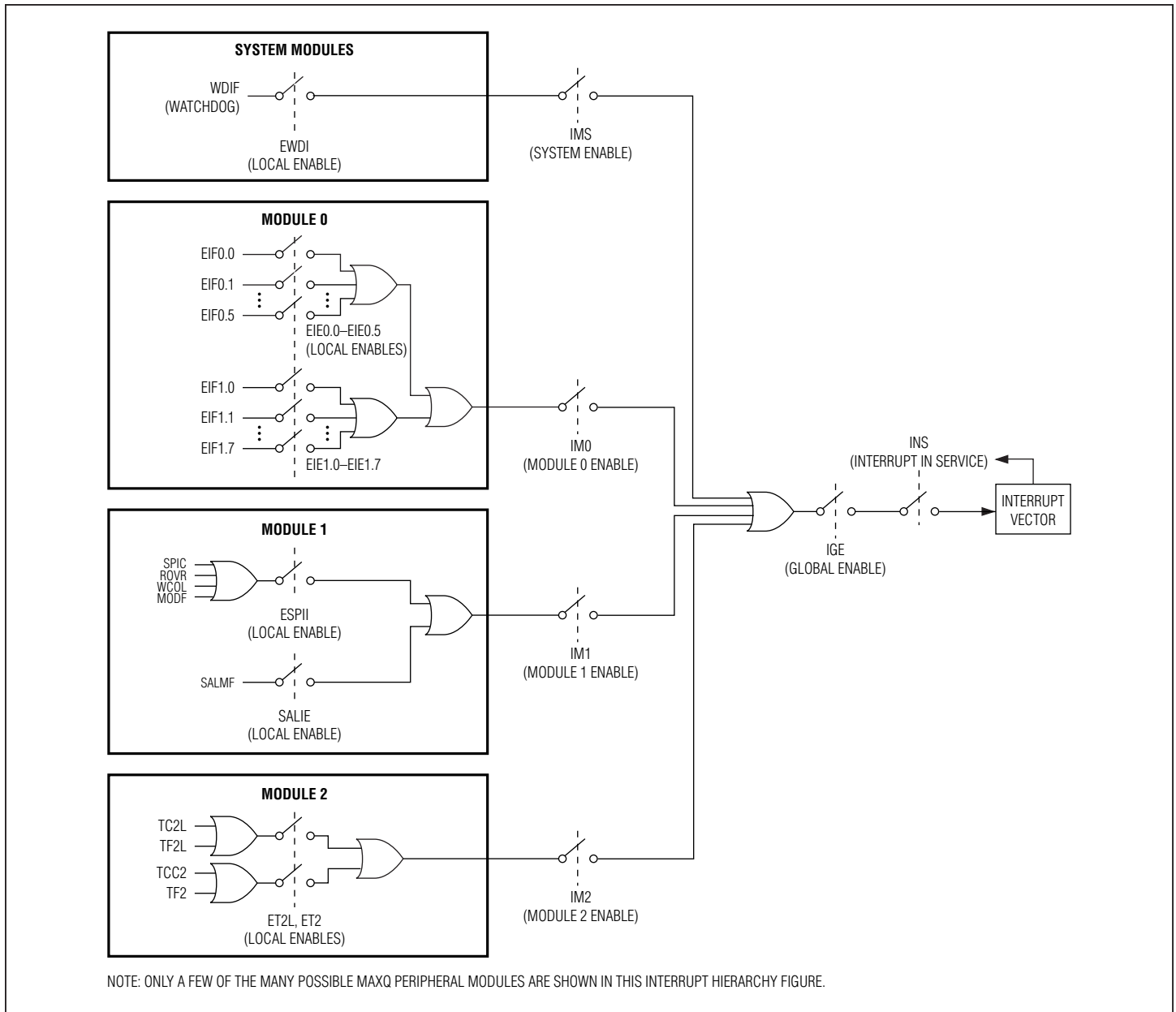


Figure 2-7. MAXQ7667 Interrupt Source Hierarchy Example

- 3) The local interrupt enable bit for that specific interrupt source must be set to 1.
- 4) The interrupt flag for that interrupt source must be set to 1. Typically, this is done by hardware when the condition that requires interrupt service occurs.
- 5) The Interrupt In Service (INS) bit must be cleared to 0. This bit is set automatically upon vectoring to the interrupt handler address and cleared automatically upon exit (RETI/POPI), so the only reason to clear this bit manually (inside the interrupt handler routine) is allow nested interrupt handling.

Table 2-3. MAXQ7667 Interrupt Sources and Control Bits

INTERRUPT	MODULE ENABLE BIT	LOCAL ENABLE BIT	INTERRUPT FLAG
External Interrupt Port 0 0	IM0 (IMR.0)	EX0 (EIE0.0)	IE0 (EIF0.0)
External Interrupt Port 0 1		EX1 (EIE0.1)	IE1 (EIF0.1)
External Interrupt Port 0 2		EX2 (EIE0.2)	IE2 (EIF0.2)
External Interrupt Port 0 3		EX3 (EIE0.3)	IE3 (EIF0.3)
External Interrupt Port 0 4		EX4 (EIE0.4)	IE4 (EIF0.4)
External Interrupt Port 0 5		EX5 (EIE0.5)	IE5 (EIF0.5)
External Interrupt Port 1 0		EX0 (EIE1.0)	IE0 (EIF1.0)
External Interrupt Port 1 1		EX1 (EIE1.1)	IE1 (EIF1.1)
External Interrupt Port 1 2		EX2 (EIE1.2)	IE2 (EIF1.2)
External Interrupt Port 1 3		EX3 (EIE1.3)	IE3 (EIF1.3)
External Interrupt Port 1 4		EX4 (EIE1.4)	IE4 (EIF1.4)
External Interrupt Port 1 5		EX5 (EIE1.5)	IE5 (EIF1.5)
External Interrupt Port 1 6		EX6 (EIE1.6)	IE6 (EIF1.6)
External Interrupt Port 1 7		EX7 (EIE1.7)	IE7 (EIF1.7)
SPI Mode Fault	IM1 (IMR.1)	ESPII (SPICF.7)	MODF (SPICN.3)
SPI Write Collision		ESPII (SPICF.7)	WCOL (SPICN.4)
SPI Receive Overrun		ESPII (SPICF.7)	ROVR (SPICN.5)
SPI Transfer Complete		ESPII (SPICF.7)	SPIC (SPICN.6)
Schedule Timer Alarm		SALIE (SCNT.7)	SALMF (SCNT.6)
Timer 0—Low Compare	IM2 (IMR.2)	ET2L (T2CNB0.7)	T2CL (T2CNB0.0)
Timer 0—Low Overflow		ET2L (T2CNB0.7)	TF2L (T2CNB0.2)
Timer 0—Capture/Compare		ET2 (T2NCA0.7)	TCC2 (T2CNB0.1)
Timer 0—Overflow		ET2 (T2NCA0.7)	TF2 (T2CNB0.3)
Timer 1—Low Compare		ET2L (T2CNB1.7)	T2CL (T2CNB1.0)
Timer 1—Low Overflow		ET2L (T2CNB1.7)	TF2L (T2CNB1.2)
Timer 1—Capture/Compare		ET2 (T2NCA1.7)	TCC2 (T2CNB1.1)
Timer 1—Overflow		ET2 (T2NCA1.7)	TF2 (T2CNB1.3)
Timer 2—Low Compare	IM3 (IMR.3)	ET2L (T2CNB2.7)	T2CL (T2CNB2.0)
Timer 2—Low Overflow		ET2L (T2CNB2.7)	TF2L (T2CNB2.2)
Timer 2—Capture/Compare		ET2 (T2NCA2.7)	TCC2 (T2CNB2.1)
Timer 2—Overflow		ET2 (T2NCA2.7)	TF2 (T2CNB2.3)
UART Mode Receive Interrupt		IE (SMD.2)	RI (SCON0.0)
UART Mode Transmit Interrupt		IE (SMD.2)	TI (SCON0.1)
LIN Mode Master or Slave Interrupt		INE (CNT0.4)	INP (STA0.1)

Table 2-3. MAXQ7667 Interrupt Sources and Control Bits (continued)

INTERRUPT	MODULE ENABLE BIT	LOCAL ENABLE BIT	INTERRUPT FLAG
SAR ADC Data Ready	IM5 (IMR.5)	SARIE (AIE.0)	SARRDY (ASR.0)
Echo Envelope Lowpass Filter Output		LPFIE (AIE.1)	LPFRDY (ASR.1)
Echo Envelope Lowpass Filter FIFO Full Interrupt		LFLIE (AIE.2)	LPFFL (ASR.2)
Echo Envelope Comparator Interrupt		CMPIE (AIE.3)	CMPI (ASR.3)
AVDD Brownout Interrupt		VABIE (AIE.4)	VABI (ASR.4)
DVDD Brownout Interrupt		VDBIE (AIE.5)	VDBI (ASR.5)
DVDDIO Brownout Interrupt		VIBIE (AIE.6)	VIBI (ASR.6)
Crystal Oscillator Failure Interrupt		XTIE (AIE.7)	XTI (ASR.7)

SECTION 3: PROGRAMMING

This section contains the following information:

3.1 Addressing Modes	3-3
3.2 Prefixing Operations	3-3
3.3 Reading and Writing Registers	3-4
3.3.1 Loading an 8-Bit Register with an Immediate Value	3-4
3.3.2 Loading a 16-Bit Register with a 16-Bit Immediate Value	3-4
3.3.3 Moving Values Between Registers of the Same Size	3-4
3.3.4 Moving Values Between Registers of Different Sizes	3-5
3.3.4.1 8-Bit Destination ← Low Byte (16-Bit Source)	3-5
3.3.4.2 8-Bit Destination ← High Byte (16-Bit Source)	3-5
3.3.4.3 16-Bit Destination ← Concatenation (8-Bit Source, 8-Bit Source)	3-5
3.3.4.4 Low (16-Bit Destination) ← 8-Bit Source	3-5
3.3.4.5 High (16-Bit Destination) ← 8-Bit Source	3-6
3.4 Reading and Writing Register Bits	3-6
3.5 Using the Arithmetic and Logic Unit	3-6
3.5.1 Selecting the Active Accumulator	3-6
3.5.2 Enabling Autoincrement and Autodecrement	3-7
3.5.3 ALU Operations Using the Active Accumulator and a Source	3-9
3.5.4 ALU Operations Using Only the Active Accumulator	3-9
3.5.5 ALU Bit Operations Using Only the Active Accumulator	3-9
3.5.6 Example: Adding Two 4-Byte Numbers Using Autoincrement	3-10
3.6 Processor Status Flag Operations	3-10
3.6.1 Sign Flag	3-10
3.6.2 Zero Flag	3-10
3.6.3 Equals Flag	3-10
3.6.4 Carry Flag	3-10
3.6.5 Overflow Flag	3-11
3.7 Controlling Program Flow	3-11
3.7.1 Obtaining the Next Execution Address	3-11
3.7.2 Unconditional Jumps	3-12

3.7.3 Conditional Jumps3-12
3.7.4 Calling Subroutines3-12
3.7.5 Looping Operations3-12
3.7.6 Conditional Returns3-13
3.8 Handling Interrupts3-13
3.8.1 Conditional Return from Interrupt3-14
3.9 Accessing the Stack3-15
3.10 Accessing Data Memory3-16

LIST OF TABLES

Table 3-1. Accumulator Pointer Control Register Settings3-8
--	------

SECTION 3: PROGRAMMING

This section provides a programming overview of the MAXQ7667. For full details on the instruction set, as well as System Register and Peripheral Register detailed bit descriptions, see the appropriate sections in this user's guide.

3.1 Addressing Modes

The instruction set for the MAXQ7667 provides three different addressing modes: direct, indirect, and immediate.

The direct addressing mode can be used to specify either source or destination registers, such as:

```

move  A[ 0] , A[ 1]          ; copy accumulator 1 to accumulator 0
push  A[ 0]                  ; push accumulator 0 on the stack
add   A[ 1]                  ; add accumulator 1 to the active accumulator

```

Direct addressing is also used to specify addressable bits within registers.

```

move  C, Acc.0              ; copy bit zero of the active accumulator
                               ; to the carry flag
move  PO0.3, #1             ; set bit three of port 0 Output register

```

Indirect addressing, in which a register contains a source or destination address, is used only in a few cases.

```

move  @DP[ 0] , A[ 0]      ; copy accumulator 0 to the data memory
                               ; location pointed to by data pointer 0
move  A[ 0] , @SP--        ; where @SP-- is used to pop the data pointed to
                               ; by the stack pointer register

```

Immediate addressing is used to provide values to be directly loaded into registers or used as operands.

```

move  A[ 0] , #10h         ; set accumulator 1 to 10h/16d

```

3.2 Prefixing Operations

All instructions on the MAXQ7667 are 16 bits long and execute in a single cycle. However, some operations require more data than can be specified in a single cycle or require that high-order register-index bits be set to achieve the desired transfer. In these cases, the prefix register module PFX is loaded with temporary data and/or required register index bits to be used by the following instruction. The PFX module only holds loaded data for a single cycle before it clears to zero.

Instruction prefixing is required for the following operations, which effectively makes them two-cycle operations.

- When providing a 16-bit immediate value for an operation (e.g., loading a 16-bit register, ALU operation, supplying an absolute program branch destination), the PFX module must be loaded in the previous cycle with the high byte of the 16-bit immediate value unless that high byte is zero. One exception to this rule is when supplying an absolute branch destination to 00xxh. In this case, PFX still must be written with 00h. Otherwise, the branch instruction would be considered a relative one instead of the desired absolute branch.
- When selecting registers with indexes greater than 07h within a module as destinations for a transfer or registers with indexes greater than 0Fh within a module as sources, the PFX[n] register must be loaded in the previous cycle. This can be combined with the previous item.

Generally, prefixing operations can be inserted automatically by the assembler as needed, so that (for example)

```
move DP[ 0] , #1234h
```

actually assembles as

```
move PFX[ 0] , #12h
move DP[ 0] , #34h
```

However, the operation

```
move DP[ 0] , #0055h
```

does not require a prefixing operation even though the register DP[0] is 16-bit. This is because the prefix value defaults to zero, so the line

```
move PFX[ 0] , #00h
```

is not required.

3.3 Reading and Writing Registers

All functions in the MAXQ7667 are accessed through registers, either directly or indirectly. This section discusses loading registers with immediate values and transferring values between registers of the same size and different sizes.

3.3.1 Loading an 8-Bit Register with an Immediate Value

Any writable 8-bit register with a subindex from 0h to 7h within its module can be loaded with an immediate value in a single cycle using the MOVE instruction.

```
move AP, #05h ; load accumulator pointer register with 5 hex
```

Writable 8-bit registers with subindexes 8h and higher can be loaded with an immediate value using MOVE as well, but an additional cycle is required to set the prefix value for the destination.

```
move WDCN, #33h ; assembles to: move PFX[ 2] , #00h
; move (WDCN-80h), #33h
```

3.3.2 Loading a 16-Bit Register with a 16-Bit Immediate Value

Any writable 16-bit register with a subindex from 0h to 07h can be loaded with an immediate value in a single cycle if the high byte of that immediate value is zero.

```
move LC[ 0] , #0010h ; prefix defaults to zero for high byte
```

If the high byte of that immediate value is not zero or if the 16-bit destination subindex is greater than 7h, an extra cycle is required to load the prefix value for the high byte and/or the high-order register index bits.

```
move LC[ 0] , #0110h ; high byte <> #00h
; assembles to: move PFX[ 0] , #01h
; move LC[ 0] , #10h
; destination sub-index > 7h
move A[ 8] , #0034h ; assembles to: move PFX[ 2] , #00h
; move (A[ 8]-80h), #34h
```

3.3.3 Moving Values Between Registers of the Same Size

Moving data between same-size registers can be done in a single-cycle MOVE if the destination register's index is from 0h to 7h and the source register index is between 0h and Fh.

```
move A[ 0] , A[ 8] ; copy accumulator 8 to accumulator 0
move LC[ 0] , LC[ 1] ; copy loop counter 1 to loop counter 0
```

If the destination register's index is greater than 7h or if the source register index is greater than Fh, prefixing is required.

```
move A[ 15] , A[ 0] ; assembles to: move PFX[ 2] , #00h
; move (A[ 15]-80h), A[ 0]
```

3.3.4 Moving Values Between Registers of Different Sizes

Before covering some transfer scenarios that might arise, a special register must be introduced that will be used in many of these cases. The 16-bit General Register (GR) is expressly provided for performing byte singulation of 16-bit words. The high and low bytes of GR are individually accessible in the GRH and GRL registers respectively. A read-only GRS register makes a byte-swapped version of GR accessible and the GRXL register provides a sign-extended version of GRL.

3.3.4.1 8-Bit Destination ← Low Byte (16-Bit Source)

The simplest transfer possibility would be loading an 8-bit register with the low byte of a 16-bit register. This transfer does not require use of GR and requires a prefix only if the destination or source register are outside of the single cycle write or read regions, 0–7h and 0–Fh, respectively.

```

move  OFFS, LC[ 0]          ; copy the low byte of LC[ 0] to the OFFS register
move  IMR,  @DP[ 1]        ; copy the low byte @DP[ 1] to the IMR register
move  WDCN, LC[ 0]          ; assembles to: move PFX[ 2], #00h
                                ;               move (WDCON-80h), LC[ 0]

```

3.3.4.2 8-Bit Destination ← High Byte (16-Bit Source)

If, however, we needed to load an 8-bit register with the high byte of a 16-bit source, it would be best to use the GR register. Transferring the 16-bit source to the GR register adds a single cycle.

```

move  GR,  LC[ 0]          ; move LC[ 0] to the GR register
move  IC,  GRH              ; copy the high byte into the IC register

```

3.3.4.3 16-Bit Destination ← Concatenation (8-Bit Source, 8-Bit Source)

Two 8-bit source registers can be concatenated and stored into a 16-bit destination by using the prefix register to hold the high-order byte for the concatenated transfer. An additional cycle may be required if either source byte register index is greater than 0Fh or the 16-bit destination is greater than 07h.

```

move  PFX[ 0], IC          ; load high order source byte IC into PFX
move  @DP[ 0], AP          ; store @DP[ 0] the concatenation of IC:AP

                                ; 16-bit destination sub-index: dst=08h
                                ; 8-bit source sub-indexes:
                                ;   high=10h, low=11h

move  PFX[ 1], #00h        ;
move  PFX[ 3], high        ; PFX=00:high
move  dst, low              ; dst=high:low

```

3.3.4.4 Low (16-Bit Destination) ← 8-Bit Source

To modify only the low byte of a given 16-bit destination, the 16-bit register should be moved into the GR register such that the high byte can be singulated and the low byte written exclusively. An additional cycle is required if the destination index is greater than 0Fh.

```

move  GR,  DP[ 0]          ; move DP[ 0] to the GR register
move  PFX[ 0], GRH          ; get the high byte of DP[ 0] via GRH
move  DP[ 0], #20h          ; store the new DP[ 0] value
                                ; 16-bit destination sub-index: dst=10h
                                ; 8-bit source sub-index: src=11h

move  PFX[ 1], #00h        ;
move  GR,  dst              ; read dst word to the GR register
move  PFX[ 5], GRH          ; get the high byte of dst via GRH
move  dst, src              ; store the new dst value

```


3.3.4.5 High (16-Bit Destination) ← 8-Bit Source

To modify only the high byte of a given 16-bit destination, the 16-bit register should be moved into the GR register such that the low byte can be singulated and the high byte can be written exclusively. Additional cycles are required if the destination index is greater than 0Fh or if the source index is greater than 0Fh.

```

move GR, DP[ 0]           ; move DP[ 0] to the GR register
move PFX[ 0], #20h        ; get the high byte of DP[ 0] via GRH
move DP[ 0], GRL          ; store the new DP[ 0] value
                           ; 16-bit destination sub-index: dst=10h
                           ; 8-bit source sub-index: src=11h

move PFX[ 1], #00h        ;
move GR, dst              ; read dst word to the GR register
move PFX[ 1], #00h        ;
move PFX[ 4], src         ; get the new src byte
move dst, GRL             ; store the new dst value

```

If the high byte needs to be cleared to 00h, the operation can be shortened by transferring only the GRL byte to the 16-bit destination (example follows):

```

move GR, DP[ 0]           ; move DP[ 0] to the GR register
move DP[ 0], GRL          ; store the new DP[ 0] value, 00h used for high byte

```

3.4 Reading and Writing Register Bits

The MOVE instruction can also be used to directly set or clear any one of the lowest 8 bits of a peripheral register in module 0h–5h or a system register in module 8h. The set or clear operation will not affect the upper byte of a 16-bit register that is the target of the set or clear operation. If a set or clear instruction is used on a destination register that does not support this type of operation, the register high byte will be written with the prefix data and the low byte will be written with the bit mask (i.e., all zeros with a single 1 for the set bit operation or all ones with a single 0 for the clear bit operation).

Register bits can be set or cleared individually using the MOVE instruction as follows.

```

move IGE, #1              ; set IGE (Interrupt Global Enable) bit
move APC.6, #0           ; clear IDS bit (APC.6)

```

As with other instructions, prefixing is required to select destination registers beyond index 07h.

The MOVE instruction may also be used to transfer any one of the lowest 8 bits from a register source or any bit of the active accumulator (Acc) to the Carry flag. There is no restriction on the source register module for the 'MOVE C, src.bit' instruction.

```

move C, IIR.3             ; copy IIR.3 to Carry
move C, Acc.7            ; copy Acc.7 to Carry

```

Prefixing is required to select source registers beyond index 15h.

3.5 Using the Arithmetic and Logic Unit

The MAXQ7667 provides a 16-bit (MAXQ20) ALU, which allows operations to be performed between the active accumulator and any other register. The ALU configuration provides 16 accumulator registers that are also 16 bits (MAXQ20) wide, of which any one may be selected as the active accumulator.

3.5.1 Selecting the Active Accumulator

Any of the 16 accumulator registers A[0] through A[15] may be selected as the active accumulator by setting the low four bits of the Accumulator Pointer Register (AP) to the index of the accumulator register you want to select.

```

move AP, #01h            ; select A[ 1] as the active accumulator
move AP, #0Fh            ; select A[15] as the active accumulator

```

The current active accumulator can be accessed as the Acc register, which is also the register used as the implicit destination for all arithmetic and logical operations.

```

move A[ 0], #55h         ; set A[ 0] = 55 hex (MAXQ10)
                           ; = 0055 hex (MAXQ20)
move AP, #00h           ; select A[ 0] as active accumulator
move Acc, #55h          ; set A[ 0] = 55 hex (MAXQ10)
                           ; = 0055 hex (MAXQ20)

```

3.5.2 Enabling Autoincrement and Autodecrement

The accumulator pointer AP can be set to automatically increment or decrement after each arithmetic or logical operation. This is useful for operations involving a number of accumulator registers, such as adding or subtracting two multibyte integers.

If autoincrement/decrement is enabled, the AP register increments or decrements after any of the following operations:

- ADD src (Add source to active accumulator)
- ADDC src (Add source to active accumulator with carry)
- SUB src (Subtract source from active accumulator)
- SUBB src (Subtract source from active accumulator with borrow)
- AND src (Logical AND active accumulator with source)
- OR src (Logical OR active accumulator with source)
- XOR src (Logical XOR active accumulator with source)
- CPL (Bit-wise complement active accumulator)
- NEG (Negate active accumulator)
- SLA (Arithmetic shift left on active accumulator)
- SLA2 (Arithmetic shift left active accumulator two bit positions)
- SLA4 (Arithmetic shift left active accumulator four bit positions)
- SRA (Arithmetic shift right on active accumulator)
- SRA2 (Arithmetic shift right active accumulator two bit positions)
- SRA4 (Arithmetic shift right active accumulator four bit positions)
- RL (Rotate active accumulator left)
- RLC (Rotate active accumulator left through Carry flag)
- RR (Rotate active accumulator right)
- RRC (Rotate active accumulator right through Carry flag)
- SR (Logical shift active accumulator right)
- MOVE Acc, src (Copy data from source to active accumulator)
- MOVE dst, Acc (Copy data from active accumulator to destination)
- MOVE Acc, Acc (Recirculation of active accumulator contents)
- XCHN (Exchange nibbles within each byte of active accumulator)
- XCH (Exchange active accumulator bytes)

The active accumulator may not be the source in any instruction where it is also the implicit destination.

There is an additional notation that can be used to refer to the active accumulator for the instruction "MOVE dst, Acc." If the instruction is instead written as "MOVE dst, A[AP]," the source value is still the active accumulator, but no AP autoincrement or autodecrement function will take place, even if this function is enabled. Note that the active accumulator may not be the destination for the MOVE dst, A[AP] instruction (i.e., MOVE Acc, A[AP] is prohibited).

So, the two instructions

```
move  A[ 7] , Acc
move  A[ 7] , A[ AP]
```

are equivalent, except that the first instruction triggers autoinc/dec (if it is enabled), while the second one will never do so.

The Accumulator Pointer Control Register (APC) controls the automatic-increment/decrement mode as well as selects the range of bits (modulo) in the AP register that will be incremented or decremented. There are nine different unique settings for the APC register, as listed in Table 3-1.

Table 3-1. Accumulator Pointer Control Register Settings

APC.2 (MOD2)	APC.1 (MOD1)	APC.0 (MOD0)	APC.6 (IDS)	APC	AUTO INCREMENT/DECREMENT SETTING
0	0	0	X	00h	No autoincrement/decrement (default mode)
0	0	1	0	01h	Increment bit 0 of AP (modulo 2)
0	0	1	1	41h	Decrement bit 0 of AP (modulo 2)
0	1	0	0	02h	Increment bits [1:0] of AP (modulo 4)
0	1	0	1	42h	Decrement bits [1:0] of AP (modulo 4)
0	1	1	0	03h	Increment bits [2:0] of AP (modulo 8)
0	1	1	1	43h	Decrement bits [2:0] of AP (modulo 8)
1	0	0	0	04h	Increment all 4 bits of AP (modulo 16)
1	0	0	1	44h	Decrement all 4 bits of AP (modulo 16)

For the modulo increment or decrement operation, the selected range of bits in AP are incremented or decremented. However, if these bits roll over or under, they simply wrap around without affecting the remaining bits in the accumulator pointer. So, the operations can be defined as follows:

- Increment modulo 2: $AP = AP[3:1] + ((AP[0] + 1) \bmod 2)$
- Decrement modulo 2: $AP = AP[3:1] + ((AP[0] - 1) \bmod 2)$
- Increment modulo 4: $AP = AP[3:2] + ((AP[1:0] + 1) \bmod 4)$
- Decrement modulo 4: $AP = AP[3:2] + ((AP[1:0] - 1) \bmod 4)$
- Increment modulo 8: $AP = AP[3] + ((AP[2:0] + 1) \bmod 8)$
- Decrement modulo 8: $AP = AP[3] + ((AP[2:0] - 1) \bmod 8)$
- Increment modulo 16: $AP = (AP + 1) \bmod 16$
- Decrement modulo 16: $AP = (AP - 1) \bmod 16$

For this example, assume that all 16 accumulator registers are initially set to zero.

```

move AP, #02h           ; select A[ 2] as active accumulator
move APC, #02h         ; auto-increment AP[ 1:0] modulo 4
                        ; AP  A[ 0]  A[ 1]  A[ 2]  A[ 3]
                        ; 02  0000  0000  0000  0000
add #01h               ; 03  0000  0000  0001  0000
add #02h               ; 00  0000  0000  0001  0002
add #03h               ; 01  0003  0000  0001  0002
add #04h               ; 02  0003  0004  0001  0002
add #05h               ; 03  0003  0004  0006  0002

```

3.5.3 ALU Operations Using the Active Accumulator and a Source

The following arithmetic and logical operations can use any register or immediate value as a source. The active accumulator Acc is always used as the second operand and the implicit destination. Also, Acc may not be used as the source for any of these operations.

```

add   A[ 4]                ; Acc = Acc + A[ 4]
addc  #32h                 ; Acc = Acc + 0032h + Carry

sub   A[ 15]               ; Acc = Acc - A[ 15]
subb  A[ 1]                ; Acc = Acc - A[ 1] - Carry
cmp   #00h                 ; If (Acc == 0000h), set Equals flag

and   A[ 0]                ; Acc = Acc AND A[ 0]
or    #55h                 ; Acc = Acc OR #0055h

xor   A[ 1]                ; Acc = Acc XOR A[ 1]

```

3.5.4 ALU Operations Using Only the Active Accumulator

The following arithmetic and logical operations operate only on the active accumulator.

```

cpl                   ; Acc = NOT Acc
neg                   ; Acc = (NOT Acc) + 1
rl                    ; Rotate accumulator left (not using Carry)
rlc                   ; Rotate accumulator left through Carry
rr                    ; Rotate accumulator right (not using Carry)
rrc                   ; Rotate accumulator right through Carry
sla                   ; Shift accumulator left arithmetically once
sla2                  ; Shift accumulator left arithmetically twice
sla4                  ; Shift accumulator left arithmetically four times
sr                    ; Shift accumulator right, set Carry to Acc.0,
                    ; set Acc.15 to zero (MAXQ20)
sra                   ; Shift accumulator right arithmetically once
sra2                  ; Shift accumulator right arithmetically twice
sra4                  ; Shift accumulator right arithmetically four times
xchn                  ; Swap low and high nibbles of each Acc byte
xch (MAXQ20 only)    ; Swap low byte and high byte of Acc

```

3.5.5 ALU Bit Operations Using Only the Active Accumulator

The following operations operate on single bits of the current active accumulator in conjunction with the Carry flag. Any of these operations may use an Acc bit from 0 to 15.

```

move  C, Acc.0        ; copy bit 0 of accumulator to Carry
move  Acc.5, C        ; copy Carry to bit 5 of accumulator
and   Acc.3           ; Acc.3 = Acc.3 AND Carry
or    Acc.0           ; Acc.0 = Acc.0 OR Carry
xor   Acc.1           ; Acc.1 = Acc.1 OR Carry

```

None of the above bit operations cause the autoincrement, autodecrement, or modulo operations defined by the accumulator pointer control (APC) register.

3.5.6 Example: Adding Two 4-Byte Numbers Using Autoincrement

```

move  A[ 0] , #5678h           ; First number - 12345678h
move  A[ 1] , #1234h
move  A[ 2] , #0AAAAh         ; Second number - 0AAAAAAAh
move  A[ 3] , #0AAAAh
move  APC, #81h               ; Active Acc = A[ 0] , increment low bit = mod 2
add   A[ 2]                    ; A[ 0] = 5678h + AAAAh = 0122h + Carry
addc  A[ 3]                    ; A[ 1] = 1234h + AAAh + 1 = 1CDFh
; 12345678h + 0AAAAAAAh = 1CDF0122h

```

3.6 Processor Status Flag Operations

The Processor Status Flag (PSF) register contains five flags that are used to indicate and store the results of arithmetic and logical operations, four of which can also be used for conditional program branching.

3.6.1 Sign Flag

The Sign flag (PSF.6) reflects the current state of the high bit of the active accumulator (Acc.15 for the MAXQ20). If signed arithmetic is being used, this flag indicates whether the value in the accumulator is positive or negative.

Since the Sign flag is a dynamic reflection of the high bit of the active accumulator, any instruction that changes the value in the active accumulator can potentially change the value of the Sign flag. Also, any instruction that changes which accumulator is the active one (including AP autoincrement/decrement) can also change the Sign flag.

The following operation uses the Sign flag:

- JUMP S, src (Jump if Sign flag is set)

3.6.2 Zero Flag

The Zero flag (PSF.7) is a dynamic flag that reflects the current state of the active accumulator Acc. If all bits in the active accumulator are zero, the Zero flag equals 1. Otherwise, it equals 0.

Since the Zero flag is a dynamic reflection of (Acc = 0), any instruction that changes the value in the active accumulator can potentially change the value of the Zero flag. Also, any instruction that changes which accumulator is the active one (including AP autoincrement/decrement) can also change the Zero flag.

The following operations use the Zero flag:

- JUMP Z, src (Jump if Zero flag is set)
- JUMP NZ, src (Jump if Zero flag is cleared)

3.6.3 Equals Flag

The Equals flag (PSF.0) is a static flag set by the CMP instruction. When the source given to the CMP instruction is equal to the active accumulator, the Equals flag is set to 1. When the source is different from the active accumulator, the Equals flag is cleared to 0.

The following instructions use the value of the Equals flag. Note that the 'src' for the JUMP E/NE instructions must be immediate.

- JUMP E, src (Jump if Equals flag is set)
- JUMP NE, src (Jump if Equals flag is cleared)

In addition to the CMP instruction, any instruction using PSF as the destination can alter the Equals flag.

3.6.4 Carry Flag

The Carry flag (PSF.1) is a static flag indicating that a carry or borrow bit resulted from the last ADD/ADDC or SUB/SUBB operation. Unlike the other status flags, it can be set or cleared explicitly and is also used as a generic bit operand by many other instructions.

The following instructions can alter the Carry flag:

- ADD src (Add source to active accumulator)
- ADDC src (Add source and Carry to active accumulator)
- SUB src (Subtract source from active accumulator)
- SUBB src (Subtract source and Carry from active accumulator)

- SLA, SLA2, SLA4 (Arithmetic shift left active accumulator)
- SRA, SRA2, SRA4 (Arithmetic shift right active accumulator)
- SR (Shift active accumulator right)
- RLC/RRC (Rotate active accumulator left/right through Carry)
- MOVE C, Acc. (Set Carry to selected active accumulator bit)
- MOVE C, #i (Explicitly set, i = 1, or clear, i = 0, the Carry flag)
- CPL C (Complement Carry)
- AND Acc.
- OR Acc.
- XOR Acc.
- MOVE C, src. (Copy bit addressable register bit to Carry)
- any instruction using PSF as the destination

The following instructions use the value of the Carry flag:

- ADDC src (Add source and Carry to active accumulator)
- SUBB src (Subtract source and Carry from active accumulator)
- RLC/RRC (Rotate active accumulator left/right through Carry)
- CPL C (Complement Carry)
- MOVE Acc., C (Set selected active accumulator bit to Carry)
- AND Acc. (Carry = Carry AND selected active accumulator bit)
- OR Acc. (Carry = Carry OR selected active accumulator bit)
- XOR Acc. (Carry = Carry XOR selected active accumulator bit)
- JUMP C, src (Jump if Carry flag is set)
- JUMP NC, src (Jump if Carry flag is cleared)

3.6.5 Overflow Flag

The Overflow flag (PSF.2) is a static flag indicating that the carry or borrow bit (Carry status Flag) resulting from the last ADD/ADDC or SUB/SUBB operation but did not match the carry or borrow of the high order bit of the active accumulator. The overflow flag is useful when performing signed arithmetic operations.

The following instructions can alter the Overflow flag:

- ADD src (Add source to active accumulator)
- ADDC src (Add source and Carry to active accumulator)
- SUB src (Subtract source from active accumulator)
- SUBB src (Subtract source and Carry from active accumulator)

3.7 Controlling Program Flow

The MAXQ7667 provides several options to control program flow and branching. Jumps may be unconditional, conditional, relative, or absolute. Subroutine calls store the return address on the hardware stack for later return. Built-in counters and address registers are provided to control looping operations.

3.7.1 Obtaining the Next Execution Address

The address of the next instruction to be executed can be read at any time by reading the Instruction Pointer (IP) register. This can be particularly useful for initializing loops. Note that the value returned is actually the address of the current instruction plus 1, so this will be the address of the next instruction executed as long as the current instruction does not cause a jump.

3.7.2 Unconditional Jumps

An unconditional jump can be relative (IP +127/-128 words) or absolute (to anywhere in program space). Relative jumps must use an 8-bit immediate operand, such as

```
Label1:                               ; must be within +127/-128 words of the JUMP
...
jump Label1
```

Absolute jumps can use a 16-bit immediate operand, a 16-bit register, or an 8-bit register.

```
jump LongJump                         ; assembles to: move PFX[0], #high(LongJump)
;                                     jump #low(LongJump)
jump DP[0]                             ; absolute jump to the address in DP[0]
```

If an 8-bit register is used as the jump destination, the prefix value is used as the high byte of the address and the register is used as the low byte.

3.7.3 Conditional Jumps

Conditional jumps transfer program execution based on the value of one of the status flags (C, E, Z, S). Except where noted for JUMP E and JUMP NE, the absolute and relative operands allowed are the same as for the unconditional JUMP command.

```
jump c, Label1                         ; jump to Label1 if Carry is set
jump nc, LongJump                      ; jump to LongJump if Carry is not set
jump z, LC[0]                          ; jump to 16-bit register destination if
; Zero is set
jump nz, Label1                        ; jump to Label1 if Zero is not set (Acc<>0)
jump s, A[2]                           ; jump to A[2] if Sign flag is set
jump e, Label1                          ; jump to Label1 if Equal is set
jump ne, Label1                         ; jump to Label1 if Equal is cleared
```

JUMP E and JUMP NE may only use immediate destinations.

3.7.4 Calling Subroutines

The CALL instruction works the same as the unconditional JUMP, except that the next execution address is pushed on the stack before transferring program execution to the branch address. The RET instruction is used to return from a normal call, and RETI is used to return from an interrupt handler routine.

```
call Label1                            ; if Label1 is relative,
; assembles to: call #immediate
call LongCall                           ; assembles to: move PFX[0], #high(LongCall)
;                                     call #low(LongCall)
call LC[0]                               ; call to address in LC[0]
LongCall:
ret                                       ; return from subroutine
```

3.7.5 Looping Operations

Looping over a section of code can be performed by using the conditional jump instructions. However, there is built-in functionality, in the form of the 'DJNZ LC[n], src' instruction, to support faster, more compact looping code with separate loop counters. The 16-bit registers LC[0], and LC[1] are used to store these loop counts. The 'DJNZ LC[n], src' instruction automatically decrements the associated loop counter register and jumps to the loop address specified by src if the loop counter has not reached 0.

To initialize a loop, set the LC[n] register to the count you wish to use before entering the loop's main body.

The desired loop address should be supplied in the src operand of the 'DJNZ LC[n], src' instruction. When the supplied loop address is relative (+127/-128 words) to the DJNZ LC[n] instruction, as is typically the case, the assembler automatically calculates the relative offset and inserts this immediate value in the object code.

```
move LC[1], #10h                       ; loop 16 times
LoopTop:                                ; loop addr relative to djnz LC[n],src instruction
call LoopSub
djnz LC[1], LoopTop                    ; decrement LC[1] and jump if nonzero
```

When the supplied loop address is outside the relative jump range, the prefix register (PFX[0]) is used to supply the high byte of the loop address as required.

```

    move   LC[ 1], #10h           ; loop 16 times
LoopTop:
    call  LoopSub                ; loop addr not relative to djnz LC[ n],src
    ...
    djnz  LC[ 1], LoopTop        ; decrement LC[ 1] and jump if nonzero
                                   ; assembles to: move PFX[ 0], #high(LoopTop)
                                   ;                   djnz LC[ 1], #low(LoopTop)

```

If loop execution speed is critical and a relative jump cannot be used, one might consider preloading an internal 16-bit register with the src loop address for the 'DJNZ LC[n], src' loop. This ensures that the prefix register will not be needed to supply the loop address and always yields the fastest execution of the DJNZ instruction.

```

    move   LC[ 0], #LoopTop       ; using LC[ 0] as address holding register
                                   ; assembles to: move PFX[ 0], #high(LoopTop)
                                   ;                   move LC[ 0], #low(LoopTop)
    move   LC[ 1], #10h           ; loop 16 times
    ...
LoopTop:
    call  LoopSub                ; loop address not relative to djnz LC[ n],src
    ...
    djnz  LC[ 1], LC[ 0]         ; decrement LC[ 1] and jump if nonzero

```

If opting to preload the loop address to an internal 16-bit register, the most time and code efficient means is by performing the load in the instruction just prior to the top of the loop:

```

    move   LC[ 1], #10h           ; Set loop counter to 16
    move   LC[ 0], IP            ; Set loop address to the next address
LoopTop:
    ...                          ; loop addr not relative to djnz LC[ n],src
    ...

```

3.7.6 Conditional Returns

Similar to the conditional jumps, the MAXQ7667 microcontroller also supports a set of conditional return operations. Based upon the value of one of the status flags, the CPU can conditionally pop the stack and begin execution at the address popped from the stack. If the condition is not true, the conditional return instruction does not pop the stack and does not change the instruction pointer. The following conditional return operations are supported:

```

RET C           ; if C=1, a RET is executed
RET NC          ; if C=0, a RET is executed
RET Z           ; if Z=1 (Acc=00h), a RET is executed
RET NZ          ; if Z=0 (Acc<>00h), a RET is executed
RET S           ; if S=1, a RET is executed

```

3.8 Handling Interrupts

Handling interrupts in the MAXQ7667 is a three-part process. First, the location of the interrupt handling routine must be set by writing the address to the 16-bit Interrupt Vector (IV) register. This register defaults to 0000h on reset, but this will usually not be the desired location since this will often be the location of reset/power-up code.

```

    move   IV, IntHandler        ; move PFX[ 0], #high(IntHandler)
                                   ; move IV, #low(IntHandler)
                                   ; PFX[ 0] write not needed if IntHandler addr=00xxh

```

Next, the interrupt must be enabled. For any interrupts to be handled, the IGE bit in the Interrupt and Control register (IC) must first be set to 1. Next, the interrupt itself must be enabled at the module level and locally within the module itself. The module interrupt enable is located in the Interrupt Mask register, while the location of the local interrupt enable will vary depending on the module in which the interrupt source is located.

Once the interrupt handler receives the interrupt, the Interrupt in Service (INS) bit will be set by hardware to block further interrupts, and execution control is transferred to the interrupt service routine. Within the interrupt service routine, the source of the interrupt must be determined. Since all interrupts go to the same interrupt service routine, the Interrupt Identification Register (IIR) must be examined to determine which module initiated the interrupt. For example, the IIO (IIR.0) bit will be set if there is a pending interrupt from module 0. These bits cannot be cleared directly; instead, the appropriate bit flag in the module must be cleared once the interrupt is handled.

INS is set automatically on entry to the interrupt handler and cleared automatically on exit (RETI).

```

IntHandler:
    push    PSF                ; save C since used in identification process
    move    C, IIR.X           ; check highest priority flag in IIR
    jump    C, ISR_X           ; if IIR.X is set, interrupt from module X
    move    C, IIR.Y           ; check next highest priority int source
    jump    C, ISR_Y           ; if IIR.Y is set, interrupt from module Y
    ...

ISR_X:
    ...
    reti
    
```

To support high priority interrupts while servicing another interrupt source, the IMR register may be used to create a user-defined prioritization. The IMR mask register should not be utilized when the highest priority interrupt is being serviced because the highest priority interrupt should never be interrupted. This is default condition when a hardware branch is made the Interrupt Vector address (INS is set to 1 by hardware and all other interrupt sources are blocked). The code below demonstrates how to use IMR to allow other interrupts.

```

ISR_Z:
    pop     PSF                ; restore PSF
    push    IMR                ; save current interrupt mask
    move    IMR, #int_mask     ; new mask to allow only higher priority ints
    move    INS, #0            ; re-enable interrupts
    ...
    (interrupt servicing code)
    ...
    pop     IMR                ; restore previous interrupt mask
    ret                       ; back to code or lower priority interrupt
    
```

Please note that configuring a given IMR register mask bit to '0' only prevents interrupt conditions from the corresponding module or system from generating an interrupt request. Configuring an IMR mask bit to '0' does not prevent the corresponding IIR system or module identification flag from being set. This means that when using the IMR mask register functionality to block interrupts, there may be cases when both the mask (IMR.x) and identifier (IIR.x) bits should be considered when determining if the corresponding peripheral should be serviced.

3.8.1 Conditional Return from Interrupt

Similar to the conditional returns, the MAXQ7667 microcontroller also supports a set of conditional return from interrupt operations. Based upon the value of one of the status flags, the CPU can conditionally pop the stack, clear the INS bit to 0, and begin execution at the address popped from the stack. If the condition is not true, the conditional return from interrupt instruction leaves the INS bit unchanged, does not pop the stack and does not change the instruction pointer. The following conditional return from interrupt operations are supported:

```

RETI C           ; if C=1, a RETI is executed
RETI NC          ; if C=0, a RETI is executed
RETI Z           ; if Z=1 (Acc=00h), a RETI is executed
RETI NZ          ; if Z=0 (Acc<>00h), a RETI is executed
RETI S           ; if S=1, a RETI is executed
    
```

3.9 Accessing the Stack

The hardware stack is used automatically by the CALL, RET and RETI instructions, but it can also be used explicitly to store and retrieve data. All values stored on the stack are 16 bits wide.

The PUSH instruction increments the stack pointer SP and then stores a value on the stack. When pushing a 16-bit value onto the stack, the entire value is stored. However, when pushing an 8-bit value onto the stack, the high byte stored on the stack comes from the prefix register. The @++SP stack access mnemonic is the associated destination specifier that generates this push behavior, thus the following two instruction sequences are equivalent:

```

move  PFX[ 0] , IC
push  PSF                ; stored on stack: IC:PSF

move  PFX[ 0] , IC
move  @++SP, PSF        ; stored on stack: IC:PSF

```

The POP instruction removes a value from the stack and then decrements the stack pointer. The @SP-- stack access mnemonic is the associated source specifier that generates this behavior, thus the following two instructions are equivalent:

```

pop   PSF
move  PSF, @SP--

```

The POPI instruction is equivalent to the POP instruction but additionally clears the INS bit to 0. Thus, the following two instructions would be equivalent:

```

popi  IP
reti

```

The @SP-- mnemonic can be used by the MAXQ microcontroller so that stack values may be used directly by ALU operations (e.g. ADD src, XOR src, etc.) without requiring that the value be first popped into an intermediate register or accumulator.

```

add  @SP--                ; sum the last three words pushed onto the stack
add  @SP--                ; with Acc, disregarding overflow
add  @SP--

```

The stack pointer SP can be set explicitly, however only those least significant bits needed to represent the stack depth for the associated MAXQ device are used. For a MAXQ device that has a stack depth of 16 words, only the lowest four bits are used and setting SP to 0Fh will return it to its reset state.

Since the stack is 16 bits wide, it is possible to store two 8-bit register values on it in a single location. This allows more efficient use of the stack if it is being used to save and restore registers at the start and end of a subroutine.

SubOne:

```

move  PFX[ 0] , IC
push  PSF                ; store IC:PSF on the stack
...
pop   GR                ; 16-bit register
move  IC, GRH           ; IC was stored as high byte
move  PSF, GRL         ; PSF was stored as low byte
ret

```

3.10 Accessing Data Memory

Data memory is accessed through the data pointer registers DP[0] and DP[1] or the Frame Pointer BP[OFFS]. Once one of these registers is set to a location in data memory, that location can be read or written as follows, using the mnemonic @DP[0], @DP[1] or @BP[OFFS] as a source or destination.

```

move DP[ 0] , #0000h      ; set pointer to location 0000h
move A[ 0] , @DP[ 0]     ; read from data memory
move @DP[ 0] , #55h      ; write to data memory

```

Either of the data pointers may be post-incremented or post-decremented following any read or may be preincremented or predecremented before any write access by using the following syntax.

```

move A[ 0] , @DP[ 0] ++  ; increment DP[ 0] after read
move @++DP[ 0] , A[ 1]   ; increment DP[ 0] before write
move A[ 5] , @DP[ 1] --  ; decrement DP[ 1] after read
move @--DP[ 1] , #00h    ; decrement DP[ 1] before write

```

The Frame Pointer (BP[OFFS]) is actually composed of a base pointer (BP) and an offset from the base pointer (OFFS). For the frame pointer, the offset register (OFFS) is the target of any increment or decrement operation. The base pointer (BP) is unaffected by increment and decrement operations on the Frame Pointer. Similar to DP[n], the OFFS register may be preincremented/decremented when writing to data memory and may be postincremented/decremented when reading from data memory.

```

move A[ 0] , @BP[ OFFS--] ; decrement OFFS after read
move @BP[ ++OFFS] , A[ 1] ; increment OFFS before write

```

All three data pointers support both byte and word access to data memory. Each data pointer has its own word/byte select (WBSn) special-function register bit to control the access mode associated with the data pointer. These three register bits (WBS2, which controls BP[OFFS] access; WBS1, which controls DP[1] access; and WBS0, which controls DP[0] access) reside in the Data Pointer Control (DPC) register. When a given WBSn control bit is configured to 1, the associated pointer is operated in the word access mode. When the WBSn bit is configured to 0, the pointer is operated in the byte access mode. Word access mode allows addressing of 64KWords of memory while byte access mode allows addressing of 64KB of memory.

Each data pointer (DP[n]) and Frame Pointer base (BP) register is actually implemented internally as a 17-bit register (e.g., 16:0). The Frame Pointer offset register (OFFS) is implemented internally as a 9-bit register (e.g., 8:0). The WBSn bit for the respective pointer controls whether the highest 16 bits (16:1) of the pointer are in use, as is the case for word mode (WBSn = 1) or whether the lowest 16 bits (15:0) are in use, as will be the case for byte mode (WBSn = 0). The WBS2 bit also controls whether the high 8 bits (8:1) of the offset register are in use (WBS2 = 1) or the low 8 bits (7:0) are used (WBS2 = 0). All data pointer register reads, writes, autoincrement/decrement operations occur with respect to the current WBSn selection. Data pointer increment and decrement operations only affect those bits specific to the current word or byte addressing mode (e.g., incrementing a byte mode data pointer from FFFFh does not carry into the internal high order bit that is utilized only for word mode data pointer access). Switching from byte to word access mode or vice versa does not alter the data pointer contents. Therefore, it is important to maintain the consistency of data pointer address value within the given access mode.

```

move DPC, #0              ; DP[ 0] in byte mode
move DP[ 0] , #2345h      ; DP[ 0]=2345h (byte mode)
                          ; internal bits 15:0 loaded

move DPC, #4              ; DP[ 0] in word mode
move DP[ 0] , #2345h      ; DP[ 0]=2345h (word mode)
                          ; internal bits 16:1 loaded

move DPC, #0              ; DP[ 0] in byte mode
move GR, DP[ 0]           ; GR = 468Bh (looking at bits 15:0)

```

The three pointers share a single read/write port on the data memory and thus, the user must knowingly activate a desired pointer before using it for data memory read operations. This can be done explicitly using the data pointer select bits (SDPS[1:0]; DPC.[1:0]), or implicitly by writing to the DP[n], BP, or OFFS registers as shown below. Any indirect memory write operation using a data pointer will set the SDPS bits, thus activating the write pointer as the active source pointer.

```

move DPC, #2              ; (explicit) selection of FP as the pointer
move DP[ 1] , DP[ 1]      ; (implicit) selection of DP[ 1]; set SDPS1:0=01b
move OFFS, src            ; (implicit) selection of FP; set SDPS1=1
move @DP[ 0] , src        ; (implicit) selection of DP[ 0]; set SDPS1:0=00b

```

Once the pointer selection has been made, it will remain in effect until:

- the source data pointer select bits are changed via the explicit or implicit methods described above (i.e., another data pointer is selected for use)
- the memory to which the active source data pointer is addressing is enabled for code fetching using the Instruction Pointer, or
- a memory write operation is performed using a data pointer other than the current active source pointer.

```

move  DP[ 1], DP[ 1]      ; select DP[ 1] as the active pointer
move  dst, @DP[ 1]       ; read from pointer
move  @DP[ 1], src       ; write using a data pointer
                                ; DP[ 0] is needed
move  DP[ 0], DP[ 0]     ; select DP[ 0] as the active pointer

```

To simplify data pointer increment/decrement operations without disturbing register data, a virtual NUL destination has been assigned to system module 6, subindex 7 to serve as a bit bucket. Data pointer increment/decrement operations can be done as follows without altering the contents of any other register:

```

move  NUL, @DP[ 0] ++    ; increment DP[ 0]
move  NUL, @DP[ 0] --    ; decrement DP[ 0]

```

The following data pointer related instructions are invalid:

```

move @++DP[ 0], @DP[ 0] ++
move @++DP[ 1], @DP[ 1] ++
move @BP[ ++Offs], @BP[ Offs++]
move @--DP[ 0], @DP[ 0] --
move @--DP[ 1], @DP[ 1] --
move @BP[ --Offs], @BP[ Offs--]
move @++DP[ 0], @DP[ 0] --
move @++DP[ 1], @DP[ 1] --
move @BP[ ++Offs], @BP[ Offs--]
move @--DP[ 0], @DP[ 0] ++
move @--DP[ 1], @DP[ 1] ++
move @BP[ --Offs], @BP[ Offs++]
move @DP[ 0], @DP[ 0] ++
move @DP[ 1], @DP[ 1] ++
move @BP[ Offs], @BP[ Offs++]
move @DP[ 0], @DP[ 0] --
move @DP[ 1], @DP[ 1] --
move @BP[ Offs], @BP[ Offs--]
move DP[ 0], @DP[ 0] ++
move DP[ 0], @DP[ 0] --
move DP[ 1], @DP[ 1] ++
move DP[ 1], @DP[ 1] --
move Offs, @BP[ Offs--]
move Offs, @BP[ Offs++]

```

SECTION 4: REGISTER DESCRIPTIONS

This section contains the following information:

4.1 System Register Descriptions	4-3
4.1.1 Accumulator Pointer Register (AP)	4-6
4.1.2 Accumulator Pointer Control Register (APC)	4-6
4.1.3 Processor Status Flags Register (PSF)	4-7
4.1.4 Interrupt and Control Register (IC)	4-8
4.1.5 Interrupt Mask Register (IMR)	4-8
4.1.6 System Control Register (SC)	4-9
4.1.7 Interrupt Identification Register (IIR)	4-10
4.1.8 System Clock Control Register (CKCN)	4-10
4.1.9 Watchdog Timer Control Register (WDCN)	4-11
4.1.10 Accumulator n Register (A[n])	4-11
4.1.11 Prefix Register (PFX[n])	4-12
4.1.12 Instruction Pointer Register (IP)	4-13
4.1.13 Stack Pointer Register (SP)	4-13
4.1.14 Interrupt Vector Register (IV)	4-14
4.1.15 Loop Counter 0 Register (LC[0])	4-14
4.1.16 Loop Counter 1 Register (LC[1])	4-15
4.1.17 Frame Pointer Offset Register (OFFS)	4-15
4.1.18 Data Pointer Control Register (DPC)	4-16
4.1.19 General Register (GR)	4-17
4.1.20 General Register Low Byte (GRL)	4-17
4.1.21 Frame Pointer Base Register (BP)	4-18
4.1.22 General Register Byte-Swapped (GRS)	4-18
4.1.23 General Register High Byte (GRH)	4-19
4.1.24 General Register Sign Extended Low Byte (GRXL)	4-19
4.1.25 Frame Pointer Register (FP)	4-20
4.1.26 Data Pointer 0 Register (DP[0])	4-20
4.1.27 Data Pointer 1 Register (DP[1])	4-21
4.2 Peripheral Register Modules	4-22

LIST OF TABLES

Table 4-1. MAXQ7667 System Register Map	4-3
Table 4-2. MAXQ7667 System Register Bit Functions and Reset Value	4-4
Table 4-3. MAXQ7667 Peripheral Register Map	4-22
Table 4-4. MAXQ7667 Module 0 Register Bit Functions and Reset Values	4-23
Table 4-5. MAXQ7667 Module 1 Register Bit Functions and Reset Values	4-24
Table 4-6. MAXQ7667 Module 2 Register Bit Functions and Reset Values	4-26
Table 4-7. MAXQ7667 Module 3 Register Bit Functions and Reset Values	4-28
Table 4-8. MAXQ7667 Module 5 Register Bit Functions and Reset Values	4-30

SECTION 4: REGISTER DESCRIPTIONS

4.1 System Register Descriptions

The MAXQ7667 system register map is shown in Table 4-1. The system register bit functions and reset value are shown in Table 4-2. Those registers defined in the MAXQ7667 system register map are described in the following sections. The address for each register are given in the format *module[index]*, where *module* is the module specifier from 8h to Fh and *index* is the register subindex from 0h to Fh.

Table 4-1. MAXQ7667 System Register Map

CYCLES TO READ	CYCLES TO WRITE	REGISTER INDEX	MODULE NAME (BASE SPECIFIER)						
			AP (8h)	A (9h)	PFX (Bh)	IP (Ch)	SP (Dh)	DPC (Eh)	DP (Fh)
1	1	0h	AP	A[0]	PFX[0]	IP			
1	1	1h	APC	A[1]	PFX[1]		SP		
1	1	2h		A[2]	PFX[2]		IV		
1	1	3h		A[3]	PFX[3]			OFFS	DP[0]
1	1	4h	PSF	A[4]	PFX[4]			DPC	
1	1	5h	IC	A[5]	PFX[5]			GR	
1	1	6h	IMR	A[6]	PFX[6]		LC[0]	GRL	
1	1	7h		A[7]	PFX[7]		LC[1]	BP	DP[1]
1	2	8h	SC	A[8]				GRS	
1	2	9h		A[9]				GRH	
1	2	Ah		A[10]				GRXL	
1	2	Bh	<i>IIR</i>	A[11]				FP	
1	2	Ch		A[12]					
1	2	Dh		A[13]					
1	2	Eh	CKCN	A[14]					
1	2	Fh	WDCN	A[15]					

Note: Names that appear in italics indicate that all bits of a register are read-only. Names that appear in bold indicate that a register is 16 bits wide.

Table 4-2. MAXQ7667 System Register Bit Functions and Reset Value

REGISTER	REGISTER BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AP 08h[00h]									—	—	—	—	AP3	AP2	AP1	AP0
APC 08h[01h]									CLR	IDS	—	—	—	MOD2	MOD1	MOD0
PSF 08h[04h]									Z	S	—	GPF1	GPF0	OV	C	E
IC 08h[05h]									—	—	CGDS	—	—	—	INS	IGE
IMR 08h[06h]									IMS	—	IM5	IM4	IM3	IM2	IM1	IM0
SC 08h[08h]									TAP	—	CDA1	CDA0	UPA	ROD	PWL	—
IIR 08h[09h]									1	0	0	0	0	0	s	0
CKCN 08h[0Eh]									IIS	—	IIS	IIS	IIS	IIS	IIS	IIS
WDCN 08h[0Fh]									0	0	0	0	0	0	0	0
A[n] (0...15) 09h[0nh]	A[n]15	A[n]14	A[n]13	A[n]12	A[n]11	A[n]10	A[n]9	A[n]8	A[n]7	A[n]6	A[n]5	A[n]4	A[n]3	A[n]2	A[n]1	A[n]0
PFX[n] (0...7) 0Bh[0nh]	PFX[n]15	PFX[n]14	PFX[n]13	PFX[n]12	PFX[n]11	PFX[n]10	PFX[n]9	PFX[n]8	PFX[n]7	PFX[n]6	PFX[n]5	PFX[n]4	PFX[n]3	PFX[n]2	PFX[n]1	PFX[n]0
IP	IP15	IP14	IP13	IP12	IP11	IP10	IP9	IP8	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0
0Ch[00h]	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SP	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0Dh[01h]	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
IV	IV15	IV14	IV13	IV12	IV11	IV10	IV9	IV8	IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0
0Dh[02h]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LC[0]	LC[0]15	LC[0]14	LC[0]13	LC[0]12	LC[0]11	LC[0]10	LC[0]9	LC[0]8	LC[0]7	LC[0]6	LC[0]5	LC[0]4	LC[0]3	LC[0]2	LC[0]1	LC[0]0
0Dh[06h]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LC[1]	LC[1]15	LC[1]14	LC[1]13	LC[1]12	LC[1]11	LC[1]10	LC[1]9	LC[1]8	LC[1]7	LC[1]6	LC[1]5	LC[1]4	LC[1]3	LC[1]2	LC[1]1	LC[1]0
0Dh[07h]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OFFS	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0Eh[03h]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
DPC	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0Eh[04h]	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

s = Bit affected only by power-on reset and not by other forms of reset. See the register description for more information.

Table 4-2. MAXQ7667 System Register Bit Functions and Reset Value (continued)

REGISTER	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GR 0Eh[05h]	GR15	GR14	GR13	GR12	GR11	GR10	GR9	GR8	GR7	GR6	GR5	GR4	GR3	GR2	GR1	GR0
GRL 0Eh[06h]									GRL7	GRL6	GRL5	GRL4	GRL3	GRL2	GRL1	GRL0
BP 0Eh[07h]	BP15	BP14	BP13	BP12	BP11	BP10	BP9	BP8	BP7	BP6	BP5	BP4	BP3	BP2	BP1	BP0
GRS 0Eh[08h]	GRS15	GRS14	GRS13	GRS12	GRS11	GRS10	GRS9	GRS8	GRS7	GRS6	GRS5	GRS4	GRS3	GRS2	GRS1	GRS0
GRH 0Eh[09h]									GRH7	GRH6	GRH5	GRH4	GRH3	GRH2	GRH1	GRH0
GRXL 0Eh[0Ah]	GRXL15	GRXL14	GRXL13	GRXL12	GRXL11	GRXL10	GRXL9	GRXL8	GRXL7	GRXL6	GRXL5	GRXL4	GRXL3	GRXL2	GRXL1	GRXL0
FP 0Eh[0Bh]	FP15	FP14	FP13	FP12	FP11	FP10	FP9	FP8	FP7	FP6	FP5	FP4	FP3	FP2	FP1	FP0
DP[0] 0Fh[03h]	DP[0]15	DP[0]14	DP[0]13	DP[0]12	DP[0]11	DP[0]10	DP[0]9	DP[0]8	DP[0]7	DP[0]6	DP[0]5	DP[0]4	DP[0]3	DP[0]2	DP[0]1	DP[0]0
DP[1] 0Fh[07h]	DP[1]15	DP[1]14	DP[1]13	DP[1]12	DP[1]11	DP[1]10	DP[1]9	DP[1]8	DP[1]7	DP[1]6	DP[1]5	DP[1]4	DP[1]3	DP[1]2	DP[1]1	DP[1]0

s = Bit affected only by power-on reset and not by other forms of reset. See the register description for more information.

4.1.1 Accumulator Pointer Register (AP)

Register Description: **Accumulator Pointer Register**
 Register Name: **AP**
 Register Address: **Module 08h, Index 00h**

Bit #	7	6	5	4	3	2	1	0
Name	—	—	—	—	AP3	AP2	AP1	AP0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 7 to 4: Reserved. Read 0, write ignored.

Bits 3 to 0: Accumulator Select 3:0 (AP[3:0]). These bits select which of the 16 accumulator registers are used for arithmetic and logical operations. If the APC register has been set to perform automatic increment/decrement of the active accumulator, this setting will be automatically changed after each arithmetic or logical operation. If a MOVE AP, Acc instruction is executed, any enabled AP inc/dec/modulo control will take precedence over the transfer of Acc data into AP.

4.1.2 Accumulator Pointer Control Register (APC)

Register Description: **Accumulator Pointer Control Register**
 Register Name: **APC**
 Register Address: **Module 08h, Index 01h**

Bit #	7	6	5	4	3	2	1	0
Name	CLR	IDS	—	—	—	MOD2	MOD1	MOD0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	r	r	r	rw	rw	rw

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bit 7: Accumulator Pointer Clear (CLR). Writing this bit to 1 clears the accumulator pointer AP to 0. Once set, this bit will automatically be reset to 0 by hardware. If a MOVE APC, Acc instruction is executed requesting that AP be set to 0 (i.e., CLR = 1), the AP clear function overrides any enabled inc/dec/modulo control. All reads from this bit return 0.

Bit 6: Accumulator Pointer Increment/Decrement Select (IDS). If this bit is set to 0, the accumulator pointer AP is incremented following each arithmetic or logical operation according to MOD[2:0]. If this bit is set to 1, the accumulator pointer AP is decremented following each arithmetic or logical operation according to MOD[2:0]. If MOD[2:0] is set to 000, the setting of this bit is ignored.

Bits 5 to 3: Reserved. Read 0, write ignored.

Bits 2 to 0: Accumulator Pointer Autoincrement/Decrement Modulus (MOD[2:0]). If these bits are set to a nonzero value, the accumulator pointer (AP[3:0]) will be automatically incremented or decremented following each arithmetic or logical operation. The mode for the autoincrement/decrement is determined as follows:

MOD[2:0]	AUTOINCREMENT/DECREMENT MODE
000	No autoincrement/decrement (default).
001	Increment/decrement AP[0] modulo 2.
010	Increment/decrement AP[1:0] modulo 4.
011	Increment/decrement AP[2:0] modulo 8.
100	Increment/decrement AP modulo 16.
101 to 111	Reserved (modulo 16 when set).

4.1.3 Processor Status Flags Register (PSF)

Register Description: **Processor Status Flags Register**
 Register Name: **PSF**
 Register Address: **Module 08h, Index 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	r	rw	rw

r = read, w = write

Note: This register is cleared to 80h on all forms of reset.

Bit 7: Zero Flag (Z). The value of this bit flag equals 1 whenever the active accumulator is equal to zero, and it equals 0 otherwise.

Bit 6: Sign Flag (S). This bit flag mirrors the current value of the high bit of the active accumulator (Acc.15).

Bit 5: Reserved. Read 0, write ignored.

Bits 4 and 3: General-Purpose Software Flag 1 and 0 (GPF[1:0]). These general-purpose register bits are provided for user software control.

Bit 2: Overflow Flag (OV). This flag is set to 1 if there is a carry out of bit 14 but not out of bit 15, or a carry out of bit 15 but not out of bit 14 from the last arithmetic operation, otherwise, the OV flag remains as 0. OV indicates a negative number resulted as the sum of two positive operands, or a positive sum resulted from two negative operands.

Bit 1: Carry Flag (C). This bit flag is set to 1 whenever an add or subtract operation (ADD, ADDC, SUB, SUBB) returns a carry or borrow. This bit flag is cleared to 0 whenever an add or subtract operation does not return a carry or borrow. Many other instructions potentially affect the carry bit. See *Section 19: Instruction Set Summary* for details.

Bit 0: Equals Flag (E). This bit flag is set to 1 whenever a compare operation (CMP) returns an equal result. If a CMP operation returns not equal, this bit is cleared.

4.1.4 Interrupt and Control Register (IC)

Register Description: **Interrupt and Control Register**
 Register Name: **IC**
 Register Address: **Module 08h, Index 05h**

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

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 7, 6, 4, 3, and 2: Reserved. Read 0, write ignored.

Bit 5: System Clock Gating Disable (CGDS). If this bit is set to 0 (default mode), system clock gating circuitry is active. If this bit is set to 1, the clock gating circuitry is disabled.

Bit 1: Interrupt In Service (INS). The INS is set by hardware automatically when an interrupt is acknowledged. No further interrupts occur as long as the INS remains set. The interrupt service routine can clear the INS bit to allow interrupt nesting. Otherwise, the INS bit is cleared by hardware upon execution of an RETI or POPI instruction.

Bit 0: Interrupt Global Enable (IGE). If this bit is set to 1, interrupts are globally enabled, but still must be locally enabled to occur. If this bit is set to 0, all interrupts are disabled.

4.1.5 Interrupt Mask Register (IMR)

The first six bits in this register are interrupt mask bits for modules 0 to 5, 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 (for the case of IMS) to generate interrupt requests. Clearing the mask bit effectively disables all interrupt sources associated with that specific module or all system interrupt sources (for the case of IMS). The interrupt mask register is intended to facilitate user-definable interrupt prioritization.

Register Description: **Interrupt Mask Register**
 Register Name: **IMR**
 Register Address: **Module 08h, Index 06h**

Bit #	7	6	5	4	3	2	1	0
Name	IMS	—	IM5	IM4	IM3	IM2	IM1	IM0
Reset	0	0	0	0	0	0	0	0
Access	rw	r	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bit 7: Interrupt Mask for System Modules (IMS)

Bit 6: Reserved. Read 0, write ignored.

Bits 5 to 0: Interrupt Mask for Register Module 5:0 (IM[5:0])

4.1.6 System Control Register (SC)

Register Description: **System Control Register**
 Register Name: **SC**
 Register Address: **Module 08h, Index 08h**

Bit #	7	6	5	4	3	2	1	0
Name	TAP	—	CDA1	CDA0	UPA	ROD	PWL	—
Reset	1	0	0	0	0	0	1	0
Access	rw	r	rw	rw	rw	rw	rw	r

r = read, w = write

Note: Bit 1 (PWL) is set to 1 on a power-on reset only.

Bit 7: Test Access (JTAG) Port Enable (TAP). This bit controls whether the Test Access Port special-function pins are enabled. The TAP defaults to being enabled. Clearing this bit to 0 disables the TAP special function pins. See *Section 11* for more information about JTAG and TAP.

Bits 6 and 0: Reserved. Read 0, write ignored.

Bits 5 and 4: Code Data Access Bits 1 and 0 (CDA[1:0]). The CDA bits are used to logically map physical program memory page to the data space for read/write access (see table below).

The logical data memory addresses of the program pages depend on whether execution is from Utility ROM or logical data memory. Note that CDA1 is not implemented if the upper 32k of the program space is not used for the user code. No CDA bits are needed if only one page of program space is incorporated. (P0 is the only memory available in the MAXQ7667.)

CDA[1:0]	BYTE MODE ACTIVE PAGE	WORD MODE ACTIVE PAGE
00	P0	P0 and P1
01	P1	P0 and P1
10	P2	P2 and P3
11	P3	P2 and P3

Bit 3: Upper Program Access (UPA). The physical program memory is logically divided into four pages; P0 and P1 occupy the lower 32KWords while P2 and P3 occupy the upper 32KWords. P0 and P1 are assigned to the lower half of the program space and are always active. P2 and P3 must be explicitly activated in the upper half of the program space by setting the UPA bit to 1. When UPA bit is cleared to 0, the upper program memory space is occupied by the utility ROM and the logical data memory, which is accessible as program memory. Note that the UPA is not implemented if the upper 32K of the program space is not used for the user code.

Bit 2: ROM Operation Done (ROD). This bit is used to signify completion of a ROM operation sequence to the control units. This allows the debug engine to determine the status of a ROM sequence. Setting this bit to logic 1 causes an internal system reset if the JTAG SPE bit is also set. Setting the ROD bit will clear the JTAG SPE bit if it is set and the ROD bit will be automatically cleared by hardware once the control unit acknowledges the done indication. See *Section 12* for more information.

Bit 1: Password Lock (PWL). This bit defaults to 1 on a power-on reset. When this bit is 1, it requires a 32-byte password to be matched with the password in the program space before allowing access to the password protected in-circuit debug or bootstrap loader ROM routines. Clearing this bit to 0 disables the password protection for these ROM routines. See *Section 13* for more information.

4.1.7 Interrupt Identification Register (IIR)

The first six bits in this register indicate interrupts pending in modules 0 to 5, one bit per module. The eighth bit, IIS, indicates a pending system interrupt, such as from the watchdog timer. The interrupt pending flags will be set only for enabled interrupt sources waiting for service. The interrupt pending flag will be cleared when the pending interrupt sources within that module are disabled or when the interrupt flags are cleared by software.

Register Description: **Interrupt Identification Register**
 Register Name: **IIR**
 Register Address: **Module 08h, Index 0Bh**

Bit #	7	6	5	4	3	2	1	0
Name	IIS	—	I15	I14	I13	I12	I11	I10
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: This register is cleared to 00h on all forms of reset.

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

Bit 6: Reserved. Read 0, write ignored.

Bits 5 to 0: Interrupt Identifier Flag for Register Module 5 to 0 (I1[5:0])

4.1.8 System Clock Control Register (CKCN)

The 8-bit CKCN register is part of the system register group and used to support system clock generation. It controls the system clock speed and power management mode selection. See *Section 5* for the description of this register.

Register Description: **System Clock Control Register**
 Register Name: **CKCN**
 Register Address: **Module 08h, Index 0Eh**

Bit #	7	6	5	4	3	2	1	0
Name	XTRC	—	RCMD	STOP	SWB	PMME	CD1	CD0
Reset	0	0	0	0	0	0	0	0
Access	rw	r	r	rw	rw	rw	rw	rw

r = read, w = write

Note: See bit descriptions in *Section 15*.

4.1.9 Watchdog Timer Control Register (WDCN)

The 8-bit WDCN register is part of the system register group and used to provide system control. It controls the watchdog timeout period and interrupt or reset generation on watchdog timeout. The watchdog timer is clocked by the internal RC oscillator. See *Section 15* for a description of this register.

Register Description: **Watchdog Timer Control Register**
 Register Name: **WDCN**
 Register Address: **Module 08h, Index 0Fh**

Bit #	7	6	5	4	3	2	1	0
Name	POR	EWDI	WD1	WD0	WDIF	WTRF	EWT	RWT
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: Bits 5, 4, 3, and 0 are cleared to 0 on all forms of reset; for others, see the individual bit descriptions.

4.1.10 Accumulator n Register (A[n])

Register Description: **Accumulator n Register**
 Register Name: **A[n]**
 Register Address: **Module 09h, Index 0nh**

Bit #	15	14	13	12	11	10	9	8
Name	A[n]15	A[n]14	A[n]13	A[n]12	A[n]11	A[n]10	A[n]9	A[n]8
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	A[n]7	A[n]6	A[n]5	A[n]4	A[n]3	A[n]2	A[n]1	A[n]0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Accumulator n Register Bits 15:0 (A[n][15:0]). This register acts as the accumulator for all ALU arithmetic and logical operations when selected by the accumulator pointer (AP). It can also be used as a general-purpose working register.

4.1.11 Prefix Register (PFX[n])

Register Description: **Prefix Register**
 Register Name: **PFX[n]**
 Register Address: **Module 0Bh, Index 0nh**

Bit #	15	14	13	12	11	10	9	8
Name	PFX[n]15	PFX[n]14	PFX[n]13	PFX[n]12	PFX[n]11	PFX[n]10	PFX[n]9	PFX[n]8
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	PFX[n]7	PFX[n]6	PFX[n]5	PFX[n]4	PFX[n]3	PFX[n]2	PFX[n]1	PFX[n]0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Prefix Register Bits 15:0 (PFX[n][15:0]). The prefix register provides a means of supplying an additional 8 bits of high-order data for use by the succeeding instruction as well as providing additional indexing capabilities. This register will only hold any data written to it for one execution cycle, after which it will revert to 0000h. Although this is a 16-bit register, only the lower 8 bits are actually used for prefixing purposes by the next instruction. Writing to or reading from any index in the Prefix module will select the same 16-bit register. However, when the prefix register is written, the index n used for the PFX[n] write also determines the high-order bits for the register source and destination specified in the following instruction.

The index selection reverts to 0 (default mode allowing selection of registers 0h to 7h for destinations) after one cycle in the same manner as the contents of the prefix register.

WRITE TO	SOURCE, DESTINATION INDEX SELECTION	
	SOURCE REGISTER RANGE	DESTINATION REGISTER RANGE
PFX[0]	0h to Fh	0h to 7h
PFX[1]	10h to 1Fh	0h to 7h
PFX[2]	0h to Fh	8h to Fh
PFX[3]	10h to 1Fh	8h to Fh
PFX[4]	0h to Fh	10h to 17h
PFX[5]	10h to 1Fh	10h to 17h
PFX[6]	0h to Fh	18h to 1Fh
PFX[7]	10h to 1Fh	18h to 1Fh

4.1.12 Instruction Pointer Register (IP)

Register Description: **Instruction Pointer Register**
 Register Name: **IP**
 Register Address: **Module 0Ch, Index 00h**

Bit #	15	14	13	12	11	10	9	8
Name	IP15	IP14	IP13	IP12	IP11	IP10	IP9	IP8
Reset	1	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	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 8000h on all forms of reset.

Bits 15 to 0: Instruction Pointer Register Bits 15:0 (IP[15:0]). This register contains the address of the next instruction to be executed and is automatically incremented by 1 after each program fetch. Writing an address value to this register will cause program flow to jump to that address. Reading from this register will not affect program flow.

4.1.13 Stack Pointer Register (SP)

Register Description: **Stack Pointer Register**
 Register Name: **SP**
 Register Address: **Module 0Dh, Index 01h**

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	—	—	—	—	SP3	SP2	SP1	SP0
Reset	0	0	0	0	1	1	1	1
Access	r	r	r	r	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 000Fh on all forms of reset.

Bits 15 to 4: Reserved. Read 0, write ignored.

Bits 3 to 0: Stack Pointer Register Bits 3 to 0 (SP[3:0]). These four bits indicate the current top of the hardware stack, from 0h to Fh. This pointer is incremented after a value is pushed on the stack and decremented before a value is popped from the stack.

4.1.14 Interrupt Vector Register (IV)

Register Description: **Interrupt Vector Register**
 Register Name: **IV**
 Register Address: **Module 0Dh, Index 02h**

Bit #	15	14	13	12	11	10	9	8
Name	IV15	IV14	IV13	IV12	IV11	IV10	IV9	IV8
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	IV7	IV6	IV5	IV4	IV3	IV2	IV1	IV0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Interrupt Vector Register Bits 15:0 (IV[15:0]). This register contains the address of the interrupt service routine. The interrupt handler will generate a CALL to this address whenever an interrupt is acknowledged.

4.1.15 Loop Counter 0 Register (LC[0])

Register Description: **Loop Counter 0 Register**
 Register Name: **LC[0]**
 Register Address: **Module 0Dh, Index 06h**

Bit #	15	14	13	12	11	10	9	8
Name	LC[0]15	LC[0]14	LC[0]13	LC[0]12	LC[0]11	LC[0]10	LC[0]9	LC[0]8
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	LC[0]7	LC[0]6	LC[0]5	LC[0]4	LC[0]3	LC[0]2	LC[0]1	LC[0]0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Loop Counter 0 Register Bits 15:0 (LC[0][15:0]). This register is used as the loop counter for the DJNZ LC[0], src operation. This operation decrements LC[0] by one and then jumps to the address specified in the instruction by src.

4.1.16 Loop Counter 1 Register (LC[1])

Register Description: **Loop Counter 1 Register**
 Register Name: **LC[1]**
 Register Address: **Module 0Dh, Index 07h**

Bit #	15	14	13	12	11	10	9	8
Name	LC[1]15	LC[1]14	LC[1]13	LC[1]12	LC[1]11	LC[1]10	LC[1]9	LC[1]8
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	LC[1]7	LC[1]6	LC[1]5	LC[1]4	LC[1]3	LC[1]2	LC[1]1	LC[1]0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Loop Counter 1 Register Bits 15:0 (LC[1][15:0]). This register is used as the loop counter for the DJNZ LC[1], src operation. This operation decrements LC[1] by one and then jumps to the address specified in the instruction by src.

4.1.17 Frame Pointer Offset Register (OFFS)

Register Description: **Frame Pointer Offset Register**
 Register Name: **OFFS**
 Register Address: **Module 0Eh, Index 03h**

Bit #	7	6	5	4	3	2	1	0
Name	OFFS7	OFFS6	OFFS5	OFFS4	OFFS3	OFFS2	OFFS1	OFFS0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 7 to 0: Frame Pointer Offset Register Bits 7:0 (OFFS[7:0]). This 8-bit register provides the frame pointer (FP) offset from the base pointer (BP). The frame pointer is formed by unsigned addition of Frame Pointer Base Register (BP) and Frame Pointer Offset Register (OFFS). The contents of this register can be postincremented or postdecremented when using the frame pointer for read operations and may be preincremented or pre-decremented when using the frame pointer for write operations. A carry out or borrow resulting from an increment/decrement operation has no effect on the Frame Pointer Base Register (BP).

4.1.18 Data Pointer Control Register (DPC)

Register Description: **Data Pointer Control Register**
 Register Name: **DPC**
 Register Address: **Module 0Eh, Index 04h**

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	—	—	—	WBS2	WBS1	WBS0	SDPS1	SDPS0
Reset	0	0	0	1	1	1	0	0
Access	r	r	r	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 001Ch on all forms of reset.

Bits 15 to 5: Reserved. Read 0, write ignored.

Bit 4: Word/Byte Select 2 (WBS2). This bit selects access mode for BP[OFFS]. When WBS2 is set to logic 1, the BP[OFFS] is operated in word mode for data memory access; when WBS2 is cleared to logic 0, BP[OFFS] is operated in byte mode for data memory access.

Bit 3: Word/Byte Select 1 (WBS1). This bit selects access mode for DP[1]. When WBS1 is set to logic 1, the DP[1] is operated in word mode for data memory access; when WBS1 is cleared to logic 0, DP[1] is operated in byte mode for data memory access.

Bit 2: Word/Byte Select 0 (WBS0). This bit selects access mode for DP[0]. When WBS0 is set to logic 1, the DP[0] is operated in word mode for data memory access; when WBS0 is cleared to logic 0, DP[0] is operated in byte mode for data memory access.

Bits 1 and 0: Source Data Pointer Select Bits 1 and 0 (SDPS[1:0]). These bits select one of the three data pointers as the active source pointer for the load operation. A new data pointer must be selected before being used to read data memory (see table below).

These bits default to 00b but do not activate DP[0] as an active source pointer until the SDPS bits are explicitly cleared to 00b or the DP[0] register is written by an instruction. Also, modifying the register contents of a data/frame pointer register (DP[0], DP[1], BP, or OFFS) will change the setting of the SDPS bits to reflect the active source pointer selection.

SDPS1	SDPS0	SOURCE POINTER SELECTION
0	0	DP[0]
0	1	DP[1]
1	0	FP (BP[OFFS])
1	1	Reserved (select FP if set)

4.1.19 General Register (GR)

Register Description: **General Register**
 Register Name: **GR**
 Register Address: **Module 0Eh, Index 05h**

Bit #	15	14	13	12	11	10	9	8
Name	GR15	GR14	GR13	GR12	GR11	GR10	GR9	GR8
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	GR7	GR6	GR5	GR4	GR3	GR2	GR1	GR0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: General Register Bits 15:0 (GR[15:0]). This register is intended primarily for supporting byte operations on 16-bit data. The 16-bit register is byte-readable, byte-writable through the corresponding GRL and GRH 8-bit registers and byte-swappable through the GRS 16-bit register.

4.1.20 General Register Low Byte (GRL)

Register Description: **General Register Low Byte**
 Register Name: **GRL**
 Register Address: **Module 0Eh, Index 06h**

Bit #	7	6	5	4	3	2	1	0
Name	GRL7	GRL6	GRL5	GRL4	GRL3	GRL2	GRL1	GRL0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 7 to 0: General Register Low Byte Bits 7:0 (GRL[7:0]). This register reflects the low byte of the GR register and is intended primarily for supporting byte operations on 16-bit data. Any data written to the GRL register will also be stored in the low byte of the GR register.

4.1.21 Frame Pointer Base Register (BP)

Register Description: **Frame Pointer Base Register**
 Register Name: **BP**
 Register Address: **Module 0Eh, Index 07h**

Bit #	15	14	13	12	11	10	9	8
Name	BP15	BP14	BP13	BP12	BP11	BP10	BP9	BP8
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	BP7	BP6	BP5	BP4	BP3	BP2	BP1	BP0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Frame Pointer Base Register Bits 15:0 (BP[15:0]). This register serves as the base pointer for the Frame Pointer (FP). The Frame Pointer is formed by unsigned addition of Frame Pointer Base Register (BP) and Frame Pointer Offset Register (OFFS). The content of this base pointer register is not affected by increment/decrement operations performed on the offset (OFFS) register.

4.1.22 General Register Byte-Swapped (GRS)

Register Description: **General Register Byte-Swapped**
 Register Name: **GRS**
 Register Address: **Module 0Eh, Index 08h**

Bit #	15	14	13	12	11	10	9	8
Name	GRS15	GRS14	GRS13	GRS12	GRS11	GRS10	GRS9	GRS8
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	GRS7	GRS6	GRS5	GRS4	GRS3	GRS2	GRS1	GRS0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: General Register Byte-Swapped Bits 15:0 (GRS[15:0]). This register is intended primarily for supporting byte operations on 16-bit data. This 16-bit read-only register returns the byte-swapped value for the data contained in the GR register.

4.1.23 General Register High Byte (GRH)

Register Description: **General Register High Byte**
 Register Name: **GRH**
 Register Address: **Module 0Eh, Index 09h**

Bit #	7	6	5	4	3	2	1	0
Name	GRH7	GRH6	GRH5	GRH4	GRH3	GRH2	GRH1	GRH0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 7 to 0: General Register High Byte Bits 7:0 (GRH[7:0]). This register reflects the high byte of the GR register and is intended primarily for supporting byte operations on 16-bit data. Any data written to the GRH register will also be stored in the high byte of the GR register.

4.1.24 General Register Sign Extended Low Byte (GRXL)

Register Description: **General Register Sign Extended Low Byte**
 Register Name: **GRXL**
 Register Address: **Module 0Eh, Index 0Ah**

Bit #	15	14	13	12	11	10	9	8
Name	GRXL15	GRXL14	GRXL13	GRXL12	GRXL11	GRXL10	GRXL9	GRXL8
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	GRXL7	GRXL6	GRXL5	GRXL4	GRXL3	GRXL2	GRXL1	GRXL0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: General Register Sign Extended Low Byte Bits 15:0 (GRXL[15:0]). This register provides the sign extended low byte of GR as a 16-bit source.

4.1.25 Frame Pointer Register (FP)

Register Description: **Frame Pointer Register**
 Register Name: **FP**
 Register Address: **Module 0Eh, Index 0Bh**

Bit #	15	14	13	12	11	10	9	8
Name	FP15	FP14	FP13	FP12	FP11	FP10	FP9	FP8
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	FP7	FP6	FP5	FP4	FP3	FP2	FP1	FP0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Frame Pointer Register Bits 15:0 (FP[15:0]). This register provides the current value of the frame pointer (BP[OFFS]).

4.1.26 Data Pointer 0 Register (DP[0])

Register Description: **Data Pointer 0 Register**
 Register Name: **DP[0]**
 Register Address: **Module 0Fh, Index 03h**

Bit #	15	14	13	12	11	10	9	8
Name	DP[0]15	DP[0]14	DP[0]13	DP[0]12	DP[0]11	DP[0]10	DP[0]9	DP[0]8
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	DP[0]7	DP[0]6	DP[0]5	DP[0]4	DP[0]3	DP[0]2	DP[0]1	DP[0]0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Data Pointer 0 Register Bits 15:0 (DP[0][15:0]). This register is used as a pointer to access data memory. DP[0] can be automatically incremented or decremented following each read operation or can be automatically incremented or decremented before each write operation.

4.1.27 Data Pointer 1 Register (DP[1])

Register Description: **Data Pointer 1 Register**
 Register Name: **DP[1]**
 Register Address: **Module 0Fh, Index 07h**

Bit #	15	14	13	12	11	10	9	8
Name	DP[1]15	DP[1]14	DP[1]13	DP[1]12	DP[1]11	DP[1]11	DP[1]9	DP[1]8
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	DP[1]7	DP[1]6	DP[1]5	DP[1]4	DP[1]3	DP[1]2	DP[1]1	DP[1]0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Data Pointer 1 Register Bits 15:0 (DP[1][15:0]). This register is used as a pointer to access data memory. DP[1] can be automatically incremented or decremented following each read operation or can be automatically incremented or decremented before each write operation.

4.2 Peripheral Register Modules

The MAXQ7667 microcontroller uses peripheral registers to control and monitor peripheral modules. These registers reside in Modules 0h to 5h, with subindex values 0h to 1Fh. The MAXQ7667 peripheral register map is shown in Table 4-3. The peripheral register module bit function and reset values are shown in Table 4-4. Each peripheral modules and its associated registers/bits are covered separately in their respective sections.

Table 4-3. MAXQ7667 Peripheral Register Map

REGISTER INDEX	MODULE NAME (BASE SPECIFIER)					
	M0	M1	M2	M3	M4	M5
00h	PO0	MCNT	T2CNA0	T2CNA2		BPH
01h	PO1	MA	T2H0	T2H2		BTRN
02h		MB	T2RH0	T2RH2		SARC
03h	EIF0	MC2	T2CH0	T2CH2		RCVC
04h	EIF1	MC1	T2CNA1			PLL
05h		MC0	T2H1	CNT1		AIE
06h		SPIB	T2RH1	SCON		CMPC
07h		SPICN	T2CH1	SBUF		CMPT
08h	PI0	SPICF	T2CNB0	T2CNB2		ASR
09h	PI1	SPICK	T2V0	T2V2		SARD
0Ah			T2R0	T2R2		LPFC
0Bh	EIE0		T2C0	T2C2		OSCC
0Ch	EIE1	MC1R	T2CNB1	FSTAT		BPFI
0Dh		MC0R	T2V1	ERRR		BPFO
0Eh		SCNT	T2R1	CHKSUM		LPFD
0Fh		STIM	T2C1	ISVEC		LPFF
10h	PD0	SALM	T2CFG0	T2CFG2		APE
11h	PD1	FPCTL	T2CFG1	STA0		
12h				SMD		FGAIN
13h	EIES0			FCON		B1COEF
14h	EIES1			CNT0		B2COEF
15h				CNT2		B3COEF
16h				IDFB		A2A
17h		RCTRM		SADDR		A2B
18h	PS0		ICDT0	SADEN		
19h	PS1		ICDT1	BT		A2D
1Ah			ICDC	TMR		
1Bh	PR0		ICDF			A3A
1Ch	PR1	ID0	ICDB			A3B
1Dh		ID1	ICDA			
1Eh			ICDD			A3D
1Fh						

Table 4-4. MAXQ7667 Module 0 Register Bit Functions and Reset Values

REGISTER	REGISTER BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PO0 00h[00h]	—	—	—	—	—	—	—	PO07	PO06	PO05	PO04	PO03	PO02	PO01	PO00	
PO1 00h[01h]	0	0	0	0	0	0	0	PO17	PO16	PO15	PO14	PO13	PO12	PO11	PO10	
EIF0 00h[03h]	—	—	—	—	—	—	—	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0	
EIF1 00h[04h]	0	0	0	0	0	0	0	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0	
PI0 00h[08h]	—	—	—	—	—	—	—	PI07	PI06	PI05	PI04	PI03	PI02	PI01	PI00	
PI1 00h[09h]	0	0	0	0	0	0	0	st	st	st	st	st	st	st	st	
EIE0 00h[0Bh]	—	—	—	—	—	—	—	EX7	EX6	EX5	EX4	EX3	EX2	EX1	EX0	
EIE1 00h[0Ch]	0	0	0	0	0	0	0	EX7	EX6	EX5	EX4	EX3	EX2	EX1	EX0	
PD0 00h[10h]	—	—	—	—	—	—	—	PD07	PD06	PD05	PD04	PD03	PD02	PD01	PD00	
PD1 00h[11h]	0	0	0	0	0	0	0	PD07	PD06	PD05	PD04	PD03	PD02	PD01	PD00	
EIES0 00h[13h]	—	—	—	—	—	—	—	IT7	IT6	IT5	IT4	IT3	IT2	IT1	IT0	
EIES1 00h[14h]	0	0	0	0	0	0	0	IT7	IT6	IT5	IT4	IT3	IT2	IT1	IT0	
PS0 00h[18h]	—	—	—	—	—	—	—	PS07	PS06	PS05	PS04	PS03	PS02	PS01	PS00	
PS1 00h[19h]	—	—	—	—	—	—	—	PS17	PS16	PS15	PS14	PS13	PS12	PS11	PS10	
PR0 00h[1Bh]	—	—	—	—	—	—	—	PR07	PR06	PR05	PR04	PR03	PR02	PR01	PR00	
PR1 00h[1Ch]	—	—	—	—	—	—	—	PR17	PR16	PR15	PR14	PR13	PR12	PR11	PR10	

st = Dependent on the pin's state.

Table 4-5. MAXQ7667 Module 1 Register Bit Functions and Reset Values

REGISTER	REGISTER BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCNT 01h[00h]	—	—	—	—	—	—	—	—	OF	MCW	CLD	SQU	OPCS	MSUB	MMAC	SUS
MA 01h[01h]	MA15	MA14	MA13	MA12	MA11	MA10	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0
MB 01h[02h]	MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8	MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0
MC2 01h[03h]	MC215	MC214	MC213	MC212	MC211	MC210	MC29	MC28	MC27	MC26	MC25	MC24	MC23	MC22	MC21	MC20
MC1 01h[04h]	MC115	MC114	MC113	MC112	MC111	MC110	MC19	MC18	MC17	MC16	MC15	MC14	MC13	MC12	MC11	MC10
MC0 01h[05h]	MC015	MC014	MC013	MC012	MC011	MC010	MC09	MC08	MC07	MC06	MC05	MC04	MC03	MC02	MC01	MC00
SPIB 01h[06h]	SPIB15	SPIB14	SPIB13	SPIB12	SPIB11	SPIB10	SPIB9	SPIB8	SPIB7	SPIB6	SPIB5	SPIB4	SPIB3	SPIB2	SPIB1	SPIB0
SPICN 01h[07h]	—	—	—	—	—	—	—	—	STBY	SPIC	ROVR	WCOL	MODF	MODFE	MSTM	SPIEN
SPICF 01h[08h]	—	—	—	—	—	—	—	—	ESPII	SAS	—	—	—	CHR	CKPHA	CKPOL
SPICK 01h[09h]	—	—	—	—	—	—	—	—	SPICK7	SPICK6	SPICK5	SPICK4	SPICK3	SPICK2	SPICK1	SPICK0
MC1R 01h[0Ch]	MC1R15	MC1R14	MC1R13	MC1R12	MC1R11	MC1R10	MC1R9	MC1R8	MC1R7	MC1R6	MC1R5	MC1R4	MC1R3	MC1R2	MC1R1	MC1R0
MC0R 01h[0Dh]	MC0R15	MC0R14	MC0R13	MC0R12	MC0R11	MC0R10	MC0R9	MC0R8	MC0R7	MC0R6	MC0R5	MC0R4	MC0R3	MC0R2	MC0R1	MC0R0
SCNT 01h[0Eh]	—	—	—	—	STDIV2	STDIV1	STDIV0	SSYNC_EN	SALIE	SALMF	—	—	—	—	SALME	STIME
STIM 01h[0Fh]	STIM15	STIM14	STIM13	STIM12	STIM11	STIM10	STIM9	STIM8	STIM7	STIM6	STIM5	STIM4	STIM3	STIM2	STIM1	STIM0
SALM 01h[10h]	SALM15	SALM14	SALM13	SALM12	SALM11	SALM10	SALM9	SALM8	SALM7	SALM6	SALM5	SALM4	SALM3	SALM2	SALM1	SALM0

P = Cleared to 00h on power-on reset and then, if required, initialized to a value stored within the flash information block.

Table 4-5. MAXQ7667 Module 1 Register Bit Functions and Reset Values (continued)

REGISTER	REGISTER BIT																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
FPCTL 01h[14h]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RCTRM 01h[17h]	—	—	—	—	—	—	—	RCTRM8	RCTRM7	RCTRM6	RCTRM5	RCTRM4	RCTRM3	RCTRM2	RCTRM1	RCTRM0	
	0	0	0	0	0	0	0	P	P	P	P	P	P	P	P	P	
ID0 01h[1Ch]	ID015	ID014	ID013	ID012	ID011	ID010	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	
	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	
ID1 01h[1Dh]	ID115	ID114	ID113	ID112	ID111	ID110	ID19	ID18	ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10	
	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	

P = Cleared to 00h on power-on reset and then, if required, initialized to a value stored within the flash information block.

Table 4-6. MAXQ7667 Module 2 Register Bit Functions and Reset Values

REGISTER	REGISTER BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2CNA0 02h[00h]	—	—	—	—	—	—	—	—	ET2	T2OE0	T2POL0	TR2L	TR2	CPRL2	SS2	G2EN
T2H0 02h[01h]	—	—	—	—	—	—	—	—	T2H07	T2H06	T2H05	T2H04	T2H03	T2H02	T2H01	T2H00
T2RH0 02h[02h]	—	—	—	—	—	—	—	—	T2RH07	T2RH06	T2RH05	T2RH04	T2RH03	T2RH02	T2RH01	T2RH00
T2CH0 02h[03h]	—	—	—	—	—	—	—	—	T2CH07	T2CH06	T2CH05	T2CH04	T2CH03	T2CH02	T2CH01	T2CH00
T2CNA1 02h[04h]	—	—	—	—	—	—	—	—	ET2	T2OE0	T2POL0	TR2L	TR2	CPRL2	SS2	G2EN
T2H1 02h[05h]	—	—	—	—	—	—	—	—	T2H17	T2H16	T2H15	T2H14	T2H13	T2H12	T2H11	T2H10
T2RH1 02h[06h]	—	—	—	—	—	—	—	—	T2RH17	T2RH16	T2RH15	T2RH14	T2RH13	T2RH12	T2RH11	T2RH10
T2CH1 02h[07h]	—	—	—	—	—	—	—	—	T2CH17	T2CH16	T2CH15	T2CH14	T2CH13	T2CH12	T2CH11	T2CH10
T2CNB0 02h[08h]	—	—	—	—	—	—	—	—	ET2L	T2OE1	T2POL1	X	TF2	TF2L	TCC2	TC2L
T2V0 02h[09h]	T2V015	T2V014	T2V013	T2V012	T2V011	T2V010	T2V09	T2V08	T2V07	T2V06	T2V05	T2V04	T2V03	T2V02	T2V01	T2V00
T2R0 02h[0Ah]	T2R015	T2R014	T2R013	T2R012	T2R011	T2R010	T2R09	T2R08	T2R07	T2R06	T2R05	T2R04	T2R03	T2R02	T2R01	T2R00
T2C0 02h[0Bh]	T2C015	T2C014	T2C013	T2C012	T2C011	T2C010	T2C09	T2C08	T2C07	T2C06	T2C05	T2C04	T2C03	T2C02	T2C01	T2C00
T2CNB1 02h[0Ch]	—	—	—	—	—	—	—	—	ET2L	—	—	X	TF2	TF2L	TCC2	TC2L
T2V1 02h[0Dh]	T2V115	T2V114	T2V113	T2V112	T2V111	T2V110	T2V19	T2V18	T2V17	T2V16	T2V15	T2V14	T2V13	T2V12	T2V11	T2V10
T2R1 02h[0Eh]	T2R115	T2R114	T2R113	T2R112	T2R111	T2R110	T2R19	T2R18	T2R17	T2R16	T2R15	T2R14	T2R13	T2R12	T2R11	T2R10
T2C1 02h[0Fh]	T2C115	T2C114	T2C113	T2C112	T2C111	T2C110	T2C19	T2C18	T2C17	T2C16	T2C15	T2C14	T2C13	T2C12	T2C11	T2C10

db = Special: read/write access only in background or debug mode.
 dw = Special: read/write by debug engine.

Table 4-6. MAXQ7667 Module 2 Register Bit Functions and Reset Values (continued)

REGISTER	REGISTER BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2CFG0 02h[10h]	—	—	—	—	—	—	—	—	T2CI	T2DIV2	T2DIV1	T2DIV0	T2MD	CCF1	CCF0	C/T2
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T2CFG1 02h[11h]	—	—	—	—	—	—	—	—	T2CI	T2DIV2	T2DIV1	T2DIV0	T2MD	CCF1	CCF0	C/T2
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ICDT0 02h[18h]	ICDT015	ICDT014	ICDT013	ICDT012	ICDT011	ICDT010	ICDT09	ICDT08	ICDT07	ICDT06	ICDT05	ICDT04	ICDT03	ICDT02	ICDT01	ICDT00
	db	db	db	db	db	db	db	db	db	db	db	db	db	db	db	db
ICDT1 02h[19h]	ICDT115	ICDT114	ICDT113	ICDT112	ICDT111	ICDT110	ICDT19	ICDT18	ICDT17	ICDT16	ICDT15	ICDT14	ICDT13	ICDT12	ICDT11	ICDT10
	db	db	db	db	db	db	db	db	db	db	db	db	db	db	db	db
ICDC 02h[1Ah]									DME	—	REGE	—	CMD3	CMD2	CMD1	CMD0
									dw	0	dw	0	dw	dw	dw	dw
ICDF 02h[1Bh]									—	—	—	—	PSS1	PSS0	SPE	TXC
									0	0	0	0	0	0	0	0
ICDB 02h[1Ch]									ICDB7	ICDB6	ICDB5	ICDB4	ICDB3	ICDB2	ICDB1	ICDB0
									0	0	0	0	0	0	0	0
ICDA 02h[1Dh]	ICDA15	ICDA14	ICDA13	ICDA12	ICDA11	ICDA10	ICDA9	ICDA8	ICDA7	ICDA6	ICDA5	ICDA4	ICDA3	ICDA2	ICDA1	ICDA0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ICDD 02h[1Eh]	ICDD15	ICDD14	ICDD13	ICDD12	ICDD11	ICDD10	ICDD9	ICDD8	ICDD7	ICDD6	ICDD5	ICDD4	ICDD3	ICDD2	ICDD1	ICDD0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

db = Special: read/write access only in background or debug mode.
dw = Special: read/write by debug engine.

Table 4-7. MAXQ7667 Module 3 Register Bit Functions and Reset Values

REGISTER	REGISTER BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2CNA2 03h[00h]	—	—	—	—	—	—	—	—	ET2	T2OE0	T2POLO	TR2L	TR2	CPRL2	SS2	G2EN
T2H2 03h[01h]	—	—	—	—	—	—	—	—	T2H27	T2H26	T2H25	T2H24	T2H23	T2H22	T2H21	T2H20
T2RH2 03h[02h]	—	—	—	—	—	—	—	—	T2RH27	T2RH26	T2RH25	T2RH24	T2RH23	T2RH22	T2RH21	T2RH20
T2CH2 03h[03h]	—	—	—	—	—	—	—	—	T2CH27	T2CH26	T2CH25	T2CH24	T2CH23	T2CH22	T2CH21	T2CH20
CNT1 03h[05h]									RTN	CK	FL5	FL4	FL3	FL2	FL1	FL0
SCON 03h[06h]									SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI
SBUF 03h[07h]									SBUF7	SBUF6	SBUF5	SBUF4	SBUF3	SBUF2	SBUF1	SBUF0
T2CNB2 03h[08h]	—	—	—	—	—	—	—	—	ET2L	T2OE1	T2POL1	X	TF2	TF2L	TCC2	TC2L
T2V2 03h[09h]	0	0	0	0	0	0	0	0	T2V27	T2V26	T2V25	T2V24	T2V23	T2V22	T2V21	T2V20
T2R2 03h[0Ah]	0	0	0	0	0	0	0	0	T2R27	T2R26	T2R25	T2R24	T2R23	T2R22	T2R21	T2R20
T2C2 03h[0Bh]	0	0	0	0	0	0	0	0	T2C27	T2C26	T2C25	T2C24	T2C23	T2C22	T2C21	T2C20
FSTAT 03h[0Ch]									—	—	TFE	TFAE	TFE	RFF	RFAF	RFE
ERRR 03h[0Dh]									—	OTE	DME	CKE	P1	P1E	P0	POE
CHKSUM 03h[0Eh]	0	0	0	0	0	0	0	0	CHKSUM7	CHKSUM6	CHKSUM5	CHKSUM4	CHKSUM3	CHKSUM2	CHKSUM1	CHKSUM0
ISVEC 03h[0Fh]									—	—	—	—	ISVEC3	ISVEC2	ISVEC1	ISVEC0
T2CFG2 03h[10h]	0	0	0	0	0	0	0	0	T2C1	T2DIV2	T2DIV1	T2DIV0	T2MD	CCF1	CCF0	C/I2
STAO 03h[11h]									—	—	—	—	—	—	INP	BUSY

Table 4-7. MAXQ7667 Module 3 Register Bit Functions and Reset Values (continued)

REGISTER	REGISTER BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMD 03h[12h]									EIR	OFS	—	—	—	IE	SMOD	FEDE
FCON 03h[13h]									FTF	FRF	TXFT1	TXFT0	RXFT1	RXFT0	OE	FEN
CNT0 03h[14h]									WU	FP1	FP0	INE	AUT	INIT	LUN1	LUN0
CNT2 03h[15h]									—	—	—	DMIS	PM	HDO	FBS	BTH
IDFB 03h[16h]	—	—	IDFBH5	IDFBH4	IDFBH3	IDFBH2	IDFBH1	IDFBH0	—	—	IDFBL5	IDFBL4	IDFBL3	IDFBL2	IDFBL1	IDFBL0
SADDR 03h[17h]	0	0	1	1	1	1	1	1	SADDR7	SADDR6	SADDR5	SADDR4	SADDR3	SADDR2	SADDR1	SADDR0
SADEN 03h[18h]									SADEN7	SADEN6	SADEN5	SADEN4	SADEN3	SADEN2	SADEN1	SADEN0
BT 03h[19h]	BT15	BT14	BT13	BT12	BT11	BT10	BT9	BT8	BT7	BT6	BT5	BT4	BT3	BT2	BT1	BT0
TMR 03h[1Ah]	TMR15	TMR14	TMR13	TMR12	TMR11	TMR10	TMR9	TMR8	TMR7	TMR6	TMR5	TMR4	TMR3	TMR2	TMR1	TMR0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-8. MAXQ7667 Module 5 Register Bit Functions and Reset Values

REGISTER	REGISTER BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BPH 05h[00h]	BSTT	BDS	—	—	—	BPH9	BPH8	BPH7	BPH6	BPH5	BPH4	BPH3	BPH2	BPH1	BPH0	
BTRN 05h[01h]	BDIV3	BDIV2	BDIV1	BDIV0	BPOL	BCKS	BTRI	BGT	BCTN7	BCTN6	BCTN5	BCTN4	BCTN3	BCTN2	BCTN1	BCTN0
SARC 05h[02h]	—	—	—	—	SARMX2	SARMX1	SARMX0	SARDIF	SARBIP	SARDUL	SARRSEL	SARASD	SARBY	SARS2	SARS1	SARS0
RCVC 05h[03h]	—	—	—	—	—	—	LNAOSEL	LNAISEL1	LNAISEL0	—	—	RCVGN4	RCVGN3	RCVGN2	RCVGN1	RCVGN0
PLLF 05h[04h]	—	—	—	—	—	PLLC0	PLLC1	PLLF8	PLLF7	PLLF6	PLLF5	PLLF4	PLLF3	PLLF2	PLLF1	PLLF0
AIE 05h[05h]	—	—	—	—	—	—	—	XTIE	VIBIE	VDBIE	VABIE	CMPIE	LFIE	LPFIE	SARIE	—
CMPC 05h[06h]	CMPP	CMPH14	CMPH13	CMPH12	CMPH11	CMPH10	CMPH9	CMPH8	CMPH7	CMPH6	CMPH5	CMPH4	CMPH3	CMPH2	CMPH1	CMPH0
CMPT 05h[07h]	CMPT15	CMPT14	CMPT13	CMPT12	CMPT11	CMPT10	CMPT9	CMPT8	CMPT7	CMPT6	CMPT5	CMPT4	CMPT3	CMPT2	CMPT1	CMPT0
ASR 05h[08h]	VIOLVL	DVLVL	AVLVL	CMPLVL	—	—	—	XTRDY	XTI	VIBI	VDBI	VABI	CMPI	LPFLL	LPFRDY	SARRDY
SARD 05h[09h]	—	—	—	—	SARD11	SARD10	SARD9	SARD8	SARD7	SARD6	SARD5	SARD4	SARD3	SARD2	SARD1	SARD0
LPFC 05h[0Ah]	FFIL3	FFIL2	FFIL1	FFILO	FFDP3	FFDP2	FFDP1	FFDP0	FFOV	—	—	—	FLLD	FLLS2	FLLS1	FLLS0
OSCC 05h[0Bh]	—	—	—	—	—	—	—	—	—	—	—	—	SARCD1	SARCD0	XTE	RCE
BPFI 05h[0Ch]	BPFI15	BPFI14	BPFI13	BPFI12	BPFI11	BPFI10	BPFI9	BPFI8	BPFI7	BPFI6	BPFI5	BPFI4	BPFI3	BPFI2	BPFI1	BPFI0
BPFO 05h[0Dh]	BPFO15	BPFO14	BPFO13	BPFO12	BPFO11	BPFO10	BPFO9	BPFO8	BPFO7	BPFO6	BPFO5	BPFO4	BPFO3	BPFO2	BPFO1	BPFO0
LPFD 05h[0Eh]	LPFD15	LPFD14	LPFD13	LPFD12	LPFD11	LPFD10	LPFD9	LPFD8	LPFD7	LPFD6	LPFD5	LPFD4	LPFD3	LPFD2	LPFD1	LPFD0
LPFF 05h[0Fh]	LPFF15	LPFF14	LPFF13	LPFF12	LPFF11	LPFF10	LPFF9	LPFF8	LPFF7	LPFF6	LPFF5	LPFF4	LPFF3	LPFF2	LPFF1	LPFF0
APE 05h[10h]	—	RBUFE	—	BGE	LRIOPD	LRDPD	LRAPD	VIBE	VDPE	VDBE	VABE	SARE	PLLE	MDE	LNAE	BIASE
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

s = Bits clear on power-on reset.

Table 4-8. MAXQ7667 Module 5 Register Bit Functions and Reset Values (continued)

REGISTER	REGISTER BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FGAIN 05h[12h]	FGAIN15	FGAIN14	FGAIN13	FGAIN12	FGAIN11	FGAIN10	FGAIN9	FGAIN8	FGAIN7	FGAIN6	FGAIN5	FGAIN4	FGAIN3	FGAIN2	FGAIN1	FGAIN0
	0x7B5C															
B1COEF 05h[13h]	B1COEF15	B1COEF14	B1COEF13	B1COEF12	B1COEF11	B1COEF10	B1COEF9	B1COEF8	B1COEF7	B1COEF6	B1COEF5	B1COEF4	B1COEF3	B1COEF2	B1COEF1	B1COEF0
	0x2492															
B2COEF 05h[14h]	B2COEF15	B2COEF14	B2COEF13	B2COEF12	B2COEF11	B2COEF10	B2COEF9	B2COEF8	B2COEF7	B2COEF6	B2COEF5	B2COEF4	B2COEF3	B2COEF2	B2COEF1	B2COEF0
	0x5820															
B3COEF 05h[15h]	B3COEF15	B3COEF14	B3COEF13	B3COEF12	B3COEF11	B3COEF10	B3COEF9	B3COEF8	B3COEF7	B3COEF6	B3COEF5	B3COEF4	B3COEF3	B3COEF2	B3COEF1	B3COEF0
	0x2410															
A2A 05h[16h]	A2A15	A2A14	A2A13	A2A12	A2A11	A2A10	A2A9	A2A8	A2A7	A2A6	A2A5	A2A4	A2A3	A2A2	A2A1	A2A0
	0x30F4															
A2B 05h[17h]	A2B15	A2B14	A2B13	A2B12	A2B11	A2B10	A2B9	A2B8	A2B7	A2B6	A2B5	A2B4	A2B3	A2B2	A2B1	A2B0
	0x3369															
A2D 05h[19h]	A2D15	A2D14	A2D13	A2D12	A2D11	A2D10	A2D9	A2D8	A2D7	A2D6	A2D5	A2D4	A2D3	A2D2	A2D1	A2D0
	0x3A28															
A3A 05h[1Bh]	A3A15	A3A14	A3A13	A3A12	A3A11	A3A10	A3A9	A3A8	A3A7	A3A6	A3A5	A3A4	A3A3	A3A2	A3A1	A3A0
	0xE20E															
A3B 05h[1Ch]	A3B15	A3B14	A3B13	A3B12	A3B11	A3B10	A3B9	A3B8	A3B7	A3B6	A3B5	A3B4	A3B3	A3B2	A3B1	A3B0
	0xE1E3															
A3D 05h[1Eh]	A3D15	A3D14	A3D13	A3D12	A3D11	A3D10	A3D9	A3D8	A3D7	A3D6	A3D5	A3D4	A3D3	A3D2	A3D1	A3D0
	0xE559															

SECTION 5: GENERAL-PURPOSE I/O MODULE

This section contains the following information:

5.1 Architecture	5-3
5.1.1 Enhanced Type D I/O Port	5-3
5.1.2 GPIO Port Pins	5-4
5.2 Port Registers	5-5
5.2.1 Port 0 Output Register (PO0)	5-5
5.2.2 Port 1 Output Register (PO1)	5-5
5.2.3 External Interrupt Flag Register (Port 0) (EIF0)	5-6
5.2.4 External Interrupt Flag Register (Port 1) (EIF1)	5-7
5.2.5 Port 0 Input Register (PI0)	5-8
5.2.6 Port 1 Input Register (PI1)	5-8
5.2.7 External Interrupt Enable Register (Port 0) (EIE0)	5-9
5.2.8 External Interrupt Enable Register (Port 1) (EIE1)	5-10
5.2.9 Port 0 Direction Register (PD0)	5-11
5.2.10 Port 1 Direction Register (PD1)	5-11
5.2.11 External Interrupt Edge Select Register (Port 0) (EIES0)	5-12
5.2.12 External Interrupt Edge Select Register (Port 1) (EIES1)	5-13
5.2.13 Pad Drive Strength Register (Port 0) (PS0)	5-14
5.2.14 Pad Drive Strength Register (Port 1) (PS1)	5-14
5.2.15 Pad Resistive Pull Direction Register (Port 0) (PR0)	5-14
5.2.16 Pad Resistive Pull Direction Register (Port 1) (PR1)	5-15
5.3 GPIO Operation	5-15
5.3.1 Port Direction Control and Input/Output	5-15
5.3.2 Port P0 and P1 External Interrupts	5-16
5.3.3 Port Pin Special and Alternate Functions	5-16
5.3.4 Port Pin Examples	5-19
5.3.4.1 Port Pin Example 1: Driving Outputs on Port 0	5-19
5.3.4.2 Port Pin Example 2: Receiving Inputs on Port 0	5-19

LIST OF FIGURES

Figure 5-1. Enhanced Type D Port Pin Schematic 5-3

LIST OF TABLES

Table 5-1. Port P0 Pins 5-4

Table 5-2. Port P1 Pins 5-4

Table 5-3. Port Pin Input/Output States (in Standard Mode) 5-15

Table 5-4. Port P0 Pin Special and Alternate Functions 5-16

Table 5-5. Port P1 Pin Special and Alternate Functions 5-18

SECTION 5: GENERAL-PURPOSE I/O MODULE

The MAXQ7667 smart data-acquisition microcontroller provides 2 ports (P0 and P1) for general-purpose I/O, each with 8 port pins. The port pins have the following features:

- All pins support alternate and special functions
- CMOS-compatible I/O levels to DVDDIO and GND rails
- User-selectable programmable drive strength when configured as output
- User-selectable resistive pull direction when configured as input
- Rising or falling edge selectable interrupt or wakeup inputs on all digital I/O pins
- Low leakage

5.1 Architecture

5.1.1 Enhanced Type D I/O Port

The MAXQ7667 supports the enhanced Type D port. Enhanced Type D is a bidirectional I/O port that incorporates Schmitt trigger receivers and full CMOS output drivers, and can support alternate functions. All enhanced Type D pins also have interrupt capability.

All port pins can support special function (SF). Enabling the special function automatically converts the pin to that function. Special function is usually implemented in another functional module and supported by individual enable or status bits.

Figure 5-1 illustrates an enhanced Type D port pin function. The pin logic of each port pin is identical.

When the ports are configured as an output, the output drive strength can be set to either 1mA or 2mA by setting the PS0 and PS1 registers. When the ports are configured as an input, the resistive pull direction (either pullup or pulldown) can be set by the PR0 and PR1 registers. The typical value is 150kΩ.

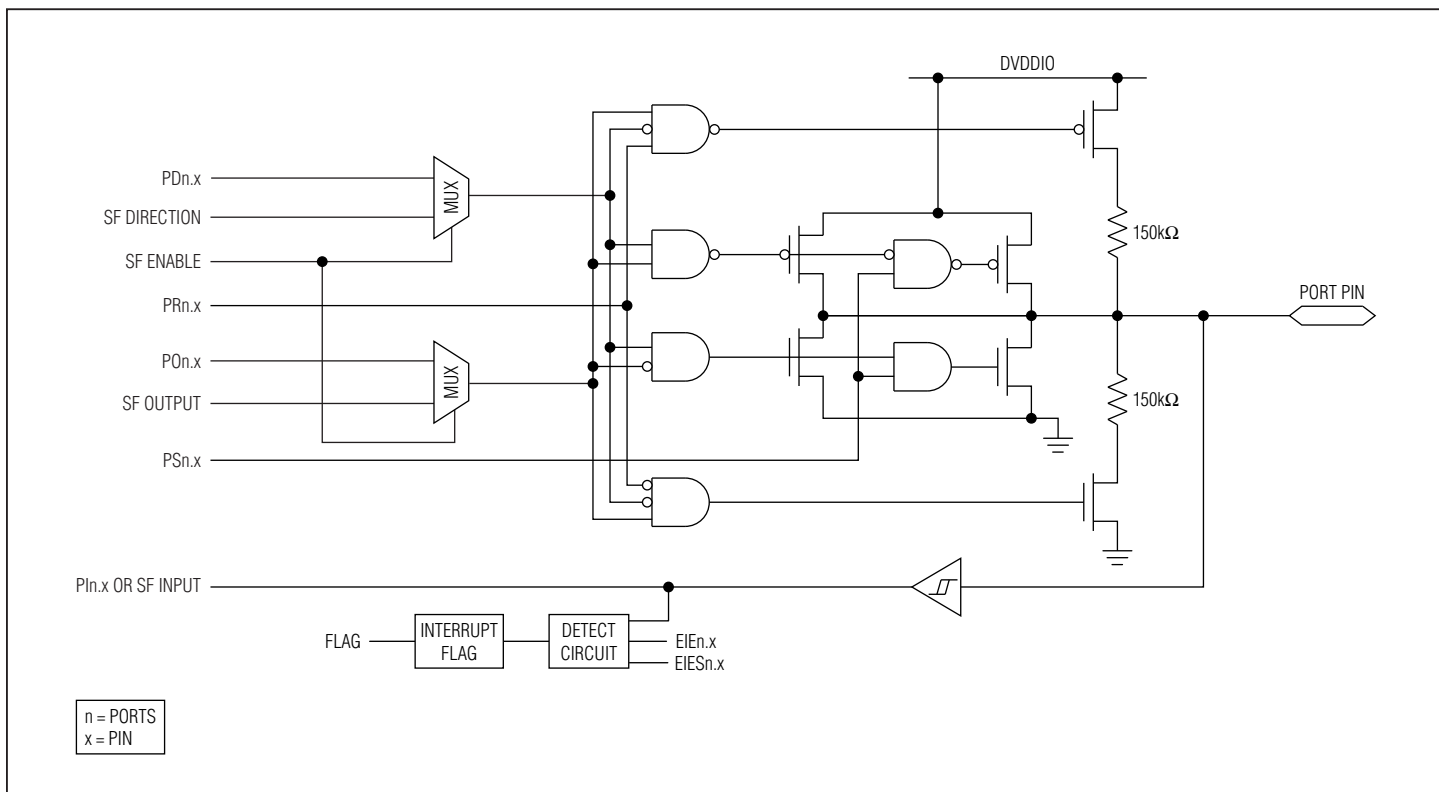


Figure 5-1. Enhanced Type D Port Pin Schematic

5.1.2 GPIO Port Pins

The MAXQ7667 port P0 and P1 pins are summarized in Table 5-1 and Table 5-2.

Table 5-1. Port P0 Pins

PORT P0 SIGNALS	PIN	FUNCTION
P0.0/URX	9	Digital GPIO and UART Receive Data Input. As URX this pin is the receive data input of the UART, which can (optionally) be connected to RXD of a LIN transceiver.
P0.1/UTX	10	Digital GPIO and UART Transmit Data Output. As UTX this pin is the transmit data output of the UART, which can (optionally) be connected to TXD of a LIN transceiver.
P0.2/ $\overline{\text{TXEN}}$	11	Digital GPIO and UART Transmit. As $\overline{\text{TXEN}}$ the pin can be used to control the transmit enable of an external driver. This pin defaults to $\overline{\text{TXEN}}$ any time the UART is used. $\overline{\text{TXEN}}$ is high when the UART is receiving and low when the UART is transmitting.
P0.3/T0/ ADCCTL	12	Digital GPIO, Timer 0 I/O, and ADC Control Input. As T0 this pin is the primary timer/PWM0 input or output. As ADCCTL this user-programmable rising or falling edge controls the SAR ADC sampling instant and start of conversion. Optionally, the other edge can be used to enable the ADC and begin acquiring prior to sampling.
P0.4/T0B	13	Digital GPIO, Timer 0 I/O, and Comparator Output. As T0B this pin is the secondary timer/PWM1 input or output. As CMPO this pin is the output of the digital comparator for the lowpass filter.
P0.5/T1	14	Digital GPIO and Timer 1 I/O. As T1 this pin is the primary timer/PWM2 input or output.
P0.6/T2	15	Digital GPIO and Timer 2 I/O. As T2 this pin is the primary timer/PWM2 input or output.
P0.7/T2B	16	Digital GPIO and Timer 2 I/O. As T2B this pin is the secondary timer/PWM2 input or output.

Table 5-2. Port P1 Pins

PORT P1 SIGNALS	PIN	FUNCTION
P1.0/TDO	46	Digital GPIO and JTAG Serial Data Output. As TDO this pin is the JTAG serial test data output.
P1.1/TMS	47	Digital GPIO and JTAG Test Mode Select Input. As TMS this pin is the JTAG test mode select input.
P1.2/TDI	48	Digital GPIO and JTAG. As TDI this pin is the JTAG serial test data input.
P1.3/TCK	1	Digital GPIO and JTAG Serial Clock Input. As TCK this pin is the JTAG serial test clock input.
P1.4/MOSI	2	Digital GPIO and SPI Serial Data I/O. As MOSI this pin is the master out-slave in for the SPI interface.
P1.5/MISO	3	Digital GPIO and SPI Serial Data I/O. As MISO this pin is the master in-slave out for the SPI interface.
P1.6/SCLK	4	Digital GPIO and SPI Serial Clock. As SCLK this pin is the serial clock for the SPI interface.
P1.7/SYNC/SS	5	Digital GPIO, System Timer Sync Input, and SPI Port Slave Select. As SYNC this pin resets the system timer.

5.2 Port Registers

The following peripheral registers control the general-purpose I/O and external interrupt features specific to the MAXQ7667.

5.2.1 Port 0 Output Register (PO0)

Register Description: **Port 0 Output Register**

Register Name: **PO0**

Register Address: **Module 00h, Index 00h**

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	PO07	PO06	PO05	PO04	PO03	PO02	PO01	PO00
Reset	1	1	1	1	1	1	1	1
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to FFh on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

Bits 7 to 0: Port 0 Output Register Bits 7:0 (PO0[7:0]). Port 0 is an enhanced Type D I/O port. The PO0 register stores output data for port 0 when it is defined as an output port and controls whether the internal pullup resistor is enabled/disabled if a port pin is defined as an input. The contents of this register can be modified by a write access. Reading from the register returns the contents of the register. Changing the direction of port 0 does not change the data contents of the register.

5.2.2 Port 1 Output Register (PO1)

Register Description: **Port 1 Output Register**

Register Name: **PO1**

Register Address: **Module 00h, Index 01h**

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	PO17	PO16	PO15	PO14	PO13	PO12	PO11	PO10
Reset	1	1	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to FFh on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

Bits 7 to 0: Port 1 Output Register Bits 7:0 (PO1[7:0]). Port 1 is an enhanced Type D I/O port. The PO1 register stores output data for port 1 when it is defined as an output port and controls whether the internal weak pullup resistor is enabled/disabled if a port pin is defined as an input. The contents of this register can be modified by a write access. Reading from the register returns the contents of the register. Changing the direction of port 1 does not change the data contents of the register.

5.2.3 External Interrupt Flag Register (Port 0) (EIF0)

Register Description: **External Interrupt Flag Register (Port 0)**
 Register Name: **EIF0**
 Register Address: **Module 00h, Index 03h**

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	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

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

Bit 7: Bit 7 Edge Detect (IE7). This bit is set when a negative edge (IT7 = 1) or a positive edge (IT7 = 0) is detected on the interrupt 7 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 6: Bit 6 Edge Detect (IE6). This bit is set when a negative edge (IT6 = 1) or a positive edge (IT6 = 0) is detected on the interrupt 6 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 5: Bit 5 Edge Detect (IE5). This bit is set when a negative edge (IT5 = 1) or a positive edge (IT5 = 0) is detected on the interrupt 5 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 4: Bit 4 Edge Detect (IE4). This bit is set when a negative edge (IT4 = 1) or a positive edge (IT4 = 0) is detected on the interrupt 4 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 3: Bit 3 Edge Detect (IE3). This bit is set when a negative edge (IT3 = 1) or a positive edge (IT3 = 0) is detected on the interrupt 3 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 2: Bit 2 Edge Detect (IE2). This bit is set when a negative edge (IT2 = 1) or a positive edge (IT2 = 0) is detected on the interrupt 2 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 1: Bit 1 Edge Detect (IE1). This bit is set when a negative edge (IT1 = 1) or a positive edge (IT1 = 0) is detected on the interrupt 1 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 0: Bit 0 Edge Detect (IE0). This bit is set when a negative edge (IT0 = 1) or a positive edge (IT0 = 0) is detected on the interrupt 0 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

5.2.4 External Interrupt Flag Register (Port 1) (EIF1)

Register Description: **External Interrupt Flag Register (Port 1)**

Register Name: **EIF1**

Register Address: **Module 00h, Index 04h**

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	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

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

Bit 7: Interrupt 7 Edge Detect (IE7). This bit is set when a negative edge (IT7 = 1) or a positive edge (IT7 = 0) is detected on the interrupt 7 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 6: Interrupt 6 Edge Detect (IE6). This bit is set when a negative edge (IT6 = 1) or a positive edge (IT6 = 0) is detected on the interrupt 6 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 5: Interrupt 5 Edge Detect (IE5). This bit is set when a negative edge (IT5 = 1) or a positive edge (IT5 = 0) is detected on the interrupt 5 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 4: Interrupt 4 Edge Detect (IE4). This bit is set when a negative edge (IT4 = 1) or a positive edge (IT4 = 0) is detected on the interrupt 4 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 3: Interrupt 3 Edge Detect (IE3). This bit is set when a negative edge (IT3 = 1) or a positive edge (IT3 = 0) is detected on the interrupt 3 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 2: Interrupt 2 Edge Detect (IE2). This bit is set when a negative edge (IT2 = 1) or a positive edge (IT2 = 0) is detected on the interrupt 2 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 1: Interrupt 1 Edge Detect (IE1). This bit is set when a negative edge (IT1 = 1) or a positive edge (IT1 = 0) is detected on the interrupt 1 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

Bit 0: Interrupt 0 Edge Detect (IE0). This bit is set when a negative edge (IT0 = 1) or a positive edge (IT0 = 0) is detected on the interrupt 0 pin. Setting this bit to 1 generates an interrupt to the CPU if enabled. This bit remains set until cleared by software or a reset. It must be cleared by software before exiting the interrupt source routine or another interrupt will be generated as long as this bit is set.

5.2.5 Port 0 Input Register (PI0)

Register Description: **Port 0 Input Register**
 Register Name: **PI0**
 Register Address: **Module 00h, Index 08h**

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	PI07	PI06	PI05	PI04	PI03	PI02	PI01	PI00
Reset	s	s	s	s	s	s	s	s
Access	r	r	r	r	r	r	r	r

r = read, s = dependent on pin's state

Note: The reset value for this register is dependent on the logical states of the pins.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

Bits 7 to 0: Port 0 Input Register Bits 7:0 (PI0[7:0]). Port 0 is an enhanced Type D I/O port. The PI0 register always reflects the logic state of its pins when read.

5.2.6 Port 1 Input Register (PI1)

Register Description: **Port 1 Input Register**
 Register Name: **PI1**
 Register Address: **Module 00h, Index 09h**

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	PI17	PI16	PI15	PI14	PI13	PI12	PI11	PI10
Reset	s	s	s	s	s	s	s	s
Access	r	r	r	r	r	r	r	r

r = read, s = dependent on pin's state

Note: The reset value for this register is dependent on the logical states of the pins.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

Bits 7 to 0: Port 1 Input Register Bits 7:0 (PI1[7:0]). Port 1 is an enhanced Type D I/O port. The PI1 register always reflects the logic state of its pins when read.

5.2.7 External Interrupt Enable Register (Port 0) (EIE0)

Register Description: **External Interrupt Enable Register (Port 0)**

Register Name: **EIE0**

Register Address: **Module 00h, Index 0Bh**

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	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

r = read, w = write

Note: The reset value for this register is dependent on the logical states of the pins.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

Bit 7: Enable External Interrupt 7 (EX7). Setting this bit to 1 enables external interrupt 7. Clearing this bit to 0 disables the interrupt function.

Bit 6: Enable External Interrupt 6 (EX6). Setting this bit to 1 enables external interrupt 6. Clearing this bit to 0 disables the interrupt function.

Bit 5: Enable External Interrupt 5 (EX5). Setting this bit to 1 enables external interrupt 5. Clearing this bit to 0 disables the interrupt function.

Bit 4: Enable External Interrupt 4 (EX4). Setting this bit to 1 enables external interrupt 4. Clearing this bit to 0 disables the interrupt function.

Bit 3: Enable External Interrupt 3 (EX3). Setting this bit to 1 enables external interrupt 3. Clearing this bit to 0 disables the interrupt function.

Bit 2: Enable External Interrupt 2 (EX2). Setting this bit to 1 enables external interrupt 2. Clearing this bit to 0 disables the interrupt function.

Bit 1: Enable External Interrupt 1 (EX1). Setting this bit to 1 enables external interrupt 1. Clearing this bit to 0 disables the interrupt function.

Bit 0: Enable External Interrupt 0 (EX0). Setting this bit to 1 enables external interrupt 0. Clearing this bit to 0 disables the interrupt function.

5.2.8 External Interrupt Enable Register (Port 1) (EIE1)

Register Description: **External Interrupt Enable Register (Port 1)**
 Register Name: **EIE1**
 Register Address: **Module 00h, Index 0Ch**

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	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

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

Bit 7: Enable External Interrupt 7 (EX7). Setting this bit to 1 enables external interrupt 7. Clearing this bit to 0 disables the interrupt function.

Bit 6: Enable External Interrupt 6 (EX6). Setting this bit to 1 enables external interrupt 6. Clearing this bit to 0 disables the interrupt function.

Bit 5: Enable External Interrupt 5 (EX5). Setting this bit to 1 enables external interrupt 5. Clearing this bit to 0 disables the interrupt function.

Bit 4: Enable External Interrupt 4 (EX4). Setting this bit to 1 enables external interrupt 4. Clearing this bit to 0 disables the interrupt function.

Bit 3: Enable External Interrupt 3 (EX3). Setting this bit to 1 enables external interrupt 3. Clearing this bit to 0 disables the interrupt function.

Bit 2: Enable External Interrupt 2 (EX2). Setting this bit to 1 enables external interrupt 2. Clearing this bit to 0 disables the interrupt function.

Bit 1: Enable External Interrupt 1 (EX1). Setting this bit to 1 enables external interrupt 1. Clearing this bit to 0 disables the interrupt function.

Bit 0: Enable External Interrupt 0 (EX0). Setting this bit to 1 enables external interrupt 0. Clearing this bit to 0 disables the interrupt function.

5.2.9 Port 0 Direction Register (PD0)

Register Description: **Port 0 Direction Register**
 Register Name: **PD0**
 Register Address: **Module 00h, Index 10h**

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	PD07	PD06	PD05	PD04	PD03	PD02	PD01	PD00
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

Bits 7 to 0: Port 0 Direction Register Bits 7:0 (PD0[7:0]). Port 0 is an enhanced Type D I/O port. The PD0 register is used to determine the direction of each pin that makes up the port. The port pins are independently controlled by their direction bit. When a bit in PD0 is set to 1, its corresponding pin is enabled as an output. The data value in the respective bit of the PO register will be driven on the pin. When a bit in PD0 is cleared to 0, its corresponding pin is available as an input, and allows an external signal to drive the pin. Note that each port pin has weak pullup and pulldown resistors when functioning as an input, which is controlled by the respective PO bit. If the PO bit is set to 1, this feature is enabled; if the PO bit is cleared to 0, this feature is disabled and the port pin is in high-impedance three-state.

5.2.10 Port 1 Direction Register (PD1)

Register Description: **Port 1 Direction Register**
 Register Name: **PD1**
 Register Address: **Module 00h, Index 11h**

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	PD17	PD16	PD15	PD14	PD13	PD12	PD11	PD10
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

Bits 7 to 0: Port 1 Direction Register Bits 7:0 (PD1[7:0]). Port 1 is an enhanced Type D I/O port. PD1 is used to determine the direction of each pin that makes up the port. The port pins are independently controlled by their direction bit. When a bit in PD1 is set to 1, its corresponding pin is used as an output. The data value in the respective bit of the PO register will be driven on the pin. When a bit in PD1 is cleared to 0, its corresponding pin is available as an input, and allows an external signal to drive the pin. Note that each port pin has weak pullup and pulldown resistors when functioning as an input, which is controlled by the respective PO bit. If the PO bit is set to 1, this feature is enabled; if the PO bit is cleared to 0, this feature is disabled and the port pin is in high-impedance three-state.

5.2.11 External Interrupt Edge Select Register (Port 0) (EIES0)

Register Description: **External Interrupt Edge Select Register (Port 0)**

Register Name: **EIES0**

Register Address: **Module 00h, Index 13h**

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	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

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

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

IT7 = 0: External interrupt 7 is positive-edge triggered.

IT7 = 1: External interrupt 7 is negative-edge triggered.

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

IT6 = 0: External interrupt 6 is positive-edge triggered.

IT6 = 1: External interrupt 6 is negative-edge triggered.

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

IT5 = 0: External interrupt 5 is positive-edge triggered.

IT5 = 1: External interrupt 5 is negative-edge triggered.

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

IT4 = 0: External interrupt 4 is positive-edge triggered.

IT4 = 1: External interrupt 4 is negative-edge triggered.

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

IT3 = 0: External interrupt 3 is positive-edge triggered.

IT3 = 1: External interrupt 3 is negative-edge triggered.

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

IT2 = 0: External interrupt 2 is positive-edge triggered.

IT2 = 1: External interrupt 2 is negative-edge triggered.

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

IT1 = 0: External interrupt 1 is positive-edge triggered.

IT1 = 1: External interrupt 1 is negative-edge triggered.

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

IT0 = 0: External interrupt 0 is positive-edge triggered.

IT0 = 1: External interrupt 0 is negative-edge triggered.

5.2.12 External Interrupt Edge Select Register (Port 1) (EIES1)

Register Description: **External Interrupt Edge Select Register (Port 1)**

Register Name: **EIES1**

Register Address: **Module 00h, Index 14h**

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	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

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

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

IT7 = 0: External interrupt 7 is positive-edge triggered.

IT7 = 1: External interrupt 7 is negative-edge triggered.

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

IT6 = 0: External interrupt 6 is positive-edge triggered.

IT6 = 1: External interrupt 6 is negative-edge triggered.

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

IT5 = 0: External interrupt 5 is positive-edge triggered.

IT5 = 1: External interrupt 5 is negative-edge triggered.

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

IT4 = 0: External interrupt 4 is positive-edge triggered.

IT4 = 1: External interrupt 4 is negative-edge triggered.

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

IT3 = 0: External interrupt 3 is positive-edge triggered.

IT3 = 1: External interrupt 3 is negative-edge triggered.

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

IT2 = 0: External interrupt 2 is positive-edge triggered.

IT2 = 1: External interrupt 2 is negative-edge triggered.

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

IT1 = 0: External interrupt 1 is positive-edge triggered.

IT1 = 1: External interrupt 1 is negative-edge triggered.

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

IT0 = 0: External interrupt 0 is positive-edge triggered.

IT0 = 1: External interrupt 0 is negative-edge triggered.

In addition to the standard enhanced Type D GPIO port features, the MAXQ7667 offers programmable drive strength and programmable direction of resistive pullup or pulldown.

5.2.13 Pad Drive Strength Register (Port 0) (PS0)

Register Description: **Pad Drive Strength Register (Port 0)**
 Register Name: **PS0**
 Register Address: **Module 00h, Index 18h**

Bit #	7	6	5	4	3	2	1	0
Name	PS07	PS06	PS05	PS04	PS03	PS02	PS01	PS00
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 7 to 0: Port 0 Drive Strength Register Bits 7:0 (PS0[7:0]). When the port direction register PD0 configures a particular I/O pin as an output, the corresponding PS0 bit configures the drive strength of that output. When a PS0 bit is set to 0, the output driver is 1mA (typical). When set to 1 the output driver is 2mA (typical).

5.2.14 Pad Drive Strength Register (Port 1) (PS1)

Register Description: **Pad Drive Strength Register (Port 1)**
 Register Name: **PS1**
 Register Address: **Module 00h, Index 19h**

Bit #	7	6	5	4	3	2	1	0
Name	PS17	PS16	PS15	PS14	PS13	PS12	PS11	PS10
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 7 to 0: Port 1 Drive Strength Register Bits 7:0 (PS1[7:0]). When the port direction register PD1 configures a particular I/O pin as an output, the corresponding PS1 bit configures the drive strength of that output. When a PS0 bit is set to 0, the output driver is 1mA (typical). When set to 1 the output driver is 2mA (typical).

5.2.15 Pad Resistive Pull Direction Register (Port 0) (PR0)

Register Description: **Pad Resistive Pull Direction Register (Port 0)**
 Register Name: **PR0**
 Register Address: **Module 00h, Index 1Bh**

Bit #	7	6	5	4	3	2	1	0
Name	PR07	PR06	PR05	PR04	PR03	PR02	PR01	PR00
Reset	1	1	1	1	1	1	1	1
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to FFh on all forms of reset.

Bits 7 to 0: Port 0 Resistive Select Bits 7:0 (PR0[7:0]). When the port direction register PD0 configures a particular I/O pin as an input, and the corresponding PO0 register bit enables a resistive pull, PR0 controls the direction of that pull. If the PR0 bit is set to 0, the I/O pin has a resistive pulldown of approximately 150kΩ (typical). When the PR0 bit is set to 1, the I/O pin has a resistive pullup of approximately 150kΩ (typical).

5.2.16 Pad Resistive Pull Direction Register (Port 1) (PR1)

Register Description: **Pad Resistive Pull Direction Register (Port 1)**
 Register Name: **PR1**
 Register Address: **Module 00h, Index 1Ch**

Bit #	7	6	5	4	3	2	1	0
Name	PR17	PR16	PR15	PR14	PR13	PR12	PR11	PR10
Reset	1	1	1	1	1	1	1	1
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to FFh on all forms of reset.

Bits 7 to 0: Port 1 Resistive Select Bits 7:0 (PR1[7:0]). When the port direction register PD1 configures a particular I/O pin as an input, and the corresponding PO1 register bit enables a resistive pull, PR1 controls the direction of that pull. If the PR1 bit is set to 0, the I/O pin has a resistive pulldown of approximately 150kΩ (typical). When the PR1 bit is set to 1, the I/O pin has a resistive pullup of approximately 150kΩ (typical).

5.3 GPIO Operation

From a software perspective, the MAXQ7667 ports appear as a group of peripheral registers with unique addresses and are addressed as a byte or 8 individual bit locations. The ports are designed to provide programming flexibility for the user application.

- All individual I/O bits are independently configured.
- Any combination of input, output, alternate, or special function in a port is permitted.
- All I/O pins have protection circuitry to DVDDIO and ground.
- When configured as input, the resistive pull direction of 150kΩ can be controlled.
- Output strength on all I/O pins can be controlled to either 1mA (typical) or 2mA (typical).

5.3.1 Port Direction Control and Input/Output

The port direction registers (PD0 and PD1) control the MAXQ7667's respective port pins' (P0 and P1) input/output direction. The port input registers (PI0 and PI1) are read-only registers that always reflect the logic state on the pins. The port output registers (PO0 and PO1) have dual function. For pins defined as output, the port output registers store output data and for pins defined as input, the registers control whether the internal weak pullup is enabled or disabled. Table 5-3 shows the input/output states of the port pins as controlled by the data direction register, output register, pad drive strength register, and pad resistive pull direction register. The table also shows the port pins' (P0 and P1) input/output states in standard mode (no special or alternate function enabled).

Table 5-3. Port Pin Input/Output States (in Standard Mode)

PORT PIN MODE	PDn.x	POn.x	PRn.x	PSn.x	PORT PIN (Pn.x) STATE
Input	0	0	X	X	Three-state
			X	X	Three-state
	0	1	0 (150kΩ pulldown)	X	150kΩ pulldown
			1 (150kΩ pullup)	X	150kΩ pullup
Output	1	0	X	0 (1mA drive strength)	Low (1mA)
			X	1 (2mA drive strength)	Low (2mA)
	1	1	X	0 (1mA drive strength)	High (1mA)
			X	1 (2mA drive strength)	High (1mA)

n = ports, x = pins

The ports (P0 and P1) can be used to support applications that require open-drain/open-source functionality. This can be achieved by using the PO and PD register of the port.

- Three-state the port pin needed to be open drain by setting the corresponding PDn.x bit to 0.
- Clear the corresponding POn.x bit to 0.
- Use the corresponding PDn.x bit to drive the port pin function, instead of the POn.x register.

Note that the internal pullup/pulldown has a relatively high impedance (typically ~150kΩ), so a particular system may require a stronger (external) pullup/pulldown to meet the system level needs.

5.3.2 Port P0 and P1 External Interrupts

Each port pin can function as an external interrupt with individual enable, flag, and active edge selection bits.

- External interrupt enable register (EIE0 and EIE1) bits determine if the external interrupt functionality at each pin is enabled or not.
- External interrupt edge select register (EIES0 and EIES1) bits determine if the external interrupt is generated on rising or falling edge of the interrupt pin input.
- External interrupt flag register (EIF0 and EIF1) bits indicate if a valid rising or falling edge has been detected on the interrupt pin input. An interrupt is generated only if the external interrupt functionality is enabled for the pin. Also, global interrupt mask bits IM0 (in the IMR register) and IGE (in the IC register) must be enabled.

Note: The detection of a valid interrupt edge on any of the external interrupt pins can act as a switchback-trigger source, causing the microcontroller to switch back from stop mode to the standard system clock frequency.

5.3.3 Port Pin Special and Alternate Functions

All the MAXQ7667's port pins are multiplexed with special functions as listed in Table 5-4. All these special functions are disabled by default with the exception of the JTAG interface pins, which are enabled by default following any reset. The behavior of these functions breaks down into two categories:

- Special functions override the data direction register (PD0 and PD1) and port output register (PO1 and PO2) 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 is disabled. Examples of special functions include timer 0 and timer 1 output.
- Alternate functions operate in parallel with the data direction register (PD0 and PD1) and port output register (PO1 and PO2) 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 5-4. Port P0 Pin Special and Alternate Functions

PORT P0 PIN	FUNCTION TYPE	FUNCTION	ENABLED WHEN	MULTIPLEXING/PRIORITIZATION
P0.0/URX	Special	URX—As URX this pin is the receive data input for the UART, which can (optionally) be connected to RXD or a LIN transceiver.	Always	This port pin defaults to a weak pullup input after a reset. This pin can be tied to P0.1/UTX for UART half-duplex operation or connected to a LIN transceiver.
	Alternate	External Interrupt 0, Input	(EIE0.0) EX0 = 1	
P0.1/UTX	Special	UTX—As UTX this pin is the transmit data output of the UART, which can (optionally) be connected to TXD or a LIN transceiver.	SBUF loaded with data	This port pin defaults to a weak pullup input after a reset. When the UART is transmitting, this pin is actively driven with the transmit data; the pin returns to the configured pullup/down state when the UART becomes idle.
	Alternate	External Interrupt 1, Input	(EIE0.1) EX1 = 1	

Table 5-4. Port P0 Pin Special and Alternate Functions (continued)

PORT P0 PIN	FUNCTION TYPE	FUNCTION	ENABLED WHEN	MULTIPLEXING/PRIORITIZATION
P0.2/ $\overline{\text{TXEN}}$	Special	$\overline{\text{TXEN}}$ —As $\overline{\text{TXEN}}$ this pin can be used to control the transmit enable of an external driver (active low). This pin defaults to $\overline{\text{TXEN}}$ any time the UART is used. $\overline{\text{TXEN}}$ is low when the UART is transmitting, and is resistively pulled high/low when it is not.	SBUF loaded with data	This port pin defaults to a weak pullup input after a reset. When the UART is transmitting, this pin is driven low (active low) to denote that the UART is transmitting. When the UART becomes idle the pin returns to the configured pullup/down state (so it should be left as a pullup in order to use this pin to enable an external transceiver).
	Alternate	External Interrupt 2, Input	(EIE0.2) EX2 = 1	
P0.3/T0/ ADCCTL	Special	T0—Timer 0 (Type 2) Output	(T2CNA0.6) T2OE0 = 1	This port pin defaults to a weak pullup input after a reset. If the T2OE0 is set, the pin is a timer output, but the ADCCTL input path still operates (i.e., output timer pulse can come back in as ADCCTL). When the SAR's conversion trigger is selected as "100" this pin is used as the ADCCTL strobe. When SARC.SARS = "101" this pin is inverted and then used as the ADCCTL strobe. The SAR's behavior then depends on SARC.SARDUL, which selects single edge (when 0) or dual edge (when 1) conversions.
	Special	T0—Timer 0 (Type 2) Counter Input	(T2CFG0.0) C/T2 = 1 (T2CFG0[2:1]) CCF[1:0] = 00b (T2CNA0.6) T2OE0 must be 0	
	Special	T0—Timer 0 (Type 2) Gate Input	(T2CNA0.0) G2EN = 1 or (T2CFG0[2:1]) CCF[1:0] = 11b and (T2CNA0.2) CPRL2 = 1 (T2CNA0.6) T2OE0 must be 0 (T2CFG0.0) C/T2 must be 0	
	Special	ADCCTL—As ADCCTL this user-programmable rising or falling edge controls the SAR ADC sampling instant and start of conversion. Optionally, the other edge can be used to enable the ADC and begin acquiring prior to sampling.	SARC.SARS[2:0] = "100" or "101"	
P0.4/T0B	Special	T0B—Timer 0 (Type 2) Secondary Output	(T2CNB0.6) T2OE1 = 1	This port pin defaults to a weak pullup input after a reset. If interrupt 4 is enabled, T0B is not permitted at this pin.
	Alternate	External Interrupt 4, Input	(EIE0.4) EX4 = 1	
P0.5/T1	Special	T1—Timer 1 (Type 2) Output	(T2CNA1.6) T2OE0 = 1	This port pin defaults to a weak pullup input after a reset. If interrupt 5 is enabled, T1 is not permitted at this pin.
	Special	T1—Timer 1 (Type 2) Counter Input	(T2CFG1.0) C/T2 = 1 (T2CFG1[2:1]) CCF[1:0] = 00b (T2CNA1.6) T2OE0 must be 0	
	Special	T1—Timer 1 (Type 2) Gate Input	(T2CNA1.0) G2EN = 1 or (T2CFG1[2:1]) CCF[1:0] = 11b and (T2CNA1.2) CPRL2 = 1 (T2CNA1.6) T2OE0 must be 0 (T2CFG1.0) C/T2 must be 0	
	Alternate	External Interrupt 5, Input	(EIE0.5) EX5 = 1	

Table 5-4. Port P0 Pin Special and Alternate Functions (continued)

PORT P0 PIN	FUNCTION TYPE	FUNCTION	ENABLED WHEN	MULTIPLEXING/PRIORITIZATION
P0.6/T2	Special	T2—Timer 2 (Type 2) Output	(T2CNA2.6) T2OE0 = 1	This port pin defaults to a weak pullup input after a reset. If interrupt 6 is enabled, T2 is not permitted at this pin.
	Special	T2—Timer 2 (Type 2) Counter Input	(T2CFG2.0) C/T2 = 1 (T2CFG2[2:1]) CCF[1:0] = ! 00b (T2CNA2.6) T2OE0 must be 0	
	Special	T2—Timer 2 (Type 2) Gate Input	(T2CNA2.0) G2EN = 1 or (T2CFG2[2:1]) CCF[1:0] = 11b and (T2CNA2.2) CPRL2 = 1 (T2CNA2.6) T2OE0 must be 0 (T2CFG2.0) C/T2 must be 0	
	Alternate	External Interrupt 6, Input	(EIE0.6) EX6 = 1	
P0.7/T2B	Special	T2B—Timer 2 (Type 2) Secondary Output	(T2CNB2.6) T2OE1 = 1	This port pin defaults to a weak pullup input after a reset. If interrupt 7 is enable, T2B is not permitted at this pin.
	Alternate	External Interrupt 7, Input	(EIE0.7) EX7 = 1	

Table 5-5. Port P1 Pin Special and Alternate Functions

PORT P1 PIN	FUNCTION TYPE	FUNCTION	ENABLED WHEN	MULTIPLEXING/PRIORITIZATION
P1.0/TD0	Special	TD0—JTAG Data Out, Output	(SC.7) TAP = 1	The JTAG is the defaulted interface and this port pin is configured as an output and is ready for JTAG interface after a reset. When this pin is used as an external interrupt, the test serial data output function on this pin is disabled. This pin is normally weakly pulled up to DVDDIO unless the JTAG interface is in Shift-DR or Shift-IR states where it is actively putting out data.
	Alternate	External Interrupt 0, Input	(EIE1.0) EX0 = 1	
P1.1/TMS	Special	TMS—JTAG Mode Select, Input	(SC.7) TAP = 1	This port pin defaults to a weak pullup input and is ready for JTAG interface after a reset. No special hardware measure is taken.
	Alternate	External Interrupt 1, Input	(EIE1.1) EX1 = 1	
P1.2/TDI	Special	TDI—JTAG Data In, Input	(SC.7) TAP = 1	This port pin defaults to a weak pullup input and is ready for JTAG interface after a reset. No special hardware measure is taken.
	Alternate	External Interrupt 2, Input	(EIE1.2) EX2 = 1	
P1.3/TCK	Special	TCK—JTAG Clock In, Input	(SC.7) TAP = 1	This port pin defaults to a weak pullup input and is ready for JTAG interface after a reset. No special hardware measure is taken.
	Alternate	External Interrupt 3, Input	(EIE1.3) EX3 = 1	
P1.4/MOSI	Special	MOSI—Master Out-Slave In, Slave Mode, Input	—	This port pin defaults to a weak pullup input after a reset. If interrupt 4 is enabled, MOSI is not permitted at this pin.
	Special	MOSI—Master Out-Slave In, Master Mode, Output	—	
	Alternate	External Interrupt 4, Input	(EIE1.4) EX4 = 1	
P1.5/MISO	Special	MISO—Master In-Slave Out, Master Mode, Input	—	This port pin defaults to a weak pullup input after a reset. If interrupt 5 is enabled, MOSI is not permitted at this pin.
	Special	MISO—Master In-Slave Out, Slave Mode, Output	—	
	Alternate	External Interrupt 5, Input	(EIE1.5) EX5 = 1	

Table 5-5. Port P1 Pin Special and Alternate Functions (continued)

PORT P1 PIN	FUNCTION TYPE	FUNCTION	ENABLED WHEN	MULTIPLEXING/PRIORITIZATION
P1.6/SCLK	Special	SCLK—Serial Clock, Master Mode, Output	—	This port pin is defaulted as a weak pullup input after a reset. If interrupt 14 is enabled, SCLK is not permitted at this pin.
	Special	SCLK—Serial Clock, Slave Mode, Input	—	
	Alternate	External Interrupt 6, Input	(EIE1.6) EX6 = 1	
P1.7/SYNC/SS	Special	SYNC—Resets the Synchronous Timer	(SCNT.8) SSYNC_EN = 1	This port pin is defaulted as a weak pullup input after a reset. If the SPI is enabled as a slave, this pin is the slave select input. If the synchronous timer has SSYNC_EN = 1, the next rising edge on this pin causes a reset of the synchronous timer's count and the clearing of SSYNC_EN. Note that the rising edge detection is performed by synchronizing the input to the CPU clock (after any clock division), and as such a high-going pulse must be greater than 2x TCK to be detected.
	Special	SS—Slave Select for the SPI, Input, Slave Mode	(SPICN.0) SPIEN = 1 (SPICN.1) MSTM = 0 (SPICN.2) MODFE = 0	
	Special	SS—Slave Select for the SPI, Input, Master Mode	(SPICN.0) SPIEN = 1 (SPICN.1) MSTM = 1 (SPICN.2) MODFE = 1	
	Alternate	External Interrupt 7, Input	(EIE1.7) EX7 = 1	

5.3.4 Port Pin Examples

5.3.4.1 Port Pin Example 1: Driving Outputs on Port 0

```

move PS0, #000h           ; Set the output drive strength to 1 mA (approx)
move PO0, #000h           ; Set all outputs low
move PD0, #0FFh           ; Set all P0 pins to output mode

```

5.3.4.2 Port Pin Example 2: Receiving Inputs on Port 0

```

move PO0, #0FFh           ; Set weak pullups ON on all pins
move PD0, #000h           ; Set all P0 pins to input mode

nop                         ; Wait for external source to drive P1

move Acc, PI0              ; Get input values from P0
                           ; (will return FF if no other source
                           ; drives the pins low)

```

SECTION 6: TYPE 2 TIMER/COUNTER MODULE

This section contains the following information:

6.1 Architecture	6-3
6.1.1 Modes of Operation for Type 2 Timer/Counter	6-3
6.1.2 Type 2 Timer/Counter I/O Pins	6-6
6.2 Type 2 Timer/Counter Peripheral Registers	6-7
6.3 Type 2 Status/Control Registers	6-8
6.3.1 Type 2 Timer/Counter Configuration Register (T2CFGx)	6-8
6.3.2 Type 2 Timer/Counter Control Register A (T2CNAX)	6-9
6.3.3 Type 2 Timer/Counter Control Register B (T2CNBx)	6-11
6.4 Type 2 Counter Registers	6-12
6.4.1 Type 2 Timer/Counter Value Register (T2Vx)	6-12
6.4.2 Type 2 Timer/Counter Value High Register (T2Hx)	6-13
6.5 Type 2 Reload Register	6-14
6.5.1 Type 2 Timer/Counter Reload Register (T2Rx)	6-14
6.5.2 Type 2 Timer/Counter Reload High Register (T2RHx)	6-15
6.6 Type 2 Capture/Compare Registers	6-16
6.6.1 Type 2 Timer/Counter Capture/Compare Register (T2Cx)	6-16
6.6.2 Type 2 Timer/Counter Capture/Compare High Register (T2CHx)	6-17
6.7 Type 2 Timer/Counter Operation Modes	6-18
6.7.1 16-Bit Timer: Autoreload/Compare	6-20
6.7.2 16-Bit Timer: Capture Mode	6-21
6.7.3 16-Bit Counter	6-21
6.7.4 Dual 8-Bit Timers	6-22
6.7.5 8-Bit Timer/8-Bit Capture Mode	6-22
6.7.6 8-Bit Timer/8-Bit Counter	6-22
6.7.7 Type 2 Timer Input Clock Selection	6-23
6.7.8 Type 2 Timer Compare Application Examples	6-23
6.7.8.1 A Simple Waveform Output	6-23
6.7.8.2 Waveform Output with Gating	6-25

6.7.9 Type 2 Timer Capture Application Examples	6-27
6.7.9.1 Measure Low-Pulse Duration	6-27
6.7.9.2 Measure High-Pulse Duration Repeatedly	6-28
6.7.9.3 Measure Period	6-29
6.7.9.4 Measure Duty Cycle Repeatedly	6-30
6.7.9.5 Overflow/Interrupt on Cumulative Time	6-31

LIST OF FIGURES

Figure 6-1. Type 2 Timer/Counter in 16-Bit Mode	6-4
Figure 6-2. Type 2 Timer/Counter in 8-Bit Mode	6-5
Figure 6-3. Type 2 Timer Mode Selection	6-19
Figure 6-4. Output Enable and Polarity Control	6-19
Figure 6-5. Type 2 Timer Clock	6-23
Figure 6-6. Simple Waveform Output	6-23
Figure 6-7. Waveform Output with Gating	6-25
Figure 6-8. Type 2 Timer Application Example—Measure Low Pulse Width	6-27
Figure 6-9. Type 2 Timer Application Example—Measure High Pulse Width	6-28
Figure 6-10. Type 2 Timer Application Example—Measure Period	6-29
Figure 6-11. Type 2 Timer Application Example—Measure Duty Cycle	6-30
Figure 6-12. Type 2 Timer Application Example—Overflow/Interrupt on Cumulative Time	6-31

LIST OF TABLES

Table 6-1. Type 2 Timer/Counter Input and Output Pins	6-6
Table 6-2. Type 2 Timer/Counter Register Map	6-7
Table 6-3. Type 2 Timer/Counter Functions and Control	6-18

SECTION 6: TYPE 2 TIMER/COUNTER MODULE

The MAXQ7667 microcontroller has three Type 2 timer/counters. The Type 2 timer/counter is an autoreload 16-bit timer/counter with the following functions:

- 8-bit/16-bit timer/counter
- Up/down autoreload
- Counter function of external pulse
- Capture
- Compare
- Input/output enhancements

The three Type 2 timer/counter modules supported in the MAXQ7667 are referred to as timer 0, timer 1, and timer 2. To simplify the discussion, the generic notation "x" is appended to register and pin names to denote to which timer/counter module they belong (x = 0, 1, 2).

6.1 Architecture

6.1.1 Modes of Operation for Type 2 Timer/Counter

16-Bit Autoreload/Compare Timer: In this mode a timer is generated that can be used internally or sent out via the primary and/or the secondary pin as a waveform or PWM. This mode allows the flexibility to control the waveform output either through firmware or through an external signal on the primary pin.

16-Bit Capture: In this mode, only the primary pin is used, as an input to time an event on a waveform (e.g., duty cycle, period, etc.). No waveform output is allowed.

16-Bit Counter: In this mode, the primary pin is used to count transitions on an input signal. This transition can be a rising edge, a falling edge, or both rising and falling edges. The secondary pin is used to output the generated waveform resulting from compare match and overflow conditions for the counter.

Dual 8-Bit Autoreload Timers: In this mode two timers are generated that can be used internally or delivered as a waveform/PWM to the primary pin and to the secondary pin; otherwise it serves as an internal timer.

8-Bit Capture and 8-Bit Timer/PWM: The primary pin can be used to time an event on a waveform. The secondary timer/compare can be either used internally or delivered as a waveform/PWM to the secondary pin (not available for timer 1).

8-Bit Counter and 8-Bit Timer/PWM: The primary pin can be used to count transitions in an input signal. This transition can be a rising edge, a falling edge, or both rising and falling edge. The secondary timer/compare can be used internally or delivered as a waveform/PWM to the secondary pin (not available for timer 1).

Figure 6-1 shows the 16-bit operation of the timer/counter and Figure 6-2 shows the 8-bit operation of the timer/counter.

The 16-bit Type 2 timer value is contained in the T2Vx register. The upper byte is always accessible through the T2Hx 8-bit register. When the Type 2 timer is operated in the dual 8-bit mode of operation, the high byte of T2Vx always reads x00h and is not write accessible. The low byte of the T2Vx is often referenced as T2Lx. Similar registers and nomenclature are used for the Type 2 timer autoreload register, T2Rx. A separate 8-bit T2RHx register allows read/write access to the high byte. The low byte of T2Rx is often referred to as T2RLx. The capture/compare functionality is supported by the timer through the 16-bit T2Cx capture register and the 8-bit T2CHx high-byte access register.

For convenience, the lower byte of T2Vx is referred to as T2Lx. Unlike T2Hx, there is no separate T2Lx register and the low byte is always accessed through T2Vx. Similarly, the lower byte of T2Rx is referred to as T2RLx and the lower byte of T2Cx is referred to as T2CLx. There are no separate T2RLx and T2CLx registers.

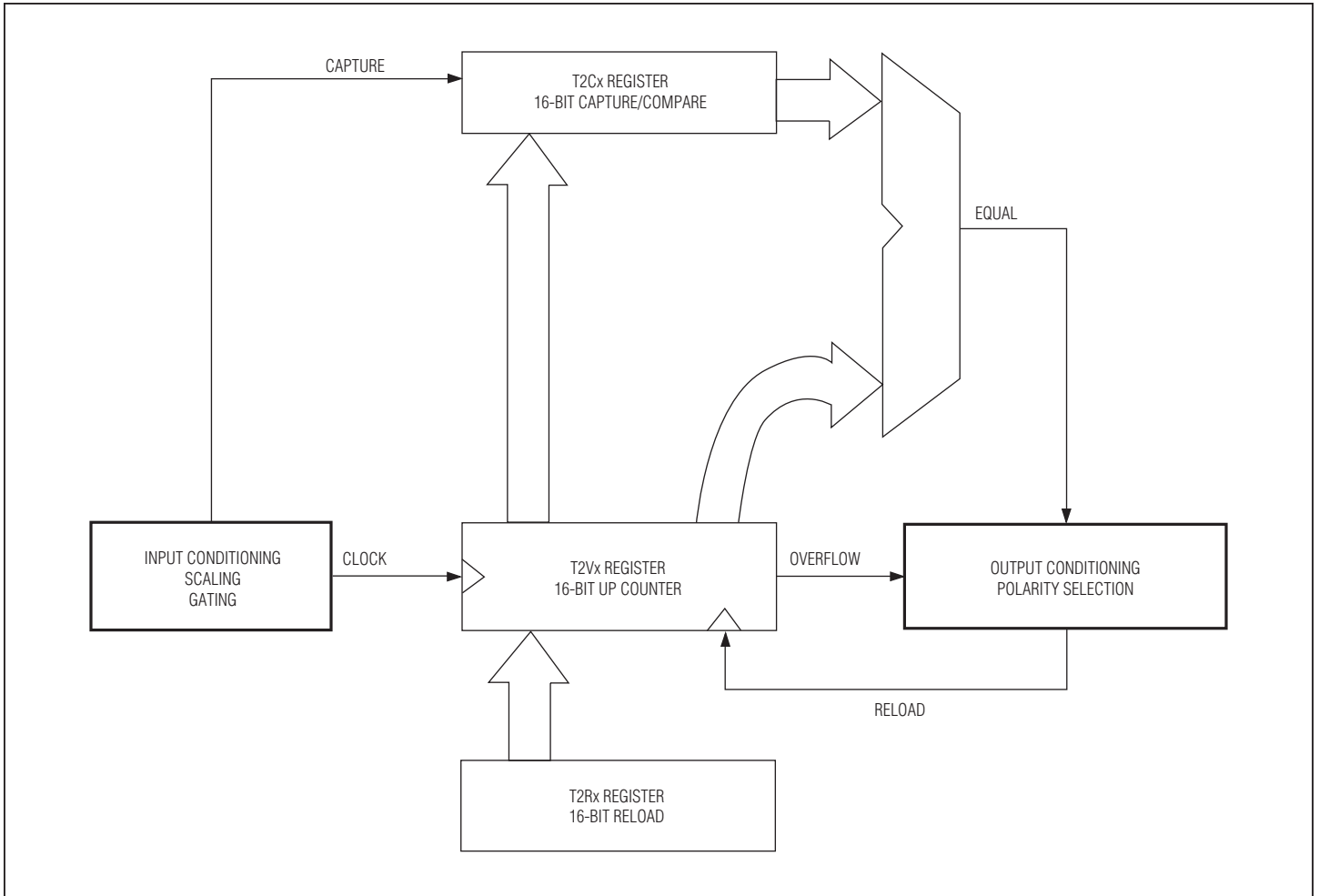


Figure 6-1. Type 2 Timer/Counter in 16-Bit Mode

The input and output conditioning in Figure 6-1 is determined by the status/control registers T2CNx (Type 2 timer/counter control register A), T2CNBx (Type 2 timer/counter control register B), and T2CFGx (Type 2 timer configuration register). See Section 6.2: Type 2 Timer/Counter Peripheral Registers for a detailed discussion of these registers.

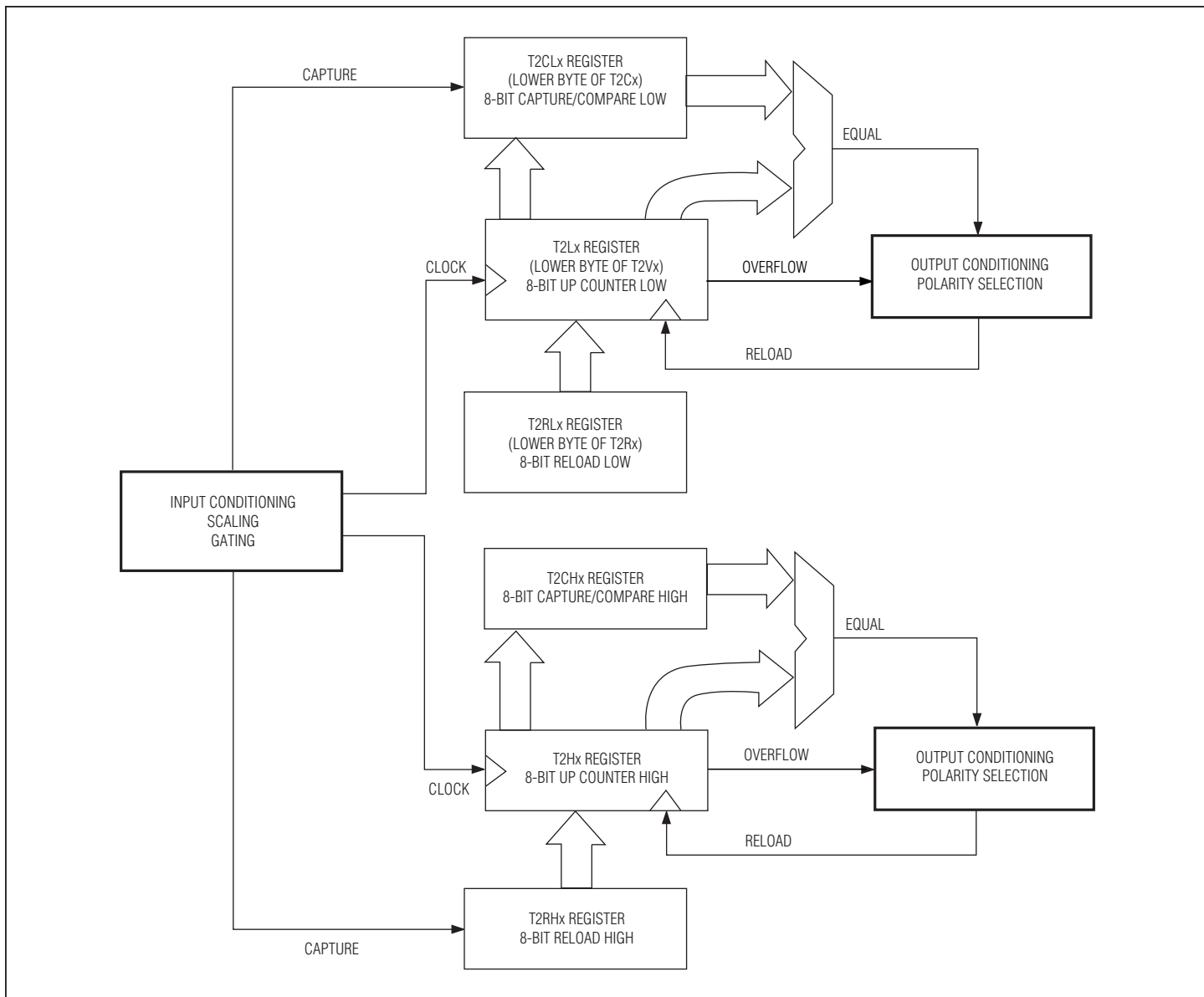


Figure 6-2. Type 2 Timer/Counter in 8-Bit Mode

In Figure 6-2, the input and output conditioning is determined by the status/control registers T2CNAx (Type 2 timer/counter control register A), T2CNBx (Type 2 timer/counter control register B), and T2CFGx (Type 2 timer configuration register). See Section 6.2: Type 2 Timer/Counter Peripheral Registers for a detailed discussion of these registers.

6.1.2 Type 2 Timer/Counter I/O Pins

Each of the three timer/counters, typically, has a pair of pins associated with it to support the enhanced input/output functionality. The pair of pins are referred to as the primary and secondary pins and are designated the symbols T2P0 and T2PB0, respectively. Because there are three Type 2 timers, the pairs are referred to as follows: T2P0, T2PB0; T2P1; T2P2, T2PB2 (secondary pin is not available for timer 1).

A slightly different naming convention is also used for the primary and secondary pins, as shown by the following:

Timer 0: T2P0, T2PB0 = T0, T0B

Timer 1: T2P1 = T1 (no secondary output on T1B)

Timer 2: T2P2, T2PB2 = T2, T2B

To generalize, T2Px, T2PBx = Tx, TxB where x = 0, 1, 2.

Both naming conventions are commonly used for the MAXQ microcontrollers.

Table 6-1. Type 2 Timer/Counter Input and Output Pins

TYPE 2 TIMER/COUNTER FUNCTION	PIN	MULTIPLEXED WITH PORT PIN	FUNCTION
Timer 0 Input/Output—T0 (T2P0)	12	P0.3/ADCCTL	Digital GPIO and ADC Control Input. As T0 this pin is the primary timer/PWM0 input or output. As ADCCTL this user-programmable rising or falling edge controls the SAR ADC sampling instant and start of conversion. Optionally, the other edge can be used to enable the ADC and begin acquiring prior to sampling.
Timer 0 Secondary Output—T0B (T2PB0)	13	P0.4	Digital GPIO, Timer 0 I/O, and Comparator Output. As T0B this pin is the secondary timer/PWM1 input or output. As CMPO this pin is the output of the digital comparator for the lowpass filter.
Timer 1 Input/Output—T1 (T2P1)*	14	P0.5	Digital GPIO and Timer 1 I/O. As T1 this pin is the primary timer/PWM2 input or output.
Timer 2 Input/Output—T2 (T2P2)	15	P0.6	Digital GPIO and Timer 2 I/O. As T2 this pin is the primary timer/PWM2 input or output.
Timer 2 Secondary Output—T2B (T2PB2)	16	P0.7	Digital GPIO and Timer 2 I/O. As T2B this pin is the secondary timer/PWM2 input or output.

*The secondary pin is not supported for timer 1 in the MAXQ7667.

6.2 Type 2 Timer/Counter Peripheral Registers

Table 6-2. Type 2 Timer/Counter Register Map

REGISTER	MODULE NUMBER	INDEX	FUNCTION
T2CFG0	02h	10h	Timer/Counter 0 (Type 2) Configuration Register. Controls counter/timer select, capture/compare function select, 8-/16-bit mode select, and clock divide modes.
T2CNA0	02h	00h	Timer/Counter 0 (Type 2) Control Register A. I/O settings, run enables, polarity modes.
T2CNB0	02h	08h	Timer/Counter 0 (Type 2) Control Register B. Contains capture, compare, overflow flags.
T2V0	02h	09h	Timer/Counter 0 (Type 2) Value Register
T2H0	02h	01h	Timer/Counter 0 (Type 2) Value MSB Register. Provides access to high byte of T2V register.
T2R0	02h	0Ah	Timer/Counter 0 (Type 2) Reload Register
T2RH0	02h	02h	Timer/Counter 0 (Type 2) Reload MSB Register. Provides access to high byte of T2R register.
T2C0	02h	0Bh	Timer/Counter 0 (Type 2) Capture/Compare Register
T2CH0	02h	03h	Timer/Counter 0 (Type 2) Capture/Compare MSB Register. Access to high byte of T2C.
T2CFG1	02h	11h	Timer/Counter 1 (Type 2) Configuration Register. Controls counter/timer select, capture/compare function select, 8-/16-bit mode select, and clock divide modes.
T2CNA1	02h	04h	Timer/Counter 1 (Type 2) Control Register A. I/O settings, run enables, polarity modes.
T2CNB1	02h	0Ch	Timer/Counter 1 (Type 2) Control Register B. Contains capture, compare, overflow flags.
T2V1	02h	0Dh	Timer/Counter 1 (Type 2) Value Register
T2H1	02h	05h	Timer/Counter 1 (Type 2) Value MSB Register. Provides access to high byte of T2V register.
T2R1	02h	0Eh	Timer/Counter 1 (Type 2) Reload Register
T2RH1	02h	06h	Timer/Counter 1 (Type 2) Reload MSB Register. Provides access to high byte of T2R register.
T2C1	02h	0Fh	Timer/Counter 1 (Type 2) Capture/Compare Register
T2CH1	02h	07h	Timer/Counter 1 (Type 2) Capture/Compare MSB Register. Access to high byte of T2C.
T2CFG2	03h	10h	Timer/Counter 2 (Type 2) Configuration Register. Controls counter/timer select, capture/compare function select, 8-bit/16-bit mode select, and clock divide modes
T2CNA2	03h	00h	Timer/Counter 2 (Type 2) Control Register A. I/O settings, run enables, polarity modes
T2CNB2	03h	08h	Timer/Counter 2 (Type 2) Control Register B. Contains capture, compare, overflow flags.
T2V2	03h	09h	Timer/Counter 2 (Type 2) Value Register
T2H2	03h	01h	Timer/Counter 2 (Type 2) Value MSB Register. Provides access to high byte of T2V register.
T2R2	03h	0Ah	Timer/Counter 2 (Type 2) Reload Register
T2RH2	03h	02h	Timer/Counter 2 (Type 2) Reload MSB Register. Provides access to high byte of T2R register.
T2C2	03h	0Bh	Timer/Counter 2 (Type 2) Capture/Compare Register
T2CH2	03h	03h	Timer/Counter 2 (Type 2) Capture/Compare MSB Register. Access to high byte of T2C.

6.3 Type 2 Status/Control Registers

6.3.1 Type 2 Timer/Counter Configuration Register (T2CFGx)

Register Description: **Type 2 Timer/Counter Configuration Register**
 Register Name: **T2CFGx (x = 0, 1, 2)**
 Register Address:

T2CFG0: **Module 02h, Index 10h**
 T2CFG1: **Module 02h, Index 11h**
 T2CFG2: **Module 03h, Index 10h**

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	T2CI	T2DIV2	T2DIV1	T2DIV0	T2MD	CCF1	CCF0	C/T2
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 8: Reserved. Read 0, write ignored.

Bit 7: Type 2 Timer Clock Input Select Bit (T2CI). Setting this bit enables an alternate input clock source to the Type 2 timer block. The alternate input clock selection is the 32kHz clock. The alternate input clock must be sampled by the system clock, which requires that the system clock be at least 4 x 32kHz for proper operation unless the system clock is also source from the 32kHz crystal.

Bits 6 to 4: Type 2 Timer Clock Divide Bits 2:0 (T2DIV[2:0]). These three bits select the divide ratio for the timer clock input clock (as a function of the system clock) when operating in timer mode with T2CI = 0.

T2DIV[2:0]			DIVIDE RATIO
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Bit 3: Type 2 Timer Mode Select (T2MD). This bit enables the dual 8-bit mode of operation. The default reset state is 0, which selects the 16-bit mode of operation. When the dual 8-bit mode is established, the primary timer/counter (T2Hx) carries all the counter/capture functionality, while the secondary 8-bit timer (T2Lx) must operate in timer compare mode, sourcing the defined internal clock.

- 0 = 16-bit mode (default)
- 1 = dual 8-bit mode

Bits 2 and 1: Capture/Compare Function Select Bits (CCF1 and CCF0). These bits, in conjunction with the C/T2 bit, select the basic operating mode of the Type 2 timer. In the dual 8-bit mode of operation (T2MD = 1), the T2Lx timer only operates in compare mode.

CCF1	CCF0	EDGE(s)	C/T2 = 0 (TIMER MODE)	C/T2 = 1 (COUNTER MODE)
0	0	None	Compare Mode	Disabled
0	1	Rising	Capture/Reload	Counter
1	0	Falling	Capture/Reload	Counter
1	1	Rising and Falling	Capture/Reload	Counter

Bit 0: Counter/Timer Select (C/T2). This bit enables/disables the edge counter mode of operation for the 16-bit counter (T2Vx) or the 8-bit counter (T2Hx) when the dual 8-bit mode of operation is enabled (T2MD = 1). The edge for counting (rising/falling/both) is defined by the CCF[1:0] bits.

- 0 = timer mode
- 1 = counter mode

6.3.2 Type 2 Timer/Counter Control Register A (T2CNAx)

Register Description: **Type 2 Timer/Counter Control Register A**
 Register Name: **T2CNAx (x = 0, 1, 2)**
 Register Address:

T2CNA0: **Module 02h, Index 00h**
 T2CNA1: **Module 02h, Index 04h**
 T2CNA2: **Module 03h, Index 00h**

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	ET2	T2OE0	T2POLO	TR2L	TR2	CPRL2	SS2	G2EN
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 8: Reserved. Read 0, write ignored.

Bit 7: Enable Type 2 Timer Interrupts (ET2). This bit serves as the local enable for the Type 2 timer interrupt sources that fall under the TF2 and TCC2 interrupt flags.

Bit 6: Type 2 Timer Output Enable 0 (T2OE0). This register bit enables the timer output function for the external T2P pin. The following table shows the timer output possibilities for the T2P, T2PB pins. (T2OE1 bit is in the T2CNBx register.)

T2OE[1:0]	T2MD	T2P	T2PB
00	X	Port latch data	Port latch data
01	0	16-bit PWM output	Port latch data
10	0	Port latch data	16-bit PWM output
11	0	16-bit PWM output	16-bit PWM output
01	1	8-bit PWM output (T2Hx)	Port latch data
10	1	Port latch data	8-bit PWM output (T2Lx)
11	1	8-bit PWM output (T2Hx)	8-bit PWM output (T2Lx)

Bit 5: Type 2 Timer Polarity Select 0 (T2POL0). When the timer output function has been enabled (T2OE0 = 1), the polarity select bit defines the starting logic level for the T2Px output waveform. When T2POL0 = 0, the starting state for the T2Px output is logic low. When T2POL0 = 1, the starting state for the T2Px output is logic high. The T2POL0 bit can be modified at any time, but takes effect on the external pin when T2OE0 is changed from 0 to 1. When the Type 2 timer pin is being used as an input (T2OE0 = 0), the polarity select bit defines which logic level can be used to gate the timer input clock (when CCF[1:0] = 11b). When CCF[1:0] = 11b, T2POL0 defines which edge can start/stop a single-shot capture and which edge reload can be skipped (if CPRL2 = 1 and G2EN = 1).

Bit 4: Type 2 Timer Low Run Enable (TR2L). This bit start/stops the low 8-bit timer (T2Lx) when dual 8-bit mode (T2MD = 1) is in effect. This bit has no effect when T2MD = 0.

- 0 = timer low stopped
- 1 = timer low run

Bit 3: Type 2 Timer Run Enable (TR2). This bit starts/stops timer 2. In the dual 8-bit mode of operation, this bit applies only to the T2Hx timer/counter. Otherwise, the bit applies to the full 16-bit T2Vx timer/counter. When the timer is stopped (TR2 = 0), the timer registers hold their count. The single-shot bit (SS2) can override and/or delay the effect of the TR2 bit.

- 0 = timer stopped
- 1 = timer run

Bit 2: Capture and Reload Enable (CPRL2). This bit enables a reload (in addition to a capture) on the edge specified by CCF1:CCF0 when operating in capture/reload mode (C/T2 = 0). If both edges are defined for capture/reload (CCF[1:0] = 11b), enabling the gating control (G2EN = 1) allows the T2POL0 bit to be used to prevent a reload on one of the edges if T2POL0 is 0, no reload on the falling edge; or, if T2POL0 is 1, no reload on the rising edge.

- 0 = capture on edge(s) specified by CCF[1:0] bits
- 1 = capture and reload on edge(s) specified by CCF[1:0] bits

Bit 1: Single-Shot (SS2). This bit is used to automatically override or delay the effect of the TR2 bit setting. The single-shot bit is only useful in the timer mode of operation (C/T2 = 0) and should not be set to 1 when the counter mode of operation is enabled (C/T2 = 1).

Compare Mode: If SS2 is written to 1 while in compare mode, one cycle of the defined waveform (reload to overflow) is output to the T2Px, T2PBx pins as prescribed by T2POL[1:0] and T2OE[1:0] controls. The only time that this does not immediately occur is when a gating condition is also defined. If a gating condition is defined, the single-shot cycle cannot occur until the gating condition is removed. If the specified nongated level is already in effect, the single-shot period starts. The gated single-shot output is not supported in dual 8-bit mode.

Capture Mode: If SS2 is written to 1 while in capture mode, the timer is halted and the single-shot capture cycle does not begin until 1) the edge specified by CCF[1:0] is detected, or 2) the defined gating condition is removed. Once running, the timer continues running (as allowed by the gate condition) until the defined capture single-shot edge is detected. In this way, the SS2 bit can be used to delay the running of a timer until an edge is detected (setting both SS2 and TR2 = 1) or override the TR2 = 0 bit setting for one capture cycle (setting only SS2 = 1). When both edges are defined for capture (CCF[1:0] = 11b), the T2POL0 bit serves to define the single-shot start/end edge: falling edge if T2POL0 = 1, rising edge if T2POL0 = 0. No interrupt flag is set when the starting edge for the single-shot capture cycle is detected. The single-shot capture cycle always ends when the next single-shot edge is detected. The start/end edge is defined by T2POL0. This bit is intended to automate pulse-width measurement (low or high) and duty cycle/period measurement.

Bit 0: Gating Enable (G2EN). This bit enables the external T2Px pin to gate the input clock to the 16-bit (T2MD = 0) or highest 8-bit (T2MD = 1) timer. Gating uses T2Px as an input, so it can only be used when T2OE0 = 0 and C/T2 = 0. Gating is not possible on the low 8-bit timer (T2Lx) when the Type 2 timer is operated in dual 8-bit mode. Gating is not supported for counter mode operation (C/T2 = 1). The G2EN bit serves a different purpose when capture and reload have been defined for both edges (CCF[1:0] = 11b and CPRL2 = 1). For this special case, setting G2EN = 1 allows the T2POL0 bit to specify which edge does not cause a reload. If T2POL0 is 0, there is no reload on the falling edge; if T2POL0 is 1, there is no reload on the rising edge.

- 0 = gating disabled
- 1 = gating enabled

6.3.3 Type 2 Timer/Counter Control Register B (T2CNBx)

Register Description: **Type 2 Timer/Counter Control Register B**
 Register Name: **T2CNBx (x = 0, 1, 2)**
 Register Address:

T2CNB0: **Module 02h, Index 08h**
 T2CNB1: **Module 02h, Index 0Ch**
 T2CNB2: **Module 03h, Index 08h**

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	ET2L	T2OE1*	T2POL1*	X	TF2	TF2L	TCC2	TC2L
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	r	rw	rw	rw	rw

r = read, w = write
 *This bit is not available for timer 1.

Bits 15 to 8: Reserved. Read 0, write ignored.

Bit 7: Enable Type 2 Timer Low Interrupts (ET2L). This bit serves as the local enable for timer low interrupt sources that fall under the TF2L and TC2L interrupt flags.

Bit 6: Type 2 Timer Output Enable 1 (T2OE1). See the table given for bit 6 (T2OE0) in the T2CNAx description. The T2OE1 bit is not implemented for single pin versions of the Type 2 timer, hence it is not available for timer 1.

Bit 5: Type 2 Timer Polarity Select 1 (T2POL1). When the T2Px output is enabled (T2OE1 = 1), this bit selects the starting logic level for the alternate pin output. The output that is driven on the T2PBx pin can be derived from the 16-bit timer or the 8-bit timer (T2Lx), depending upon whether operating in the 16-bit mode or the dual 8-bit mode. The T2POL1 bit can be modified at any time, but takes effect on the external pin when T2OE1 is changed from 0 to 1. This bit is not available for timer 1.

Bit 4: Don't Care (X). Reserved. Read 0, write ignored.

Bit 3: Type 2 Timer Overflow Flag (TF2). This flag becomes set anytime there is an overflow of the full 16-bit T2Vx timer/counter (when T2MD = 0) or an overflow of the 8-bit T2Hx timer/counter when the dual 8-bit mode of operation is selected (T2MD = 1).

Bit 2: Type 2 Timer Low Overflow Flag (TF2L). This flag is meaningful only when in the dual 8-bit mode of operation (T2MD = 1). It is set whenever there is an overflow of the T2Lx 8-bit timer.

Bit 1: Type 2 Timer Capture/Compare Flag (TCC2). This flag is set on any compare match between the Type 2 timer value and compare register (T2Vx = T2Cx or T2Hx = T2CHx, respectively, for 16-bit and 8-bit compare modes) or when a capture event is initiated by an external edge.

Bit 0: Type 2 Timer Low Compare Flag (TC2L). This flag is meaningful only for the dual 8-bit mode of operation (T2MD = 1). It is set only when a compare match occurs between T2CLx and T2Lx. The Type 2 timer low does not have an associated capture function.

6.4 Type 2 Counter Registers

6.4.1 Type 2 Timer/Counter Value Register (T2Vx)

Register Description: **Type 2 Timer/Counter Value Register**
 Register Name: **T2Vx (x = 0, 1, 2)**
 Register Address:
 T2V0: **Module 02h, Index 09h**
 T2V1: **Module 02h, Index 0Dh**
 T2V2: **Module 03h, Index 09h**

Bit #	15	14	13	12	11	10	9	8
Name	T2Vx15	T2Vx14	T2Vx13	T2Vx12	T2Vx11	T2Vx10	T2Vx9	T2Vx8
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	T2Vx7	T2Vx6	T2Vx5	T2Vx4	T2Vx3	T2Vx2	T2Vx1	T2Vx0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 0: Type 2 Timer/Counter Value Register (T2Vx[15:0]). The T2Vx register is a 16-bit register that holds the current timer value. When operating in 16-bit mode (T2MD = 0), the full 16 bits are read/write accessible. If the dual 8-bit mode of operation (T2MD = 1) is selected, the upper byte of T2Vx is inaccessible. T2Vx reads while in the dual 8-bit mode return 00h as the high byte and writes to the upper byte of T2Vx are blocked. A separate T2Hx register is provided to facilitate high-byte access for dual 8-bit mode.

Note: For convenience, the lower byte of T2Vx (T2Vx[7:0]) is referred to as T2Lx. Unlike T2Hx, there is no separate T2Lx register and the low byte is always accessed through T2Vx.

6.4.2 Type 2 Timer/Counter Value High Register (T2Hx)

Register Description: **Type 2 Timer/Counter Value High Register**

Register Name: **T2Hx (x = 0, 1, 2)**

Register Address:

T2H0: **Module 02h, Index 01h**

T2H1: **Module 02h, Index 05h**

T2H2: **Module 03h, Index 01h**

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	T2Hx7	T2Hx6	T2Hx5	T2Hx4	T2Hx3	T2Hx2	T2Hx1	T2Hx0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 8: Reserved.

Bits 7 to 0: Type 2 Timer/Counter Value High Register (T2Hx[7:0]). This register is used to load and read the most significant 8-bit value in the timer.

6.5 Type 2 Reload Register

6.5.1 Type 2 Timer/Counter Reload Register (T2Rx)

Register Description: **Type 2 Timer/Counter Reload Register**
 Register Name: **T2Rx (x = 0, 1, 2)**
 Register Address:

T2R0: **Module 02h, Index 0Ah**
 T2R1: **Module 02h, Index 0Eh**
 T2R2: **Module 03h, Index 0Ah**

Bit #	15	14	13	12	11	10	9	8
Name	T2Rx15	T2Rx14	T2Rx13	T2Rx12	T2Rx11	T2Rx10	T2Rx9	T2Rx8
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	T2Rx7	T2Rx6	T2Rx5	T2Rx4	T2Rx3	T2Rx2	T2Rx1	T2Rx0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 0: Type 2 Timer/Counter Reload Register (T2Rx[15:0]). This 16-bit register holds the reload value for the timer. When operating in 16-bit mode (T2MD = 0), the full 16 bits are read/write accessible. If the dual 8-bit mode of operation is selected, the upper byte of T2Rx is inaccessible. T2Rx reads while in the dual 8-bit mode return 00h as the high byte and writes to the upper byte of T2Rx are blocked. A separate T2RHx register is provided to facilitate high-byte access for the dual 8-bit mode.

Note: For convenience, the lower byte of T2Rx (T2Rx[7:0]) is referred to as T2RLx. Unlike T2RHx, there is no separate T2RLx register and the low byte is always accessed through T2Rx.

6.5.2 Type 2 Timer/Counter Reload High Register (T2RHx)

Register Description: **Type 2 Timer/Counter Reload High Register**

Register Name: **T2RHx (x = 0, 1, 2)**

Register Address:

T2RH0: **Module 02h, Index 02h**

T2RH1: **Module 02h, Index 06h**

T2RH2: **Module 03h, Index 02h**

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	T2RHx7	T2RHx6	T2RHx5	T2RHx4	T2RHx3	T2RHx2	T2RHx1	T2RHx0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 8: Reserved.

Bits 7 to 0: Type 2 Timer/Counter Reload High Register (T2RHx[7:0]). This register is used to load and read the most significant 8-bit reload value in the timer.

6.6 Type 2 Capture/Compare Registers

6.6.1 Type 2 Timer/Counter Capture/Compare Register (T2Cx)

Register Description: **Type 2 Timer/Counter Capture/Compare Register**
 Register Name: **T2Cx (x = 0, 1, 2)**
 Register Address:

T2C0: **Module 02h, Index 0Bh**
 T2C1: **Module 02h, Index 0Fh**
 T2C2: **Module 03h, Index 0Bh**

Bit #	15	14	13	12	11	10	9	8
Name	T2Cx15	T2Cx14	T2Cx13	T2Cx12	T2Cx11	T2Cx10	T2Cx9	T2Cx8
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	T2Cx7	T2Cx6	T2Cx5	T2Cx4	T2Cx3	T2Cx2	T2Cx1	T2Cx0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 0: Type 2 Timer/Counter Capture/Compare Register (T2Cx[15:0]). This 16-bit register holds the compare value when operating in compare mode and gets the capture value when operating in capture mode. When operating in 16-bit mode (T2MD = 0), the full 16 bits are read/write accessible. If the dual 8-bit mode of operation is selected, the upper byte of T2Cx is inaccessible. T2Cx reads while in the dual 8-bit mode return 00h as the high byte and writes to the upper byte of T2Cx are blocked. A separate T2CHx register is provided to facilitate high-byte access.

Note: For convenience, the lower byte of T2Cx (T2Cx[7:0]) is referred to as T2CLx. Unlike T2CHx, there is no separate T2CLx register and the low byte is always accessed through T2Cx.

6.6.2 Type 2 Timer/Counter Capture/Compare High Register (T2CHx)

Register Description: **Type 2 Timer/Counter Capture/Compare High Register**

Register Name: **T2CHx (x = 0, 1, 2)**

Register Address:

T2CH0: **Module 02h, Index 03h**

T2CH1: **Module 02h, Index 07h**

T2CH2: **Module 03h, Index 03h**

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	T2CHx7	T2CHx6	T2CHx5	T2CHx4	T2CHx3	T2CHx2	T2CHx1	T2CHx0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 8: Reserved.

Bits 7 to 0: Type 2 Timer/Counter Capture/Compare High (T2CHx[7:0]). This register is used to load and read the most significant 8-bit capture/compare value of the timer.

6.7 Type 2 Timer/Counter Operation Modes

The MAXQ7667 Type 2 timer/counter supports six operation modes. Table 6-3 summarizes the modes supported by the Type 2 timer and the peripheral register bits associated with those modes.

The Type 2 timer operating mode selection is illustrated in Figure 6-3. Figure 6-4 shows the PWM timer output possibilities.

Table 6-3. Type 2 Timer/Counter Functions and Control

MODE	T2MD	C/T2	CCF[1:0]	CONTROL BITS
16-Bit Autoreload/Compare Timer	0	0	00	T2OE1:0: Output enables (PWM out) T2POL1:0: Output polarity select T2POL0: Defines gating level (when set to input) SS2: Single-shot pulse control G2EN: Gated PWM output on secondary pin, primary pin is used for gating signal
16-Bit Capture (CCF[1:0] bits define capture edge)	0	0	01, 10, or 11	T2OE0 = 0 T2POL0: Gate level/reload edge select SS2: Single-shot capture G2EN: Gate timer clock (or gate reload) CPRL2: Reload enable T2OE1: Not used
16-Bit Counter (CCF[1:0] bits define count edge)	0	1	01, 10, or 11	T2OE0 = 0 T2OE1: Pulse counter output enable T2POL1: Output polarity select
Dual 8-Bit Autoreload Timers	1	0	00	T2OE1:0: Output enables (PWM out) T2POL1:0: Output polarity select (primary for gating) <u>T2Hx Only:</u> SS2: Single-shot pulse control
8-Bit Capture and 8-Bit Timer/PWM	1	0	01, 10, or 11	<u>T2Lx Only:</u> T2OE1: Output enable T2POL1: Output polarity select <u>T2Hx Only:</u> T2OE0 = 0 T2POL0: Gate level/reload edge select SS2: Single-shot capture G2EN: Gate timer (or gate reload) CPRL2: Reload enable
8-Bit Counter and 8-Bit Timer/PWM	1	1	01, 10, or 11	<u>T2Lx Only:</u> T2OE1: Output enable T2POL1: Output polarity select <u>T2Hx Only:</u> T2OE[0] = 0

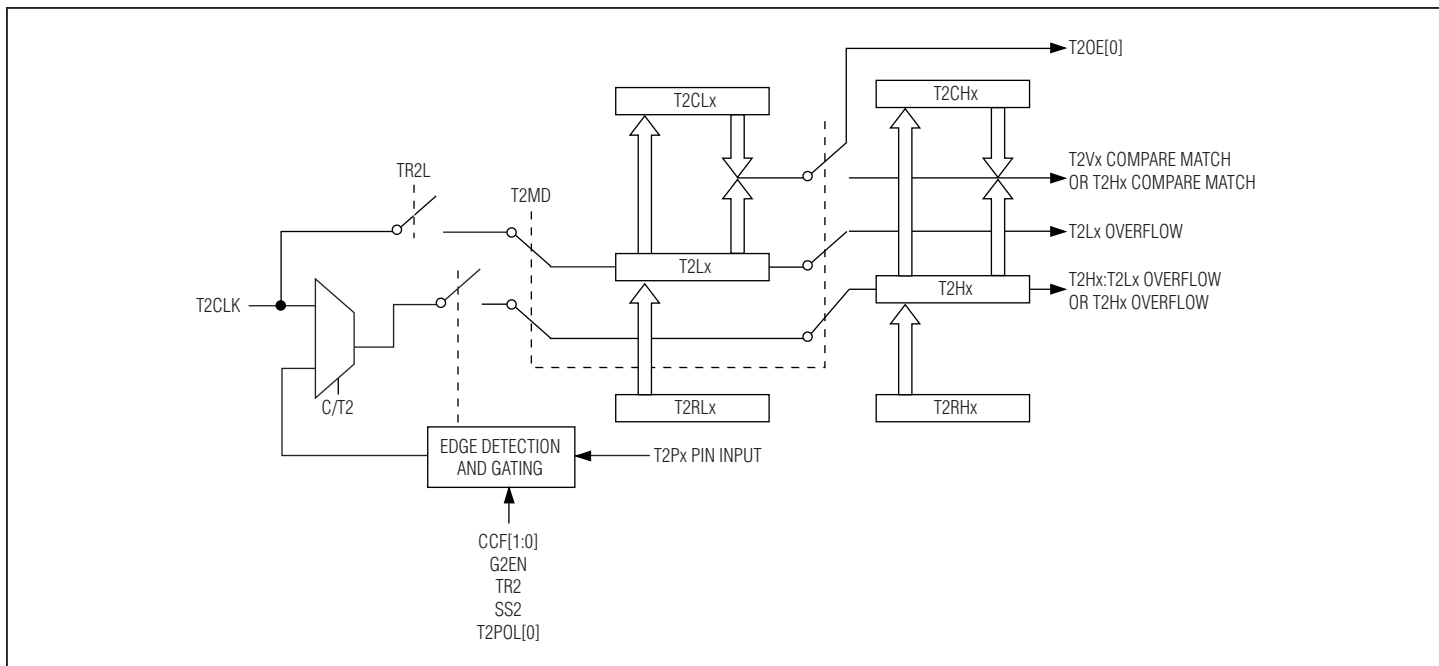


Figure 6-3. Type 2 Timer Mode Selection

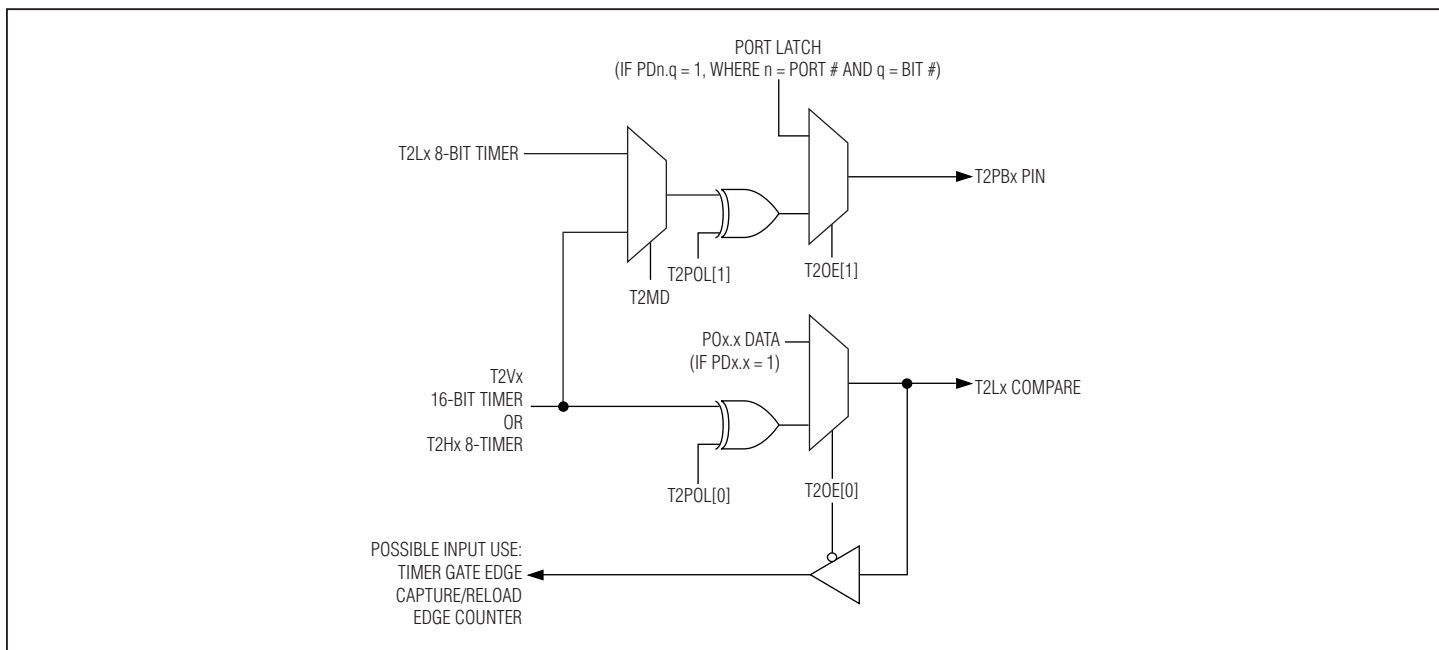


Figure 6-4. Output Enable and Polarity Control

6.7.1 16-Bit Timer: Autoreload/Compare

The 16-bit autoreload/compare mode for the Type 2 timer is in effect when the timer-mode select bit (T2MD) is cleared and the capture/compare function definition bits are both cleared (CCF[1:0] = 00b). The timer value is contained in the T2Vx register. The timer run control bit (TR2) starts and stops the 16-bit timer. The input clock for the 16-bit Type 2 timer is defined as the system clock divided by the ratio specified by the T2DIV[2:0] prescale bits. The timer begins counting from the value contained in the T2Vx register until an overflow occurs. When an overflow occurs, the reload value is reloaded instead of the x0000h state. The timer overflow flag (TF2) is set every time that an overflow condition (T2Vx = 0xFFFFh) is detected. If the Type 2 timer interrupts have been enabled (ET2 = 1), the TF2 flag can generate an interrupt request. When operating in compare mode, the capture/compare register, T2Cx, is compared versus the timer value registers. Whenever a compare match occurs, the capture/compare status flag (TCC2) is set. If the Type 2 timer interrupts have been enabled (ET2 = 1), this event can generate an interrupt request. If the capture/compare register is set to a value outside of the timer counting range, a compare match is not signaled and the TCC2 flag is not set. Internally, a timer output clock is generated that toggles on the cycle following any compare match or overflow, unless the compare match value has been set equal to the overflow condition, in which case, only one toggle occurs. This clock is sourced out on the designated timer/counter pins of the microcontroller.

- **Output Enable (PWM Out).** The output enable bits (T2OE[1:0]) enable the timer output clock to be presented on the pins associated with the respective bits. If the Type 2 timer has a single I/O pin, as in the case of timer 1, the T2OE0 bit is associated with the T2Px pin and the T2OE1 bit is not implemented (as it would serve no purpose as in the case of timer 1).
- **Polarity Control.** The polarity control bits (T2POL[1:0]) can be used to modify (invert) the enabled clock outputs to the pin(s). The enabled clock outputs (defined by T2OE[1:0]) will toggle on each compare match or overflow. The T2POL[1:0] bits are logically XORed with the timer output signal, therefore setting a given T2POLn bit will result in a high starting state. The T2POLn bit can be changed any time, however, the assigned T2POLn state will take effect on the external pin only when the corresponding T2OEn bit is changed from 0 to 1. (Timer 1 has no secondary pin, hence polarity control is not available.)
- **Gated.** To use the T2P pin as a timer input clock gate, the T2OE0 bit must be cleared to 0, i.e., it is an input, and the G2EN bit must be set to 1. When T2OE0 = 1, the G2EN bit setting has no effect. When T2OE0 is cleared to 0, the respective polarity control bit is used to modify the polarity of the input signal to the timer. In the gated mode, the timer input clock is gated anytime the external signal matches the state of the T2POL0 bit. This means that the default clock gating condition for the T2P pin is logic low (since T2POL0 = 0 default). Setting T2POL0 = 1 results in the timer input clock being gated when the T2P pin is high. Note that if multiple pins are allocated for the Type 2 timer (i.e., T2P, T2PB), the primary pin can be used for clock gating, while the secondary pin can be used to output the gated PWM output signal (if T2OE1 = 1).
- **Single-Shot (and Gating).** When operating in 16-bit compare mode, the single-shot is used to automate the generation of single pulses under software control or in response to an external signal (single-shot gated). To generate single-shot output pulses solely under software control, the G2EN bit should be cleared to 0, the output enables and polarity controls should be configured as desired, and the single-shot bit should be set to 1. Writing the single-shot bit effectively overrides the TR2 = 0 condition until timer overflow/reload occurs. The single-shot bit is automatically cleared once the overflow/reload occurs.

Writing SS2 and TR2 = 1 at the same time still causes the SS2 bit to stay in effect until an overflow/reload occurs, however, since TR2 was also written to a 1, the specified PWM output continues even after SS2 becomes clear.

If two pins are available for the Type 2 timer implementation, an additional mode is supported: single-shot gated. Single-shot gated requires that the T2P pin be used as an input (T2OE0 = 0). It also requires that G2EN = 1, thus differentiating it from the software-controlled single-shot mode on the second output pin. If G2EN is enabled and SS2 is written to 1, the gating condition must first be removed in order for the single-shot enabled output to occur on the pin. When the clock gate is removed, the single-shot output occurs. Just as previously described, the SS2 bit = 1 state remains in effect until overflow/reload. Note that this makes it possible for the single-shot to span multiple gated/nongated intervals. Once the SS2 = 1 conditions completes, if TR2 = 1, the gated PWM mode is in effect. Otherwise (TR2 = 0), the timer is stopped.

- **Capture/Reload Control.** For the 16-bit compare operating mode, the CPRL2 bit is not used.

6.7.2 16-Bit Timer: Capture Mode

The 16-bit capture mode requires that some event trigger the capture. Normally, this event will be an external edge. The CCF[1:0] bits define which edge(s) causes a capture to occur. If CCF[1:0] = 01b, a rising edge causes a capture. If CCF[1:0] = 10b, a falling edge causes a capture. If CCF[1:0] = 11b, rising and falling edges both cause a capture to occur. The CPRL2 bit enables both capture and reload to occur on the specified edge(s).

- **Output Enable.** In 16-bit capture mode, the output enables are meaningless. No output waveform is allowed since the capture/compare registers are being used for the purpose of capturing the timer value.
- **Polarity Control.** The polarity control bits (T2POL[1:0]) have no specific meaning as related to the output function since there is no output function. The T2POL0 bit is used to establish the gating condition for the single-edge capture mode when gating is enabled (G2EN = 1). If capture and reload are defined (CPRL2 = 1 and CCF[1:0] = 11b) for both edges, the T2POL0 bit can be used to specify which edge does not have an associated edge reload when gating has also been enabled (G2EN bit = 1). When the SS2 bit is used to delay the timer run (for both edge capture), the T2POL0 bit also defines which edge starts/ends the single-shot process.
- **Edge Detection.** Edge detection was described above (CCF[1:0] controlled).
- **Gated.** If gating is specified, it uses the T2POL0 bit to define when the input clock to the timer is gated (just as described for the compare mode). This mode can easily be used to measure or incrementally capture high or low pulse durations. If a pre-defined high/low duration is required to generate an interrupt, the gated compare mode can also be used. Note that if capture is defined for both rising and falling edges, gating would serve no useful purpose because it would result in redundant capture data/interrupts. For this reason, when G2EN = 1 and CCF[1:0] = 11b, the T2POL0 bit is used to specify which edge is a capture-only edge when CPRL2 = 1 (gating of the reload event).
- **Single-Shot.** The single-shot bit overrides the TR2 = 0 bit setting for a single edge-to-edge capture cycle (as defined by the CCF[1:0] bits). The single-shot takes effect (starting the timer) only when 1) the edge defined by CCF[1:0] is detected or 2) the defined gating condition is removed. While a capture and/or reload may occur on this starting edge, the interrupt flag will not be set since a single-shot event has been requested. When rising or falling edge capture is defined, the single-shot mode is useful for measuring single periods. If gating is also specified for the single-shot, the high/low pulse widths are easily measured. If rising **and** falling edges are defined, the T2POL0 bit designates which edge starts/ends the single-shot cycle, but the starting edge will not cause the interrupt flag to set. If G2EN = 1 for the two-edge capture, the alternate edge (opposite of defined start/end edge can only be used for capture, not capture and reload). For T2POL0 = 1, the falling edge starts and stops the single shot. This is important for combined duty cycle and period measurement.
- **Capture and Reload.** The CPRL2 bit enables both capture and reload on the specified edge(s). The only exception to this rule is when the G2EN bit is set to a logic 1. When G2EN is set to 1, a reload does not occur on the edge specified by T2POL0. When T2POL0 = 0, the falling edge does not cause a reload; if T2POL0 = 1, the rising edge does not cause a reload.

6.7.3 16-Bit Counter

The 16-bit counter mode is enabled by setting the C/T2 bit to logic 1. When C/T2 = 1, rising, falling, or both rising and falling edges are counted as determined by the CCF[1:0] bits. If CCF[1:0] = 00b, neither edge is defined as a counted edge, and the T2Vx counter will hold its count since no edge is defined as the counting edge. When an overflow occurs, the reload value (T2Rx) is reloaded instead of the x0000h state. The timer/counter 2 overflow flag (TF2) is set every time that an overflow occurs. If timer/counter 2 interrupts have been enabled (ET2 = 1), the TF2 flag can generate an interrupt request. In counter mode, the capture/compare register (T2Cx) is compared versus the timer/counter 2 value register. Whenever a compare match occurs, the capture/compare status flag (TCC2) is set. If timer/counter 2 interrupts have been enabled (ET2 = 1), this event can generate an interrupt request. If the capture/compare register is set to a value outside of the Type 2 timer counting range, a compare match is not signaled and the TCC2 flag is not set.

- **Output Enable.** For the timer to serve as a counter, the T2Px pin must be used as an input. Thus, when C/T2 = 1, the T2OE0 bit is ignored. The T2OE1 bit can be used to output the generated waveform on T2PB (not available for timer 1) resulting from compare match and overflow conditions for the counter.
- **Polarity Control.** Only the T2POL1 bit is meaningful. It can define the starting state of the T2PBx pin when the T2PB output has been enabled. The T2POL1 bit can be changed any time, however, the assigned T2POL1 state will take effect on the external pin only when the corresponding T2OE1 bit is changed from 0 to 1.
- **Gating and Single-Shot.** Neither gating nor single-shot modes are supported when operating in 16-bit counter mode. The G2EN and SS2 bits should not be set to 1 when operating in the counter mode (C/T2 = 1).

6.7.4 Dual 8-Bit Timers

The dual 8-bit timer mode of operation is initiated by setting the T2MD bit to logic 1. When T2MD = 1, each 16-bit register associated with the Type 2 timer is split into separate upper and lower 8-bit registers to support dual 8-bit timers. Thus, the primary 8-bit timer is composed of T2Hx (value), T2RHx (reload), T2CHx (capture/compare), and the secondary 8-bit timer is composed of T2Lx (value), T2RLx (reload), and T2CLx (capture/compare). There is but a single internal Type 2 timer input clock that can be sourced by either of these two 8-bit timers. In the dual 8-bit mode of operation, both timer output clocks (from T2Lx and T2Hx) are available to internal peripherals as required by a given product. The secondary 8-bit timer/counter has its own run control bit (TR2L) and interrupt flags (TF2L, TC2L). Timer 1 does not support the secondary pin, therefore, it is available internally only.

- **Output Enable (PWM out).** The output enable bits (T2OE[1:0]) enable the respective 8-bit timer outputs to be presented on the pins associated with the respective bits. The T2Hx timer output onto the T2Px pin is controlled by the T2OE0 bit and the T2Lx timer output onto the T2PB pin is controlled by the T2OE1 bit. If the Type 2 timer has a single I/O pin, as in the case of timer 1, only the T2OE0 bit is required as the secondary timer. T2Lx cannot be output to a pin and can only serve as an internal timer.
- **Polarity Control.** The polarity control bits (T2POL[1:0]) can be used to modify (invert) the enabled clock outputs to the pin(s). The starting state of the enabled clock outputs (defined by T2OE[1:0]) is the logic state of T2POL[1:0] and toggles on each compare match or overflow. The T2POL[1:0] bits are logically XORed with the timer output signal, therefore setting a given T2POLn bit results in a high starting state. The T2POLn bit can be changed any time, however the assigned T2POLn state will take effect on the external pin only when the corresponding T2OEn bit is changed from 0 to 1. T2POL1 is not required for a single-pin Type 2 timer implementation.
- **Gated.** To use the T2Px pin as a timer input clock gate, the T2OE0 bit must be cleared to 0 and the G2EN bit must be set to 1. When T2OE0 = 1, the G2EN bit setting has no effect. When T2OE0 is cleared to 0, the respective polarity control bit is used to modify the polarity of the input signal to the timer. In the gated mode, the input clock to T2Hx is gated anytime that the external signal matches the state of the T2POL0 bit. This means that the default clock gating condition is associated with the T2Px pin being low (T2POL0 = 0). Note that the secondary 8-bit timer, T2Lx, cannot be gated. Also, since the output enables, T2OE[1:0] apply to each individual 8-bit timer, there is no gated PWM mode available.
- **Single-Shot.** The single-shot bit and mode apply only to the primary 8-bit timer (T2Hx). The single-shot mode is used to automate the generation of single pulses under software control. To generate single-shot output pulses under software control, the G2EN bit should be cleared to 0, the output enables and polarity controls should be configured as desired, and the single-shot bit should be set to 1. Writing the single-shot bit effectively overrides the TR2 = 0 condition until the timer overflow/reload occurs. Writing SS2 and TR2 = 1 at the same time still causes the SS2 bit to stay in effect until an overflow/reload occurs, however, the specified PWM output continues since TR2 was also written to 1.

6.7.5 8-Bit Timer/8-Bit Capture Mode

When the CCF[1:0] bits are configured to a state other than 00b, the edge capture mode is enabled for the primary timer (T2Hx). The secondary timer (T2Lx) always remains in the timer/compare mode and does not support any capture functionality. The capture controls for the 8-bit mode are identical to those specified for the 16-bit mode, however, they apply only to the upper timer, T2Hx.

One obvious difference is that the secondary timer (T2Lx), operable only in compare mode, can be used to generate a PWM output with valid T2OE1 and T2POL1 controls, while the primary timer is operating in capture mode. (**Note:** Timer 1 does not have a secondary pin, therefore, it is not available.)

6.7.6 8-Bit Timer/8-Bit Counter

Just as in the 16-bit mode, setting the C/T2 bit to logic 1 enables the external T2Px pin to function as a counter input. The edges that are counted are determined by the CCF[1:0] bits. The counter mode of operation applies only to the primary timer/counter (T2Hx). In a similar fashion to the 16-bit counter mode, when an overflow occurs, an autoreload of T2RHx occurs and the TF2 flag is set. The TCC2 flag is also set on a compare match between the T2Hx counter and the T2CHx compare register (except for the case where T2CHx is outside of the T2RHx to 0xFFh counting range). The secondary timer (T2Lx) always continues to operate in 8-bit compare mode. Just as in the above split 8-bit timer/8-bit capture mode, this allows the secondary timer (T2Lx) to function in the PWM output capacity if a T2PBx pin is provided, therefore, it is not available for timer 1. The T2POL1 control still applies to the 8-bit T2Lx PWM output when T2OE1 = 1.

6.7.7 Type 2 Timer Input Clock Selection

The Type 2 timer clock source is illustrated in Figure 6-5. The timer input clock is selected by the T2CI bit while the clock prescale is determined by the T2DIV bits in the T2CFGx register. Note that when T2CI is configured to a 1, the alternate clock source (32kHz) is sampled by the current system clock selection. The maximum frequency that can be sampled on the alternate clock frequency is (system clock/4).

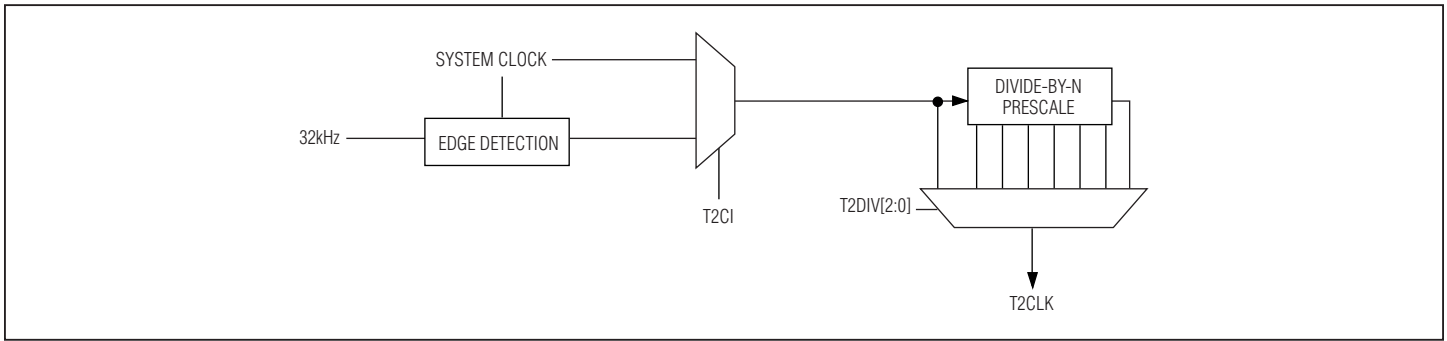


Figure 6-5. Type 2 Timer Clock

6.7.8 Type 2 Timer Compare Application Examples

6.7.8.1 A Simple Waveform Output

The following code will output a waveform on T2P0 and T2P0B of the Type 2 timer that has a duty cycle of 2/3 on the primary pin and 1/3 on the secondary pin. The T2V0 and T2R0 could have been loaded with 0s as well, without affecting the duty cycle but it would change the period and hence the frequency. The code can be enhanced to dynamically change the duty cycle and the frequency of the output signal by continuously updating the values of T2R0 and T2V0 in the body of the code.

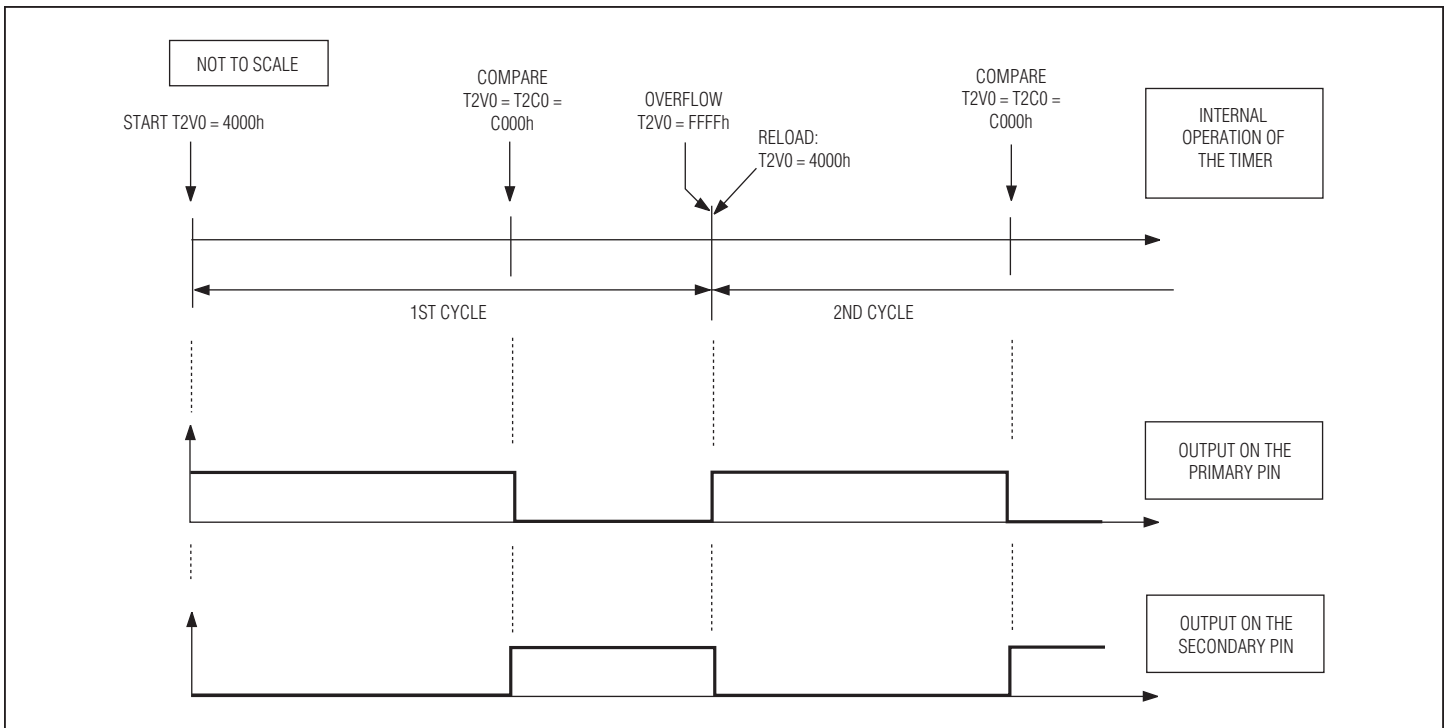


Figure 6-6. Simple Waveform Output

```
org 0
;
main_top:

    move    T2V0, #04000h    ; Set to reload value to keep first pulse from
                            ; being extra long
    move    T2R0, #04000h    ; Reload Value
    move    T2C0, #0C000h    ; Compare Value

    move    T2CFG0, #000h
    ;
    ; 0000 0000
    ; C/T2      0,          Timer Mode
    ; CCF1:0    00,        Compare Mode
    ; T2MD      0,          16-bit Mode
    ; T2DIV2:0  000,       No freq pre-scaling
    ;T2CI      0,          use system clock
move    T2CNB0, #040h
    ;
    ; 0100 0000
    ; TC2L      0,          Not used
    ; TCC2      0          Not used
    ; TF2L      0,          Not used
    ; TF2       0,          Not used
    ; X         x,          reserved
    ; T2POL1    0,          Low Starting value on secondary pin
    ; T2OE1     1,          Secondary pin is enabled as output
    ; ET2L      0,          Not used

move    T2CNA0, #068h
    ;
    ; 0110 1000
    ; G2EN      0,          Gating disabled
    ; SS2       0,          Single shot disabled
    ; CPRL2     0,          Not used
    ; TR2       1,          Run enabled
    ; TR2L      0,          Not used
    ; T2PO10    1,          High Starting value on primary pin
    ; T2OE0     1,          Primary pin is enabled as output
    ; ET2       0,          Not used

Loop1:
    ; body of the code, loop here indefinitely
    jump    Loop1

end
```

6.7.8.2 Waveform Output with Gating

This example uses the code from the previous example, with the modification that the primary pin is now the gating input. The output on the secondary pin is gated when the primary pin is low. The 1/3 duty cycle waveform is output on the secondary pin whenever the primary pin is held high.

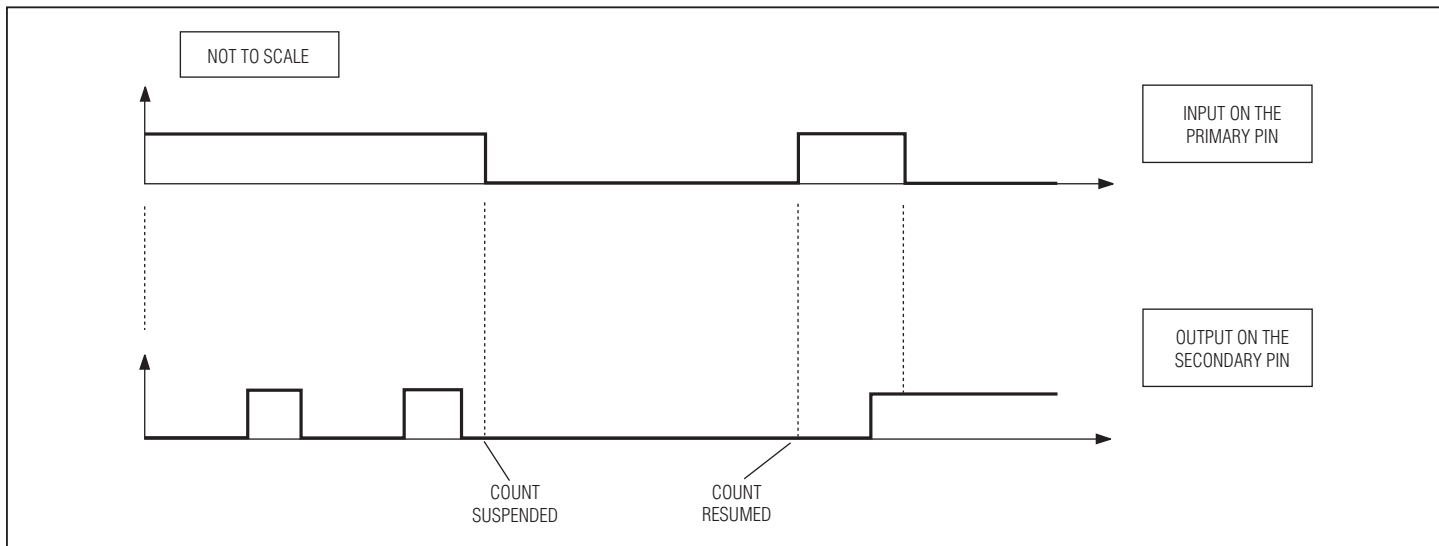


Figure 6-7. Waveform Output with Gating

```

org 0
;
main_top:

    move    T2V0, #04000h    ;
    move    T2R0, #04000h    ;
    move    T2C0, #0C000h    ;

    move    T2CFG0, #000h

        ; 0010 0000
        ; C/T2      0,      Timer Mode
        ; CCF1:0    00,     Compare Mode
        ; T2MD      0,      16-bit Mode
        ; T2DIV2:0  000,    No freq. pre-scaling
        ; T2CI      0,      use system clock

    move    T2CNB0, #040h
    
```

```
    ; 0100 0000
    ; TC2L      0,          Not used
    ; TCC2      0,          Not used
    ; TF2L      0,          Not used
    ; TF2       0,          Not used
    ; T2POL1    0,          Low Starting value on secondary pin
    ; T2OE1     1,          Secondary pin is enabled as output
    ; ET2L      0,          Not used

move  T2CNA0, #009h

    ; 0000 1001
    ; G2EN      1,          Gating enabled
    ; SS2       0,          Single shot disabled
    ; CPRL2     0,          Not used
    ; TR2       1,          Run enabled
    ; TR2L      0,          Not used
    ; T2PO10    0,          gated when primary pin is low
    ; T2OE0     0,          Primary pin is used for gating
    ; ET2       0,          Not used

loop1:
    ; body of the code, loop here indefinitely
    jump Loop1

end
```


6.7.9 Type 2 Timer Capture Application Examples

The following examples are used to demonstrate some of the Type 2 timer capture functions. All examples assume that pulse and/or period measurements do not exceed 2^{16} (i.e., 65,536) input clocks and that capture register holds the desired result.

6.7.9.1 Measure Low-Pulse Duration

To measure the duration of the first full low pulse seen on the T2P0 input pin, the Type 2 timer is configured for a single-shot capture, gating enabled for logic high, capture on the rising edge. The CPRL2 bit can optionally be set to generate a reload on the same rising edge as the capture if the preconfigured T2R0 value is expected to be needed next.

```

; ----- Reset State:  T2R0 = T2V0 = T2C0 = 0000h -----
MOVE T2CFG0, #00000010b      ; T2CI          =0          (sysclk/N input)
                               ; T2DIV2:0        =000          (/1)
                               ; T2MD           =0          (16-bit)
                               ; CCF1:0         =01          (rising edge)
                               ; C/T2          =0          (timer/capture)
MOVE T2CNA0, #10100111b     ; ET2           =1          (enable Type 2 Timer ints)
                               ; T2OE0         =0          (input)
                               ; T2POL0        =1          (gating level = '1')
                               ; TR2L:TR2      =00          (don't start timer)
                               ; CPRL2         =1          (reload on capture edge)
                               ; SS2           =1          (single shot mode)
                               ; G2EN         =1          (gating enabled)
; ----- TCC2 Interrupt : DURATION = T2C0 -----
    
```

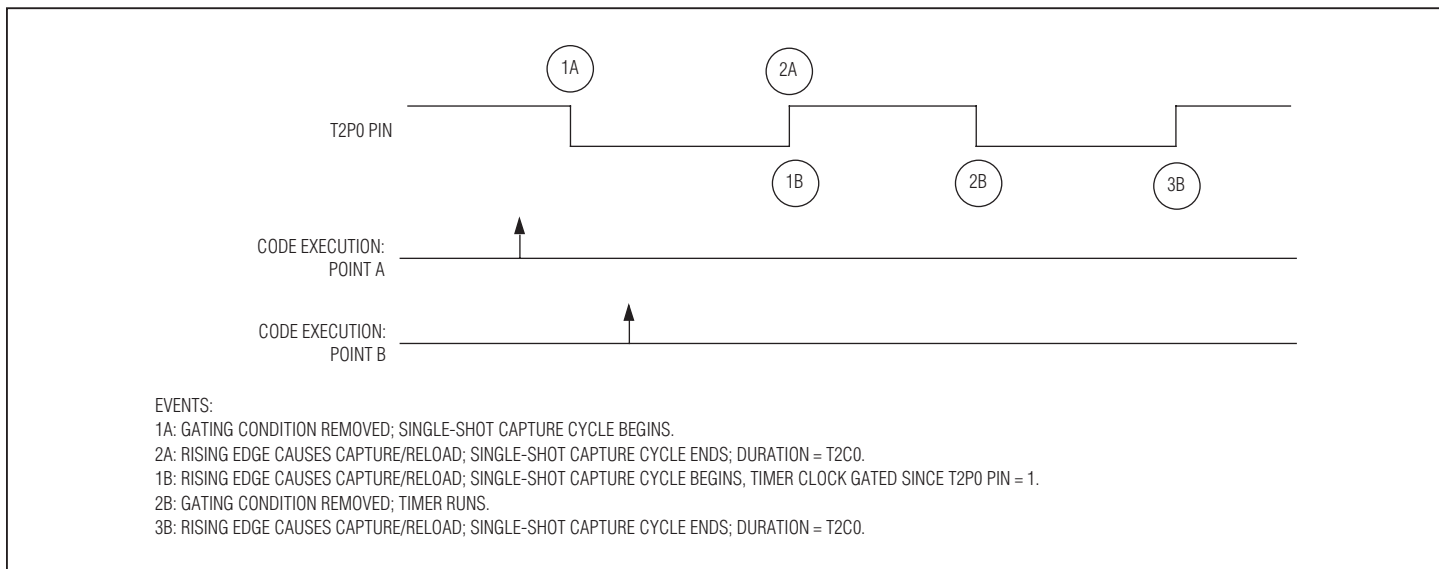


Figure 6-8. Type 2 Timer Application Example—Measure Low Pulse Width

6.7.9.2 Measure High-Pulse Duration Repeatedly

To measure the duration of high pulses seen on the T2P0 input pin repeatedly, the Type 2 timer is configured for a single-shot delayed run, gating enabled for logic low, capture on the falling edge. The CPRL2 bit can be set to generate a reload on each falling edge.

```

; ----- Reset State:  T2R0 = T2V0 = T2C0 = 0000h -----
MOVE T2CFG0, #00000100b      ; T2CI      =0      (sysclk/N input)
                               ; T2DIV2:0   =000      (/1)
                               ; T2MD       =0      (16-bit)
                               ; CCF1:0     =10      (falling edge)
                               ; C/T2       =0      (timer/capture)
MOVE T2CNA0, #10001111b     ; ET2      =1      (enable Type 2 Timer ints)
                               ; T2OE0      =0      (input)
                               ; T2POL0     =0      (gating level = '0')
                               ; TR2L:TR2   =01      (start timer on single shot
                               ;              condition)
                               ;
                               ; CPRL2      =1      (reload on capture edge)
                               ; SS2        =1      (single shot mode)
                               ; G2EN       =1      (gating enabled)
; ----- TCC2 Interrupt : DURATION = T2C0
    
```

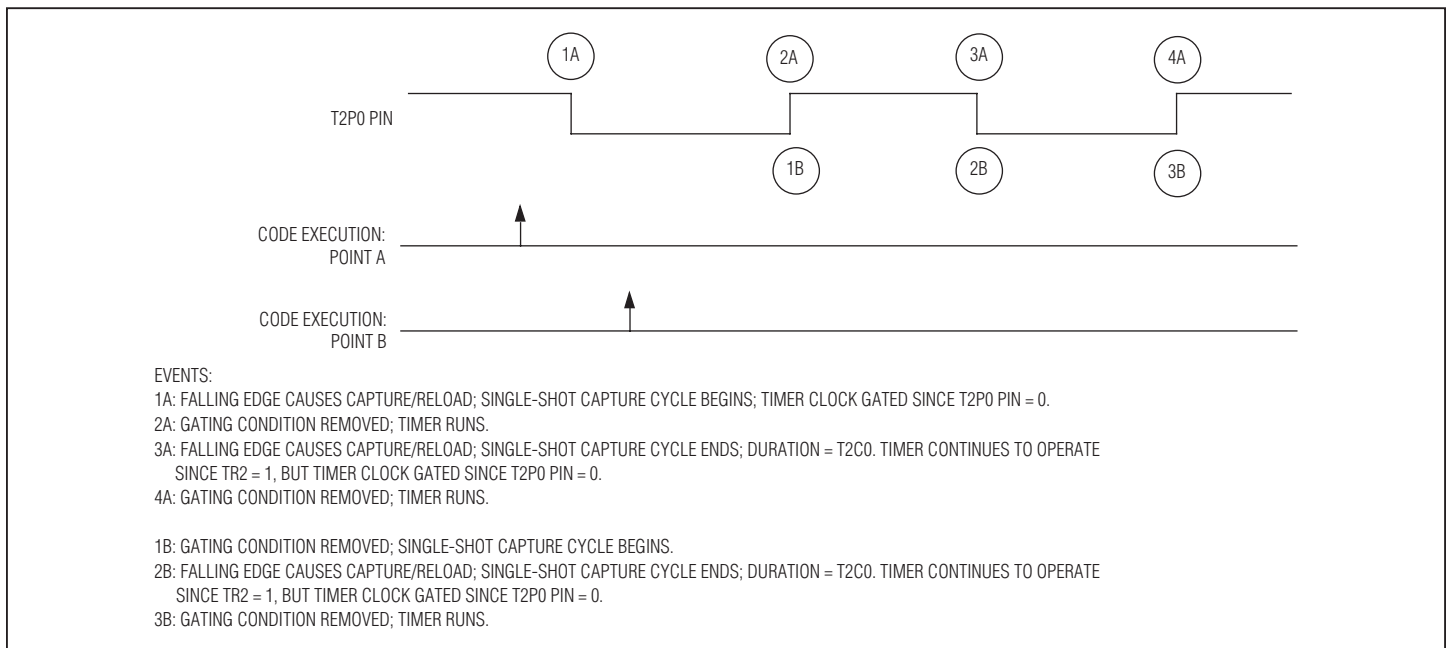


Figure 6-9. Type 2 Timer Application Example—Measure High Pulse Width

6.7.9.3 Measure Period

To measure the period of the signal seen on the T2P0 input pin, the Type 2 timer is configured for a single-shot capture, no gating, either edge (selected by the CCF[1:0] bits). The CPRL2 bit can be set to generate a reload on each capture edge.

```

; ----- Reset State:  T2R0 = T2V0 = T2C0 = 0000h -----
MOVE T2CFG0, #00000100b      ; T2CI          =0      (sysclk/N input)
                               ; T2DIV2:0       =000      (/1)
                               ; T2MD          =0      (16-bit)
                               ; CCF1:0        =10      (falling edge)
                               ; C/T2         =0      (timer/capture)
MOVE T2CNA0, #10000110b     ; ET2          =1      (enable Type 2 Timer ints)
                               ; T2OE0        =0      (input)
                               ; T2POL0       =0      (gating level = '0')
                               ; TR2L:TR2     =00      (don't start timer)
                               ; CPRL2        =1      (reload on capture edge)
                               ; SS2          =1      (single shot mode)
                               ; G2EN         =0      (gating disabled)
; ----- TCC2 Interrupt : PERIOD = T2C0
    
```

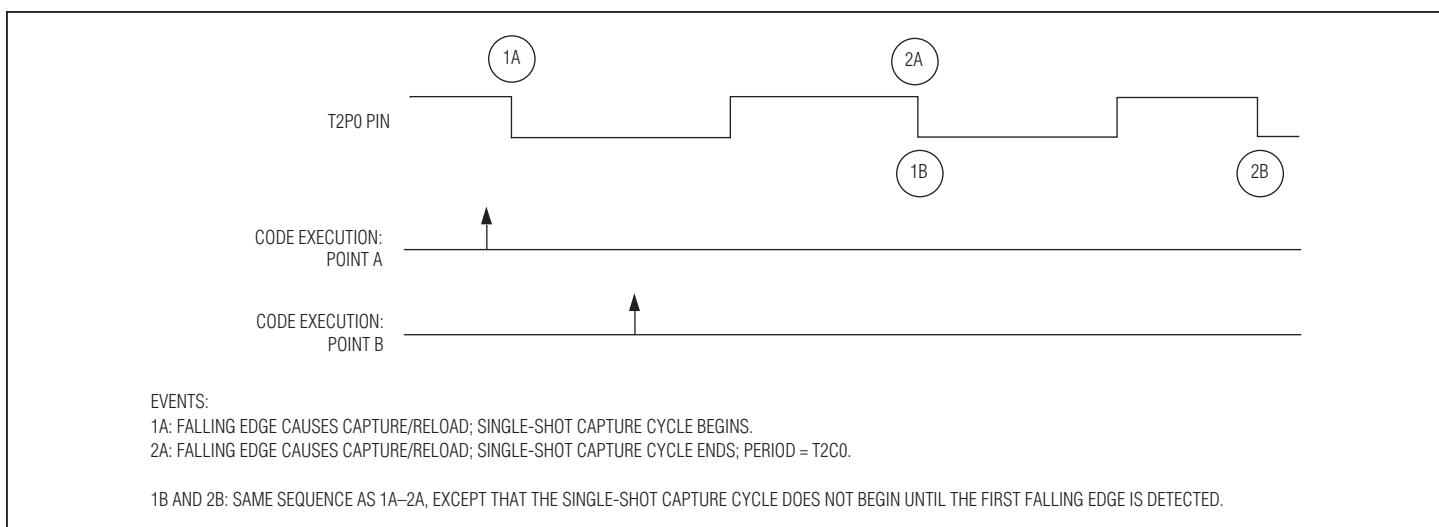


Figure 6-10. Type 2 Timer Application Example—Measure Period

6.7.9.4 Measure Duty Cycle Repeatedly

To measure the duty cycle of the signal seen on the T2P0 input pin, the Type 2 timer is configured for a single-shot delayed run with both edges defined for capture. The CPRL2 bits should be configured to 1 to request reloads on each edge. To prevent reloads on one of the edges, gating should be enabled. The T2POL0 bit specifies which edge starts/ends the capture cycle and which edge does not have a reload associated with it.

```

; ----- Reset State: T2R0 = T2V0 = T2C0 = 0000h -----
MOVE T2CFG0, #00000110b      ; T2CI      =0      (sysclk/N input)
                               ; T2DIV2:0   =000      (/1)
                               ; T2MD       =0      (16-bit)
                               ; CCF1:0     =11      (both edges)
                               ; C/T2       =0      (timer/capture)
MOVE T2CNA0, #10101111b     ; ET2      =1      (enable Type 2 Timer ints)
                               ; T2OE0     =0      (input)
                               ; T2POL0    =1      (no reload on rising edge
                               ;                                     single shot start/end on falling edge)
                               ; TR2L:TR2  =01      (start timer on single shot condition)
                               ; CPRL2     =1      (reload on capture edge)
                               ; SS2       =1      (single shot mode)
                               ; G2EN      =1      (gating enabled)
; ----- TCC2 Interrupt : LOW TIME=T2C0
;----- TCC2 Interrupt : PERIOD = T2C0
    
```

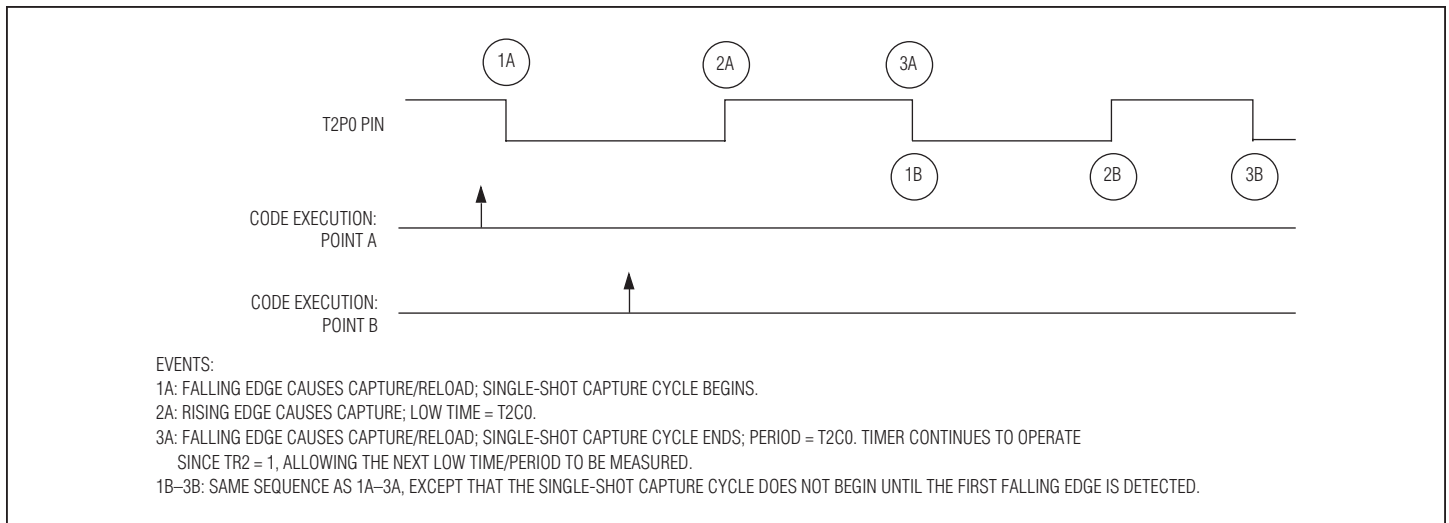


Figure 6-11. Type 2 Timer Application Example—Measure Duty Cycle

6.7.9.5 Overflow/Interrupt on Cumulative Time

To cause an overflow only when the T2P0 pin has been low for some cumulative duration, the Type 2 timer can be configured to the gated compare mode of operation with an initial starting value appropriate for the cumulative duration to be detected.

```

; ----- Reset State:  T2R0 = T2V0 = T2C0 = 0000h -----
MOVE T2V0, #1234h      ; Overflow after T2P0 input low for (10000h-01234h) CLKs
MOVE T2CFG0, #01110000b ; T2CI          =0          (sysclk/N input)
                        ; T2DIV2:0       =111         (/128)
                        ; T2MD           =0          (16-bit)
                        ; CCF1:0         =00         (no edges)
                        ; C/T2           =0          (timer/compare)
MOVE T2CNA0, #10101001b ; ET2            =1          (enable Type 2 Timer ints)
                        ; T2OE0          =0          (input)
                        ; T2POL0         =1          (gating level = '1')
                        ; TR2L:TR2       =01         (start timer)
                        ; CPRL2          =0          (no capture possible)
                        ; SS2            =0          (not single shot mode)
                        ; G2EN           =1          (gating enabled)
; ----- TF2 Interrupt : Cumulative low duration reached
    
```

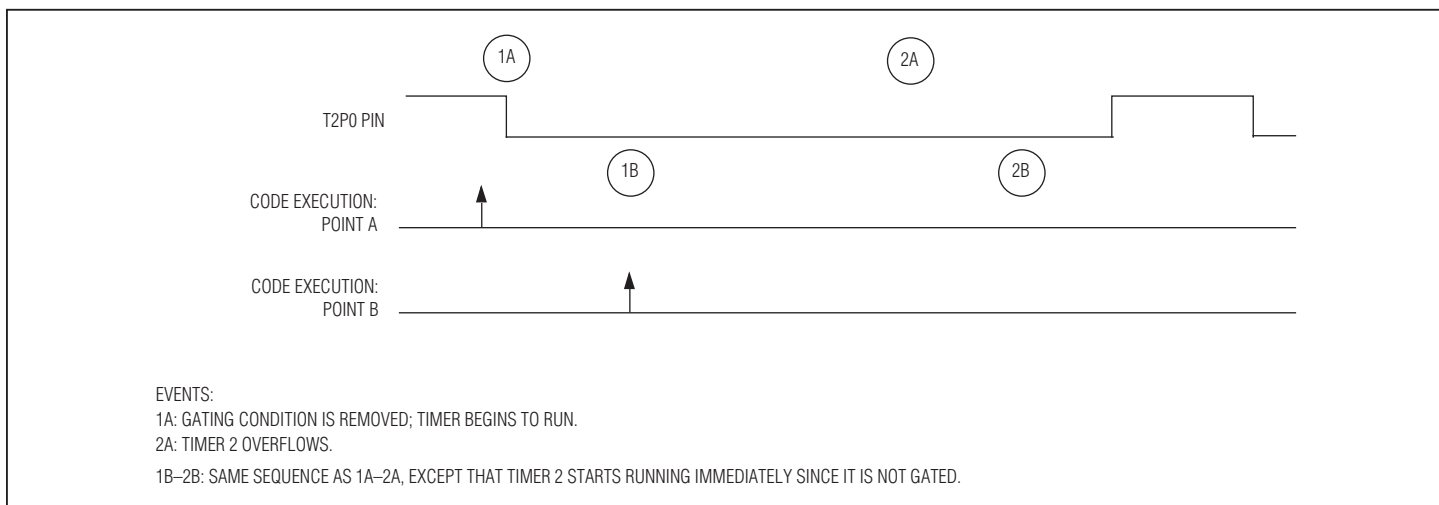


Figure 6-12. Type 2 Timer Application Example—Overflow/Interrupt on Cumulative Time

SECTION 7: SCHEDULE TIMER

This section contains the following information:

7.1 Architecture	7-2
7.2 Schedule Timer Register Descriptions	7-3
7.2.1 Schedule Timer Control Register (SCNT)	7-3
7.2.2 Schedule Timer Register (STIM)	7-4
7.2.3 Schedule Alarm Register (SALM)	7-4
7.3 Schedule Timer Operation	7-5

LIST OF FIGURES

Figure 7-1. Schedule Timer Module Block Diagram	7-2
---	-----

LIST OF TABLES

Table 7-1. STDIV Setting to Select the SYSCLK Divisor	7-5
---	-----

SECTION 7: SCHEDULE TIMER

The MAXQ7667's schedule timer is a much simpler implementation of the real-time clock module found in many MAXQ microcontrollers. The schedule timer provides a means for general timekeeping and software synchronization to the external I/O.

The schedule timer features include the following:

- 16-bit autoreload up-counter for the timer
- Programmable 16-bit alarm register
- Alarm interrupts
- Schedule timer is incremented by a prescaled system clock (SYSCLK, see *Section 15*). The system clock can be prescaled to 1, 2, 4, 8, 16, 32, 64, and 128.
- Schedule timer up-counter can be reset through an external I/O pin, thus allowing synchronization of the schedule timer to an external event.
- Wake-up alarm to pull the system clock from stop mode to normal operation (divide-by-1)

7.1 Architecture

The MAXQ7667 implements the following features:

- A 16-bit counter (STIM), clocked by a divided-down version of SYSCLK. Software can read/write STIM to access/modify the time-stamp. STIM is enabled by SCNT.0:STIME.
- A 16-bit alarm register (SALM). It is readable/writable from software. When STIM equals SALM, interrupts can be generated to CPU. SALM is enabled by SCNT.1:SALME. **Note:** SALM is writable by software ONLY when either STIME or SALME is 0.
- STIM is cleared to 0 by hardware when one of the following conditions occurs:
 - 1) STIM wraps around from FFFFh to 0.
 - 2) STIM equals SALM and the alarm function is enabled (i.e., SALME = 1). (Note that when SCNT.6:SALMF is set to 1, an interrupt is generated.)
 - 3) An external rising-edge synchronization event from input pad P1.7 and is enabled by SCNT.8:SSYNC_EN. SSYNC_EN clears when such an event occurs.

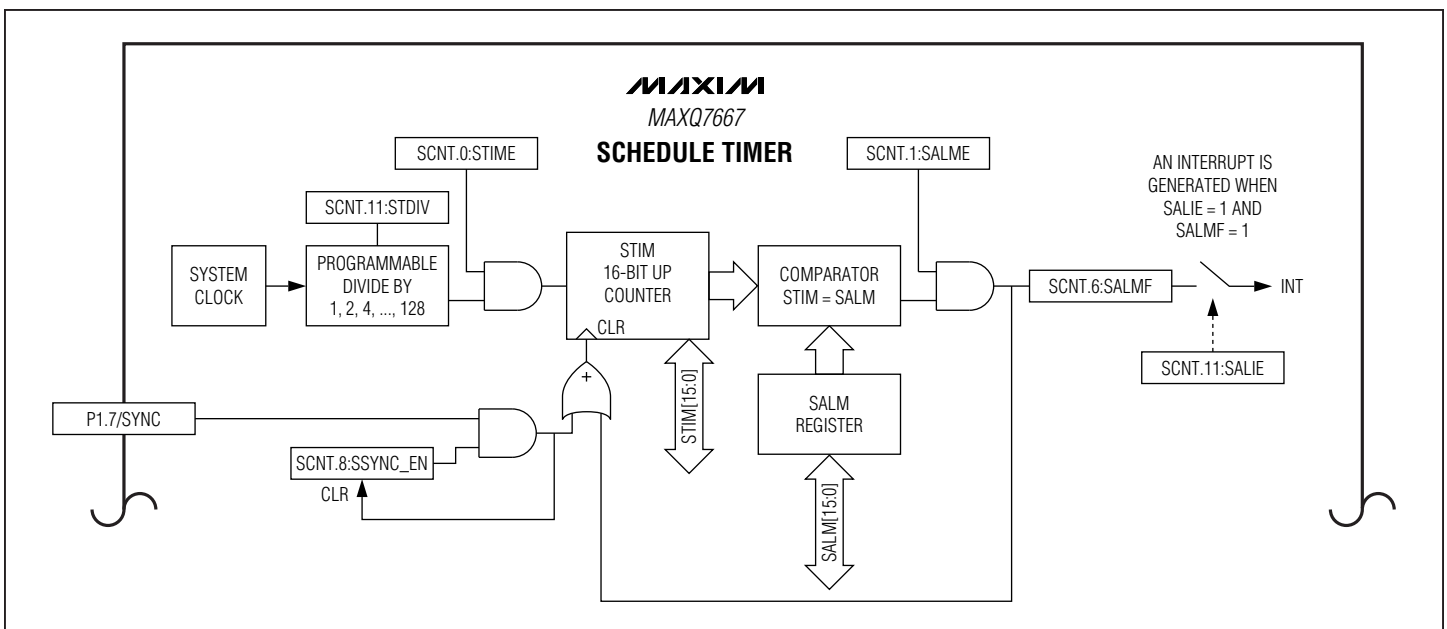


Figure 7-1. Schedule Timer Module Block Diagram

7.2 Schedule Timer Register Descriptions

7.2.1 Schedule Timer Control Register (SCNT)

Register Description: **Schedule Timer Control Register**

Register Name: **SCNT**

Register Address: **Module 01h, Index 0Eh**

Bit #	15	14	13	12	11	10	9	8
Name	—	—	—	—	STDIV2	STDIV1	STDIV0	SSYNC_EN
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	SALIE	SALMF	—	—	—	—	SALME	STIME
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	r	r	r	r	rw	rw

r = read, w = write

Note: This register is cleared to 000h on all forms of reset.

Bits 15 to 12, 5 to 2: Reserved. Read as 0.

Bits 11 to 9: Schedule Timer Prescaler Division Ratio (STDIV[2:0]). The clock used by the schedule timer is divided down from system clock, by the following factors:

- 0: divided by 1
- 1: divided by 2
- 2: divided by 4
- 3: divided by 8
- 4: divided by 16
- 5: divided by 32
- 6: divided by 64
- 7: divided by 128

Bit 8: Schedule Synchronization Enable (SSYNC_EN). When set to 1, external synchronization event (from pin SYNC) is enabled. It is cleared after such event occurs.

Bit 7: Schedule Alarm Interrupt Enable (SALIE). When set to 1 the schedule alarm interrupt gets enabled. Setting this bit to 0 disables the schedule alarm interrupt. When enabled and SALMF = 1, an interrupt is generated.

Bit 6: Schedule Alarm Flag (SALMF). This bit is set when both STIME and SALME are 1 and STIME equals SALM.

Bit 1: Schedule Timer Alarm Enable (SALME). When set to 1 the schedule alarm function is enabled. Setting this bit to 0 disables the schedule alarm function.

Bit 0: Schedule Timer Enable (STIME). When set to 1 the schedule timer is enabled. Setting this bit to 0 disables the schedule timer.

7.2.2 Schedule Timer Register (STIM)

Register Description: **Schedule Timer Register**
 Register Name: **STIM**
 Register Address: **Module 01h, Index 0Fh**

Bit #	15	14	13	12	11	10	9	8
Name	STIM15	STIM14	STIM13	STIM12	STIM11	STIM10	STIM9	STIM8
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	STIM7	STIM6	STIM5	STIM4	STIM3	STIM2	STIM1	STIM0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 00h on all forms of reset.

Bits 15 to 0: Schedule Timer Register Bits 15:0 (STIM[15:0]). When STIME is 1, STIM increases by 1 every prescaled system clock, and is automatically cleared to 0 when any one of the following conditions happens:

- 1) SSYNC_EN is 1 and an external synchronization event occurs.
- 2) SALME is 1 and STIM equals SALM. (Note that SALMF is set to 1 in this case.)

7.2.3 Schedule Alarm Register (SALM)

Register Description: **Schedule Alarm Register**
 Register Name: **SALM**
 Register Address: **Module 01h, Index 10h**

Bit #	15	14	13	12	11	10	9	8
Name	SALM15	SALM14	SALM13	SALM12	SALM11	SALM10	SALM9	SALM8
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	SALM7	SALM6	SALM5	SALM4	SALM3	SALM2	SALM1	SALM0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read (This register is writable when either STIME or SALME is 1.)

Note: This register is cleared to 00h on all forms of reset.

Bits 15 to 0: Schedule Alarm Register Bits 15:0 (SALM[15:0]). This register holds the value of the schedule alarm.

7.3 Schedule Timer Operation

The MAXQ7667 has a schedule timer that can be used for general timekeeping and interval alarms and wake-up alarms. The timer is a 16-bit up-counter that is incremented by the system clock after the system clock has been divided by a prescaler. The counter value is read through the STIM register. Writing to the STIM register sets the counter to the written value.

The divisor used in the system clock prescaler is set by STDIV and divides the system clock by one of the following values: 1, 2, 4, 8, 16, 32, 64, or 128. Table 7-1 shows the STDIV setting used to select each divisor. When operating with a 16MHz system clock, the prescaler provides full-scale timer values that range from 4.096ms $[(1/16\text{MHz}) \times 1 \times 65,536]$ to 524ms $[(1/16\text{MHz}) \times 128 \times 65,536]$. If measurement times longer than 524ms are needed, a software counter can be used to tally the number of times the schedule timer has alarmed or rolled over. This technique can be expanded to accommodate any desired length of time.

Table 7-1. STDIV Setting to Select the SYSCLK Divisor

STDIV[2:0]			PRESCALE DIVIDER
2	1	0	
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

The usefulness of the schedule timer is increased by its alarm feature. If enabled ($\text{SALME} = 1$ and $\text{STIME} = 1$), an alarm event is initiated when $\text{STIM} = \text{SALM}$. The alarm event sets the alarm flag, SALMF ; resets STIM to 0; and generates an interrupt if enabled. The alarm interrupt is enabled by setting the SALIE bit to 1. It is also important that the module interrupt, IMR1 , and the global interrupt, IGE , are also enabled for the interrupt event to occur successfully.

The schedule timer, alarm value, alarm flag, and all the enable bits are cleared to zero by all forms of reset.

The schedule timer can be synchronized to an external event through pin P1.7/SYNC. This is the only external pin that is directly associated with the schedule timer. When enabled ($\text{SSYNC_EN} = 1$), a low-to-high transition on P1.7/SYNC causes the next system clock to clear the timer. The signal on P1.7/SYNC must remain high for at least one system clock cycle. After being high for one clock cycle, P1.7/SYNC can remain high or be taken low again. Synchronizing the schedule timers on multiple MAXQ7667s is useful when triangulation is used to locate a target's position.

When switchback is enabled (see *Section 15*), the interrupt from schedule timer is a valid source to switch out of the stop mode.

SECTION 8: UART AND LIN

This section contains the following information:

8.1 Architecture	8-4
8.2 UART/LIN Pins	8-4
8.3 UART and LIN Register Descriptions	8-5
8.3.1 Control Register 1 (UART) (CNT1)	8-5
8.3.2 Serial Control Register (UART) (SCON)	8-5
8.3.3 Serial Port Buffer Register (SBUF)	8-7
8.3.4 FIFO Status Register (UART) (FSTAT)	8-7
8.3.5 Error Register (UART) (ERRR)	8-8
8.3.6 Checksum Register (UART) (CHKSUM)	8-9
8.3.7 Interrupt State Vector Register (ISVEC)	8-10
8.3.8 Status Register 0 (UART) (STA0)	8-11
8.3.9 Serial Mode Register (UART) (SMD)	8-11
8.3.10 FIFO Control Register (UART) (FCON)	8-12
8.3.11 Control Register 0 (UART) (CNT0)	8-13
8.3.12 Control Register 2 (UART) (CNT2)	8-14
8.3.13 Identifier Boundary Register (UART) (IDFB)	8-15
8.3.14 Serial Address Register (UART) (SADDR)	8-15
8.3.15 Serial Address Mask Register (UART) (SADEN)	8-16
8.3.16 Bit Timing Register (UART) (BT)	8-16
8.3.17 Timer Register (UART) (TMR)	8-17
8.4 MAXQ7667 LIN	8-17
8.4.1 MAXQ7667 LIN Features	8-17
8.4.2 LIN Frame	8-18
8.4.3 LIN Peripheral	8-18
8.4.4 LIN Master	8-19
8.4.4.1 LIN Master Setup Example	8-19
8.4.4.2 LIN Master Sending Protected ID and Break/Sync Example	8-20
8.4.4.3 LIN Master Polling for Transmit Complete Example	8-20
8.4.4.4 LIN Master Setup to Receive Acknowledge from Slave Example	8-20

8.4.5 LIN Slave	.8-21
8.4.5.1 LIN Slave Setup Example	.8-21
8.4.5.2 LIN Slave Receiving Protected ID and Break/Sync Example	.8-21
8.4.5.3 LIN Slave Polling for Receive Complete Example	.8-22
8.4.5.4 LIN Slave Setup to Transmit Data from Slave Example	.8-22
8.4.5.5 LIN Slave Setup to Transmit Multiple Bytes from Slave Example	.8-22
8.4.6 Receive Filter	.8-22
8.4.7 Identifier Boundary Register (IDFB)	.8-22
8.4.8 LIN Transmit and Receive FIFO	.8-22
8.4.9 Setting LIN Baud Rate	.8-23
8.4.10 LIN Interrupts	.8-23
8.4.10.1 LIN Interrupt Example	.8-23
8.4.10.2 LIN Master/Slave Interrupt Example	.8-23
8.4.11 Power Features	.8-23
8.4.12 LIN Error Handling	.8-24
8.5 MAXQ7667 UART	.8-24
8.5.1 MAXQ7667 UART Features	.8-24
8.5.2 MAXQ7667 UART Modes	.8-25
8.5.2.1 UART Mode 0	.8-25
8.5.2.2 UART Mode 1	.8-25
8.5.2.3 UART Mode 2	.8-28
8.5.2.4 UART Mode 3	.8-28
8.5.3 Framing Error Detection	.8-28
8.5.4 UART Mode 1 Asynchronous Full-Duplex Setup Example	.8-31
8.5.5 UART Interrupts Example	.8-31
8.5.6 UART Transmit Data Example	.8-31
8.5.7 UART Receive Data Example	.8-32
8.5.8 Setting UART Baud Rate	.8-32
8.5.9 UART FIFO	.8-32

LIST OF FIGURES

Figure 8-1. UART/LIN Block Diagram	8-4
Figure 8-2. LIN Frame Format	8-18
Figure 8-3. LIN Block Diagram	8-18
Figure 8-4. LIN Bus Master Communication	8-19
Figure 8-5. LIN Bus Slave Communication	8-21
Figure 8-6. UART Block Diagram	8-24
Figure 8-7. UART Mode 0	8-26
Figure 8-8. UART Mode 1	8-27
Figure 8-9. UART Mode 2	8-29
Figure 8-10. UART Mode 3	8-30

LIST OF TABLES

Table 8-1. GPIO Port 0 UART/LIN Pins	8-4
Table 8-2. Serial Port Operating Modes	8-6
Table 8-3. LIN Priority Encoded Interrupts	8-10

SECTION 8: UART AND LIN

The MAXQ7667 contains a standard UART for serial communication and dedicated hardware for support of the LIN bus. The dedicated LIN hardware simplifies the application code and requires less processor intervention. The MAXQ7667 can be configured to use either the UART or LIN. Both the UART and LIN require external bus transceivers to connect to the physical layer.

8.1 Architecture

Figure 8-1 shows the various blocks that comprise the UART/LIN interface. The interface to the UART/LIN hardware is accessed through the MAXQ7667 core. The MAXQ7667 memory-mapped registers for the UART/LIN are found in Module 3 of the peripheral register map.

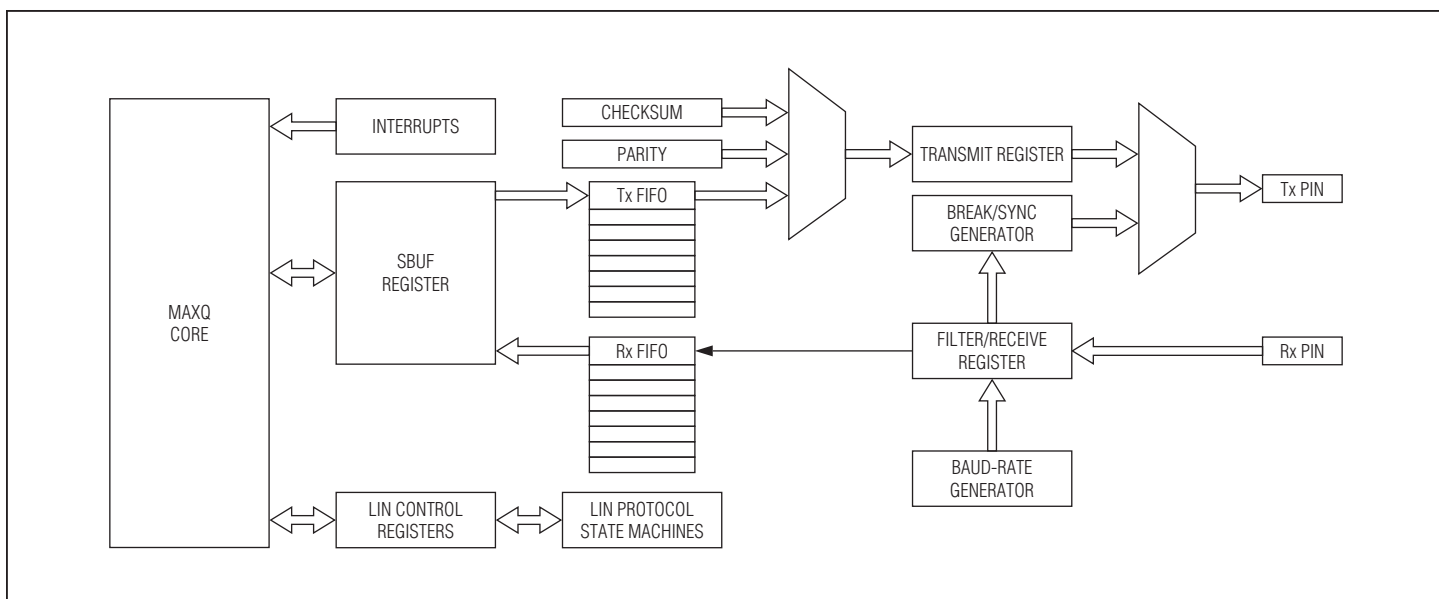


Figure 8-1. UART/LIN Block Diagram

8.2 UART/LIN Pins

The UART/LIN pins are multiplexed with GPIO port 0 pins.

Table 8-1. GPIO Port 0 UART/LIN Pins

PORT P0 SIGNALS	PORT 0 PIN	FUNCTION
P0.0/URX	9	Port 0, Bit 0, General-Purpose Digital I/O Bit. Alternately, it is used for UART/LIN receive data.
P0.1/UTX	10	Port 0, Bit 1, General-Purpose Digital I/O Bit. Alternately, it is used for UART/LIN transmit data.

8.3 UART and LIN Register Descriptions

The following sections describe the MAXQ7667 registers that control the UART and LIN hardware.

8.3.1 Control Register 1 (UART) (CNT1)

Attention: All the bits in this register should be written simultaneously in one write instruction.

Register Description: **Control Register 1 (UART)**
 Register Name: **CNT1**
 Register Address: **Module 03h, Index 05h**

Bit #	7	6	5	4	3	2	1	0
Name	RTN	CK	FL5	FL4	FL3	FL2	FL1	FL0
Reset	1	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: CNT1 is cleared to 80h on all forms of reset. The host must always write to this register after a valid identifier has been received.

Bit 7: Receive or Transmit Mode (RTN). When a valid LIN header is received, the host must set this field to indicate whether the peripheral should transmit (RTN = 0) or receive (RTN = 1) a LIN frame.

Bit 6: Checksum Type (CK). If the AUT bit (CNT0.3) is set to 1, this bit is set by the peripheral according to the checksum type for the most recent frame. If AUT is cleared to 0, the MAXQ7667 must set this flag to indicate the checksum type to be used.

Bits 5 to 0: Frame Length (FL[5:0]). This field indicates the number of bytes in the frame. The frame length is always one greater than the value specified in this field. For example, writing FL = 00h indicates a frame length of one byte and writing FL = 3Fh indicates a frame length of 64 bytes.

8.3.2 Serial Control Register (UART) (SCON)

Register Description: **Serial Control Register (UART)**
 Register Name: **SCON**
 Register Address: **Module 03h, Index 06h**

Bit #	7	6	5	4	3	2	1	0
Name	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: SCON is cleared to 00h on all forms of reset.

Bit 7: Serial Port Mode Bit 0/Framing Error Flag (SM0/FE). When the FEDE bit (SMD.0) is cleared to 0, this bit reflects the state of the SM0 mode control bit. The host can write to this bit to configure the operating mode for legacy UART. When FEDE is set to 1, this bit reflects the state of the framing error flag. The host can read this bit to determine the framing error status and can write this bit to set or clear the framing error status. The serial port operating mode is defined in Table 8-2.

When the peripheral is configured for LIN master or LIN slave mode, the UART is always operated in mode 1 and the SM0 bit serves no purpose. The FE flag reflects the framing error status in all LIN and UART modes.

Bit 6: Serial Port 0 Mode Bit 1 (SM1). If the peripheral is operating in legacy UART mode, this bit is the second mode control bit as defined in Table 8-2.

Bit 5: Serial Port Mode Bit 2 (SM2). The function of this bit is dependent on the operating mode of the serial port. In mode 0, it determines the speed of the serial clock signal (either 1/4 or 1/12 the system clock rate). In mode 1, setting this bit to 1 disables the receive interrupt if an invalid stop bit is received. In modes 2 and 3, setting this bit to 1 prevents the receive interrupt from being set if the 9th received bit is a 0.

SM2 = 0: clock is divided by 12

SM2 = 1: clock is divided by 4

Bit 4: Receive Enable (REN). In legacy UART mode, this bit enables the receiver. In LIN master or LIN slave mode, the receiver is controlled by the LIN protocol state machine and this bit serves no purpose.

Bit 3: 9th Transmission Bit State (TB8). In legacy UART mode, the host writes this bit to reflect the state of the 9th transmitted bit. If the FIFO is enabled, the host must set this bit to the correct state before writing to the SBUF register. This bit serves no purpose in LIN master or LIN slave mode.

Bit 2: 9th Received Bit State (RB8). In legacy UART mode, this bit is written by the peripheral to reflect the state of the 9th received bit. If the FIFO is enabled, the host must read the SBUF register before reading this bit. This bit serves no purpose in LIN master or LIN slave mode.

Bit 1: Transmit Interrupt Flag (TI). In legacy UART mode, this bit is set by the peripheral to indicate that a transmit interrupt condition occurred. If the FIFO is enabled, this bit is set when the amount of data in the FIFO falls below the threshold determined by the TXFT[1:0] field (FCON.5:4). If the FIFO is disabled, this flag is set every time a byte has been transmitted. This bit serves no purpose in LIN master or LIN slave mode.

Bit 0: Receive Interrupt Flag (RI). In legacy UART mode, the peripheral sets this bit to indicate that a receive interrupt condition occurred. If the FIFO is enabled, it is set when the amount of data in the receive FIFO exceeds the threshold determined by the RXFT[1:0] field (FCON.3:2). If the FIFO is disabled, this bit is set every time a byte is received. This bit serves no purpose in LIN master or LIN slave mode.

Table 8-2. Serial Port Operating Modes

MODE	SM0	SM1	SM2	FUNCTION
0	0	0	0	Synchronous communication (mode 0). The peripheral transmits and receives 8-bit data using a synchronous protocol. The clock rate is 1/12 the system clock rate.
0	0	0	1	Same as above, except the clock rate is 1/4 the system clock rate.
1	0	1	X	Asynchronous communication (mode 1). The peripheral transmits and receives 8-bit data using an asynchronous protocol with one start bit and one stop bit. In this mode, the on-chip baud-rate generator determines the communication speed.
2	1	0	0	Asynchronous communication (mode 2). The peripheral transmits and receives 9-bit data using an asynchronous protocol with one start bit and one stop bit. In this mode, the communication speed is either 1/32 or 1/64 the system clock rate, depending on the state of the SMOD bit (SMD.1). Automatic address recognition (legacy multiprocessor communication) is disabled.
2	1	0	1	Same as above, except the automatic address recognition is enabled.
3	1	1	X	This is the same as mode 2, except the communication speed is determined by the baud-rate generator. Automatic address recognition (legacy multiprocessor communication) is enabled when SM2 = 1.

8.3.3 Serial Port Buffer Register (SBUF)

Register Description: **Serial Port Buffer Register**
 Register Name: **SBUF**
 Register Address: **Module 03h, Index 07h**

Bit #	7	6	5	4	3	2	1	0
Name	SBUF7	SBUF6	SBUF5	SBUF4	SBUF3	SBUF2	SBUF1	SBUF0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: SBUF is cleared to 00h on all forms of reset.

Bits 7 to 0: Serial Port Buffer Register (SBUF[7:0]). This register is the access point for inserting data into the transmit buffer or retrieving data from the receive buffer. To load the transmit FIFO, software will normally write to this register once for each byte of data to be transmitted. Similarly, when a frame has been received, software will normally read this register until the receive FIFO is empty.

8.3.4 FIFO Status Register (UART) (FSTAT)

Register Description: **FIFO Status Register**
 Register Name: **FSTAT**
 Register Address: **Module 03h, Index 0Ch**

Bit #	7	6	5	4	3	2	1	0
Name	—	—	TFF	TFAE	TFE	RFF	RFAF	RFE
Reset	0	0	0	0	1	0	0	1
Access	r	r	r	r	r	r	r	r

r = read

Note: FSTAT is cleared to 09h on all forms of reset.

Bits 7 and 6: Reserved. Read returns 0.

Bit 5: Transmit FIFO Full (TFF). This bit is set to 1 by the peripheral when the transmit FIFO is completely full. The bit is automatically cleared to 0 when a byte of data has been transmitted.

Bit 4: Transmit FIFO Almost Empty (TFAE). This bit is set to 1 by the peripheral when the amount of data in the transmit FIFO exceeds the threshold configured by the TXFT[1:0] field (FCON.5:4).

Bit 3: Transmit FIFO Empty (TFE). This bit is set to 1 by the peripheral whenever the transmit FIFO is completely empty.

Bit 2: Receive FIFO Full (RFF). This bit is set to 1 by the peripheral whenever the receive FIFO is completely full.

Bit 1: Receive FIFO Almost Full (RFAF). This bit is set to 1 by the peripheral when the amount of data in the receive FIFO exceeds the threshold configured by the RXFT[1:0] field (FCON.3:2).

Bit 0: Receive FIFO Empty (RFE). This bit is set to 1 by the peripheral whenever the receive FIFO is completely empty.

8.3.5 Error Register (UART) (ERRR)

Register Description: **Error Register**
 Register Name: **ERRR**
 Register Address: **Module 03h, Index 0Dh**

Bit #	7	6	5	4	3	2	1	0
Name	—	OTE	DME	CKE	P1	P1E	P0	P0E
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: ERRR is cleared to 00h on all forms of reset.

Bit 7: Reserved. Read returns 0.

Bit 6: Other Communications Error (OTE). This bit is set to 1 by the peripheral whenever a communication error occurs that is not covered by one of the other error flags.

Bit 5: Data Mismatch Error (DME). This bit is set to 1 by the peripheral when a transmitted byte is not read back correctly.

Bit 4: Checksum Error (CKE). This bit is set to 1 by the peripheral when a received checksum does not match the calculated checksum for that frame.

Bit 3: Parity Bit 1 (P1). This bit reflects the status of the P1 parity bit for the most recently received frame. It is updated by the peripheral every time a header is received.

Bit 2: Parity Bit 1 Error (P1E). This bit is set to 1 by the peripheral whenever a header is received with an error in the P1 parity bit. It is cleared to 0 by the peripheral when a header is received with the correct value in the P1 parity bit.

Bit 1: Parity Bit 0 (P0). This bit reflects the status of the P0 parity bit for the most recently received frame. It is updated by the peripheral every time a header is received.

Bit 0: Parity Bit 0 Error (P0E). This bit is set to 1 by the peripheral whenever a header is received with an error in the P0 parity bit. It is cleared to 0 by the peripheral when a header is received with the correct value in the P0 parity bit.

8.3.6 Checksum Register (UART) (CHKSUM)

Register Description: **Checksum Register**
 Register Name: **CHKSUM**
 Register Address: **Module 03h, Index 0Eh**

Bit #	15	14	13	12	11	10	9	8
Name	CHKSUM15	CHKSUM14	CHKSUM13	CHKSUM12	CHKSUM11	CHKSUM10	CHKSUM9	CHKSUM8
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	CHKSUM7	CHKSUM6	CHKSUM5	CHKSUM4	CHKSUM3	CHKSUM2	CHKSUM1	CHKSUM0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: CHKSUM is cleared to 0000h on all forms of reset.

Bits 15 to 8: Received Checksum (CHKSUM[15:8]). This register reflects the checksum that was received on the bus during the most recent LIN frame. This register is updated by hardware whenever a complete frame is transmitted or received.

Bits 7 to 0: Calculated Checksum (CHKSUM[7:0]). This register reflects the checksum that was calculated by the peripheral for the most recent LIN frame. This register is updated by hardware whenever a complete frame is transmitted or received. When the peripheral is receiving a frame, the checksum is calculated from the data received on the bus. When the peripheral is transmitting a frame, the checksum is calculated from the data that is being transmitted.

8.3.7 Interrupt State Vector Register (ISVEC)

Register Description: **Interrupt State Vector Register**
 Register Name: **ISVEC**
 Register Address: **Module 03h, Index 0Fh**

Bit #	7	6	5	4	3	2	1	0
Name	—	—	—	—	ISVEC3	ISVEC2	ISVEC1	ISVEC0
Reset	0	0	0	0	1	1	1	1
Access	r	r	r	r	r	r	r	r

r = read

Note: ISVEC is cleared to 0Fh on all forms of reset.

Bits 7 to 4: Reserved.

Bits 3 to 0: Interrupt State Vector 3:0 (ISVEC[3:0]). This field is a priority encoded state vector that can be read by the host and used as an index into an interrupt service routine table. If an interrupt is pending, the value in this field corresponds to the highest priority pending interrupt. The state vector encoding is defined as shown in Table 8-3.

Table 8-3. LIN Priority Encoded Interrupts

ISVEC[3:0]	INTERRUPT STATUS
0	A wake-up request was detected on the LIN bus while the peripheral was in the low-power sleep mode.
1	Activity was detected on the LIN bus while the peripheral was in the low-power sleep state. This condition is only set if the PM bit (CNT2.3) is set to 1.
2	In slave mode, this vector indicates that a header was received with a protected identifier that was within a valid range (i.e., an always valid frame or one that passed the filter check specified by the IDFBH and IDFBL fields). In master mode, this vector indicates that the protected identifier has been successfully transmitted.
3	Buffer overrun. This condition is set when a byte of data is received and there is no room in the receive FIFO.
4	Other communications error. This condition is set when a communication error occurs that is not explicitly assigned to another state vector.
5	Reserved. This condition is never set.
6	Partial frame received successfully. This condition is set whenever the amount of data in the receive FIFO exceeds the threshold configured by the RXFT[1:0] field (FCON.3:2).
7	Reserved. This condition is never set.
8	Complete frame received successfully. This condition is set when the last byte of a frame has been received and the checksum is valid.
9	Reserved. This condition is never set.
10	Partial frame transmitted successfully. This condition is set whenever the amount of data in the transmit FIFO falls below the threshold configured by the TXFT[1:0] field (FCON.5:4).
11	Complete frame transmitted successfully. This condition is set whenever the last byte of a frame has been transmitted and no errors occurred on the bus during the transmission.
12	No master response. This condition is set if the peripheral is issuing wake-up requests on the bus and there is no response from the master.
13	No bus activity. This condition is set if the bus is idle for a period of more than 4 seconds.
14	Reserved. This condition is never set.
15	No pending interrupt. This condition is set when there are no pending interrupts.

8.3.8 Status Register 0 (UART) (STA0)

Register Description: **Status Register 0**
 Register Name: **STA0**
 Register Address: **Module 03h, Index 11h**

Bit #	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INP	BUSY
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: STA0 is cleared to 00h on all forms of reset.

Bits 7 to 2: Reserved.

Bit 1: Interrupt Pending (INP). This bit is set to 1 by the peripheral in LIN master or LIN slave mode when an interrupt condition occurs.

Bit 0: State Machine Busy (BUSY). This bit is set to 1 by the peripheral in LIN master or LIN slave mode when the state machine is actively communicating on the bus.

8.3.9 Serial Mode Register (UART) (SMD)

Register Description: **Serial Mode Register**
 Register Name: **SMD**
 Register Address: **Module 03h, Index 12h**

Bit #	7	6	5	4	3	2	1	0
Name	EIR	OFS	—	—	—	IE	SMOD	FEDE
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	r	r	r	rw	rw	rw

r = read, w = write

Note: SMD is cleared to 00h on all forms of reset.

Bit 7: Enable Infrared Modulation (EIR). Setting this bit to 1 enables the peripheral to modulate the transmitted data with the waveform supplied by an on-chip timer. If this bit is cleared to 0, the infrared modulation is disabled.

If the CONFIG_IR parameter is set to 0, this bit is not implemented and always reads 0.

Bit 6: Output Format Select (OFS). This bit determines the polarity of the output waveform when infrared modulation is enabled. If OFS = 1, the modulated waveform is normally high. If OFS = 0, the modulated waveform is normally low.

If the CONFIG_IR parameter is set to 0, this bit is not implemented and always reads 0.

Bits 5 to 3: Reserved. Read returns 0.

Bit 2: Serial Port Interrupt Enable (IE). Setting this bit to 1 enables the peripheral to issue interrupts. No interrupts are issued if this bit is cleared to 0.

Bit 1: Serial Port Baud Rate Select (SMOD). In legacy UART mode, this bit enables a prescaler for the baud-rate generator. This bit serves no purpose in LIN master or LIN slave mode. The SMOD selects the final baud rate for the asynchronous mode:

SMOD = 1: 16 times the baud clock for mode 1 and 3

32 times the system clock for mode 2

SMOD = 0: 64 times the baud clock for mode 1 and 3

64 times the system clock for mode 2

Bit 0: Framing Error Detection Enable (FEDE). Setting this bit to 1 enables access to the framing error detection flag through the SM0 bit (SCON.7).

8.3.10 FIFO Control Register (UART) (FCON)

Register Description: **FIFO Control Register**
 Register Name: **FCON**
 Register Address: **Module 03h, Index 13h**

Bit #	7	6	5	4	3	2	1	0
Name	FTF	FRF	TXFT1	TXFT0	RXFT1	RXFT0	OE	FEN
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: FCON is cleared to 00h on all forms of reset.

Bit 7: Flush Transmit FIFO (FTF). If the host sets this bit to 1, the transmit FIFO pointer is reset to 0 and all transmit FIFO status flags return to a value consistent with a system reset. Note that this does not affect the contents of the FIFO itself.

Bit 6: Flush Receive FIFO (FRF). If the host sets this bit to 1, the receive FIFO pointer is reset to 0 and all receive FIFO status flags return to a value consistent with a system reset. Note that this does not affect the contents of the FIFO itself.

Bits 5 and 4: Transmit FIFO Threshold 1:0 (TXFT[1:0]). This field is used to select the “almost empty” threshold for the transmit FIFO. In LIN slave mode, this field also determines the timing of the “partial frame transmitted” interrupt. The almost empty condition is set according to the following table. Note that this field has no effect if the FIFO is disabled (FEN = 0).

TXFT[1:0]	INTERRUPT TIMING
00	The almost empty flag is never set by hardware. In LIN master or LIN slave mode, the partial frame transmitted interrupt is issued when the transmit FIFO is full.
01	The almost empty flag is set and the partial frame transmitted interrupt is issued when the transmit FIFO is more than 25% full.
10	The almost empty flag is set and the partial frame transmitted interrupt is issued when the transmit FIFO is more than 50% full.
11	The almost empty flag is set and the partial frame transmitted interrupt is issued when the transmit FIFO is more than 75% full.

Bits 3 and 2: Receive FIFO Threshold 1:0 (RXFT[1:0]). This field is used to select the “almost full” threshold for the receive FIFO. In LIN master or LIN slave mode, this field also selects the timing of the “partial frame received” interrupt. The “almost full” condition is set according to the following table. Note that this field has no effect if the FIFO is disabled (FEN = 0).

RXFT[1:0]	INTERRUPT TIMING
00	The almost full flag is never set by hardware. In LIN master or LIN slave mode, the partial frame received interrupt is issued when the receive FIFO is full.
01	The almost full flag is set and the partial frame receive interrupt is when the receive FIFO is more than 25% full.
10	The almost full flag is set and the partial frame received interrupt is issued when the receive FIFO is more than 50% full.
11	The almost full flag is set and the partial frame received interrupt is issued when the receive FIFO is more than 75% full.

Bit 1: Overflow Error (OE). This bit is set to 1 by the peripheral when a byte of data is received and the receive FIFO is full.

Bit 0: FIFO Enable (FEN). This bit enables or disables the transmit and receive FIFO buffers. If this bit is set to 1, both FIFO buffers are enabled. If this bit is cleared to 0, both FIFO buffers behave as though a single byte of storage is available.

8.3.11 Control Register 0 (UART) (CNT0)

Register Description: **Control Register 0**
 Register Name: **CNT0**
 Register Address: **Module 03h, Index 14h**

Bit #	7	6	5	4	3	2	1	0
Name	WU	FP1	FP0	INE	AUT	INIT	LUN1	LUN0
Reset	1	0	0	0	1	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: CNT0 is cleared to 8Bh on all forms of reset.

Bit 7: Wake Up (WU). This bit enables the host to monitor and control the low-power sleep mode status of the peripheral. It can also be used to communicate to the host the status of the activity on the LIN bus.

If the host changes this bit from 1 to 0, the peripheral enters a low-power sleep mode. In this mode, most of the clocks internal to the peripheral are shut down to conserve power. If the peripheral is in the low-power sleep mode and a wake-up condition is detected on the LIN bus, hardware clears this flag to 0 and issues an interrupt.

If the peripheral is in the low-power sleep state and the host sets this bit to 1, the peripheral issues wake-up requests on the bus according to the LIN 2.0 specification. If the peripheral is unable to wake up the master, an error is generated and the peripheral returns to the low-power sleep state.

This bit is automatically cleared to 0 by hardware if the bus is inactive for at least 4 seconds.

Bits 6 and 5: Receive Filter Prescaler Mode 1:0 (FP[1:0]). This field selects the operating mode of the receive filter according to the following table.

FP[1:0]	MODE
00	Filter disabled.
01	In this mode, the filter can reject a noise pulse up to 2 system clock cycles wide. A total of 3 clock cycles of latency are added to the receive filter output.
10	In this mode, the filter can reject a noise pulse up to 4 system clocks wide. A total of 7 clock cycles of latency are added to the receive filter output.
11	In this mode, the filter can reject a noise pulse up to 6 system clocks wide. A total of 11 clock cycles of latency are added to the receive filter output.

Bit 4: Interrupt Enable (INE). This bit enables the peripheral to issue an interrupt to the host. If this bit is cleared to 0, the interrupt flags are set but no interrupt is generated.

Bit 3: Automatic Checksum Type (AUT). This bit enables or disables the automatic detection of the checksum type in a LIN frame. If this bit is set to 1, the peripheral automatically determines the checksum type based on the identifier in the LIN header. If this bit is cleared to 0, the CK bit (CNT1.7) is used to select the checksum type for the current transmitted or received frame.

Bit 2: Initialization (INIT). This bit is set to 1 by hardware on any system reset. Software must clear it to 0 after all peripheral initialization is complete. Software can also set this bit to 1 at any time to force the peripheral into the reset initialization state. This bit has no effect in legacy UART mode.

Bit 1: LIN or UART Mode Select (LUN[1:0]). This field selects between LIN master, LIN slave, and legacy UART mode according to the following table.

LUN[1:0]	MODE
00	Legacy UART Mode (default)
01	Legacy UART Mode
10	LIN Master Mode
11	LIN Slave Mode

8.3.12 Control Register 2 (UART) (CNT2)

Register Description: **Control Register 2**
 Register Name: **CNT2**
 Register Address: **Module 03h, Index 15h**

Bit #	7	6	5	4	3	2	1	0
Name	—	—	—	DMIS	PM	HDO	FBS	BTH
Reset	0	0	0	0	0	0	0	0
Access	r	r	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: CNT2 is cleared to 00h on all forms of reset.

Bits 7 to 5: Reserved. Read returns 0.

Bit 4: Data Mismatch Disable (DMIS). When this bit is set to 1 data mismatch detection is disabled (ERRR.5).

Bit 3: Power Management (PM). This bit controls the behavior of interrupt generation when the peripheral is in the low-power sleep mode. If this bit is set to 1, the peripheral issues an interrupt as soon as any bus activity is detected in sleep mode. If this bit is cleared to 0, the peripheral does not issue any interrupts until a wake-up request is detected.

Bit 2: Header Only (HDO). When this bit is set to 1, the peripheral receives a LIN header only and ignores all data in the frame. If this bit is set to 1 before a LIN header has been received, it automatically is cleared to 0 by the peripheral when a header is received. If this bit is set to 1 by the host after the peripheral has been processed a valid header, the receive buffer is emptied and the peripheral returns to the dominant state.

Bit 1: Force Break/Sync (FBS). If this bit is set to 1 while a peripheral is configured in LIN master mode, the peripheral issues a break/sync sequence on the LIN bus. The host must write the identifier for the frame into the transmit buffer. Parity bits for the transmitted identifier are computed automatically.

Bit 0: Break Threshold (BTH). This bit is used to select between a 10-bit (BTH = 0) or 11-bit (BTH = 1) detection threshold for a break symbol on the LIN bus.

8.3.13 Identifier Boundary Register (UART) (IDFB)

Register Description: **Identifier Boundary Register**
 Register Name: **IDFB**
 Register Address: **Module 03h, Index 16h**

Bit #	15	14	13	12	11	10	9	8
Name	—	—	IDFBH5	IDFBH4	IDFBH3	IDFBH2	IDFBH1	IDFBH0
Reset	0	0	1	1	1	1	1	1
Access	r	r	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	—	—	IDFBL5	IDFBL4	IDFBL3	IDFBL2	IDFBL1	IDFBL0
Reset	0	0	0	0	0	0	0	0
Access	r	r	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: IDFB is cleared to 3F00h on all forms of reset.

Bits 15, 14, 7, and 6: Reserved. Read returns 0.

Bits 13 to 8: Identifier Boundary High 13:8 (IDFBH[5:0]). This field selects the upper limit for a valid identifier. If the peripheral receives a header with an identifier that is higher than this limit, the remainder of the frame is ignored. Note that frames with an identifier in the inclusive range 38h–3Fh are always considered valid (the 3Ch–3Fh range is used by LIN 2.0 control frames and the 38h–3Bh range is used by SAE J2602 control frames).

Bits 5 to 0: Identifier Boundary Low 5:0 (IDFBL[5:0]). This field selects the lower limit for a valid identifier. If the peripheral receives a header with an identifier below this limit, the remainder of the frame is ignored.

8.3.14 Serial Address Register (UART) (SADDR)

Register Description: **Serial Address Register**
 Register Name: **SADDR**
 Register Address: **Module 03h, Index 17h**

Bit #	7	6	5	4	3	2	1	0
Name	SADDR7	SADDR6	SADDR5	SADDR4	SADDR3	SADDR2	SADDR1	SADDR0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: SADDR is cleared to 00h on all forms of reset.

Bits 7 to 0: Serial Address Register 7:0 (SADDR[7:0]). In legacy UART mode, this register can be written by the host to configure the address of the peripheral in one of the 9-bit communication modes (mode 2 or mode 3). This register serves no function if the peripheral is configured for LIN master or LIN slave mode.

8.3.15 Serial Address Mask Register (UART) (SADEN)

Register Description: **Serial Address Mask Register**
 Register Name: **SADEN**
 Register Address: **Module 03h, Index 18h**

Bit #	7	6	5	4	3	2	1	0
Name	SADEN7	SADEN6	SADEN5	SADEN4	SADEN3	SADEN2	SADEN1	SADEN0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: SADEN is cleared to 00h on all forms of reset.

Bits 7 to 0: Serial Address Mask Register 7:0 (SADEN[7:0]). In legacy UART mode, this register can be written by the host to configure the address mask for the peripheral in one of the 9-bit communication modes (mode 2 or mode 3). This register serves no function if the peripheral is configured for LIN master or LIN slave mode.

8.3.16 Bit Timing Register (UART) (BT)

Register Description: **Bit Timing Register**
 Register Name: **BT**
 Register Address: **Module 03h, Index 19h**

Bit #	15	14	13	12	11	10	9	8
Name	BT15	BT14	BT13	BT12	BT11	BT10	BT9	BT8
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	BT7	BT6	BT5	BT4	BT3	BT2	BT1	BT0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: BT is cleared to 0000h on all forms of reset.

Bits 15 to 0: Bit Timing Register 15:0 (BT[15:0]). In legacy UART mode, the host initializes this register to configure the baud-rate generator for the correct communication speed. In LIN mode, the host writes this register during system initialization to configure the normal bit timing on the bus. Hardware updates this register with the measured bit timing whenever a break/sync sequence is detected. The host can always read this register to determine the measured bit timing for the most recent LIN frame.

In legacy UART mode this register is the phase register.

Bits 15 to 0: Phase Register 15:0 (BT[15:0]). This register is used to load and read the 16-bit value in the phase register that determines the baud rate for the serial port 0.

8.3.17 Timer Register (UART) (TMR)

Register Description: **Timer Register**
 Register Name: **TMR**
 Register Address: **Module 03h, Index 1Ah**

Bit #	15	14	13	12	11	10	9	8
Name	TMR15	TMR14	TMR13	TMR12	TMR11	TMR10	TMR9	TMR8
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	TMR7	TMR6	TMR5	TMR4	TMR3	TMR2	TMR1	TMR0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: TMR is cleared to 0000h on all forms of reset.

Bits 15 to 0: Timer Register 15:0 (TMR[15:0]). This register is written by the host during system initialization to indicate the number of system clock cycles in a 50µs period.

8.4 MAXQ7667 LIN

8.4.1 MAXQ7667 LIN Features

The MAXQ7667 LIN provides the following features:

- Supports LIN 1.3, LIN 2.0, and SAE J2602
- Automatic baud-rate detection and LIN frame synchronization
- Automatic calculation of standard (LIN 1.3) and enhanced (LIN 2.0) checksums
- Frame lengths up to 64 bytes
- Support for LIN power management modes
- 8-byte transmit and receive FIFOs to reduce processor intervention
- Configurable digital filter for receiver

8.4.2 LIN Frame

The LIN protocol uses a single message frame to synchronize and address the nodes and to exchange data between them. The master sets the transmission speed and sends the message header shown in Figure 8-2. The header starts with the sync/break sequence followed by the sync field. The slave recognizes the sync/break and sets up to receive the sync field. Once the slave receives the sync field, it adjusts its internal baud rate to match the master. The slave then examines the identifier field and acknowledges the message if the identifier field is within the slave's identifier range. The slave then sends data to the master.

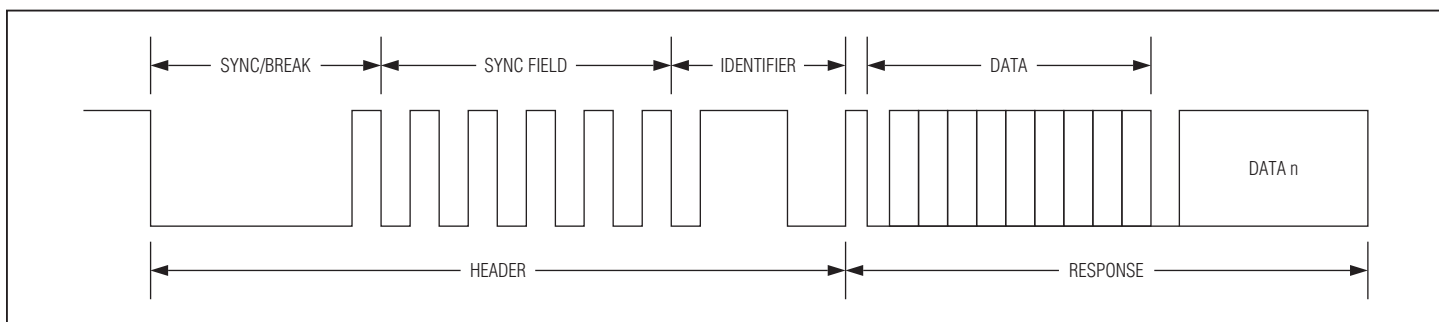


Figure 8-2. LIN Frame Format

8.4.3 LIN Peripheral

Figure 8-2 shows the dedicated LIN hardware. The MAXQ core interfaces with the LIN peripheral through the registers in Module 3. The application software configures the LIN controller for master or slave operation and monitors the LIN interrupts for status. The application software writes and reads the SBUF register for sending data over the LIN bus. The checksum and parity calculations are done in hardware and require no software intervention.

The LIN control registers and bit descriptions are detailed in Module 3 of the MAXQ7667 register specification.

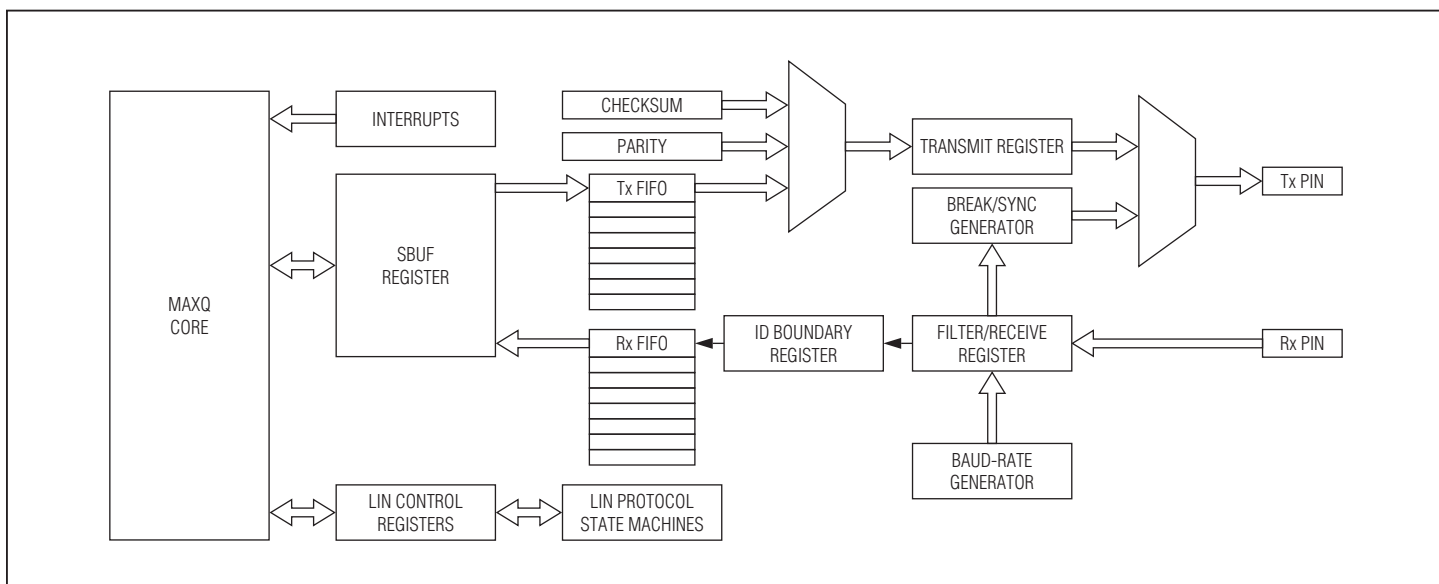


Figure 8-3. LIN Block Diagram

8.4.4 LIN Master

The MAXQ7667 can be used as the LIN bus master as shown in Figure 8-4. As the LIN bus master node, the MAXQ7667 controls all traffic on the bus. The master sets the communication speed by sending the break and sync sequence. It also sends the identification field and checksum for specific slaves to respond to.

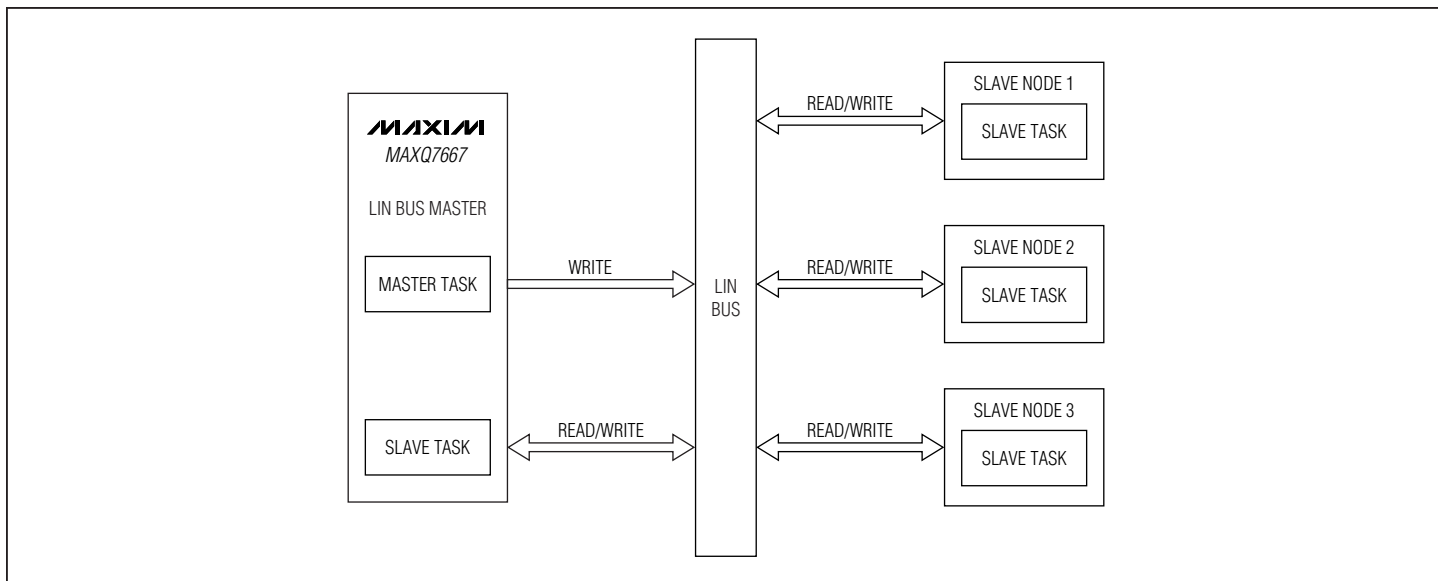


Figure 8-4. LIN Bus Master Communication

8.4.4.1 LIN Master Setup Example

To set the master mode, the LIN or UART mode select (LUN[1:0]) bits in CNT0 are set for master mode. To set up the master mode, the following registers should be set as follows.

- Set the INIT bit (CNT0.2) to 1 to force LIN into initialization state.
- Set the LUN[1:0] bits (CNT0.1:0) to 0x03 to set LIN master mode.
- Set the AUT bit (CNT0.3) to 1 to enable automatic checksum.
- Set the INE bit (CNT0.4) to 1 to enable interrupts.
- Set the FP[1:0] bits (CNT0.6:5) to 0 to disable receive filter.
- Set the WU bit (CNT0.7) to 1 to set the peripheral to wake up.
- Set CNT1 to 0x00 to enable transmit.
- Set the CNT2 to 0x00 (default).
- Set SCON to 0x60, i.e., mode 1.
- Clear the INIT bit (CNT0.2) by setting it to 0, to clear the LIN init state.
- Set BT (0x0320 sets it for 20kBd).

8.4.4.2 LIN Master Sending Protected ID and Break/Sync Example

The LIN master is now configured and ready to send the identifier and break/sync sequence to initiate communication. To have the master send the identifier and break/sync sequence, a write to SBUF and CNT2 is required as follows.

- Load SBUF with the master protected identifier in the range 0x00–0x3F.
- Set the FBS bit (CNT2.1) to 1 to transmit a break/sync.

Once the LIN controller has been set up to transmit the identifier and break/sync, the MAXQ processor must allow sufficient time for the data to be shifted out the transmit and break/sync serial registers shown in Figure 8-3. At the 20kbps rate, this takes a very long time (16,000 instructions at 16MHz) relative to the MAXQ processor speed. Therefore, the MAXQ processor must wait until the break/sync and identifier has been transmitted before the LIN controller can be set to receive.

8.4.4.3 LIN Master Polling for Transmit Complete Example

One method to wait until the data has been transmitted is to use the LIN interrupts to monitor the LIN state in the ISVEC register. The LIN ISVEC register state 2 indicates when the protected identifier has been successfully transmitted in master mode. If a flag is set in ISR when state 2 occurs, this flag can be tested to determine if the LIN has finished sending the break/sync. See *Section 8.4.10: LIN Interrupts* for details on the interrupt.

- In the interrupt handler a flag (e.g., TX_OK = 1) is set for state 2
- Wait until TX_OK = 1 before proceeding.

This is just one method and is inefficient because the processor must wait for the transmit sequence to be complete instead of doing other tasks. For example, the LIN message frame shown in Figure 8-1 contains the sync break, sync field, and the identifier. If we assume a nominal sync break of 7 bits, the sync field of 6 bits, and the identifier field of 6 bits, the total is 20 bits of data. At the 20kbps rate each bit is 50 μ s. Therefore, 50 μ s per bit multiplied by 20 bits takes approximately 1ms to shift out the LIN data. The MAXQ7667 could do another task for 1ms and then check the TX_OK flag to verify whether the LIN master is done sending the frame rather than using the previously mentioned polling method.

8.4.4.4 LIN Master Setup to Receive Acknowledge from Slave Example

The LIN master has transmitted the break/sync and is waiting for a LIN slave to acknowledge the message and reply by sending data and a checksum. The LIN master controller needs to be set up to receive a reply using the master's slave task as shown in Figure 8-4. This is done by writing the RTN (CNT1.7) bit and writing the FL (CNT1.[5:0]) bits. In this example, the RTN bit is set for receive and the frame length is set for one byte. (**Note that the frame length is always one byte greater than the frame length bit settings.**) Additionally, the write to CNT1 must be done with one write for proper LIN state machine operation.

- Set CNT1 to 0x80 to receive and specify the frame length.

Again the MAXQ processor must wait for the LIN data to be transmitted from the slave to the master before reading the slave data in SBUF. ISVEC is again used with a flag to indicate when the data has been received from the slave. ISVEC state 8 indicates when a completed frame has successfully been received.

- In the interrupt handler a flag (e.g., RX_OK = 1) is set for state 8.
- Wait until RX_OK = 1 before proceeding.
- Read slave data from SBUF.

The wait loop forces the processor to wait until the data has been received and prevents the processor from doing other tasks. After the slave data and checksum have been received, the processor can retrieve the data by reading SBUF. As was previously mentioned in the transmit section, the wait loop can be replaced with another task, and the RX_OK flag can be tested to check if the receive is complete.

8.4.5 LIN Slave

The MAXQ7667 can be used as a LIN slave as shown in Figure 8-5. As the LIN slave, the bus master controls all traffic on the bus. The slave must synchronize its baud rate to match the master and respond if the master's identifier is within the slave's address range.

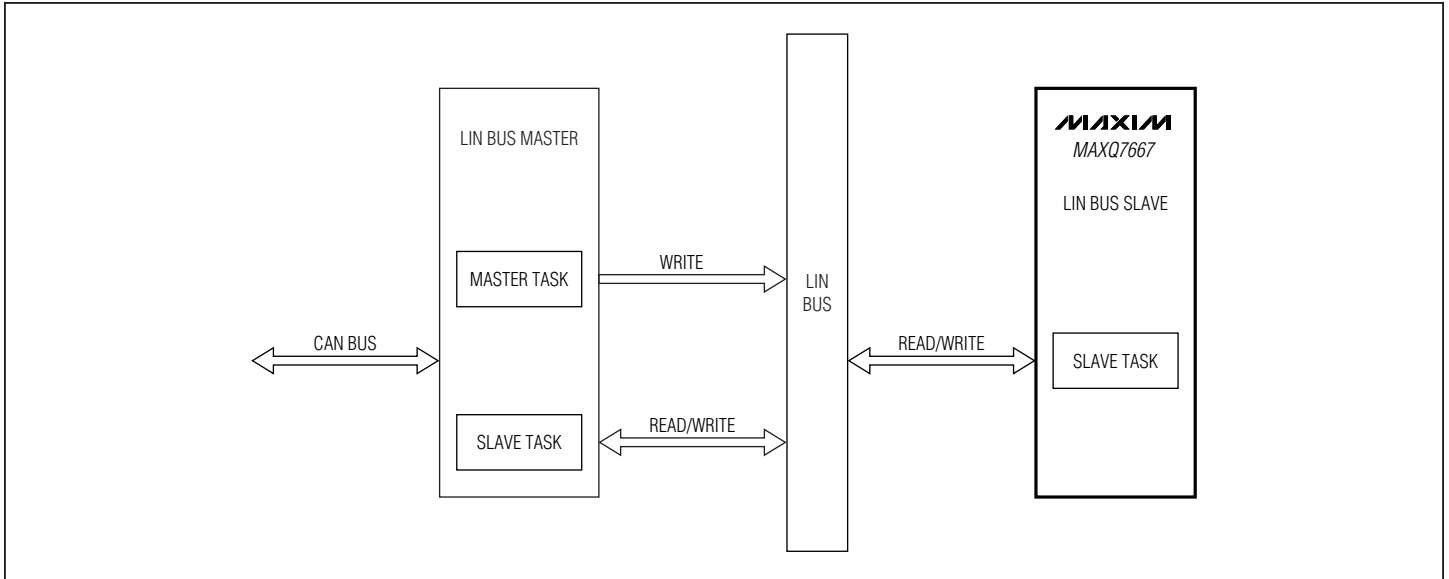


Figure 8-5. LIN Bus Slave Communication

8.4.5.1 LIN Slave Setup Example

To set the slave mode, the LIN or UART mode select (LUN[1:0]) bits in CNT0 are set for slave mode. To set up the slave mode, the following registers are set as shown.

- Set INIT bit (CNT0.2) to 1 to force LIN into initialization state.
- Set LUN[1:0] bits (CNT0.1:0) to 0x02 to set LIN slave mode.
- Set AUT bit (CNT0.3) to 1 to enable automatic checksum.
- Set INE bit (CNT0.4) to 1 to enable interrupts.
- Set FP[1:0] bits (CNT0.6:5) to 0 to disable receive filter.
- Set WU bit (CNT0.7) to 1 to set the peripheral to wake up.
- Set CNT1 to 0x00.
- Set CNT2 to 0x00.
- Set SCON to 0x60, i.e., mode 1.
- Clear the INIT bit (CNT0.2) by setting it to 0.
- Set BT (0x0320 sets it for 20kBd).

8.4.5.2 LIN Slave Receiving Protected ID and Break/Sync Example

The LIN slave is now configured and ready to receive and send the break/sync sequence and identifier. The MAXQ processor must wait until the break/sync sequence has been successfully received.

8.4.5.3 LIN Slave Polling for Receive Complete Example

One method to accomplish this is to use the LIN interrupts to monitor the LIN state in the ISVEC register similar to what the master does. Using the LIN interrupt handler, ISVEC state 2 indicates when the protected identifier has been successfully received in slave mode.

- In the interrupt handler a flag (e.g., RX_OK = 1) is set for state 2.
- Wait until RX_OK = 1 before proceeding.

This method is inefficient because the processor must wait for the receive data to be complete instead of doing other tasks.

8.4.5.4 LIN Slave Setup to Transmit Data from Slave Example

Once the break/sync has been received with a valid protected identifier, the slave can respond to the master by sending data and the checksum. To send the master data, the frame length and the RTN bit must be set and the data must be loaded to SBUF. (Note that changes to CNT1 should happen with one single write to the register.) The following example sends one data byte to the master. Note that the number of bytes of data transmitted is one greater than what is sent in CNT1. The checksum is automatically calculated and sent by the LIN controller, and requires no software intervention from the MAXQ processor.

- Set CNT1 to transmit 1 byte (CNT1 = 0x00).
- Load transmit data into SBUF.

8.4.5.5 LIN Slave Setup to Transmit Multiple Bytes from Slave Example

If more than one byte is to be sent, the FIFO can be enabled to send up to eight bytes (see *Section 8.4.8: LIN Transmit and Receive FIFO*) or the SBUF can be written successively. If using the SBUF to write successively, the MAXQ processor must wait until the previous SBUF contents have been shifted out over the LIN bus. One method to wait is to detect if the LIN state machine is busy by monitoring the BUSY bit in the STA0 register as shown in the following example. The example demonstrates the monitoring of the BUSY bit between write to SBUF.

- Load transmit data into SBUF.
- Wait until STA0 busy bit is no longer set (STA0 = 1).
- Load transmit data into SBUF.
- Wait until STA0 busy bit is no longer set (STA0 = 1).

8.4.6 Receive Filter

The LIN receive path shown in Figure 8-3 contains a noise filter to reject spurious bus noise. The filter is programmable to reject between two and six system clocks. To enable the receive filter and set the filter parameters, the receive filter prescaler mode (FP[1:0]) bits in CNT0 are used.

8.4.7 Identifier Boundary Register (IDFB)

The Identifier Boundary (IDFB) register follows the receive filter (see Figure 8-3). IDFB is used to filter the incoming message identifier. The master transmits the identifier in the range of 0x00–0x3F. The slave can set the lower limit and the upper limit of identifiers that the slave recognizes. By setting these limits, the slave can reply to a range of valid identifiers or to a single identifier. If the identifier transmitted by the master is not within the range of valid identifiers set on the slave, the message is ignored by the slave. IDFB has a power-on reset value of 0x3F00 that accepts all identifiers.

8.4.8 LIN Transmit and Receive FIFO

To reduce the load on the processor, both transmit and receive paths have 8-byte FIFOs. The transmit and receive FIFOs are both enabled or disabled with the FIFO enable (FEN) bit in FCON. The FIFOs can also be cleared independently with the flush receive FIFO (FRF) and the flush transmit FIFO (FTF) bits. Both the transmit and receive FIFO have independent thresholds that are programmable to signal the processor when the receive FIFO is almost full or the transmit FIFO is almost empty. The receive FIFO threshold (RXFT) bits set the number of bytes that are in the receive FIFO when the almost full flag causes an interrupt. Conversely, the transmit FIFO threshold (TXFT) bits set the number of bytes that are in the transmit FIFO when the almost empty flag causes an interrupt. The transmit and receive FIFO status can be monitored by reading FSTAT. The following steps can be used to enable the FIFO.

- Set FEN bit (FCON.0) to 1 to enable LIN FIFO.
- Set FRF bit (FCON.6) to 1 to reset receive FIFO pointer and flags.
- Set FTF bit (FCON.7) to 1 to reset the transmit FIFO pointer and flags.

8.4.9 Setting LIN Baud Rate

The LIN baud rate is set using the BT register. The LIN master uses the baud-rate bit timing when it issues a break/sync sequence. The LIN slave captures the sync timing and set its BT to match the master's.

To set the LIN baud rate the following formula is used:

$$BT = \text{System Clock} / \text{LIN Baud Rate}$$

For example, for the LIN baud rate of 20kBd, with a 16MHz crystal as the clock source.

$$BT = (16\text{MHz} / 20\text{kHz}) \geq 800 \geq 0x0320$$

The BT register is then loaded with 0x0320, which sets the LIN communication speed for 20kBd.

8.4.10 LIN Interrupts

After the LIN controller has been set up, the ISVEC register can be used to monitor the state of the LIN controller. To use the interrupts, the global, Module 3, and LIN interrupts need to be enabled. Once enabled, the ISVEC register interrupt is used to determine the LIN status. After the status is determined, an interrupt service routine (ISR) can take the appropriate action. The following steps are used to enable the LIN interrupts.

- Set the interrupt global enable (IGE) bit in the Interrupt Control (IC) register.
- Set the interrupt mask 3 (IM3) bit in the Interrupt Mask (IMR) register.
- Set the interrupt enable (INE) bit in the CNT0 register.

Once the interrupts have been enabled for LIN, the ISVEC register contains the priority encoded interrupt vector states as shown in Table 8-3 (**Note that the highest priority is 0 and the lowest priority is 15.**) The application software should monitor the ISVEC states for LIN status and take action if necessary.

8.4.10.1 LIN Interrupt Example

See the following example on how to use the ISVEC register in an ISR. This example assumes an interrupt is caused by LIN and determines the status from the ISVEC register.

- Clear the interrupt ID 3 (I13) bit in the Interrupt Identification (IIR) register.
- Preserve the ISVEC register by copying it to another variable such as ISV_Test.
- Test the ISV_Test variable to see which condition from Table 8-3 caused the interrupt.
- Take the appropriate action to resolve the interrupt cause.

8.4.10.2 LIN Master/Slave Interrupt Example

In the following example, flags are used for the LIN master and slave examples mentioned previously. The ISR sets a flag when the transmission or reception of data is complete.

- Test the ISV_Test variable and if ISV_Test = 2, either receive or transmit complete.
- If LIN set for slave (CNT0 bits LUN = 2), set RX_OK = 1.
- If LIN set for master (CNT0 bits LUN = 3), set TX_OK = 1.

8.4.11 Power Features

To reduce the power, the system clock can be removed by placing the MAXQ7667 in stop mode. To place the MAXQ7667 in stop mode, the CKCN register bit STOP is written.

The MAXQ7667 has internal circuitry to detect activity on the LIN bus and wake up from either of the low-power modes. To enable the wake-up circuitry, the wake-up bit (WU) in CNT0 is set to detect activity on the LIN bus. In addition, an interrupt can occur by setting the power management (PM) bit in CNT2.

8.4.12 LIN Error Handling

When the MAXQ7667 is used as a LIN master or slave, the application software should monitor the status of all LIN communication with the ISVEC and ERRR registers. In the previous master and slave sections, no provision was provided to check for errors. When the LIN controller communicates on the LIN bus, the ISVEC register should be monitored for error conditions and action should be taken to handle the error. Also the ERRR register should be monitored after communication to detect any error conditions that occurred as a result of transmitting or receiving data.

8.5 MAXQ7667 UART

The MAXQ7667 has a UART for serial communication. The UART can be used for standard RS-232 and RS-485 communication by adding an external transceiver. Figure 8-6 shows the UART hardware.

8.5.1 MAXQ7667 UART Features

The MAXQ7667 UART provides the following features:

- Synchronous and asynchronous transfers
- Independent baud-rate generator
- Two-wire interface
- Full-duplex operation for asynchronous data transfers
- Half-duplex operation for synchronous data transfers
- Programmable interrupts for transmit and receive
- Programmable 9th bit parity support
- Start/stop bit support
- Transmit and receive FIFOs

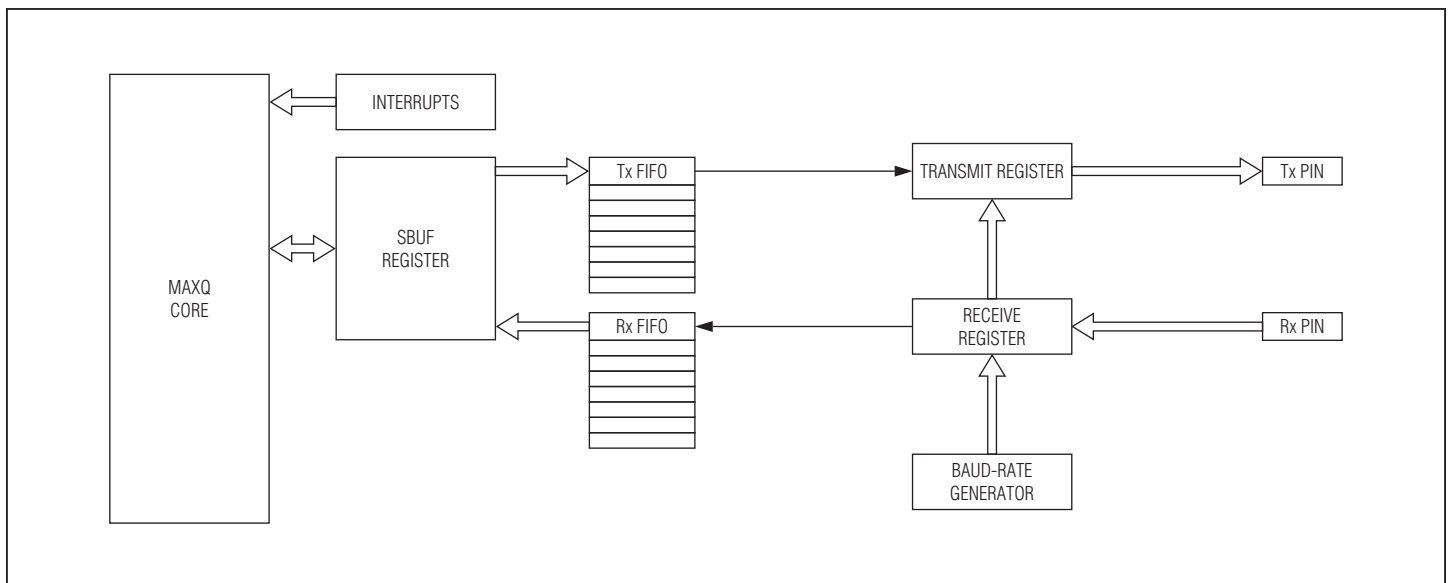


Figure 8-6. UART Block Diagram

8.5.2 MAXQ7667 UART Modes

8.5.2.1 UART Mode 0

This mode is used to communicate in synchronous, half-duplex format with devices that accept the MAXQ7667 microcontroller as a master. Figure 8-7 shows a functional diagram and basic timing of this mode. As can be seen, there is one bidirectional data line (Rx) and one shift clock line (Tx) used for communication. Mode 0 requires that the MAXQ7667 be the master since it generates the serial shift clock for data transfers that occur in either direction.

The Rx signal is used for both transmission and reception. Data bits enter and exit least significant bit first. The Tx pin provides the shift clock. The baud rate is equal to the shift clock frequency. When not using power management mode, the baud rate in mode 0 is equivalent to the system clock divided by either 12 or 4, as selected by SM2 bit in the SCON register.

The UART begins transmitting when a write is performed on SBUF. The internal shift register then begins to shift data out. The clock is activated and transfers data until the 8-bit value is complete. Data is presented one clock prior to the falling edge of the shift clock (TXD) so that an external device can latch the data using the rising edge of the shift clock.

The UART begins to receive data when the REN bit in the SCON register is set to logic 1 and the RI bit is set to logic 0. This condition indicates that there is data to be shifted in on the Rx pin. The shift clock (TXD) is activated and data is latched on the rising edge. The external device should therefore present data on the falling edge. This process continues until all eight bits have been received. The RI bit is automatically set to logic 1, one clock cycle following the last rising edge of the shift clock on TXD. This causes reception to stop until the SBUF has been read and the RI bit is cleared. When RI is cleared, another byte can be shifted in, if available.

8.5.2.2 UART Mode 1

This mode provides asynchronous, full-duplex communication. A total of 10 bits is transmitted, consisting of a start bit (logic 0), 8 data bits, and 1 stop bit (logic 1), as illustrated in Figure 8-8. The data is transferred least significant bit first. The baud rate is programmable through the baud-clock generator and is discussed later.

Following a write to SBUF, the UART begins transmission five clock cycles after the first baud clock from the baud-clock generator. Transmission takes place on the Tx pin. It begins with the start bit being placed on the pin. Data is then shifted out onto the pin, least significant bit first. The stop bit follows. The TI bit is set by hardware after the stop bit is placed on the pin. All bits are shifted out at the rate determined by the baud clock generator.

Once the baud-clock generator is active, reception can begin at any time. The REN bit must be set to logic 1 to allow reception. The detection of a falling edge on the Rx pin is interpreted as the beginning of a start bit, and begins the reception process. Data is shifted in at the selected baud rate. At the middle of the stop bit time, certain conditions must be met to load SBUF with the received data in the receive shift register:

- RI must = 0, and either
 - if SM2 = 0, the state of the stop bit does not matter
 - or
 - if SM2 is 1, the state of the stop bit must be 1

If these conditions are true, the SBUF is loaded with the received byte, the RB8 bit is loaded with the stop bit, and the RI bit is set. If these conditions are false, the received data is lost (SBUF and RB8 not loaded) and RI is not set. Regardless of the receive word status, after the middle of the stop bit time, the receiver resumes looking for a 1-to-0 transition on the Rx pin.

Each data bit received is sampled on the 7th, 8th, and 9th clock used by the divide-by-16 counter. Using majority voting, two equal samples out of the three determine the logic level for each received bit. If the start bit was determined to be invalid (= 1), the receive logic resumes looking for a 1-to-0 transition on the Rx pin to start the reception of data.

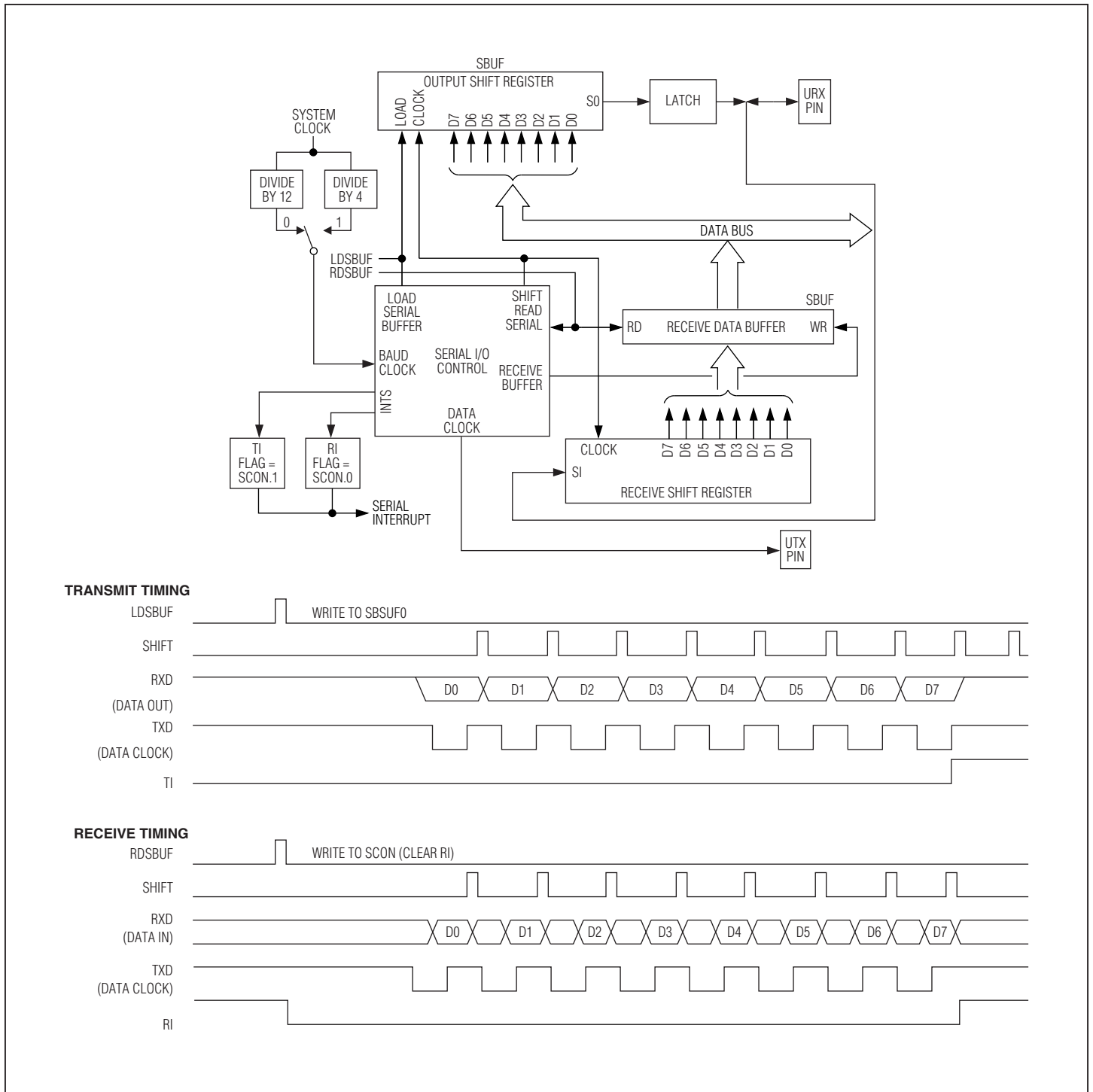


Figure 8-7. UART Mode 0

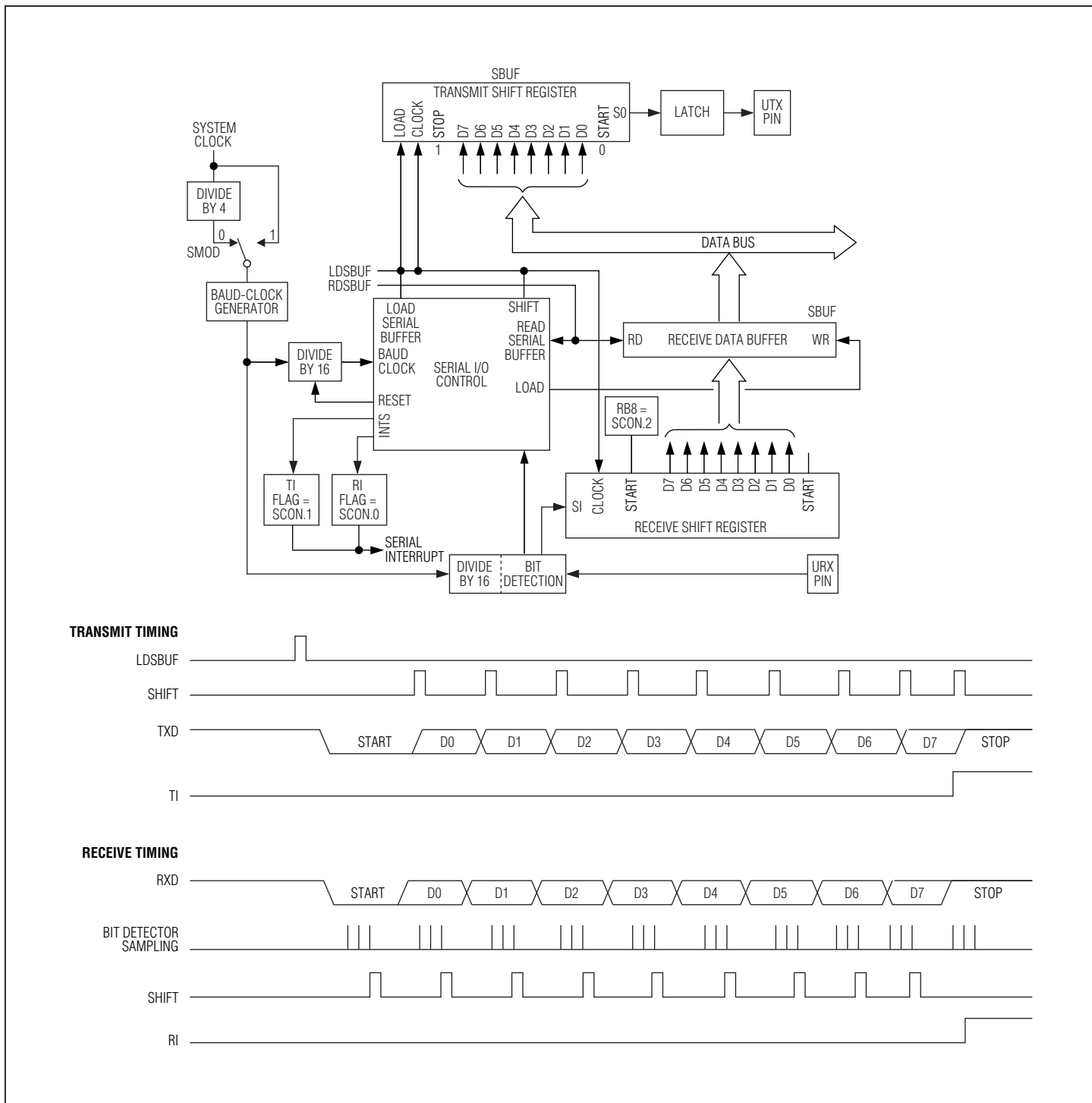


Figure 8-8. UART Mode 1

8.5.2.3 UART Mode 2

This mode uses a total of 11 bits in asynchronous, full-duplex communication as illustrated in Figure 8-9. The 11 bits consist of one start bit (a logic 0), 8 data bits, a programmable 9th bit, and one stop bit (a logic 1). Like mode 1, the transmissions occur on the Tx signal pin and receptions on Rx.

For transmission purposes, the 9th bit can be stuffed as logic 0 or 1. A common use is to put the parity bit in this location. The 9th bit is transferred from the TB8 bit position in the SCON register following a write to SBUF to initiate a transmission. The UART transmission begins five clock cycles after the first rollover of the divide-by-16 counter following a software write to SBUF. It begins with the start bit being placed on the Tx pin. The data is then shifted out onto the pin, least significant bit first, followed by the 9th bit, and finally the stop bit. The TI bit is set when the stop bit is placed on the pin.

Once the baud-rate generator is active and the REN bit has been set to logic 1, reception can begin at any time. Reception begins when a falling edge is detected as part of the incoming start bit on the Rx pin. The Rx pin is then sampled according to the baud rate speed. The 9th bit is placed on the RB8 bit location in the SCON register. At the middle of the 9th bit time, certain conditions must be met to load SBUF with the received data:

- RI must = 0, and either
 - if SM2 = 0, the state of the 9th bit does not matter
 - or
 - if SM2 is 1, the state of the 9th bit must be 1

If these conditions are true, the SBUF is loaded with the received byte, the RB8 bit is loaded with the 9th bit, and the RI bit is set. If these conditions are false, the received data is lost (SBUF and RB8 not loaded) and RI is not set. Regardless of the receive word status, after the middle of the stop bit time, the receiver resumes looking for a 1-to-0 transition on the Rx pin.

Data is sampled in a similar fashion to mode 1 with the majority voting on three consecutive samples. Mode 2 uses the sample divide-by-16 counter with either the system clock divided by 2 or 4, thus resulting in a baud clock of either system clock/32 or system clock/64.

8.5.2.4 UART Mode 3

This mode has the same operation as mode 2, except for the baud-rate source. As shown in Figure 8-10, mode 3 generates the baud rates through the baud-clock generator. The bit shifting and protocol are the same. The baud-clock generator is discussed in another section.

8.5.3 Framing Error Detection

A framing error occurs when a valid stop bit is not detected. This results in the possible improper reception of the serial word. The UART can detect a framing error and notify the software. Typical causes of framing errors are noise and contention. The framing error condition is reported in the SCON register for the UART.

The framing error bit, FE, is located in SCON. Note that this bit normally serves as SM0 and is described as SM0/FE in the register description. Framing error information is made accessible by the FEDE (framing error detection enable) bit located at SMD.0. When FEDE is set to logic 1, the framing error information is shown in SM0/FE (SCON). When FEDE is set to logic 0, the SM0 function is accessible. The information for bits SM0 and FE is actually stored in different registers. Changing FEDE only modifies which register is accessed, not the contents of either.

The FE bit is set to 1 when a framing error occurs. It must be cleared by software. Note that the FEDE state must be 1 while reading or writing the FE bit. Also note that receiving a properly framed serial word does not clear the FE bit. This must be done in software.

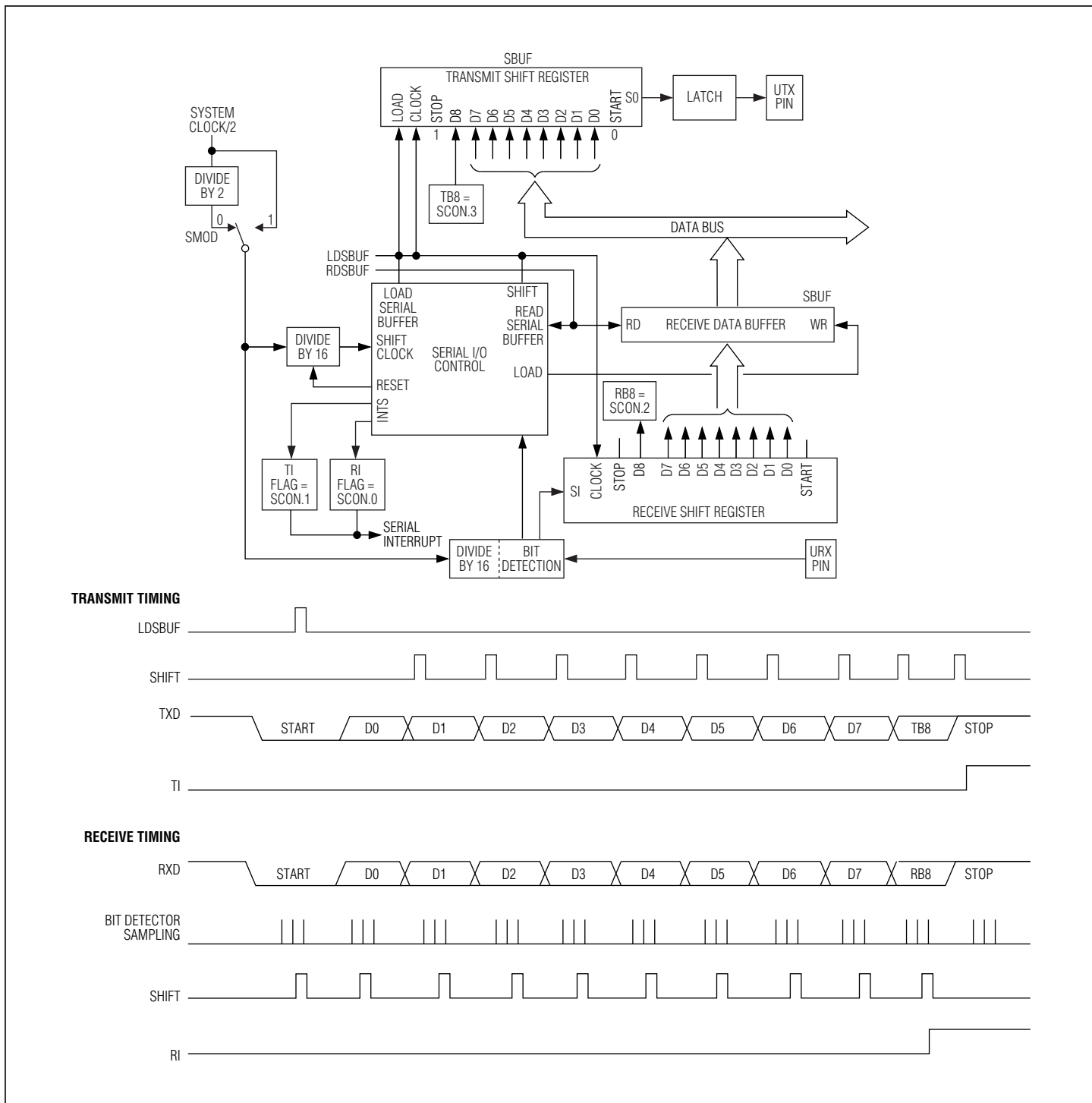


Figure 8-9. UART Mode 2

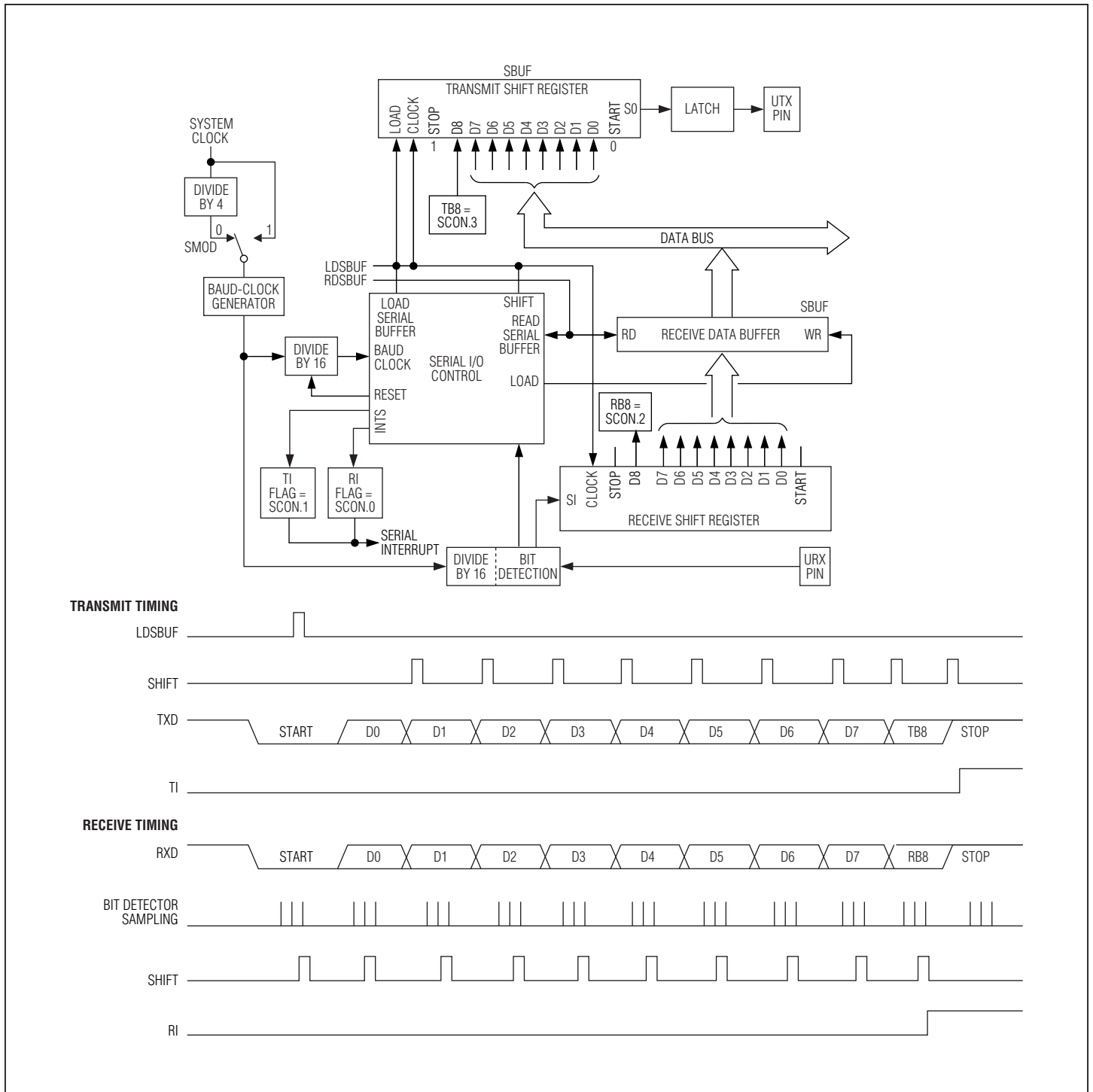


Figure 8-10. UART Mode 3

8.5.4 UART Mode 1 Asynchronous Full-Duplex Setup Example

To set up the UART/LIN controller for UART operation, the LIN or UART mode select (LUN[1:0]) bits in CNT0 are set for UART mode. To set up the UART mode, set the registers as shown in the following example. In this example the UART is set for asynchronous communication using mode 1 with 115,200 baud.

- Set LUN bits (CNT0.1:0) to 0 to select the UART mode.
- Set FEDE bit (SMD.0) to 0 to disable the framing error detection.
- Set SMOD bit (SMD.1) to 1 (16 times the baud clock for mode 1).
- Set IE bit (SMD.2) to 0.
- Set OFS bit (SMD.6) to 1.
- Disable infrared modulation by setting EIR bit (SMD.7) to 0.
- Set TMR (0x800).
- Set SM2 (SCON.5) to 0 for mode 1.
- Set SM1 (SCON.6) to 1 for mode 1.
- Set SM0 (SCON.7) to 0 for mode 1.
- Set BT to 0x3AFB to set baud rate divider with 16MHz clock for 115,200 baud.

The UART is now configured and ready to transmit and receive. To transmit or receive data, the SBUF register is written or read. The MAXQ core must wait until the transmit buffer shifts out the data before writing more data to the SBUF register. To receive, the MAXQ core must wait until the data has been completely shifted into the receive register before reading. The preferred method to control the flow to and from the SBUF register is to use transmit and receive interrupts.

8.5.5 UART Interrupts Example

To use the UART interrupts, the global, Module 3, and UART interrupts must be enabled. The following steps show how to enable the interrupts.

- Set the IGE bit in the Interrupt Control (IC.0) register.
- Set the IM3 bit in the Interrupt Mask (IMR.3) register.
- Set the IE bit in the Serial Mode (SMD.2) register.

Once the UART interrupts have been enabled, the interrupt flags should be cleared to ensure proper initialization as shown in the following example.

- Clear the RI bit (SCON.0) by setting it to 0.
- Clear the TI bit (SCON.1) by setting it to 0.

8.5.6 UART Transmit Data Example

To transmit data using the SBUF register without overwriting it, the application code should examine the transmit interrupt (TI) flag immediately after writing the SBUF register. When the transmit buffer is shifting out the data, the TI flag is set and the application code should wait until complete. The following example details the method to prevent overwriting.

- Clear the TI bit (SCON.1) by setting it to 0.
- Send the character by writing to the SBUF register.
- Wait until the TI bit is set before sending next character.

8.5.7 UART Receive Data Example

To receive serial data using the UART, the SBUF register is read. To ensure the data has been written, the application code should examine the receive interrupt (RI) flag immediately before reading the SBUF register. The application code can wait until the data has been completely shifted in to the receive buffer and the RI flag is set indicating new data has been received. The following example details the method to prevent reading before the register is full.

- Wait until the RI bit (SCON.0) is set.
- Clear the RI bit by setting it to 0.
- Read data from SBUF register.

8.5.8 Setting UART Baud Rate

The UART baud rate is set with the BT register. The baud rate is derived from the system clock that can be the external crystal for baud-rate accuracy. To set baud rate, the following equation is used.

$$\text{Baud rate} = (\text{BT} \times \text{FREQ}) / (2^{23} / 2^{(\text{SMOD} \times 2)})$$

where BT is the Bus Timing register and FREQ is the system clock frequency.

For example, to set the baud rate for 115,200 with the system clock set to the external 16MHz crystal. The mode is asynchronous, therefore, SMOD = 1. Using the above equation and solving for BT we get the following:

$$\text{Baud} = (\text{BT} \times 16,000,000) / (2^{23} / 2^2) \geq \text{BT} \times (16,000,000 / 2,097,152) = 115,200 \text{ then}$$

$$\text{BT} = 115,200 \times 2,097,152 / 16,000,000 \geq 15,099.4944$$

Round down to 15,099 as the BT register only holds integers.

Then, 15,099 = 0x3AFB hex

Load the BT register with = 0x3AFB

To verify, plug the 15,099 value back into the equation to compute the exact baud rate. The exact baud rate using the BT divider is 115,196.228 baud (well within 0.01% tolerance) and should work perfectly.

8.5.9 UART FIFO

The UART FIFO operates exactly the same as the LIN FIFO. The setup and use is detailed in *Section 8.4.8: LIN Transmit and Receive FIFO*.

SECTION 9: ENHANCED SERIAL PERIPHERAL INTERFACE (SPI) MODULE

This section contains the following information:

9.1 Architecture	9-4
9.1.1 SPI Status and Control Registers	9-5
9.1.1.1 SPI Data Buffer Register (SPIB)	9-6
9.1.1.2 SPI Control Register (SPICN)	9-7
9.1.1.3 SPI Configuration Register (SPICF)	9-9
9.1.1.4 SPI Clock Register (SPICK)	9-10
9.1.2 Master Mode	9-11
9.1.3 Slave Mode	9-13
9.1.4 SPI Clocking Modes	9-14
9.1.5 Baud-Rate Determination	9-15
9.1.6 Data Format	9-15
9.2 SPI Port Errors	9-15
9.2.1 Receive Overrun Flag (ROVR)	9-15
9.2.2 Write Collision Flag (WCOL)	9-16
9.2.3 Mode Fault Flag (MODF)	9-16
9.3 SPI Interrupts	9-16
9.4 Resetting the SPI Port	9-17
9.5 Oscillator/Clock Power-Saving Management Modes	9-17
9.5.1 Stop Mode	9-17

LIST OF FIGURES

Figure 9-1. SPI Bus Configuration	9-3
Figure 9-2. SPI Port Functional Diagram	9-4
Figure 9-3. Master Mode Transfer Cycle Operation	9-12
Figure 9-4. Slave Mode Transfer Cycle Operation	9-13
Figure 9-5. SPI Clocking Modes	9-14
Figure 9-6. MAXQ Interrupt Enabling Scheme	9-16

LIST OF TABLES

Table 9-1. SPI Port I/O Pins	9-5
Table 9-2. SPI Clocking Mode Configuration	9-9

SECTION 9: ENHANCED SERIAL PERIPHERAL INTERFACE (SPI) MODULE

The MAXQ7667 has a powerful hardware serial peripheral interface (SPI) module that allows for serial communication with a variety of external devices. The SPI port on the MAXQ is a fully independent module that can be accessed under program control. This module supports access to the full 4-wire, full-duplex serial bus, and can be operated in either master mode or slave mode. The enhanced SPI provides the ability to select the state of the slave select in slave mode.

The SPI bus is typically implemented with one master device and multiple slave devices. Each slave device has a unique SS pin that is used to enable transfers to that device. Figure 9-1 shows a typical SPI bus configuration. Auxiliary digital I/O pins are used as additional SS.

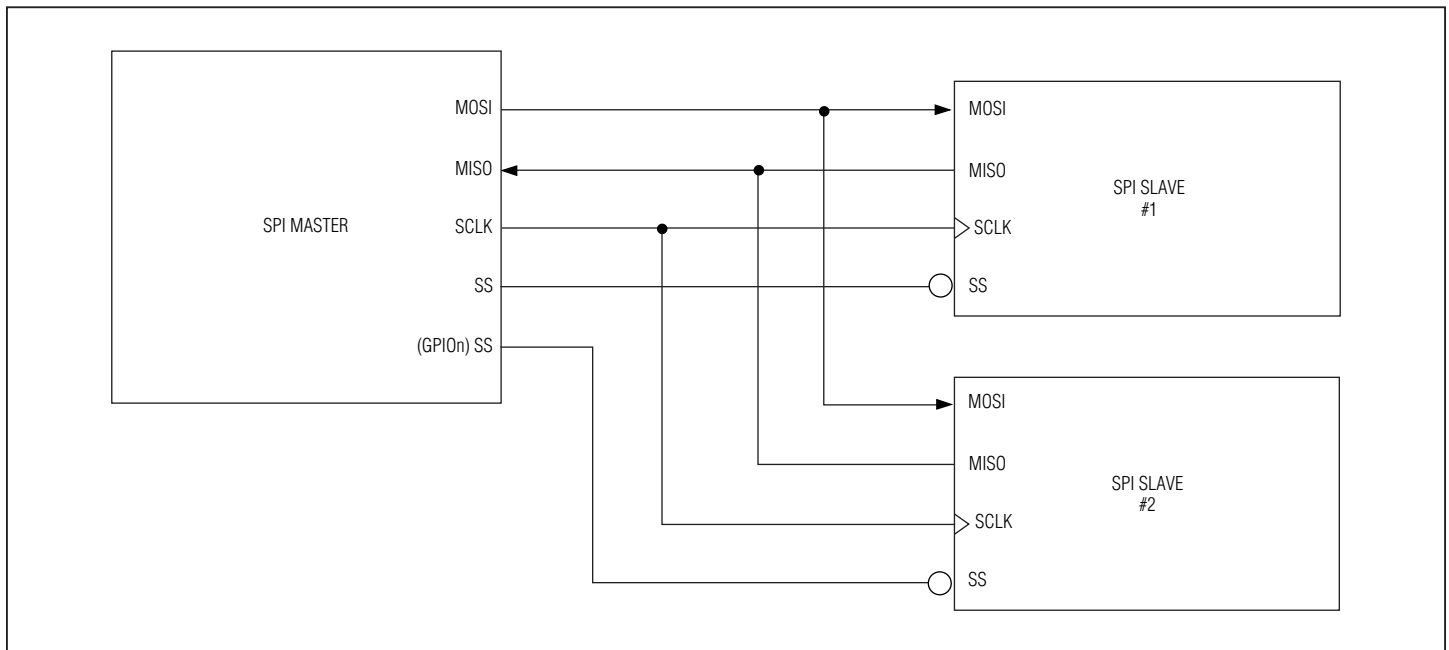


Figure 9-1. SPI Bus Configuration

9.1 Architecture

The MAXQ7667 contains a shift register, an independent read buffer, a programmable baud-rate generator, and numerous flags to control and check the status of the port through interrupts or in a polled fashion. The SPI port shift register handles both transmit and receive data transfers. The read path is double buffered to free up the shift register for the next SPI transfer. Once a SPI read transfer is complete, data is loaded into the read buffer and the SPI port is ready to accept another character for input or output. The SPI port is single buffered in the transmit direction. The SPI port should not be written to until a previous transfer is completed as signaled by the SPI transfer complete flag (SPIC) found in the SPI control register (SPICN). A data overrun occurs if the previous character is not read out of the data buffer before the next incoming character is completely shifted into the port. The SPI data buffer register (SPIB) in the SFR bank provides access for both transmit and receive data for the CPU. The **maximum** data rate of the SPI port is:

- 1/2 of SYSCLK for master mode
- 1/8 of SYSCLK for slave mode

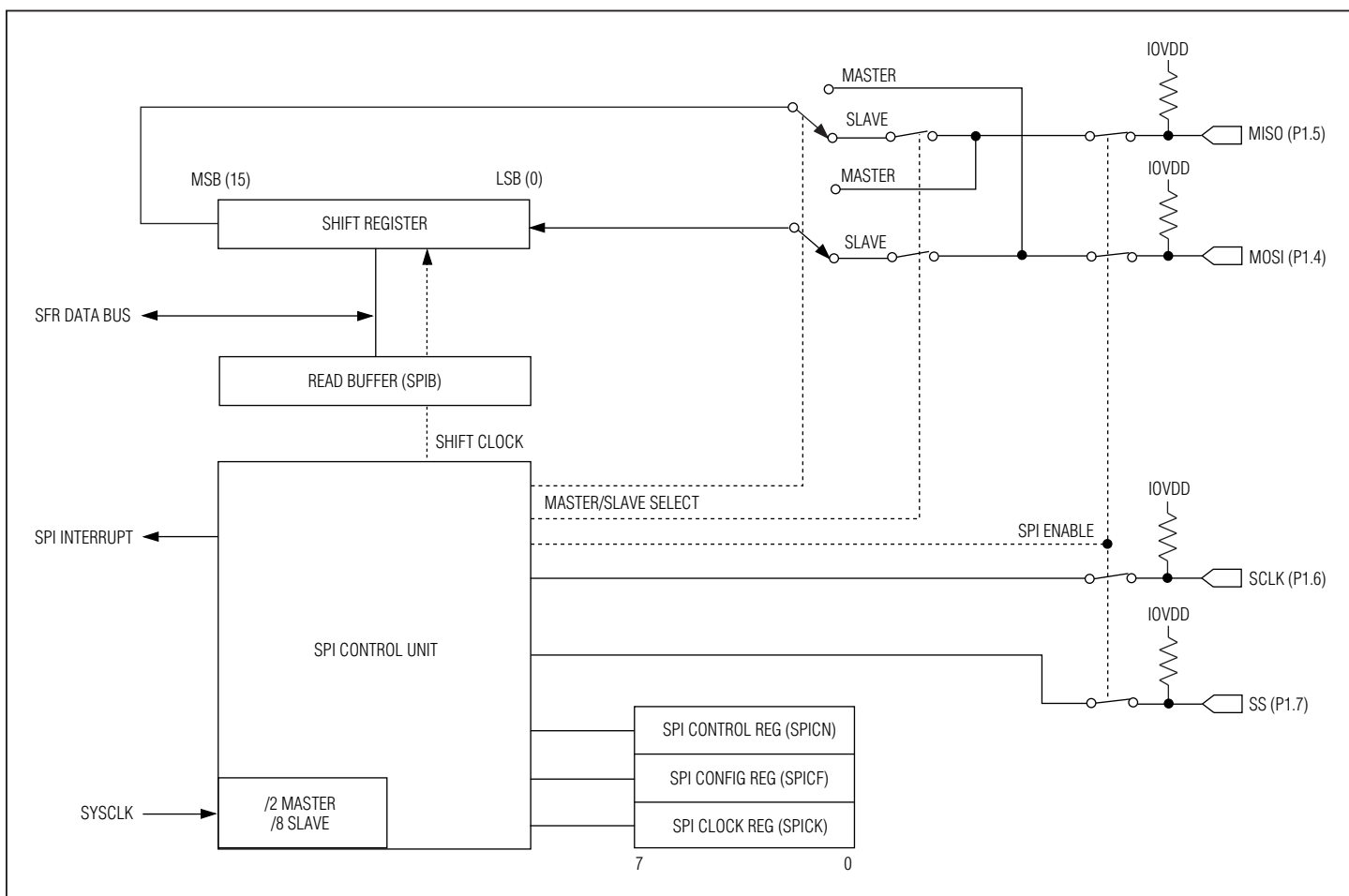


Figure 9-2. SPI Port Functional Diagram

The MAXQ7667 supports:

- 8-bit or 16-bit character lengths
- Master or slave mode
- Four standard SPI clocking modes
- Programmable master SPICK baud-rate generator
- Configurable SS pin
- MSB first data shifting
- Mode-fault detection
- Data overrun and collision detection
- Interrupt or polled operation

The MAXQ7667 does not support:

- Free-running SPICK and frame sync transfer modes
- LSB first shift

Peripherals used should be compatible with the MAXQ7667-supported SPI modes.

Table 9-1. SPI Port I/O Pins

PIN	MULTIPLEXED WITH PORT PIN	INTERFACE SIGNAL	FUNCTION
3	P1.5	MISO	Master In-Slave Out. This signal is an output from an SPI slave device and an input for an SPI master device. It is used to serially transfer data from the slave to the master. Data is transferred MSB first.
2	P1.4	MOSI	Master Out-Slave In. This signal is an output from an SPI master device and an input for an SPI slave device. It is used to serially transfer data from a master to a slave device. Data is transferred MSB first. When in slave mode, the MAXQ default configuration for MISO is an input with a weak pullup.
4	P1.6	SCLK	SPI Clock. This serial clock is an output from an SPI master device and an input for an SPI slave device. It is used to shift the data between a master and a slave device. The clock polarity select (CKPOL) and the clock phase select (CKPHA) bits configure the SCLK mode used. The MAXQ7667 supports all four SPI clocking modes.
5	P1.7	SS	Slave Select. The slave-select signal can be configured (SPICF register) as an active-low or active-high input. In the slave mode, SS is asserted and held low or high (based on its configuration) for the entire transfer. Shifting out, or in, is stopped when SS is deasserted. This pin can also be configured to detect a mode-fault condition in master mode.

Additional SS pins in master mode can be defined under firmware control using the digital I/O pins.

9.1.1 SPI Status and Control Registers

Four registers control the SPI port configuration, report status, and operation in the MAXQ7667. These registers are found in the special function register bank in Module 1, indexes 6, 7, 8, and 9. The registers are:

- SPI Data Buffer Register (SPIB): Module 1, Index 6
- SPI Control Register (SPICN): Module 1, Index 7
- SPI Configuration Register (SPICF): Module 1, Index 8
- SPI Clock Register (SPICK): Module 1, Index 9

9.1.1.1 SPI Data Buffer Register (SPIB)

The SPI data buffer register (SPIB) uses the SPI port shift register for write operations and a separate read buffer for read operations. Bit 15 is the MSB of the register and bit 0 is the LSB of the register. This register is word- or byte-access enabled and the data is shifted towards the MSB only. The port is double buffered on read and single buffered on write. See Figure 9-2. Write access to this register is allowed only outside of an active transfer cycle (STBY = 0).

SPIB writes are blocked if the shift register is busy (STBY = 1) and the write collision flag will be set.

This register is accessed through *direct* read and write addressing and is cleared on all forms of *reset* (power-on reset, brownout reset, external reset, watchdog reset, and internal system reset).

Register Description: **SPI Data Buffer Register**
 Register Name: **SPIB**
 Register Address: **Module 01h, Index 06h**

Bit #	15	14	13	12	11	10	9	8
Name	SPIB15	SPIB14	SPIB13	SPIB12	SPIB11	SPIB10	SPIB9	SPIB8
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	SPIB7	SPIB6	SPIB5	SPIB4	SPIB3	SPIB2	SPIB1	SPIB0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 0: SPIB Data Bits 15:0 (SPIB[15:0]). The SPIB data port is the register location for the SPI read and write data. Write access is only allowed when the port has completed a transfer. The read port is double buffered to allow time for firmware to read data into the μ C. SPIB7 is the MSB when 8-bit data transfers are used; SPIB15 is the MSB for 16-bit data.

9.1.1.2 SPI Control Register (SPICN)

The SPI control register (SPICN) contains both mode control and status flags to monitor the operation of the SPI port. This register is byte accessed

Register Description: **SPI Control Register**
 Register Name: **SPICN**
 Register Address: **Module 01h, Index 07h**

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	STBY	SPIC	ROVR	WCOL	MODF	MODFE	MSTM	SPIEN
Reset	0	0	0	0	0	0	0	0
Access	r	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 8: Reserved. Read 0, write ignored.

Bit 7: SPI Transfer Busy Flag (STBY). This flag indicates when an SPI transfer cycle is in progress. This bit is set and cleared under hardware control. All forms of reset clear this bit.

- 0 = SPI module is idle—no transfer in progress.
- 1 = SPI transfer in progress.

Bit 6: SPI Transfer Complete Flag (SPIC). This flag detects the completion of an SPI transfer. For a write transfer, the SPIC flag is set when the last bit is shifted out of the SPIB. For a read transfer, this flag is set when the SPI port read buffer is loaded with new data. This flag reports:

- 0 = No SPI transfer, since the bit was cleared.
- 1 = SPI transfer complete.

All forms of reset clear this bit. Writing a 0 to the bit also clears it. This bit is read- and write-access enabled and can be set by individual bit write operations.

Bit 5: Receive Overrun Flag (ROVR). This flag detects if a received character was lost because a previous piece of data had not been read out of the SPIB register. The data in the SPI port shift register is overrun and lost. This bit is read- and write-access enabled and can be set by individual bit write operations.

- 0 = No receive overrun has occurred.
- 1 = Receive overrun occurred.

Bit 4: Write Collision Flag (WCOL). This flag detects a write to the SPIB register, while the port is shifting out data from a previous operation. If a transfer is in progress (STBY = 1), an attempt to write the SPIB sets the WCOL flag, and the operation is ignored so as not to corrupt the contents of the SPIB. Accesses to the SPIB are only allowed when the STBY flag is 0. All forms of reset clear this bit. Writing a 0 to it also clears the bit. This bit is read- and write-access enabled and can be set by individual bit write operations.

- 0 = No write collision has been detected.
- 1 = Write collision detected.

Bit 3: Mode-Fault Flag (MODF). This bit is the mode-fault flag for SPI master mode operation. A mode fault occurs when the SS line is asserted (0) on a device that is configured in SPI master mode. When mode-fault detection is enabled (MODFE = 1) in master mode, detection of a high-to-low transition on the SS pin signifies a mode fault. All forms of reset clear this bit. Writing a 0 to this bit also clear it. This bit is read- and write-access enabled and can be set by individual bit write operations.

- 0 = No mode fault has been detected.
- 1 = Mode fault detected while operating as a master (MSTM = 1).

Bit 2: Mode-Fault Enable (MODFE). When MODFE is set to logic 1, the SS input pin is enabled to detect a mode fault during SPI master mode operation. A mode fault occurs when the SS line is asserted (0) on a device that is configured in master mode. When this bit is programmed to 0, mode-fault detection is disabled. All forms of reset clear this bit. This bit is read- and write-access enabled and can be set by individual bit write operations.

Bit 1: Master Mode Enable (MSTM). The MSTM bit functions as the master mode enable bit for the SPI module. Note that this bit can only be set if the SPI port is inactive and SS is deasserted. All forms of reset clear this bit. A mode-fault condition also clears this bit if the mode-fault-enable bit is set. This bit is read- and write-access enabled and can be set by individual bit write operations.

- 0 = SPI module operates in slave mode.
- 1 = SPI module operates in master mode.

Bit 0: SPI Enable (SPIEN). The SPIEN bit enables the SPI port for operation. All forms of reset clear this bit. The bit is also cleared on a mode-fault condition if the mode-fault-enable bit is set. This bit is read- and write-access enabled and can be set by individual bit write operations.

- 0 = SPI module and the baud-rate generator are disabled.
- 1 = SPI module and the baud-rate generator are enabled.

9.1.1.3 SPI Configuration Register (SPICF)

The SPI configuration register (SPICF) contains the port configuration selects. The SPI port supports 8-bit and 16-bit character transfers using the SS pin and an internal bit counter to control transfer completion. Data is shifted on the rising or falling edge of the SCLK as selected by the clock polarity select (CKPOL) and the clock phase select (CKPHA) bits found in the SPICF. These registers can be word, byte, or bit accessed.

Table 9-2 shows the four SPI clocking modes supported by the MAXQ7667.

Table 9-2. SPI Clocking Mode Configuration

SPI MODE	CLOCK POLARITY (CKPOL)	CLOCK PHASE (CKPHA)	TRANSFER METHOD
0	0	0	SPICK rising edge transfer
1	0	1	SPICK falling edge transfer
2	1	0	SPICK falling edge transfer
3	1	1	SPICK rising edge transfer

Register Description: **SPI Configuration Register**
 Register Name: **SPICF**
 Register Address: **Module 01h, Index 08h**

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	ESPII	SAS	—	—	—	CHR	CKPHA	CKPOL
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	r	r	r	rw	rw	rw

r = read, w = write

Bits 15 to 8, 5 to 3: Reserved. Read 0, write ignored.

Bit 7: Enable SPI Interrupt (ESPII). This bit enables any of the SPI interrupt source flags (MODF, WCOL, ROVR, SPIC) to generate interrupt requests.

- 0 = SPI interrupt sources disabled.
- 1 = SPI interrupt sources enabled.

Bit 6: Slave-Active Select (SAS). This bit is used to determine the SS active state. When the SAS is cleared to 0, the SS is active low and responds to an external low signal. When the SAS is set to 1, the SS pin is active high. Note: The SAS selection is only available in the enhanced SPI peripheral.

Bit 2: Character Length Select (CHR). This bit determines the character length for an SPI transfer cycle. A character can be 8 bits in length or 16 bits in length.

- 0 = 8-bit character length specified.
- 1 = 16-bit character length specified.

Bit 1: Clock Phase Select (CKPHA). This bit selects the clock phase and is used with the CKPOL bit to define the SPI data transfer format.

- 0 = Data sampled on the active clock edge.
- 1 = Data sampled on the inactive clock edge.

Bit 0: Clock Polarity Select (CKPOL). This bit selects the clock polarity and is used with the CKPHA bit to define the SPI data transfer format.

- 0 = Clock idles in the logic 0 state (rising = active clock edge).
- 1 = Clock idles in the logic 1 state (falling = active clock edge).

9.1.1.4 SPI Clock Register (SPICK)

The SPI clock register (SPICK) contains the divider ratio from the system clock to set the SPI port baud rate. In master mode, the maximum clock frequency allowed is one-half the system clock rate. The SPI baud rate is selected by:

$$\text{SPI master baud rate} = (0.5 \times \text{system clock frequency}) / (\text{SPICK}[7:0] + 1)$$

In slave mode, the SPI clock is an input driven by an external master device. That rate is limited to a maximum of one-eighth the system clock.

Register Description: **SPI Clock Register**
 Register Name: **SPICK**
 Register Address: **Module 01h, Index 09h**

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	SPICK7	SPICK6	SPICK5	SPICK4	SPICK3	SPICK2	SPICK1	SPICK0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 15 to 8: Reserved. Read 0, write ignored.

Bits 7 to 0: SPI Clock Divider Ratio (SPICK[7:0]). This value is a number that ranges from 0 to 255.

9.1.2 Master Mode

The master mode selection enables the MAXQ7667 to control transfer cycles on the SPI bus. An SPI master device drives the SCLK and the SS pins on the SPI bus. Transfers are initiated when firmware writes a character to the SPIB. The CHR bit selects between 8-bit or 16-bit transfer cycles. Data is shifted out serially, with MSB first on the MOSI pin using the SCLK as the shift clock. The SPI bus is a full-duplex bus, so as data is shifted out of the MOSI pin, new data is returned from the slave device into the master shift register on the MISO pin. When the transfer cycle begins, the STBY flag is set and the SS pin is asserted, followed by clearing of the SPIC bit by the user. At the completion of the transfer the SPIC flag is set and STBY flag clears. If auxiliary digital I/O pins are used as additional SS lines, these must be asserted under software control immediately preceding the SPIB register data write. The SPI port flags can be monitored by a polling routine or by interrupts. The SPI port baud rate is determined by the master clock setting, as programmed in the SPICK register. The MAXQ7667 can run up to one-half the system clock rate.

The following steps should be taken to initialize and set up the SPI port for master mode operation.

- 1) If SPI port is enabled:
 - a. Wait until STBY clears.
 - b. Disable SPI port.
- 2) If interrupts are used:
 - a. Disable interrupts in the SPICF (ESPIC, bit 7).
 - b. Clear flags SPICK, ROVR, WCOL, MODF.
- 3) Set SPI port baud rate in the SPICK register (from 1/2 to 1/512th of SYSCLK).
- 4) Set SPI clocking mode, character length, and enable interrupts in the SPICF register.
- 5) Set SPI master mode, which automatically configures the SPI port as follows: MISO (port 1.5 as input), MOSI (port 1.4 as output), and SCLK (port 1.6 as output). Mode-fault enable can be enabled here, if desired.
- 6) Enable the SPI port in the SPICN.

Once the port is configured as an SPI bus master, do the following to initiate a transfer cycle:

- 1) Assert the SS pin low. (Any GPIO can be used in place of the SS pin.)
- 2) Write a character to the SPIB. STBY is set (1).
- 3) A series of SCLKs shifts the contents of the SPIB to the selected slave device, while the contents of the slave device are shifted in through the LSB of the master shift register. The number of serial shift clocks required is set by the CHR bit, enabling either 8-bit or 16-bit transfer cycles.
- 4) Once the transfer is complete:
 - a. The serial register loads the completed incoming data into the read buffer.
 - b. The SPIC flag sets, marking the end of the transfer cycle and notifying the μ C that the read buffer has valid data.
 - c. If interrupts are enabled, an interrupt to the μ C is issued.
- 5) Data should be read from the SPIB, and the SPIC flag can be cleared in the SPICN.
- 6) Additional data transfer is possible by repeating steps 2 to 5.
- 7) Once done, deassert the SS pin high.

Data can be written to the serial register once the STBY flag is cleared and the SPIC flag is set. This allows the next transfer to begin while the read operation is pending. In this overlapped mode, the read buffer must be emptied before the next character is completely loaded into the shift register. Should the next transfer cycle complete, the ROVR flag is set if the SPIB contents have not been read.

Figure 9-3 shows a typical master mode transfer cycle with SPIB read and write cycles.

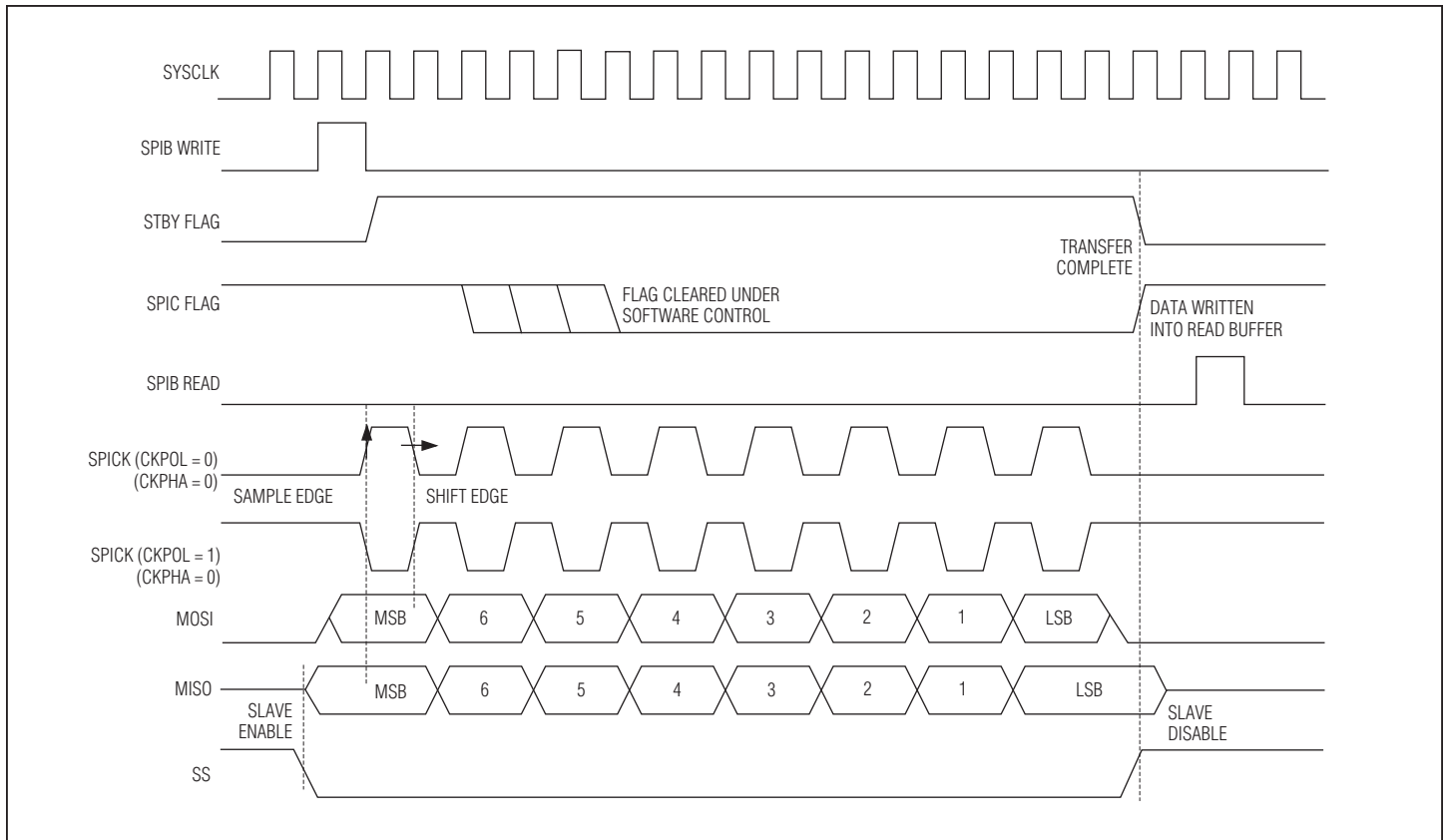


Figure 9-3. Master Mode Transfer Cycle Operation

9.1.3 Slave Mode

The MAXQ7667's SPI port can also be configured as an SPI slave device. This is achieved by enabling the SPI port (SPIEN) with MSTM = 0. The SCLK and SS lines are then driven by the SPI bus master. Any master device driving the SCLK on the MAXQ7667 SPI port should limit the maximum shift rate to one-eighth the MAXQ SYSCLOCK rate. After the MAXQ7667 SPI port has been configured as an SPI slave device, asserting the SS pin initiates the transfer cycle between the MAXQ to the bus master. Once SS is asserted, the STBY flag is set and the SPI cycle is underway. STBY remains set until SS is deasserted (1). Upon completion of the transfer cycle, the SPIC flag is set and the STBY flag is cleared. The transfer cycle is complete when the shift register count is equal to the setting of the CHR length bit (8 or 16 bits). SS can be held low across multiple transfer cycles, when CKPHA = 0, and the MAXQ slave port loads read data into the read buffer when the CHR setting equals the shift count.

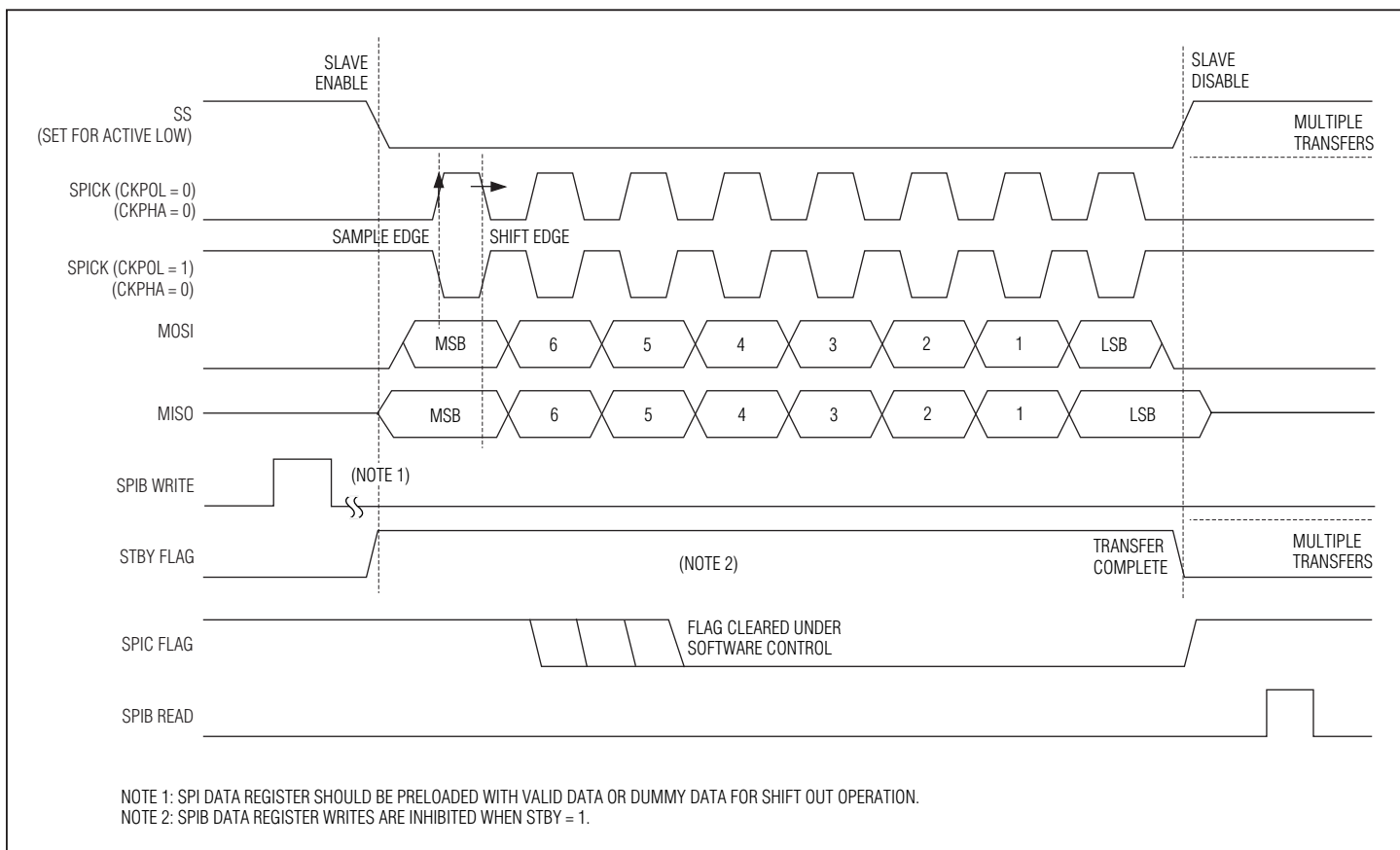


Figure 9-4. Slave Mode Transfer Cycle Operation

The following steps should be taken to initialize and set up the SPI port for slave mode operation.

- 1) Clear all SPI port flags (SPIC, ROVR, WCOL, MODF).
- 2) Set the polarity of SS in the SPICF register.
- 3) Set the SPI clocking mode (CKPOL, CKPHA) and character length (CHR) to match that of the SPI bus master configuration and enable interrupts (if desired) in the SPICF.
- 4) Set SPI slave mode, which will configure the ports to MISO (port 1.5 as output), MOSI (port 1.4 as input), SS (port 1.7 as input), and SCLK (port 1.6 as input).
- 5) Enable the SPI port in the SPICN.
- 6) Write data, if available, into the SPIB in preparation for the first transfer cycle.

Once the port is configured as an SPI bus slave, either polling or interrupts can be used to monitor the progress of the transfer cycle. The STBY flag is set once a transfer cycle is initiated from the SPI master to the slave device. Once the transfer cycle is complete, the SPIC flag is set and data should be read from the SPIB.

- 1) Poll SPIC or use interrupts to monitor the status of the transfer cycle.
- 2) Check to see the cause of the interrupt if interrupts used.
- 3) Read the contents of the SPIB register and take the appropriate action to:
 - a. Interpret the data as an instruction or data.
 - b. Take the appropriate actions as deemed by the defined protocol used.
- 4) Clear the SPIC flag by doing a bit write to 0.
- 5) Write a new character to the SPIB in response to and in preparation for the next transfer cycle.

9.1.4 SPI Clocking Modes

The SPI bus is a full-duplex bus where data is simultaneously transmitted and received synchronous to the SPICK on the MISO and MOSI data pins. The CKPHA and CKPOL bits control the SPICK data transfer mode when data is shifted and sampled on the serial data bus. Figure 9-5 illustrates the four selectable SPI transfer modes. When CKPOL is cleared (0), the rising edge of SPICK is the sampling edge; when CKPOL is set (1), the falling edge of SPICK is the active edge.

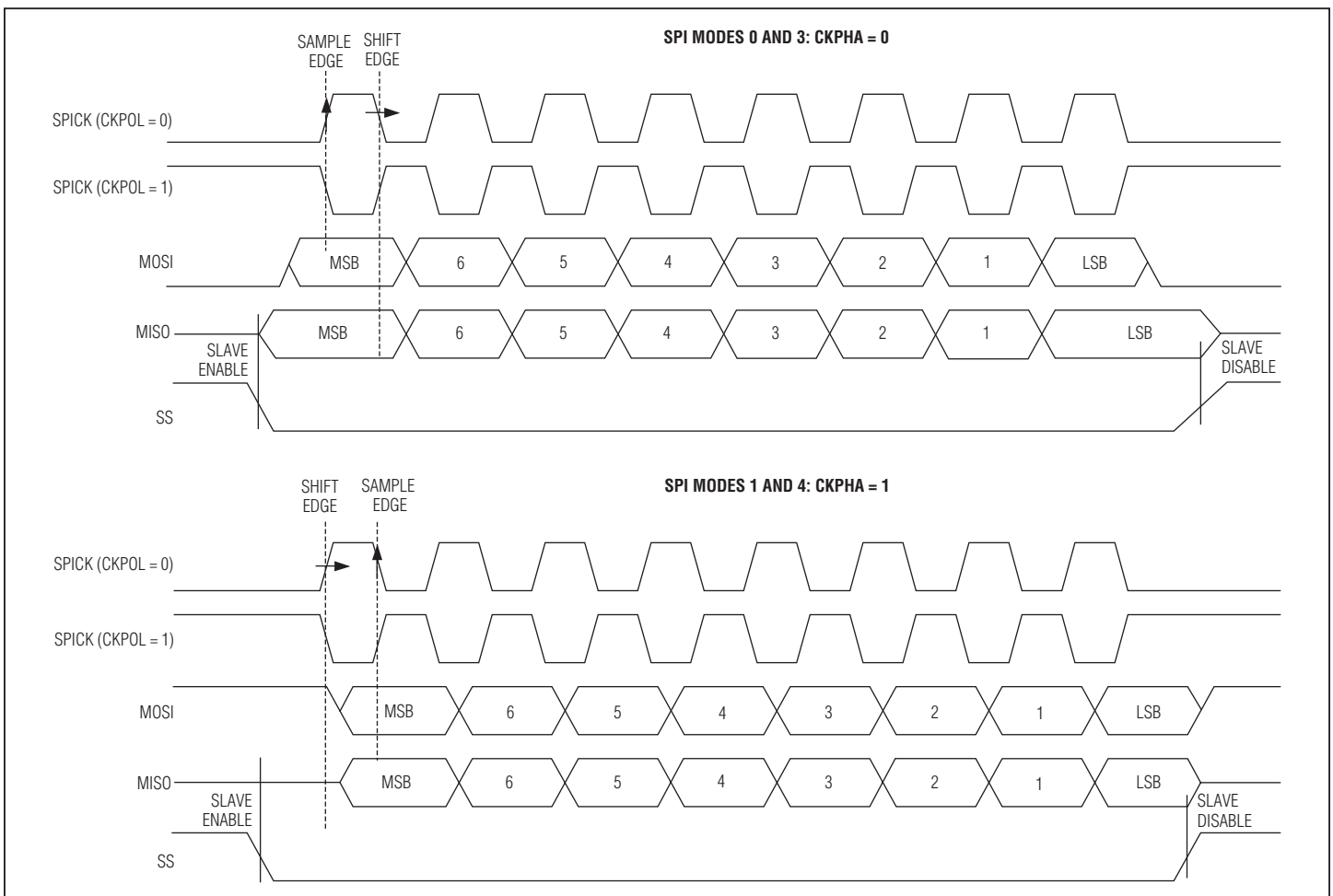


Figure 9-5. SPI Clocking Modes

When CKPHA is cleared (0), SS must be cycled to mark the beginning of each transfer cycle.

When CKPHA is set (1), SS can remain low between successive transfer cycles.

The MAXQ7667's flexible clocking schemes allow peripheral devices with different transfer formats to communicate with each other, however, the clock polarity and clock phase must be consistent for the master and selected slave device.

9.1.5 Baud-Rate Determination

The SPICK module supports a programmable shift clock rate in master mode. The master mode clock rate can be determined by the value loaded into the SPICK register. The SPICK rate is determined by the following:

$$\text{SPI master baud rate} = (0.5 \times \text{SYSCLK}) / (\text{CKR}[7:0]) + 1$$

where

SYSCLK is the frequency of the clock module. See Section 15 for more information.

CKR[7:0] are the 8-bit value loaded into the SPICK register.

In master mode, the SPICK rate cannot exceed one-half the MAXQ7667 SYSCLK.

In slave mode, the SPICK rate cannot exceed one-eighth the MAXQ7667 SYSCLK.

Writing to the SPIB during a master mode transfer cycle should not be attempted, as the cycle in progress will be corrupted and data will be unreliable.

9.1.6 Data Format

The character length select bit (CHR) in the SPICF specifies the data length transferred to either an 8-bit or 16-bit character for a transfer cycle. The SPI port state machine marks the end of a transfer cycle when the number of bits shifted in and out of the SPI shift register equals the CHR mode. Data is always shifted out MSB first and LSB last. When the CHR bit is set (1), the SPI port is configured for 16-bit transfer cycles, while when the CHR bit is cleared (0), the SPI port is configured for 8-bit transfer cycles. When set for 8-bit transfer cycles, the lower byte of the read buffer contains the data, with bit 7 being the MSB of the SPI port data.

9.2 SPI Port Errors

The SPI module detects three types of SPI system errors. Each flag reports an error condition on the SPI port that causes data to be lost or corrupted.

- Read Overrun Flag (ROVR)
- Write Collision Flag (WCOL)
- Mode Fault Flag (MODF)

9.2.1 Receive Overrun Flag (ROVR)

The receive path of the SPI port module is double buffered with the read buffer as the storage location for the first piece of data and the shift register as the storage location for the second piece of read data. An overrun condition exists when:

- The read buffer is unread (or full) AND
- The shift register is full from the previous transfer cycle AND
- A new transfer cycle begins with the first bit of the transfer cycle shifting into the SPI shift register

The ROVR flag is set (1) at this time. The data in the read buffer remains valid in this special condition while the data in the shift register is corrupted. Any error handling to correct this condition should:

- 1) Read the contents of the read buffer and handle it as valid data.
- 2) Clear the ROVR flag.

The assertion of the ROVR flag causes an interrupt if enabled by the ESPII bit. The ROVR bit is cleared by writing a 0 to it or by any reset condition.

9.2.2 Write Collision Flag (WCOL)

A write collision occurs if a write to the SPIB data register is attempted during a transfer cycle. The MAXQ7667's write path is a single-buffered path with the shift register acting as both the write buffer and shift register. If the STBY flag is set (1), this marks that a transfer cycle is in progress. Any writes to the SPIB with STBY set are blocked to maintain the integrity of the shift register contents. The transfer cycle in progress is completed normally. But the WCOL bit is set (1), marking an error condition. When a write collision is detected, the error-handling routine should clear the WCOL bit. The WCOL flag is set by the SPI port module but must be cleared by writing a 0 to this bit or by any reset condition.

9.2.3 Mode Fault Flag (MODF)

The MODF detects if the SS line is asserted on the MAXQ7667 when the SPI port is configured as an SPI bus master. In the SPI port protocol, the master device handles control of the SS line to enable target slave devices for SPI transfers. When more than one SPI master is on the bus this bus clash condition can occur. To enable mode-fault detection, the mode fault enable (MODFE) must be set (1). When the SS pin is asserted (0) while the SPI port is in master mode, the MODF flag is set and the following occurs:

- 1) The master MSTM bit is cleared (0) to reconfigure the device as a slave.
- 2) The SPIEN bit is cleared (0), disabling the SPI port.
- 3) The mode fault flag (MODF) is set (1).

The MODF flag is set by the SPI port module but must be cleared by writing a 0 to this bit or by any reset condition.

9.3 SPI Interrupts

The SPI port has four flags that generate interrupts to the μC :

- SPI Transfer Complete Flag (SPIC)
- Receive Overrun Flag (ROVR)
- Write Collision Flag (WCOL)
- Mode Fault Flag (MODF)

All the flags can be enabled to generate an interrupt by using the enable SPI interrupts (ESPII) bit (Figure 9-6). The MODF has an additional enable called the mode-fault-enable bit that must be set (1) to enable mode-fault detection.

All status flags must be cleared in the interrupt service routine by writing a 0 to them, or on any form of reset. Clearing the ESPII bit disables the interrupts but does not clear the underlying error flag condition or the status flag.

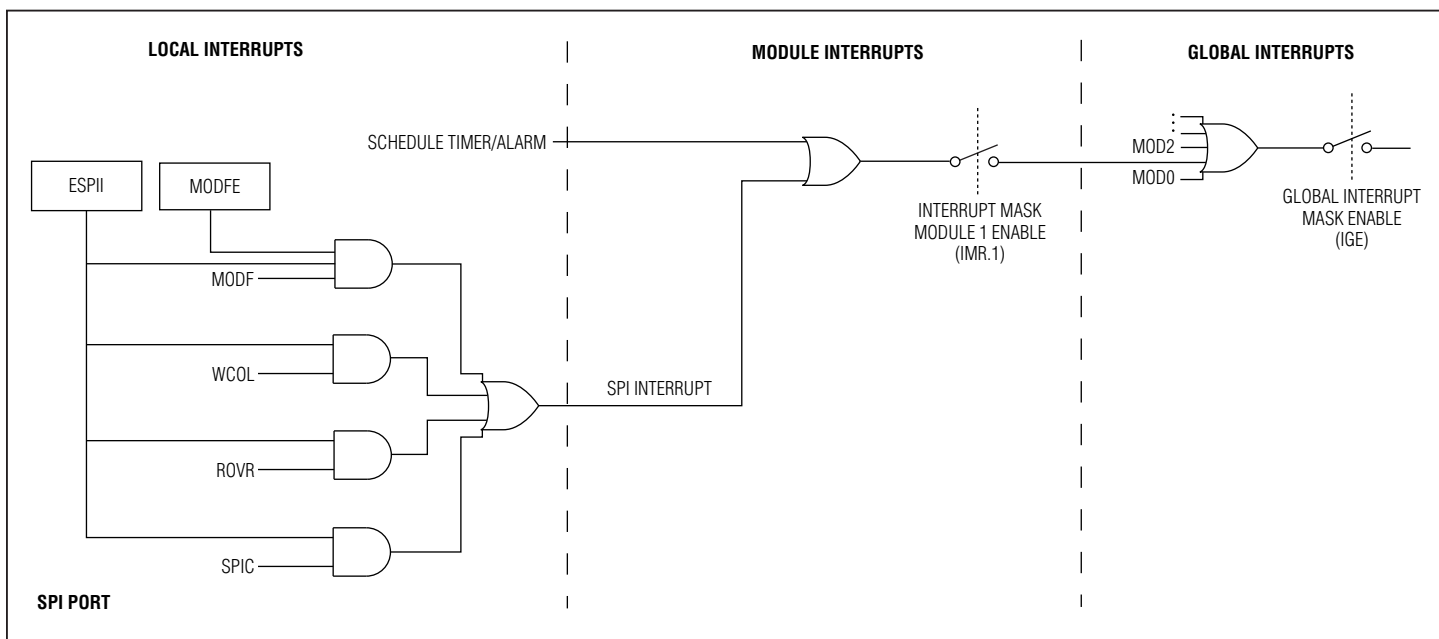


Figure 9-6. MAXQ Interrupt Enabling Scheme

9.4 Resetting the SPI Port

Any system reset completely resets the SPI. Partial resets occur whenever the SPI-enable bit (SPIE) is cleared. Whenever SPIE is cleared, the following occurs:

- Any transmission currently in progress is aborted.
- The shift register is cleared.
- The SPI state counter is cleared, making it ready for a new transmission.
- All the SPI port logic is defaulted back to being general-purpose I/O pins.

These items are reset only by a system reset:

- All control bits in the SPICN register.
- All control bits in the SPICF register.
- The status flags SPIC, ROVRF, WCOL, and MODF.

A mode-fault condition (MODF = 1) also resets flags and mode settings in the SPI port. The following bits are reset by a mode-fault condition:

- 1) The master MSTM bit is cleared (0) to reconfigure the device as a slave.
- 2) The SPIEN bit is cleared (0), disabling the SPI port.
- 3) MODF is set (1).

9.5 Oscillator/Clock Power-Saving Management Modes

Once the complete source and mode configuration of the μC is complete, the program can continue to configure all the modules (ADC, timers, DIO, interrupt, etc.). Once the μC is completely configured and operating, the low-power modes can then be enacted to reduce power consumption. The MAXQ power-saving modes are designed to place the μC in a stop mode to reduce power until a task must be completed. There are several methods for leaving the power management mode and returning to normal operation, including:

- External interrupt from an external pin configured to interrupt the μC
- Detection of SPI bus activity
- An interval timer interrupt

These wake-up events allow the μC to return to normal operation. Once the task is complete, the μC can then be put back into stop mode until the next task need be completed.

9.5.1 Stop Mode

The MAXQ stop mode is the lowest power mode. When stop (sleep) mode is used, the clock source is gated off from the μC . The external clock ports are shut down and only the internal RC oscillator continues to run, to monitor an event that would wake the μC . The following events wake the μC from stop mode and resume normal operation.

- External interrupt from an external pin is configured to interrupt the μC
- Detection of SPI bus activity
- An interval timer interrupt

SECTION 10: HARDWARE MULTIPLIER MODULE

This section contains the following information:

10.1 Hardware Multiplier Organization	10-2
10.2 Hardware Multiplier Peripheral Registers	10-3
10.2.1 Hardware Multiplier Control Register (MCNT)	10-3
10.2.2 Multiplier Operand A Register (MA)	10-4
10.2.3 Multiplier Operand B Register (MB)	10-5
10.2.4 Multiplier Accumulator 2 Register (MC2)	10-5
10.2.5 Multiplier Accumulator 1 Register (MC1)	10-6
10.2.6 Multiplier Accumulator 0 Register (MC0)	10-6
10.2.7 Multiplier Read Register 1 (MC1R)	10-7
10.2.8 Multiplier Read Register 0 (MC0R)	10-7
10.3 Hardware Multiplier Controls	10-8
10.4 Register Output Selection	10-8
10.5 Signed-Unsigned Operand Selection	10-8
10.6 Operand Count Selection	10-8
10.7 Hardware Multiplier Operations	10-8
10.8 Accessing the Multiplier	10-9
10.9 MAXQ7667 Hardware Multiplier Examples	10-10

LIST OF FIGURES

Figure 10-1. MAXQ7667 Multiplier Organization	10-2
---	------

LIST OF TABLES

Table 10-1. MAXQ7667 Hardware Multiplier Operations	10-9
---	------

SECTION 10: HARDWARE MULTIPLIER MODULE

The MAXQ7667 microcontroller includes a hardware multiplier module to support high-speed multiplications. The hardware multiplier module is equipped with two 16-bit operand registers, a 32-bit read-only result register, and an accumulator of 48-bit width. The multiplier can complete a 16-bit x 16-bit multiply-and-accumulate/subtract operation in a single cycle. The hardware multiplier module supports the following operations without interfering with the normal core functions:

- Signed or Unsigned Multiply (16 bit x 16 bit)
- Signed or Unsigned Multiply-Accumulate (16 bit x 16 bit)
- Signed or Unsigned Multiply-Subtract (16 bit x 16 bit)
- Signed Multiply and Negate (16 bit x 16 bit)

10.1 Hardware Multiplier Organization

The hardware multiplier consists of two 16-bit, parallel-load operand registers (MA, MB); a read-only result register formed by two parallel 16-bit registers (MC1R and MC0R); an accumulator, which is formed by three 16-bit parallel registers (MC2, MC1, and MC0); and a status/control register (MCNT). Figure 10-1 shows a block diagram of the hardware multiplier.

The main arithmetic unit is the 16-bit x 16-bit multiplier that processes operands feeding from the MA and MB registers and generates a 32-bit final product. The multiplier unit includes an adder that can be used to perform a final accumulate/subtract operation of the multiplier output with the MC[2:0] registers. The MCNT register must be configured to select the desired operation and operand count prior to loading the operand(s) to trigger the multiplier operation.

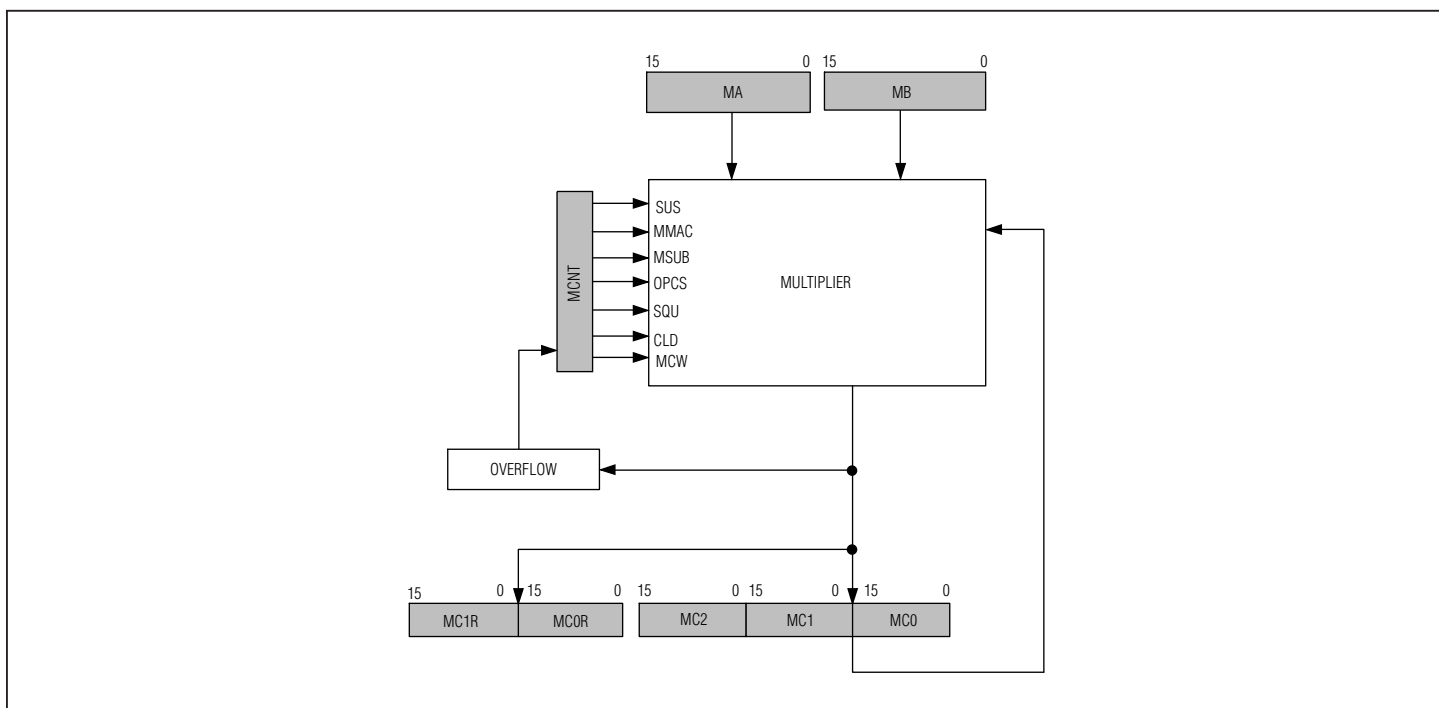


Figure 10-1. MAXQ7667 Multiplier Organization

10.2 Hardware Multiplier Peripheral Registers

10.2.1 Hardware Multiplier Control Register (MCNT)

Register Description: **Hardware Multiplier Control Register**
 Register Name: **MCNT**
 Register Address: **Module 01h, Index 00h**

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	OF	MCW	CLD	SQU	OPCS	MSUB	MMAC	SUS
Reset	0	0	0	0	0	0	0	0
Access	r	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0, write ignored.

Bit 7: Overflow Flag (OF). This bit is set to logic 1 when an overflow occurred for the last operation. This bit can be set for accumulation/subtraction operations or unsigned multiply-negate attempts. This bit is automatically cleared to 0 following a reset, starting a multiplier operation, or setting of the CLD bit to 0.

Bit 6: MC Register Write Select (MCW). The state of the MCW bit determines if an operation result will be placed into the accumulator registers (MC).

0 = The result is written to the MC registers.

1 = The result is not written to the MC registers (MC register content is unchanged).

Bit 5: Clear Data Registers (CLD). This bit initializes the operand registers and the accumulator of the multiplier. When it is set to 1, the contents of all data registers and the OF bit are cleared to 0 and the operand load counter is reset immediately. This bit is cleared by hardware automatically. Writing a 0 to this bit has no effect.

Bit 4: Square-Function Enable (SQU). This bit supports the hardware square function. When this bit is set to logic 1, a square operation is initiated after an operand is written to either the MA or the MB register. Writing data to either of the operand registers writes to both registers and triggers the specified square or square-accumulate/subtract operation. Setting this bit to 1 also overrides the OPCS bit setting. When SQU is cleared to logic 0, the hardware square function is disabled.

0 = Square function disabled.

1 = Square function enabled.

Bit 3: Operand Count Select (OPCS). This bit defines how many operands must be loaded to trigger a multiply or multiply-accumulate/subtract operation (except when SQU = 1 since this implicitly specifies a single operand). When this bit is cleared to logic 0, both operands (MA and MB) must be written to trigger the operation. When this bit is set to 1, the specified operation is triggered once either operand is written.

- 0 = Both operands (MA and MB) must be written to trigger the multiplier operation.
- 1 = Loading one operand (MA or MB) triggers the multiplier operation.

Bit 2: Multiply Negate (MSUB). Configuring this bit to logic 1 enables negation of the product for signed multiply operations and subtraction of the product from the accumulator (MC[2:0]) when MMAC = 1. When MSUB is configured to logic 0, the product of multiply operations will not be negated and accumulation is selected when MMAC = 1.

Bit 1: Multiply-Accumulate Enable (MMAC). This bit enables the accumulate or subtract operation (as per MSUB) for the hardware multiplier. When this bit is cleared to logic 0, the multiplier will perform only multiply operations. When this bit is set to logic 1, the multiplier will perform a multiply-accumulate or multiply-subtract operation based upon the MSUB bit.

- 0 = Accumulate/subtract operation disabled.
- 1 = Accumulate/subtract operation enabled.

Bit 0: Signed-Unsigned Select (SUS). This bit determines the data type of the operands. When this bit is cleared to logic 0, the operands are treated as two's complement values and the multiplier performs a signed operation. When this bit is set to logic 1, the operands are treated as absolute magnitudes and the multiplier performs an unsigned operation.

- 0 = Signed operands.
- 1 = Unsigned operands.

10.2.2 Multiplier Operand A Register (MA)

Register Description: **Multiplier Operand A Register**
 Register Name: **MA**
 Register Address: **Module 01h, Index 01h**

Bit #	15	14	13	12	11	10	9	8
Name	MA15	MA14	MA13	MA12	MA11	MA10	MA9	MA8
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	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Multiplier Operand A Register Bits 15:0 (MA[15:0]). This operand A register is used by the application code to load 16-bit values for multiplier operations.

10.2.3 Multiplier Operand B Register (MB)

Register Description: **Multiplier Operand B Register**
 Register Name: **MB**
 Register Address: **Module 01h, Index 02h**

Bit #	15	14	13	12	11	10	9	8
Name	MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8
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	MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Multiplier Operand B Register Bits 15:0 (MB[15:0]). This operand B register is used by the application code to load 16-bit values for multiplier operations.

10.2.4 Multiplier Accumulator 2 Register (MC2)

Register Description: **Multiplier Accumulator 2 Register**
 Register Name: **MC2**
 Register Address: **Module 01h, Index 03h**

Bit #	15	14	13	12	11	10	9	8
Name	MC215	MC214	MC213	MC212	MC211	MC210	MC29	MC28
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	MC27	MC26	MC25	MC24	MC23	MC22	MC21	MC20
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Multiplier Accumulator 2 Register Bits 15:0 (MC2[15:0]). The MC2 register represents the two most significant bytes of the accumulator register. The 48-bit accumulator is formed by MC2, MC1, and MC0. For a signed operation, the most significant bit of this register is the sign bit.

10.2.5 Multiplier Accumulator 1 Register (MC1)

Register Description: **Multiplier Accumulator 1 Register**
 Register Name: **MC1**
 Register Address: **Module 01h, Index 04h**

Bit #	15	14	13	12	11	10	9	8
Name	MC115	MC114	MC113	MC112	MC111	MC110	MC109	MC108
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	MC107	MC106	MC105	MC104	MC103	MC102	MC101	MC100
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Multiplier Accumulator 1 Register Bits 15:0 (MC1[15:0]). The MC1 register represents bytes 3 and 2 of the accumulator register. The 48-bit accumulator is formed by MC2, MC1, and MC0.

10.2.6 Multiplier Accumulator 0 Register (MC0)

Register Description: **Multiplier Accumulator 0 Register**
 Register Name: **MC0**
 Register Address: **Module 01h, Index 05h**

Bit #	15	14	13	12	11	10	9	8
Name	MC015	MC014	MC013	MC012	MC011	MC010	MC009	MC008
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	MC007	MC006	MC005	MC004	MC003	MC002	MC001	MC000
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Multiplier Accumulator 0 Register Bits 15:0 (MC0[15:0]). The MC0 register represents the two least significant bytes of the accumulator register. The 48-bit accumulator is formed by MC2, MC1, and MC0.

10.2.7 Multiplier Read Register 1 (MC1R)

Register Description: **Multiplier Read Register 1**
 Register Name: **MC1R**
 Register Address: **Module 01h, Index 0Ch**

Bit #	15	14	13	12	11	10	9	8
Name	MC1R15	MC1R14	MC1R13	MC1R12	MC1R11	MC1R10	MC1R9	MC1R8
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	MC1R7	MC1R6	MC1R5	MC1R4	MC1R3	MC1R2	MC1R1	MC1R0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Multiplier Read Register 1 Bits 15:0 (MC1R[15:0]). The MC1R register represents bytes' 3 and 2 result from the last operation when MCW = 1 or the last operation was a multiply or multiply-negate. When MCW = 0 and the last operation was a multiply-accumulate/subtract, the contents of this register may or may not agree with the contents of MC1 due to the combinatorial nature of the adder. The contents of this register may change if MCNT, MA, MB, or MC2:MC0 is changed.

10.2.8 Multiplier Read Register 0 (MC0R)

Register Description: **Multiplier Read Register 0**
 Register Name: **MC0R**
 Register Address: **Module 01h, Index 0Dh**

Bit #	15	14	13	12	11	10	9	8
Name	MC0R15	MC0R14	MC0R13	MC0R12	MC0R11	MC0R10	MC0R9	MC0R8
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	MC0R7	MC0R6	MC0R5	MC0R4	MC0R3	MC0R2	MC0R1	MC0R0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: This register is cleared to 0000h on all forms of reset.

Bits 15 to 0: Multiplier Read Register 0 Bits 15:0 (MC0R[15:0]). The MC0R register represents bytes' 1 and 0 result from the last operation when MCW = 1 or the last operation was a multiply or multiply-negate. When MCW = 0 and the last operation was a multiply-accumulate/subtract, the contents of this register may or may not agree with the contents of MC0 due to the combinatorial nature of the adder. The contents of this register may change if MCNT, MA, MB, or MC[2:0] is changed.

10.3 Hardware Multiplier Controls

The selection of operation to be performed by the multiplier is determined by four control bits in the MCNT register: SUS, MSUB, MMAC, and SQU. The number of operands that must be loaded to trigger the specified operation is dictated by the OPCS bit setting, except when the square function is enabled (SQU = 1). Enabling the square function implicitly defines that only a single operand (either MA or MB) needs to be loaded to trigger the square operation, independent of the OPCS bit setting. The MCNT register bits must be configured to select the desired operation and operand count prior to loading the operand(s) to trigger the multiplier operation. Any write to MCNT automatically resets the operand load counter of the multiplier, but does not affect the operand registers, unless such action is requested using the clear data registers (CLD) control bit. Once the desired operation has been specified via the MCNT register bits, loading the prescribed number of operands triggers the respective multiply, multiply-accumulate/subtract, or multiply-negate operation.

10.4 Register Output Selection

The hardware multiplier implements the MC register write select (MCW) control bit so that writing of the result to the MC[2:0] registers can be blocked to preserve the MC registers (accumulator). When the MCW bit is configured to logic 1, the result for the given operation is not written to the MC registers. When the MCW bit is configured to logic 0, the MC registers are updated with the result of the operation. The MC1R, MC0R read-only register pair is updated independent of the MCW bit setting. This register pair always reflects the output that would normally be placed in MC[1:0], given that MCW = 1 or MMAC = 0. When MCW = 0 and MMAC = 1, the MC1R:MC0R content may not match the MC[1:0] register content, but it will be predictable and may be useful in certain situations. See Table 10-1 for details.

10.5 Signed-Unsigned Operand Selection

The operands can be either signed or unsigned numbers, but the data type must be defined by the user software via the signed-unsigned (SUS) bit prior to triggering the operation. For an unsigned operation, the SUS bit in the MCNT register must be set to 1; for a signed operation, the SUS bit must be cleared to 0. The multiplier treats unsigned numbers as absolute magnitude. For a 16-bit positional binary number, this represents a value in the range 0 to $2^{16} - 1$ (xFFFFh). The signed number representation is a two's complement value, where the most significant bit is defined as a sign bit. The range of a 16-bit two's complement number is $-2^{(16-1)}$ (x8000h) to $+2^{(16-1)} - 1$ (x7FFFh). The product of any signed operation will be sign extended before being stored or accumulated/subtracted into the MC registers. The SUS bit should always be configured to logic 0 (i.e., signed operands) for the multiply-negate operation. Attempting an unsigned multiply-negate operation results in incorrect results and setting of the OF bit. Modifying the operand data type selection via the SUS bit does not alter the contents of the MC registers. The MC registers are read/write accessible and can be modified by user code when necessary.

10.6 Operand Count Selection

The OPCS bit allows selection of single operand or two operands operation for the multiply and multiply-accumulate/subtract operations. When the OPCS bit is cleared to 0, the multiply or multiply-accumulate/subtract operation established by the SUS, MSUB, and MMAC bits is triggered once two operands are loaded, one to each of the MA and MB registers. When OPCS is set to 1, the operation commences once data is loaded to either MA or MB. The OPCS bit is ignored when the square operation is enabled (SQU), since loading of data to the MA or MB register actually writes to both registers.

10.7 Hardware Multiplier Operations

The control bits, which specify data type (SUS), operand count (OPCS or SQU), and destination control (MCW), have already been described. However, there are two additional MCNT register bits that serve to define the hardware multiplier operation. The multiply-accumulate/subtract and multiply-negate operations are enabled by the multiply-accumulate enable (MMAC) and multiply negate (MSUB) bits in the MCNT register. When the MMAC bit is set to 1, the multiplier performs a multiply-accumulate (if MSUB = 0) or a multiply-subtract (if MSUB = 1). If MMAC is configured to 0, the multiplier result is not accumulated or subtracted, but can be stored directly (if MSUB = 0) or negated (if MSUB = 1) before storage. The multiply-negate operation (MMAC = 0, MSUB = 1) is only allowable for signed data operands (SUS = 0). For unsigned multiply-accumulate/subtract operations, the OF bit is set when a carry-out/borrow-in from the most significant bit of the MC register occurs. For a signed two's complement multiply-accumulate/subtract operations, the OF bit is set when the carry-out/borrow-in from the most significant magnitude position of the MC register is different from the carry-out/borrow-in of the sign position of the MC register. Since there is no overflow condition for multiply and multiply-negate operations, the OF bit is always cleared for these operations with one exception. The OF bit will be set to logic 1 if an unsigned multiply-negate (invalid operation) is requested. Table 10-1 shows the operations supported by the multiplier and associated MCNT control bit settings.

10.8 Accessing the Multiplier

There are no restrictions on how quickly data is entered into the operand registers or on the order of data entry. The only requirement to do a calculation is to perform the loading of MA and/or MB registers having specified data type and operation in the MCNT register. The multiplier keeps track of the writes to the MA and MB registers, and starts calculation immediately after the prescribed number of operands is loaded. If two operands are specified for the operation, the multiplier waits for the second operand to be loaded into the other operand register before starting the actual calculation. If for any reason software needs to reload the first operand, it should either reload that same operand register or use the CLD bit in the MCNT register to reinitialize the multiplier; otherwise, loading data to another operand register triggers the calculation. The CLD bit is a self-clearing bit that can be used for multiplier initialization. When it is set, it clears all data registers and the OF bit to zero and resets the multiplier operand write counter.

The specified hardware multiplier operation begins when the final operand(s) is loaded and will complete in a single cycle. The read-only MC1R, MC0R result registers can be accessed in the very next cycle unless accumulation/subtraction with MC[2:0] is requested (MCW = 0 and MMAC = 1), in which case, one cycle is required so that stable data can be read. When MCW = 0, the MC[2:0] registers always require one wait cycle before the operation result is accessible. The single wait cycle needed for updating the MC[2:0] registers with a calculated result does not prevent initiating another calculation. Back-to-back operations can be triggered (independent of data type and operand count) without the need of wait state between loading of operands.

Table 10-1. MAXQ7667 Hardware Multiplier Operations

MCW:MSUB: MMAC	OPERATION	MC2	MC1	MC0	MC1R:MC0R	OF STATUS
000	Multiply	MA x MB			MA x MB	No
001	Multiply-Accumulate	MC + (MA x MB)			32 LSb of [MC + 2 x (MA x MB)]	Yes
010	Multiply-Negate (SUS = 0 Only)	-(MA x MB)			-(MA x MB)	No
011	Multiply-Subtract	MC - (MA x MB)			32 LSb of [MC - 2 x (MA x MB)]	Yes
100	Multiply	MC2	MC1	MC0	MA x MB	No
101	Multiply-Accumulate	MC2	MC1	MC0	32 LSb of [MC + (MA x MB)]	No
110	Multiply-Negate (SUS = 0 Only)	MC2	MC1	MC0	-(MA x MB)	No
111	Multiply-Subtract	MC2	MC1	MC0	32 LSb of [MC - (MA x MB)]	No

10.9 MAXQ7667 Hardware Multiplier Examples

The following are code examples of multiplier operations.

```
;Unsigned Multiply 16-bit x 16-bit
move  MCNT, #21h           ; CLD=1, SUS=1 (unsigned)
move  MA, #0FFFh          ; MC2:0=0000_0000_0000h
move  MB, #1001h         ; MC1R:MC0R= 00FF_FFFFh
                           ; MC2:0=0000_00FF_FFFFh

;Signed Multiply 16-bit x 16-bit
move  MCNT, #20h          ; CLD=1, SUS=0 (signed)
move  MA, #F001h         ; MC2:0=0000_0000_0000h
move  MB, #1001h         ; MC1R:MC0R= FF00_0001h
                           ; MC2:0=FFFF_FF00_0001h

;Unsigned Multiply-Accumulate 16-bit x 16-bit
                           ; MC2:0=0000_0100_0001h
move  MCNT, #03h         ; MMAC=1, SUS=1 (unsigned)
move  MA, #0FFFh         ;
move  MB, #1001h         ;
                           ; MC1R:MC0R=02FF_FFFFh
                           ; MC2:0=0000_0200_0000h

;Signed Multiply-Accumulate 16-bit x 16-bit
                           ; MC2:0=0000_0100_0001h
move  MCNT, #02h         ; SUS=0 (signed)
move  MA, #F001h         ;
move  MB, #1001h         ;
                           ; MC1R:MC0R= FF00_0003h
                           ; MC2:0=0000_0000_0002h

;Unsigned Multiply-Subtract 16-bit x 16-bit
                           ; MC2:0=0000_0100_0001h
move  MCNT, #07h         ; MMAC=1, MSUB=1, SUS=1 (unsigned)
move  MA, #0FFFh         ;
move  MB, #1001h         ;
                           ; MC1R:MC0R=FF00_0003h
                           ; MC2:0=0000_0000_0002h

;Signed Multiply-Subtract 16-bit x 16-bit
                           ; MC2:0=0000_0100_0001h
move  MCNT, #06h         ; MMAC=1, MSUB=1, SUS=0 (signed)
move  MA, #F001h         ;
move  MB, #1001h         ;
                           ; MC1R:MC0R= 02FF_FFFFh
                           ; MC2:0=0000_0200_0000h

;Signed Multiply Negate 16-bit x 16-bit
move  MCNT, #24h         ; CLD=1, MSUB=1, SUS=0 (signed)
move  MA, #F001h         ; MC2:0=0000_0000_0000h
move  MB, #1001h         ; MC1R:MC0R =00FF_FFFFh
                           ; MC2:0=0000_00FF_FFFFh
```

SECTION 11: TEST ACCESS PORT (TAP)

This section contains the following information:

11.1 TAP Overview	11-2
11.2 Architecture	11-2
11.2.1 TAP Pins	11-3
11.3 TAP Interface Control	11-4
11.3.1 System Control Register (SC)	11-4
11.4 TAP Controller Operation	11-5
11.4.1 Communication via TAP	11-6
11.4.2 Test-Logic-Reset	11-6
11.4.3 Run-Test-Idle	11-6
11.4.4 IR-Scan Sequence	11-6
11.4.5 DR-Scan Sequence	11-7
11.4.6 TAP Communication Examples—IR-Scans and DR-Scans	11-8

LIST OF FIGURES

Figure 11-1. Simplified MAXQ7667 TAP and TAP Controller	11-3
Figure 11-2. TAP Controller State Diagram	11-5
Figure 11-3. TAP Controller Debug Mode—IR-Scan Example	11-8
Figure 11-4. TAP Controller Debug Mode—DR-Scan Example	11-9

LIST OF TABLES

Table 11-1. MAXQ7667 TAP Pins	11-3
Table 11-2. Instruction Register Content vs. TAP Controller State	11-6
Table 11-3. Instruction Register Commands	11-7

NOTE: BOUNDARY SCAN IS NOT AVAILABLE IN THE MAXQ7667.

SECTION 11: TEST ACCESS PORT (TAP)

Note: This section is only relevant to those users who plan to make their own debugging and programming tools through JTAG; most users can skip this section. Refer to *Section 12* and *Section 13* for JTAG in-circuit debug mode and in-system programming/bootloader.

11.1 TAP Overview

The MAXQ7667 incorporates a test access port (TAP) and TAP controller for communication with a host device across a 4-wire synchronous serial interface. The MAXQ7667 uses the TAP to support in-system flash programming, in-circuit debug, and device test functions. The MAXQ7667 TAP features include the following:

- 4-wire synchronous communication
- TAP signals compatible with JTAG IEEE Standard 1149.1
- Maximum TAP clock frequency limited to 1/8 the system clock

For detailed information on the TAP and TAP controller, refer to IEEE STD 1149.1 "*IEEE Standard Test Access Port and Boundary-Scan Architecture.*"

11.2 Architecture

The MAXQ7667 TAP controller is a synchronous state machine that responds to changes at the TMS and TCK signals. The TAP state control is achieved through host manipulation of the test mode select (TMS) and test clock (TCK) signals. Based on its state transition, the controller provides the clock and control sequence for TAP operation. The performance of the TAP is dependent on the TCK clock frequency. The maximum TCK clock frequency should be limited to 1/8 the system clock frequency. Figure 11-1 shows a simplified functional block diagram of the MAXQ7667 TAP and TAP controller.

The TAP provides an independent serial channel to communicate synchronously with the host system. The TMS signal is sampled at the rising edge of TCK and decoded by the TAP controller to control movement between the TAP states. The TDI input and TDO outputs are meaningful once the TAP is in a serial shift state.

The TAP controller block has four working registers that control the operation of the port.

- TAP Debug Register
- TAP System Programming Register
- TAP Instruction Register
- TAP Bypass Register

These registers are accessed through the TAP port only and control the sequencing of the TAP state machine. These registers are not accessible from the CPU.

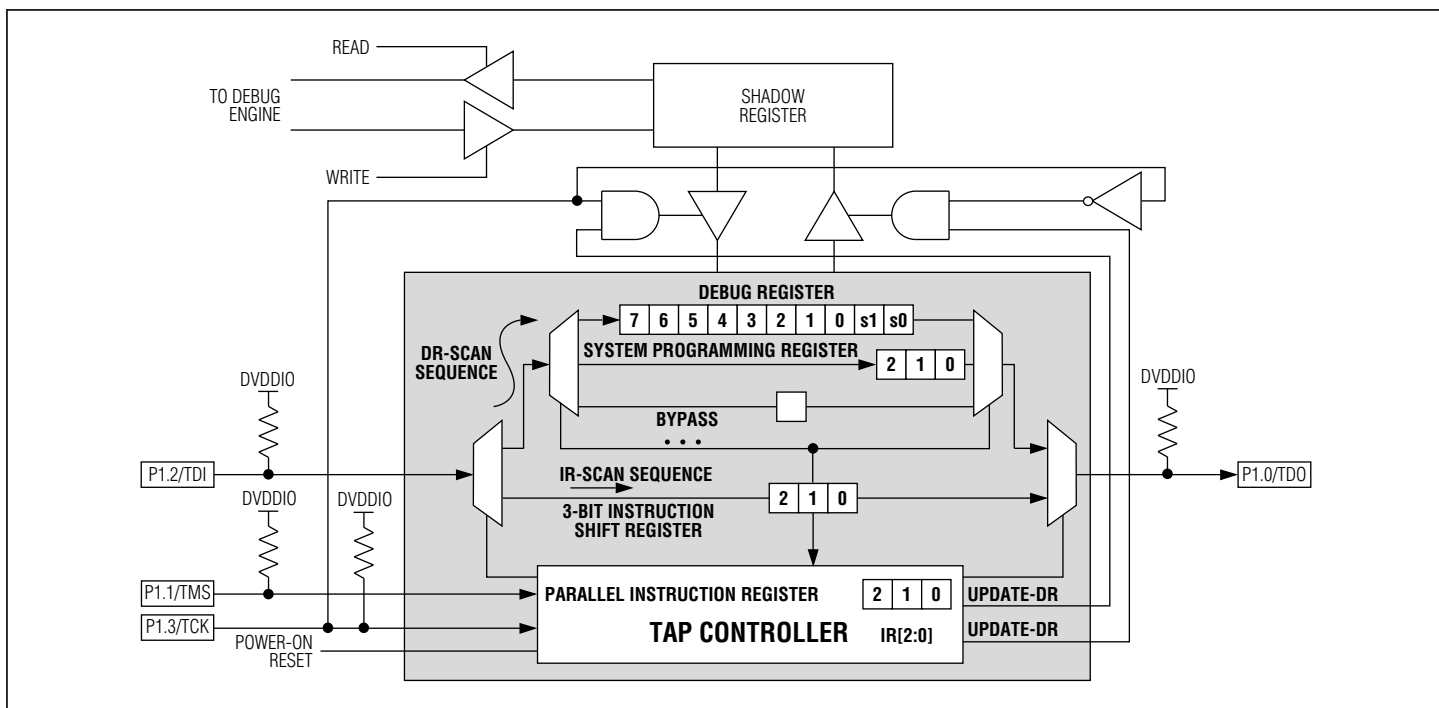


Figure 11-1. Simplified MAXQ7667 TAP and TAP Controller

11.2.1 TAP Pins

The TAP is formed by four interface signals as described in Table 11-1. The TAP signals are multiplexed with port pins P1.0, P1.1, P1.2, and P1.3. These pins default to their JTAG TAP function on reset, which means that the MAXQ7667 device will always be ready for in-circuit debugging or in-circuit programming 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 P1.0, P1.1, P1.2, and P1.3 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.

Table 11-1. MAXQ7667 TAP Pins

PIN	NAME	MULTIPLEXED WITH PORT SIGNAL	FUNCTION
48			
46	TDO	P1.0	JTAG Serial Test Data Output. This signal is used to serially transfer internal data to the external host. Data is transferred least significant bit first. Data is driven out only on the falling edge of TCK, only during TAP Shift-IR or Shift-DR states and is otherwise inactive. This pin is weakly pulled high internally when inactive and/or when SC.7 (TAP) = 1. After power-up or a reset this pin defaults to JTAG TDO pin.
47	TMS	P1.1	JTAG Test Mode Select Input. This signal is sampled at the rising edge of TCK and controls movement between TAP states. TMS is weakly pulled high internally when TAP = 1. After power-up or a reset this pin defaults to JTAG TMS pin.
48	TDI	P1.2	JTAG Serial Test Data Input. This signal is used to receive data serially transferred by the host. Data is received least significant bit first and is sampled on the rising edge of TCK. TDI is weakly pulled high internally when TAP = 1. After power-up or a reset this pin defaults to JTAG TDI pin.
1	TCK	P1.3	JTAG Serial Test Clock Input. Provided by the host. When this signal is stopped at 0, storage elements in the TAP logic retain their data indefinitely. TCK is weakly pulled high internally when TAP = 1. After power-up or a reset this pin defaults to JTAG TCK pin.

11.3 TAP Interface Control

Once an application has been loaded and starts running, the MAXQ7667 JTAG TAP interface can be controlled by the TAP bit in the system control register as described in *Section 11.3.1*.

11.3.1 System Control Register (SC)

Register Description: **System Control Register**

Register Name: **SC**

Register Address: **Module 08h, Index 08h**

Bit #	7	6	5	4	3	2	1	0
Name	TAP	—	CDA1	CDA0	UPA	ROD	PWL	—
Reset	0	0	0	0	0	0	1	0
Access	rw	r	rw	rw	rw	rw	rw	r

r = read, w = write

**This register defaults to 80h on all forms of reset except after power-on reset. After power-on reset, the PWL bit is also set and this register defaults to 82h.*

Bit 7: Test Access (JTAG) Port Enable (TAP). This bit controls whether the TAP special function pins are enabled. The TAP defaults to being enabled.

0 = JTAG/TAP functions are disabled and P1.0–P1.3 can be used as general-purpose I/O pins

1 = TAP special function pins P1.0–P1.3 are enabled to act as JTAG inputs and outputs

Bits 6 and 0: Reserved.

Bits 5 and 4: Code Data Access Bits 1 and 0 (CDA[1:0]). See *Section 4* for more information on these bits.

Bit 3: Upper Program Access (UPA). See *Section 2* for more information on this bit.

Bit 2: ROM Operation Done (ROD). See *Section 12* for more information on this bit.

Bit 1: Password Lock (PWL). See *Section 12* for more information on this bit.

11.4 TAP Controller Operation

The MAXQ7667 TAP controller is formed by a finite state machine that provides 16 operational states for access control. The TAP state control is achieved through host manipulation of the TMS and TCK signals. The TMS signal is sampled at the rising edge of TCK and decoded by the TAP controller to control movement between the TAP states. The TDI input and TDO output are meaningful once the TAP is in a serial shift state. This section provides a brief description of the TAP controller state machine and its state transitions.

The state diagram in Figure 11-2 summarizes the transitions caused by the TMS signal sampling on the rising edge at TCK. The TMS signal value is shown adjacent to each state transition in the figure.

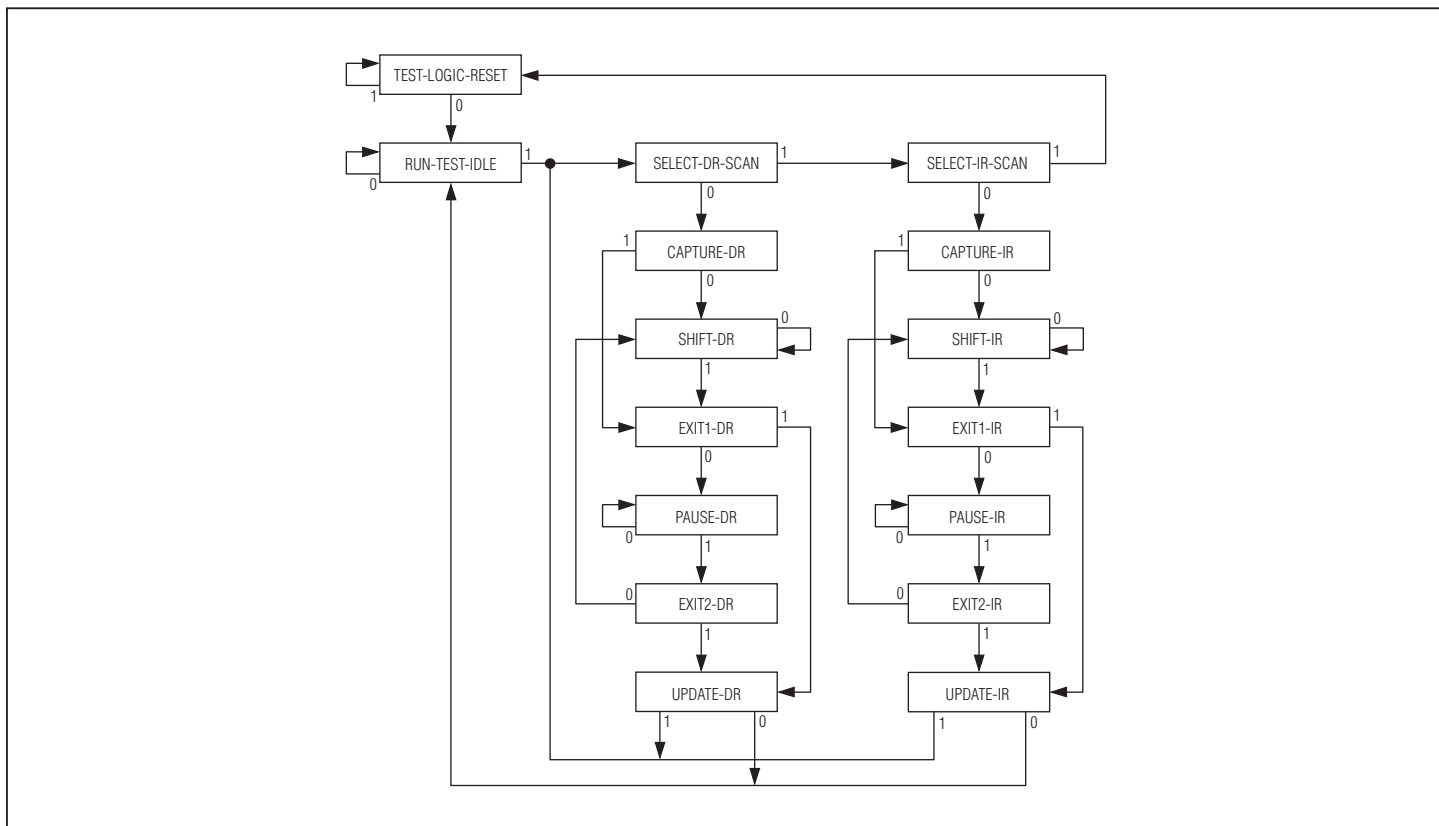


Figure 11-2. TAP Controller State Diagram

11.4.1 Communication via TAP

The TAP controller is in test-logic-reset state after a power-on-reset. During this initial state, the instruction register contains bypass instruction and the serial path defined between the TDI and TDO pins for the shift-DR state is the 1-bit bypass register. All TAP signals (TCK, TMS, TDI, and TDO) default to being weakly pulled high internally on any reset. The TAP controller remains in the test-logic-reset state as long as TMS is held high. The TCK and TMS signals can be manipulated by the host to transition to other TAP states. The TAP controller remains in a given state whenever TCK is held low.

For the host to establish a specific data communication link, a private instruction must be loaded into the IR[2:0] register. Once the instruction is latched in the instruction parallel buffer at the update-IR state, it is recognized by the TAP controller and the communication channel is established by putting the appropriate working register between TDI and TDO.

After a register has been placed between TDI and TDO, the in-circuit debug or in-system programming commands and data can be exchanged between the host and the MAXQ7667 by operating in the data register portion of the state sequence (i.e., DR-scan). The TAP retains the private instruction that was loaded into IR[2:0] until a new instruction is shifted in or until the TAP controller returns to the test-logic-reset state. Essentially, the IR-scan sequence selects the appropriate registers between TDI and TDO for debug, in-system programming, and bypass mode. Once the appropriate register is selected, the DR-scan sequence can be executed.

11.4.2 Test-Logic-Reset

On a power-on reset, the TAP controller is initialized to the test-logic-reset state and the instruction register (IR[2:0]) is initialized to the bypass instruction so that it does not affect normal system operation. No matter what the state of the controller, it enters test-logic-reset when TMS is held high for at least five rising edges of TCK. The controller remains in the test-logic-reset state if TMS remains high. An erroneous low signal on the TMS can cause the controller to move into the run-test-idle state, but no disturbance is caused to system operation if the TMS signal is returned and kept at the intended logic high for three rising edges of TCK since this returns the controller to the test-logic-reset state.

11.4.3 Run-Test-Idle

As illustrated in Figure 11-2, the run-test-idle state is an intermediate state for getting to one of the two state sequences in which the TAP controller performs meaningful operations:

- Controller state sequence (IR-scan)
- Data register state sequence (DR-scan)

11.4.4 IR-Scan Sequence

The MAXQ7667 supports a 3-bit TAP instruction register to allow certain device specific instructions (e.g., "Debug" or "System Programming") to be supported. The IR-Scan sequence allows instructions (e.g., "Debug" and "System Programming") to be shifted into the instruction register starting from the select-IR-scan state. In the TAP, the instruction register is connected between the TDI input and the TDO output. Inside the IR-scan sequence, the capture-IR state loads a fixed binary pattern (001b) into the 3-bit shift register and the shift-IR state causes shifting of TDI data into the shift register and serial output to TDO, least significant bit first. Once the desired instruction is in the shift register, the instruction can be latched into the parallel instruction register (IR[2:0]) on the falling edge of TCK in the update-IR state. The contents of the 3-bit instruction shift register and parallel instruction register (IR[2:0]) are summarized with respect to the TAP controller states in Table 11-2.

Table 11-2. Instruction Register Content vs. TAP Controller State

TAP CONTROLLER STATE	INSTRUCTION SHIFT REGISTER	PARALLEL (3-BIT) INSTRUCTION REGISTER (IR2:IR0)
Test-Logic-Reset	Undefined	Set to bypass (011b) instruction
Capture-IR	Load 001b at the rising edge of TCK	Retain last state
Shift-IR	Input data via TDI and shift towards TDO at the rising edge of TCK	Retain last state
Exit1-IR, Exit2-IR, Pause-IR	Retain last state	Retain last state
Update-IR	Retain last state	Load from shift register at the falling edge of TCK
All other states	Undefined	Retain last state

When the parallel instruction register (IR[2:0]) is updated, the TAP controller decodes the instruction and performs any necessary operations, including activation of the data shift register to be used for the particular instruction during data register shift sequences (DR-scan). The length of the activated shift register depends upon the value loaded to the instruction register (IR[2:0]). The supported instruction register encodings and associated data-register selections are shown in Table 11-3.

The extest (IR[2:0] = 000b) and sample/preload (IR[2:0] = 001b) instructions are mandated by the JTAG standard; however, the MAXQ7667 does not make use of these instructions. These instructions are treated as no operations and may be entered into the instruction register without affecting the on-chip system logic or pins and does not change the existing serial data register selection between TDI and TDO.

The bypass (IR[2:0] = 011b, 101b, or 111b) instruction is also mandated by the JTAG standard. The bypass instruction is fully implemented by the MAXQ7667 to provide a minimum length serial data path between the TDI and the TDO pins. This is accomplished by providing a single-cell bypass shift register. When the instruction register is updated with the bypass instruction, a single bypass register bit is connected serially between TDI and TDO in the shift-DR state. The instruction register automatically defaults to the bypass instruction when the TAP is in the test-logic-reset state. The bypass instruction has no effect on the operation of the on-chip system logic.

The debug (IR[2:0] = 010b) and system programming (IR[2:0] = 100b) instructions are private instructions that are intended solely for in-circuit debug and in-system programming operations, respectively. If the instruction register is updated with the debug instruction, a 10-bit serial shift register is formed between the TDI and TDO pins in the shift-DR state. If the system programming instruction is entered into the instruction register (IR[2:0]), a 3-bit serial data shift register is formed between the TDI and TDO pins in the shift-DR state.

Instruction register (IR[2:0]) settings other than those listed and previously described are reserved for internal use. As can be seen in Figure 11-1, the instruction register serves to select the length of the serial data register between TDI and TDO during the shift-DR state.

Table 11-3. Instruction Register Commands

IR[2:0]			INSTRUCTION	FUNCTION	SERIAL DATA SHIFT REGISTER SELECTION
0	0	0	Extest	No operation	Unchanged (retain previous selection)
0	0	1	Sample/Preload	No operation	Unchanged (retain previous selection)
0	1	0	Debug	In-circuit debug mode*	10-bit shift register
0	1	1	Bypass	No operation (default)	1-bit shift register
1	0	0	System Programming	Bootstrap function**	3-bit shift register
1	0	1	Bypass	No operation (default)	1-bit shift register
1	1	0	Reserved		
1	1	1	Bypass	No operation (default)	1-bit shift register

*Also supports bootloader functions in the MAXQ7667.

**Provides information about programming source and enables the MAXQ7667 for programming.

11.4.5 DR-Scan Sequence

Once the instruction register has been configured, using the IR-scan sequence, to a desired state (mode), transactions are performed via a data shift register associated with that mode (as shown in Table 11-3). These data transactions are executed serially in a manner analogous to the process used to load the instruction register. The transactions are grouped in the TAP controller state sequence starting from the select-DR-scan state. In the TAP controller state sequence, the shift-DR state allows internal data to be shifted out through the TDO pin while the external data is shifted in simultaneously via the TDI pin. Once a complete data pattern is shifted in, input data can be latched into the parallel buffer of the selected register on the falling edge of TCK in the update-DR state. On the same TCK falling edge, in the update-DR state, the internal parallel buffer is loaded to the data shift register for output. This shift-DR/update-DR process serves as the basis for passing information between the external host and the MAXQ7667. These data register transactions occur in the data register portion of the TAP controller state sequence diagram and have no effect on the instruction register.

11.4.6 TAP Communication Examples—IR-Scans and DR-Scans

Figures 11-3 and 11-4 illustrate examples of communication between the host JTAG controller and the TAP of the MAXQ7667. The host controls the TCK and TMS signals to move through the desired TAP states while accessing the selected shift register through the TDI input and TDO output pair.

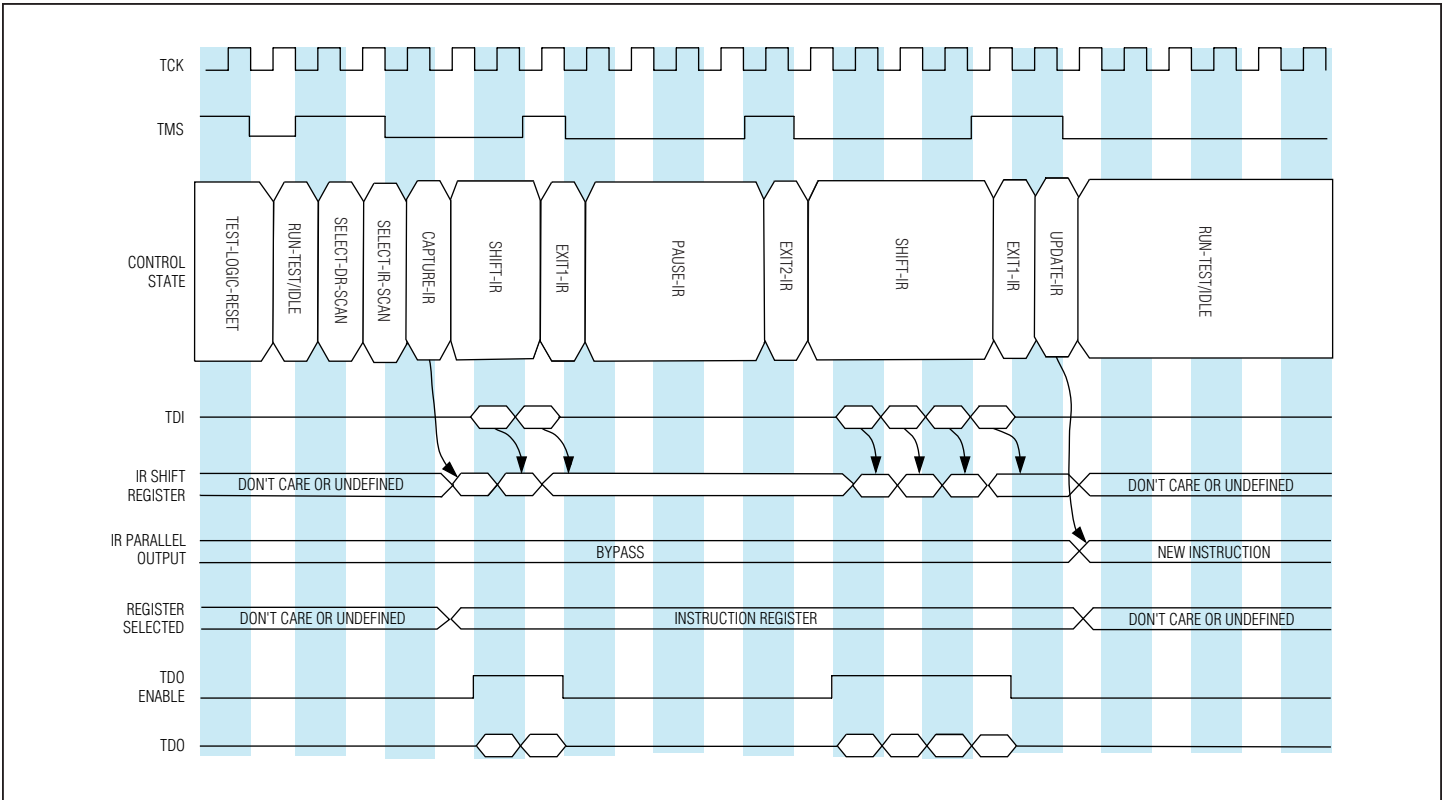


Figure 11-3. TAP Controller Debug Mode—IR-Scan Example

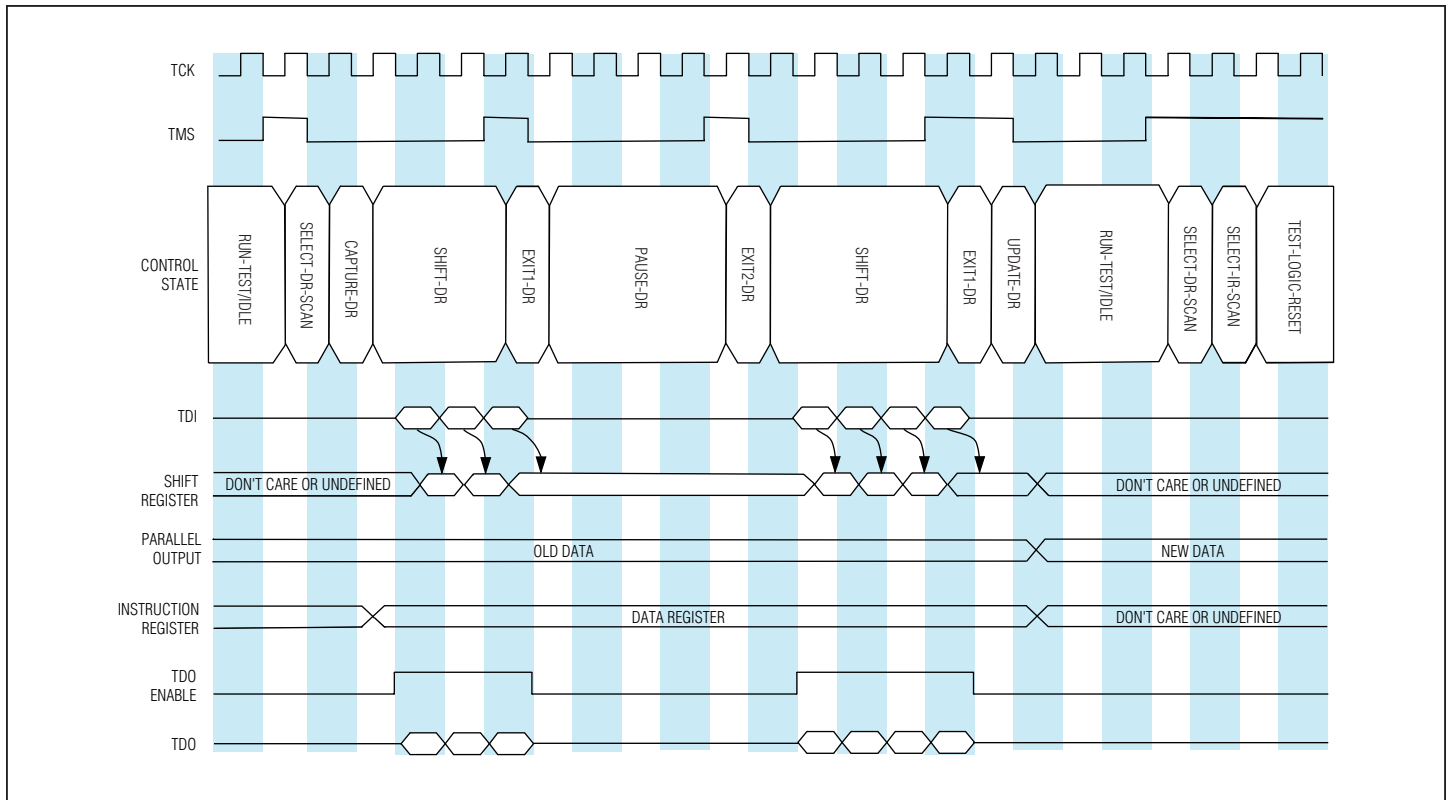


Figure 11-4. TAP Controller Debug Mode—DR-Scan Example

SECTION 12: IN-CIRCUIT DEBUG MODE

This section contains the following information:

12.1 Architecture	12-3
12.2 In-Circuit Debug Peripheral Registers	12-4
12.2.1 In-Circuit Debug Temporary 0 Register (ICDT0)	12-4
12.2.2 In-Circuit Debug Temporary 1 Register (ICDT1)	12-5
12.2.3 In-Circuit Debug Control Register (ICDC)	12-5
12.2.4 In-Circuit Debug Flag Register (ICDF)	12-7
12.2.5 In-Circuit Debug Buffer Register (ICDB)	12-7
12.2.6 In-Circuit Debug Address Register (ICDA)	12-8
12.2.7 In-Circuit Debug Data Register (ICDD)	12-8
12.2.8 System Control Register (SC)	12-9
12.3 Debug Engine Operation	12-10
12.3.1 Background Mode Operation	12-10
12.3.2 Breakpoint Registers	12-12
12.3.2.1 Breakpoint Registers 0 to 3 (BP0 to BP3)	12-12
12.3.2.2 Breakpoint Register 4 (BP4)	12-13
12.3.2.3 Breakpoint Register 5 (BP5)	12-14
12.3.3 Using Breakpoints	12-14
12.3.4 Debug Mode	12-15
12.3.5 Debug Mode Commands	12-15
12.3.6 Read-Register Map Command Host-ROM Instruction	12-17
12.3.7 Single-Step (Trace) Operation	12-17
12.3.8 Return	12-17
12.3.9 Debug Mode Special Considerations	12-18
12.3.10 Debug Command Operation	12-19
12.3.10.1 Register Read and Write Commands	12-19
12.3.10.2 Data Memory Read Command	12-19
12.3.10.3 Data Memory Write Command	12-19
12.3.10.4 Program Stack Read Command	12-19
12.3.10.5 Read Register Map Command	12-19

LIST OF FIGURES

Figure 12-1. In-Circuit Debugger12-3
Figure 12-2. Data Transmit and Receive During DR-Scan Sequence12-10

LIST OF TABLES

Table 12-1. Background Mode Commands12-11
Table 12-2. Background Mode Debug Commands12-16
Table 12-3. Output from Read Register Map Command12-20

SECTION 12: IN-CIRCUIT DEBUG MODE

Note: This section is only relevant to users who are planning to make their own debugging tools, hence this section can be skipped by most users. However, those users intending to implement in-system programming (see *Section 13*) should read this section.

The MAXQ7667 is equipped with embedded debug hardware and embedded ROM firmware developed for the purpose of providing in-circuit debugging capability to the user application. The in-circuit debug mode uses the JTAG-compatible TAP as its means of communication between the host and MAXQ7667 microcontroller. The in-circuit debug hardware and software features include the following:

- A debug engine
- A set of registers providing the ability to set breakpoints on register, code, or data
- A set of debug service routines stored in a ROM

Collectively, these hardware and software features allow two basic modes of in-circuit debugging:

- Background mode allows the host to configure and set up the in-circuit debugger while the CPU continues to execute the normal program. Debug mode can be invoked from background mode.
- Debug mode allows the debug engine to take control of the CPU, providing read-write access to internal registers and memory, and single-step trace operation.

12.1 Architecture

Figure 12-1 shows a simplified functional block diagram of the MAXQ7667 in-circuit debugger. The embedded hardware debug engine is implemented as a stand-alone hardware block in the MAXQ7667 microcontroller. The debug engine can be enabled for monitoring internal activities and interacting with selected internal registers while the CPU is executing user code. This capability allows the user to employ the embedded debug engine to debug the actual system, in place of the in-circuit emulator that uses external hardware to duplicate operation of the microcontroller outside of the real application environment.

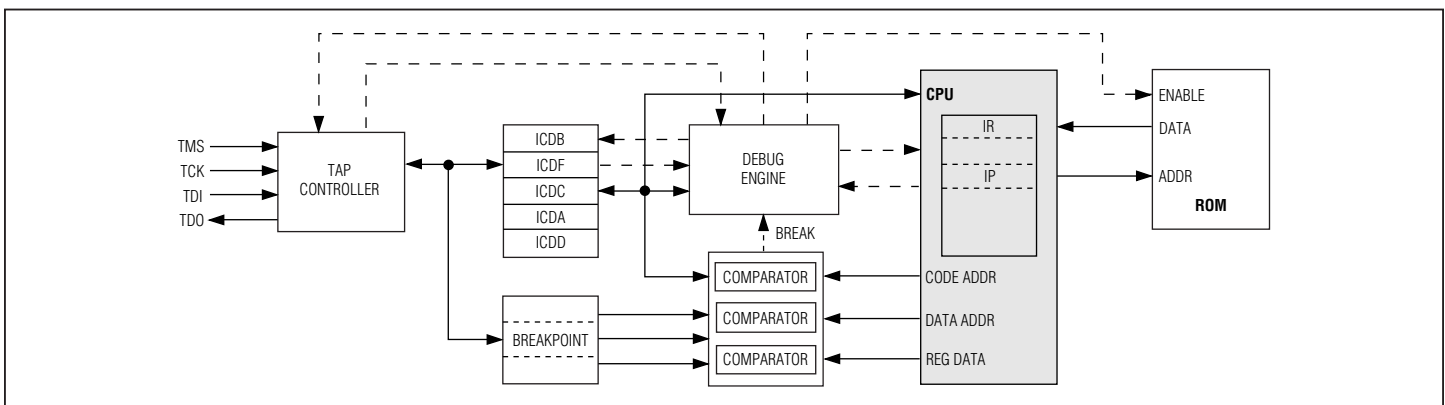


Figure 12-1. In-Circuit Debugger

The embedded debug engine is a state machine that takes commands from the host device and performs the necessary tasks to complete the debug function. While the TAP is running at the TCK clock frequency, the debug engine and all its associated hardware are clocked by the system clock. The debug engine is not operated in stop mode.

All debug engine activities are originated by the external host through 8-bit commands. The debug engine decodes the command in the ICDB register directly, and 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 break points
- Register read and write
- Program stack read
- Data memory read and write
- Optional password protection

The debug engine is supported by five functional registers:

- **ICDB:** The ICDB register is an 8-bit data register that supports exchanging command/data between the host system and the in-circuit debugger. The register functions as an 8-bit parallel buffer for the debug shift register in the TAP. The ICDB register is mapped to the peripheral register space and is read/write accessible by the CPU and the debug engine.
- **ICDC:** The ICDC register is an 8-bit control register for the in-circuit debugger. All bits in this register are set/reset by the debug engine. It is mapped to the peripheral register space and is read only by the CPU.
- **ICDF:** The ICDF register is an 8-bit register and is used to provide system status to the host system, the debug engine, and the CPU during debug operation. This register is mapped to the peripheral register space and read/write accessible by the CPU and the debug engine.
- **ICDA:** The ICDA register is a 16-bit register that is primarily used to specify an address for ROM assisted operations. The ICDA is mapped to the peripheral register space and is read only by the CPU. It is read/write accessible by the debug engine. The ICDA may also be used as a bit mask for register access breakpoints (REGE = 1).
- **ICDD:** The ICDD register is a 16-bit register that is used to store data for ROM assisted operations. The ICDD is mapped to the peripheral register space and is read only by the CPU. It is read/write accessible by the debug engine. The ICDD may also be used as the bit compare match data for register access breakpoints (REGE = 1).

12.2 In-Circuit Debug Peripheral Registers

The MAXQ7667 in-circuit debug peripheral registers are described here. All the in-circuit debug peripheral registers are directly accessible by the microcontroller through the module/index address.

12.2.1 In-Circuit Debug Temporary 0 Register (ICDT0)

The ICDT0 register is read/write accessible by the CPU only in background mode or debug mode. This register is intended for use by the utility ROM routine as temporary storage to save registers that might otherwise have to be placed in the stack. This register is cleared after a power-on reset or by a test-logic-reset TAP state.

Register Description: **In-Circuit Debug Temporary 0 Register**
 Register Name: **ICDT0**
 Register Address: **Module 02h, Index 18h**

Bit #	15	14	13	12	11	10	9	8
Name	ICDT015	ICDT014	ICDT013	ICDT012	ICDT011	ICDT010	ICDT09	ICDT08
Reset	0	0	0	0	0	0	0	0
Access	s	s	s	s	s	s	s	s

Bit #	7	6	5	4	3	2	1	0
Name	ICDT07	ICDT06	ICDT05	ICDT04	ICDT03	ICDT02	ICDT01	ICDT00
Reset	0	0	0	0	0	0	0	0
Access	s	s	s	s	s	s	s	s

s = special (read/write access only in background or debug mode)

Bits 15 to 0: In-Circuit Debug Temporary 0 Register Bits 15:0 (ICDT0[15:0])

12.2.2 In-Circuit Debug Temporary 1 Register (ICDT1)

The ICDT1 register is read/write accessible by the CPU only in background mode or debug mode. This register is intended for use by the utility ROM routines as temporary storage to save registers that might otherwise have to be placed in the stack. This register is cleared after a power-on reset or by a test-logic-reset TAP state.

Register Description: **In-Circuit Debug Temporary 1 Register**
 Register Name: **ICDT1**
 Register Address: **Module 02h, Index 19h**

Bit #	15	14	13	12	11	10	9	8
Name	ICDT115	ICDT114	ICDT113	ICDT112	ICDT111	ICDT110	ICDT19	ICDT18
Reset	0	0	0	0	0	0	0	0
Access	s	s	s	s	s	s	s	s

Bit #	7	6	5	4	3	2	1	0
Name	ICDT17	ICDT16	ICDT15	ICDT14	ICDT13	ICDT12	ICDT11	ICDT10
Reset	0	0	0	0	0	0	0	0
Access	s	s	s	s	s	s	s	s

s = special (read/write access only in background or debug mode)

Bits 15 to 0: In-Circuit Debug Temporary 1 Register Bits 15:0 (ICDT1[15:0])

12.2.3 In-Circuit Debug Control Register (ICDC)

The ICDC register is read/write accessible by the debug engine and is read only by the CPU. This register is cleared after a power-on reset or by a test-logic-reset TAP state.

Register Description: **In-Circuit Debug Control Register**
 Register Name: **ICDC**
 Register Address: **Module 02h, Index 1Ah**

Bit #	7	6	5	4	3	2	1	0
Name	DME	—	REGE	—	CMD3	CMD2	CMD1	CMD0
Reset	0	0	0	0	0	0	0	0
Access	rs	r	rs	r	rs	rs	rs	rs

r = read, s = special (write only by debug engine)

Bit 7: Debug Mode Enable (DME). When this bit is cleared to 0, background mode commands can be executed but breakpoints are disabled. When this bit is set to 1, breakpoints are enabled while background mode commands can still be entered. This bit can only be set or cleared from debug mode. This bit has no meaning for the ROM code.

Bits 6 and 4: Reserved. Read 0, write ignored.

Bit 5: Break-On Register Enable (REGE). The REGE bit is used to enable the break-on register function. When the REGE bit is set to 1, BP4 and BP5 are used as register breakpoints. A break occurs when the content of BP4 is matched with the destination address of the current instruction. For BP5, a break occurs only on a selected data pattern for a selected destination register addressed by BP5. The data pattern is determined by the contents in the ICDA and ICDD register. The REGE bit alone does not enable register breakpoints, but simply changes the manner in which BP4 and BP5 are used. The DME bit still must be set to logic 1 for any breakpoint to occur. This bit has no meaning for the ROM code.

Bits 3 to 0: Command Bits 3:0 (CMD[3:0]). These bits reflect the current host command in debug mode. These bits are set by the debug engine and allow the ROM code to determine the course of action.

CMD3	CMD2	CMD1	CMD0	ACTION
0	0	0	0	No Operation
0	0	0	1	Read Register Map
0	0	1	0	Read Data Memory
0	0	1	1	Read Stack Memory
0	1	0	0	Write Register
0	1	0	1	Write Data Memory
0	1	1	0	Trace, Single-Step the CPU
1	0	0	0	Unlock Password
1	0	0	1	Read Register
X	X	X	X	Reserved

12.2.4 In-Circuit Debug Flag Register (ICDF)

Register Description: **In-Circuit Debug Flag Register**
 Register Name: **ICDF**
 Register Address: **Module 02h, Index 1Bh**

Bit #	7	6	5	4	3	2	1	0
Name	—	—	—	—	PSS1	PSS0	SPE	TXC
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	rw	rw	rw	rw

r = read, w = write

Bits 7 to 4: Reserved. Read 0, write ignored.

Bits 3 and 2: Programming Source Select Bits 1 and 0 (PSS[1:0]). See *Section 13* for information on these bits.

Bit 1: System Program Enable (SPE). See *Section 13* for information on this bit.

Bit 0: Serial Transfer Enable (TXC). This bit is set by hardware at the end of a transfer cycle at the TAP communication link. The TXC bit helps the debug engine to recognize host requests, either command or data. This bit is normally set by ROM code to signify or request the sending or receiving of data. Once set, the debug engine clears the TXC bit. CPU writes to the TXC bit result in the clearing of the JTAG PSS[1:0] bits.

12.2.5 In-Circuit Debug Buffer Register (ICDB)

The ICDB register serves as the parallel holding buffer for the debug shift register of the TAP. Data is read from or written to ICDB for serial communication between the debug routines and the external host. This register is cleared to 00h after a power-on reset or a test-logic-reset TAP state.

Register Description: **In-Circuit Debug Buffer Register**
 Register Name: **ICDB**
 Register Address: **Module 02, Index 1Ch**

Bit #	7	6	5	4	3	2	1	0
Name	ICDB7	ICDB6	ICDB5	ICDB4	ICDB3	ICDB2	ICDB1	ICDB0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bits 7 to 0: In-Circuit Debug Buffer Register Bits 7:0 (ICDB[7:0])

12.2.6 In-Circuit Debug Address Register (ICDA)

The debug engine uses the ICDA register to store addresses so that ROM code may view that information. This register is also used by the debug engine as a mask register to mask out don't care bits in the ICDD register when BP5 is used as a register breakpoint. When a bit in this register is set to 1, the corresponding bit location in the ICDD register is compared to the data being written to the destination register to determine if a break should be generated. When a bit in this register is cleared, the corresponding bit in the ICDD register are don't cares and are not compared against the data being written. When all bits in this register are cleared, any updated data pattern causes a break when the BP5 register matches the destination register address of the current instruction. This register is cleared to 0000h after a power-on reset or a test-logic-reset TAP state.

Register Description: **In-Circuit Debug Address Register**

Register Name: **ICDA**

Register Address: **Module 02h, Index 1Dh**

Bit #	15	14	13	12	11	10	9	8
Name	ICDA15	ICDA14	ICDA13	ICDA12	ICDA11	ICDA10	ICDA9	ICDA8
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	ICDA7	ICDA6	ICDA5	ICDA4	ICDA3	ICDA2	ICDA1	ICDA0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Bits 15 to 0: In-Circuit Debug Address Register Bits 15:0 (ICDA[15:0])

12.2.7 In-Circuit Debug Data Register (ICDD)

The debug engines uses the ICDD register to store data/read count so that ROM code can view that information. The debug engine also uses this register as a data register for content matching when BP5 is used as a register breakpoint. In this case, only data bits in this register with their corresponding mask bits in the ICDA register set are compared with the updated destination data to determine if a break should be generated. This register is cleared to 0000h after a power-on reset and or a test-logic-reset sequence TAP state.

Register Description: **In-Circuit Debug Data Register**

Register Name: **ICDD**

Register Address: **Module 02h, Index 1Eh**

Bit #	15	14	13	12	11	10	9	8
Name	ICDD15	ICDD14	ICDD13	ICDD12	ICDD11	ICDD10	ICDD9	ICDD8
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	ICDD7	ICDD6	ICDD5	ICDD4	ICDD3	ICDD2	ICDD1	ICDD0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Bits 15 to 0: In-Circuit Debug Data Register Bits 15:0 (ICDD[15:0])

12.2.8 System Control Register (SC)

Register Description: **System Control Register**
 Register Name: **SC**
 Register Address: **Module 08h, Index 08h**

Bit #	7	6	5	4	3	2	1	0
Name	TAP	—	CDA1	CDA0	UPA	ROD	PWL	—
Reset	0	0	0	0	0	0	1	0
Access	rw	r	rw	rw	rw	rw	rw	r

r = read, w = write

**This register defaults to 80h on all forms of reset except after power-on reset. After power-on reset, the PWL bit is also set and this register defaults to 82h.*

Bit 7: Test Access (JTAG) Port Enable (TAP). This bit controls whether the TAP special function pins are enabled. The TAP defaults to being enabled. See *Section 11* for more information on this bit.

0 = JTAG/TAP functions are disabled and P0.0–P0.3 can be used as general-purpose I/O pins

1 = TAP special function pins P0.0–P0.3 are enabled to act as JTAG inputs and outputs

Bits 6 and 0: Reserved

Bits 5 and 4: Code Data Access Bits 1 and 0 (CDA[1:0]). See *Section 2* for more information on these bits.

Bit 3: Upper Program Access (UPA). See *Section 2* for more information on this bit.

Bit 2: ROM Operation Done (ROD). This bit is used to signify completion of a ROM operation sequence. The utility ROM signals that it has completed a requested task by setting the ROD (ROM operation done) bit of the SC register to logic 1. The ROD bit is reset by the debug engine when it recognizes the done condition. Setting the ROD bit to 1 when the SPE bit is also set causes the system to reset.

Bit 1: Password Lock (PWL). This bit defaults to 1 on a power-on reset. When this bit is 1, it requires a 32-byte password to be matched with the password in the program space before allowing access to the password protected in-circuit debug or bootstrap loader ROM routines. Clearing this bit to 0 disables the password protection for these ROM routines. See *Section 13* for more information on this bit.

12.3 Debug Engine Operation

To enable a communication link between the host and the MAXQ7667 debug engine, the debug instruction (010b) must be loaded into the TAP instruction register using the IR-scan sequence (see *Section 11*). Once the instruction is latched in the instruction parallel buffer (IR[2:0]) and is recognized by the TAP controller in the update-IR state, the 10-bit data shift register is activated as the communication channel for DR-scan sequences. The TAP instruction register retains the debug instruction until a new instruction is shifted via an IR-scan or the TAP controller returns to the test-logic-reset state.

The host now can transmit and receive serial data through the 10-bit data shift register that exists between the TDI input and TDO output during DR-scan sequences. All background and debug mode communication (commands, data input/output, and status) occurs via this serial channel. Each 10-bit exchange of data between the host and the MAXQ7667 internal hardware is composed of two status bits and a single byte of command or data. The 10-bit word is always transmitted least significant bit first according to the following format.

The data byte portion of the 10-bit shift register is interfaced directly to the ICDB parallel register. The ICDB register functions as the holding data register for both transmit and receive operations. On the falling edge of TCK in the update-DR state, the outgoing data is loaded from the ICDB parallel register to the outgoing data register, while the incoming shift register data is latched in the ICDB parallel register.

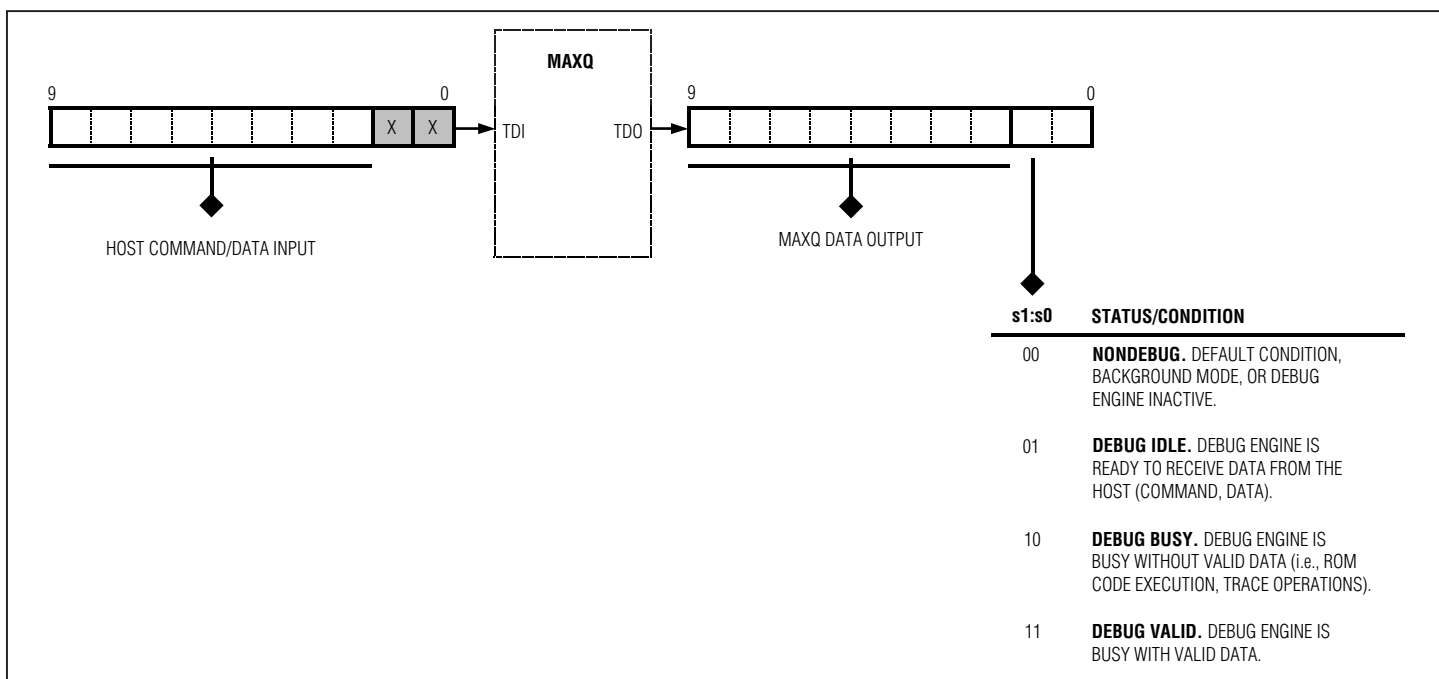


Figure 12-2. Data Transmit and Receive During DR-Scan Sequence

12.3.1 Background Mode Operation

When the instruction register is loaded with the debug instruction (IR[2:0] = 010b), the host can communicate with the MAXQ7667 in a background mode using TAP DR-scan sequences without disturbing CPU operation. Note, however, that JTAG in-system programming also requires use of the 10-bit debug shift register and, if enabled (SPE, PSS[1:0] = 100b), takes precedence over background mode communication (see *Section 13*). When operating in background mode, the status bits (Figure 12-2) are always cleared to 00b (nondebug), which indicates that the MAXQ7667 is ready to receive background mode commands. The host can perform the following operations from background mode:

- Read/write internal breakpoint registers (BP0–BP5).
- Read/write internal in-circuit debug registers (ICDC, ICDF, ICDA, ICDD).
- Monitor to determine when a breakpoint match has occurred.
- Directly invoke debug mode.

Table 12-1 shows the background mode commands supported by the MAXQ7667. Encodings (op code) not listed in this table are not supported in background mode and are treated as no operations.

A command can consist of multiple-byte transactions between the external host and the debug engine via the TAP. However, a command code is always 8 bits and is always transmitted first, followed by address and/or data when needed.

Table 12-1. Background Mode Commands

OP CODE	COMMAND	OPERATION
0000-0000	No Operation	No Operation. Default state for Debug Shift register.
0000-0001	Read ICDC	Read Control Data from the ICDC. The contents of the ICDC register are loaded into the Debug Shift Register via the ICDB register for host read. This command requires one follow-on transfer cycle.
0000-0010	Read ICDF	Read Flags from the ICDF. The contents of the ICDF register (one byte) are loaded into the Debug Shift Register via the ICDB register for host read. This command requires one follow-on transfer cycle.
0000-0011	Read ICDA	Read Data from the ICDA. The contents of the ICDA register are loaded into the Debug Shift Register via the ICDB register for host read. This command requires two follow-on transfer cycles with the least significant byte first.
0000-0100	Read ICDD	Read Data from the ICDD. The contents of the ICDD register are loaded into the Debug Shift Register via the ICDB register for host read. This command requires two follow-on transfer cycles with the least significant byte first.
0000-0101	Read BP0	Read Data from the BP0. The contents of the BP0 register are loaded into the Debug Shift Register via the ICDB register for host read. This command requires two follow-on transfer cycles with the least significant byte first.
0000-0110	Read BP1	Read Data from the BP1. The contents of the BP1 register are loaded into the Debug Shift Register via the ICDB register for host read. This command requires two follow-on transfer cycles with the least significant byte first.
0000-0111	Read BP2	Read Data from the BP2. The contents of the BP2 register are loaded into the Debug Shift Register via the ICDB register for host read. This command requires two follow-on transfer cycles with the least significant byte first.
0000-1000	Read BP3	Read Data from the BP3. The contents of the BP3 register are loaded into the Debug Shift Register via the ICDB register for host read. This command requires two follow-on transfer cycles with the least significant byte first.
0000-1001	Read BP4	Read Data from the BP4. The contents of the BP4 register are loaded into the Debug Shift Register via the ICDB register for host read. This command requires two follow-on transfer cycles with the least significant byte first.
0000-1010	Read BP5	Read Data from the BP5. The contents of the BP5 register are loaded into the Debug Shift Register via the ICDB register for host read. This command requires two follow-on transfer cycles with the least significant byte first.
0001-0001	Write ICDC	Write Control Data to the ICDC. The contents of ICDB are loaded into the ICDC register by the debug engine at the end of the data transfer cycle.
0001-0011	Write ICDA	Write Data to the ICDA. The contents of ICDB are loaded into the ICDA register by the debug engine at the end of the data transfer cycles. Data is transferred with the least significant byte first.
0001-0100	Write ICDD	Write Data to the ICDD. The contents of ICDB are loaded into the ICDD register by the debug engine at the end of data transfer cycles. Data is transferred with the least significant byte first.
0001-0101	Write BP0	Write Data to the BP0. The contents of ICDB are loaded into the BP0 register by the debug engine at the end of data transfer cycles. Data is transferred with the least significant byte first.
0001-0110	Write BP1	Write Data to the BP1. The contents of ICDB are loaded into the BP1 register by the debug engine at the end of data transfer cycles. Data is transferred with the least significant byte first.
0001-0111	Write BP2	Write Data to the BP2. The contents of ICDB are loaded into the BP2 register by the debug engine at the end of data transfer cycles. Data is transferred with the least significant byte first.
0001-1000	Write BP3	Write Data to the BP3. The contents of ICDB are loaded into the BP3 register by the debug engine at the end of data transfer cycles. Data is transferred with the least significant byte first.
0001-1001	Write BP4	Write Data to the BP4. The contents of ICDB are loaded into the BP4 register by the debug engine at the end of data transfer cycles. Data is transferred with the least significant byte first.
0001-1010	Write BP5	Write Data to the BP5. The contents of ICDB are loaded into the BP5 register by the debug engine at the end of data transfer cycles. Data is transferred with the least significant byte first.
0001-1111	Debug	Debug Command. This command forces the debug engine into debug mode and halts the CPU operation at the completion of the current instruction after the debug engine recognizes the debug command.

12.3.2 Breakpoint Registers

The MAXQ7667 incorporates six host-configurable breakpoint registers (BP0–BP5) for establishing different types of breakpoint mechanisms. The first four breakpoint registers (BP0–BP3) are 16-bit registers that are configurable as program memory address breakpoints. When enabled, the debug engine forces a break when a match between the breakpoint register and the program memory execution address occurs. The final two 16-bit breakpoint registers (BP4 and BP5) are configurable in one of two ways. They can be configured as data memory address breakpoints or can be configured to support register-access breakpoints. In either case, if breakpoints are enabled and the defined breakpoint match occurs, the debug engine generates a break condition. The six breakpoint registers are detailed in the following sections.

12.3.2.1 Breakpoint Registers 0 to 3 (BP0 to BP3)

The BP0 to BP3 registers are accessible only via background mode read/write debug commands. These four registers serve as program memory address breakpoints. When the DME bit is set, the debug engine monitors the program address bus activity while the CPU is executing the user program. A break occurs when the address pattern matches with the contents of these registers, allowing the debug engine to take control of the CPU and enter debug mode.

These registers default to FFFFh after a power-on reset or test-logic-reset TAP state.

Register Description: **Breakpoint Register x (where x = 0, 1, 2, 3)**

Register Name: **BPx**

Bit #	15	14	13	12	11	10	9	8
Name	BPx15	BPx14	BPx13	BPx12	BPx11	BPx10	BPx9	BPx8
Reset	1	1	1	1	1	1	1	1
Access	s	s	s	s	s	s	s	s

Bit #	7	6	5	4	3	2	1	0
Name	BPx7	BPx6	BPx5	BPx4	BPx3	BPx2	BPx1	BPx0
Reset	1	1	1	1	1	1	1	1
Access	s	s	s	s	s	s	s	s

s = special (accessible only by background mode read/write commands)

Bits 15 to 0: Breakpoint Register x Bits 15:0 (BPx[15:0])

12.3.2.2 Breakpoint Register 4 (BP4)

Register Description: **Breakpoint Register 4**

Register Name: **BP4**

This register is accessible only via background mode read/write commands.

When (REGE = 0): This register serves as one of the two data memory address breakpoints. When DME is set in background mode, the debug engine monitors the data memory address bus activity while the CPU is executing the user program. If an address match is detected, a break occurs, allowing the debug engine to take over control of the CPU and enter debug mode.

When (REGE = 1): This register serves as one of the two register breakpoints. A break occurs when the destination register address for the executed instruction matches with the specified module and index. When used as register breakpoint, the bits BP4[3:0] are recognized as module specifier and bits BP4[8:4] are recognized as the register index within the module. The bits BP4[15:9] are ignored.

This register defaults to FFFFh after a power-on reset or test-logic-reset TAP state.

Bit #	15	14	13	12	11	10	9	8
Name	BP415	BP414	BP413	BP412	BP411	BP410	BP49	BP48
Reset	1	1	1	1	1	1	1	1
Access	s	s	s	s	s	s	s	s

Bit #	7	6	5	4	3	2	1	0
Name	BP47	BP46	BP4 5	BP44	BP43	BP42	BP41	BP40
Reset	1	1	1	1	1	1	1	1
Access	s	s	s	s	s	s	s	s

s = special (accessible only by background mode read/write commands)

Bits 15 to 0: Breakpoint Register 4 Bits 15:0 (BP4[15:0])

12.3.2.3 Breakpoint Register 5 (BP5)

This register is accessible only through background mode read/write commands.

When (REGE = 0): This register serves as one of the two data memory address breakpoints. When DME is set in background mode, the debug engine monitors the data memory address bus activity while the CPU is executing the user program. If an address match is detected, a break occurs, allowing the debug engine to take control of the CPU and enter debug mode.

When (REGE = 1): This register serves as one of the two register breakpoints. A break occurs when the following two conditions are met:

- Condition 1: The destination register address for the executed instruction matches with the specified module and index. When used as register breakpoint, the bits BP5[3:0] are recognized as module specifier and bits BP5[8:4] are recognized as the register index within the module. Bits BP5[15:9] are ignored.
- Condition 2: The bit pattern written to the destination register matches those bits specified for comparison by the ICDD data register and ICDA mask register. Only those ICDD data bits with their corresponding ICDA mask bits are compared. When all bits in the ICDA register are cleared, Condition 2 becomes a don't care.

This register defaults to FFFFh after a power-on reset or test-logic-reset TAP state.

Register Description: **Breakpoint Register 5**
 Register Name: **BP5**

Bit #	15	14	13	12	11	10	9	8
Name	BP515	BP514	BP513	BP512	BP511	BP510	BP59	BP58
Reset	1	1	1	1	1	1	1	1
Access	s	s	s	s	s	s	s	s

Bit #	7	6	5	4	3	2	1	0
Name	BP57	BP56	BP55	BP54	BP53	BP52	BP51	BP50
Reset	1	1	1	1	1	1	1	1
Access	s	s	s	s	s	s	s	s

s = special (accessible only by background mode read/write commands)

Bits 15 to 0: Breakpoint Register 5 Bits 15:0 (BP5[15:0])

12.3.3 Using Breakpoints

All breakpoint registers (BP0–BP5) default to the FFFFh state on power-on reset or when the test-logic-reset TAP state is entered. The breakpoint registers are accessible only with background mode read/write commands issued over the TAP communication link. The breakpoint registers are not read/write accessible to the CPU.

Setting the debug-mode enable (DME) bit in the ICDC register to logic 1 enables all six breakpoint registers for breakpoint match comparison. The state of the break-on register enable (REGE) bit in the ICDC register determines whether the BP4 and BP5 breakpoints should be used as data memory address breakpoints (REGE = 0) or as register breakpoints (REGE = 1).

When using the register matching breakpoints, it is important to realize that debug mode operations (e.g., read data memory, write data memory, etc.) require the use of ICDA and ICDD for passing information between the host and MAXQ7667 ROM routines. It is advised that these registers be saved and restored, or be reconfigured before returning to the background mode if register breakpoints are to remain enabled.

When a breakpoint match occurs, the debug engine forces a break and the MAXQ7667 enters debug mode. If a breakpoint match occurs on an instruction that activates the PFX register, the break is held off until the prefixed operation completes. The host can assess whether debug mode has been entered by monitoring the status bits of the 10-bit word shifted out of the TDO pin. The status bits change from the nondebug (00b) state associated with background mode to the debug-idle (01b) state when debug mode is entered. Debug mode can also be manually invoked by host issuance of the debug background command.

12.3.4 Debug Mode

There are two ways to enter debug mode from background mode: 1) issuance of the debug command directly by the host through the TAP communication port, or 2) the breakpoint matching mechanism.

The host can issue the debug background command to the debug engine. The response time varies dependent on system conditions when the command is issued. The breakpoint mechanism provides a more controllable response, but requires that the breakpoints be initially configured in background mode. No matter the method of entry, the debug engine takes control of the CPU in the same manner. Debug mode entry is similar to the state machine flow of an interrupt except that the target execution address is 8010h, which resides in the utility ROM instead of the address specified by the IV register that is used for interrupts. On debug mode entry, the following actions occur:

- 1) Block the next instruction fetch from program memory.
- 2) Push the return address onto the stack.
- 3) Set the contents of IP to 8010h.
- 4) Clear the IGE bit to 0 to disable interrupt handler if it is not already clear.
- 5) Halt CPU operation.

Once in debug mode, further breakpoint matches or host issuance of the debug command are treated as no operations and do not disturb debug engine operation. Entering debug mode also stops the clocks to all timers, including the watchdog timer. Temporarily disabling these functions allows debug mode operations without disrupting the relationship between the original user program code and hardware-timed functions. No interrupt request can be granted because the interrupt handler is also halted as a result of IGE = 0.

12.3.5 Debug Mode Commands

The debug engine sets the data shift-register status bits to 01b (debug-idle) to indicate that it is ready to accept debug commands from the host. The host can perform the following operations from debug mode:

- Read register map
- Read program stack
- Read/write register
- Read/write data memory
- Single step of CPU (trace)
- Return to background mode
- Unlock password

The only operations directly controlled by the debug engine are single step and return. All other operations are assisted by debug service routines contained in the utility ROM. These operations require that multiple bytes be transmitted and/or received by the host; however, each operation always begins with host transmission of a command byte. The debug engine decodes the command byte to determine the quantity, sequence, and destination for follow-on bytes received from the host. Even though there is no timing window specified for receiving the complete command and follow-on data, the debug engine must receive the correct number of bytes for a particular command before executing that command. If command and follow-on data are transmitted out of byte order or proper sequence, the only way to resolve this situation is to disable the debug engine by changing the instruction register (IR[2:0]) and reloading the debug instruction. Once the debug engine has received the proper number of command and follow-on bytes for a given ROM assisted operation, it responds with the following actions:

- Update the command bits (CMD[3:0]) in the ICDC register to reflect the host request.
- Enable the ROM if it has not been enabled.
- Force a jump to ROM address 8010h.
- Set the data shift register status bits to 10b (debug-busy).

The ROM code performs a read of the ICDC register CMD[3:0] bits to determine its course of action. The ROM can process some commands without receiving data from the host beyond the initially supplied follow-on bytes, while others (e.g., unlock password) require additional data from the host. Some commands need only provide an indication of completion to the host, while others (read register map) need to supply multiple bytes of output data. To accomplish data flow control between the host and ROM, the status bits should be used by the host to assess when the ROM is ready for additional data and/or when the ROM is providing valid data output (Figure 12-2).

Internally, the ROM can ascertain when new data is available or when it can output the next data byte via the TXC flag. The TXC flag is an important indicator between the debug engine and the utility ROM debug routines. The utility ROM firmware sets the TXC flag to 1 to indicate that valid data has been loaded to the ICDB register. The debug engine clears the TXC flag to 0 to indicate completion of a data shift cycle, thus allowing the ROM to continue execution of a requested task that is still in progress. The utility ROM signals that it has completed a requested task by setting the ROM operation done (ROD) bit of the SC register to logic 1. The ROD bit is reset by the debug engine when it recognizes the done condition. Table 12-2 shows the debug mode commands supported by the MAXQ7667. Note that background mode commands are supported inside debug mode. Encodings not listed in this table are not supported in debug mode and are treated as no operations.

Table 12-2. Background Mode Debug Commands

OP CODE	COMMAND	OPERATION
0010-0000	No Operation	No Operation
0010-0001	Read Register Map	Read Data from Internal Registers. This command forces the debug engine to update the CMD[3:0] bits in the ICDC to 0001b and perform a jump to ROM code at 8010h. The ROM debug service routine will load register data to ICDB for host capture/read, starting at the lowest register location in module 0, one byte at a time in a successive order until all internal registers are read and output to the host.
0010-0010	Read Data Memory	Read Data from Data Memory. This command requires four follow-on transfer cycles, two for the starting address and two for the word read count, starting with the LSB address and ending with the MSB read count. The address is moved to the ICDA register and the word read count is moved to the ICDD register by the debug engine. This information is directly accessible by the ROM code. At the completion of this command period, the debug engine updates the CMD[3:0] bits to 0010b and performs a jump to ROM code at 8010h. The ROM debug service routine will load ICDB from data memory according to address and count information provided by the host.
0010-0011	Read Program Stack	Read Data from Program Stack. This command requires four follow-on transfer cycles, two for the starting address and two for the read count, starting with the LSB address and ending with the MSB read count. The address is moved to the ICDA register and the read count is moved to the ICDD register by the debug engine. This information is directly accessible by the ROM code. At the completion of this command period, the debug engine updates the CMD[3:0] bits to 0011b and performs a jump to ROM code at 8010h. The ROM Debug service routine will pop data out from the stack according to the information received in the ICDA and ICDD register. The stack pointer is pre-decremented for each pop operation.
0010-0100	Write Register	Write Data to a Selected Register. This command requires four follow-on transfer cycles, two for the register address and two for the data, starting with the LSB address and ending with the MSB data. The address is moved to the ICDA register and the data is moved to the ICDD register by the debug engine. This information is directly accessible by the ROM code. At the completion of this command period, the debug engine updates the CMD[3:0] bits to 0100b and performs a jump to ROM code at 8010h. The ROM Debug service routine will update the select register according to the information received in the ICDA and ICDD registers.
0010-0101	Write Data Memory	Write Data to a Selected Data Memory Location. This command requires four follow-on transfer cycles, two for the memory address and two for the data, starting with the LSB address and ending with the MSB data. The address is moved to the ICDA register and the data is moved to the ICDD register by the debug engine. This information is directly accessible by the ROM code. At the completion of this command period, the debug engine updates the CMD[3:0] bits to 0101b and performs a jump to ROM code at 8010h. The ROM Debug service routine will update the selected data memory location according to the information received in the ICDA and ICDD registers.
0010-0110	Trace	Trace Command. This command allows single stepping the CPU and requires no follow-on transfer cycle. The trace operation is a 'debug mode exit, one cycle CPU execution, debug mode entry' sequence.
0010-0111	Return	Return Command. This command terminates the debug mode and returns the debug engine to background mode. This allows the CPU to resume its normal operation at the point where it has been last interrupted.
0010-1000	Unlock Password	Unlock the Password Lock. This command requires 32 follow-on transfer cycles each containing a byte value to be compared with the program memory password for the purpose of clearing the PWL bit and granting access to protected debug and loader functions. When this command is received, the debug engine updates the CMD[3:0] bits to 1000b and performs a jump to ROM code at 8010h. Data is loaded to the ICDB register when each byte of data is received, beginning with the LSB of the least significant word first and end with the MSB of the most significant word.
0010-1001	Read Register	Read from a Selected Internal Register. This command requires two follow-on transfer cycles, starting with the LSB address and ending with the MSB address. The address is moved to ICDA register by the debug engine. This information is directly accessible by the ROM code. At the completion of this command period, the debug engine updates the CMD[3:0] bits to 1001b and performs a jump to ROM code at 8010h. The ROM Debug service routine will always assume a 16-bit register length and return the requested data LSB first.

12.3.6 Read-Register Map Command Host-ROM Instruction

A read-register map command reads out data contents for all implemented system and peripheral registers. The host does not specify a target register but instead should expect register data output in successive order, starting with the lowest order register in register module 0. Data is loaded by the ROM to the 8-bit ICDB register and is output one byte per transfer cycle. Thus, for a 16-bit register, two transfer cycles are necessary. The host initiates each transfer cycle to shift out the data bytes and will find valid data output tagged with a debug-valid (status = 11b). At the end of each transfer cycle, the debug engine clears the TXC flag to signal the ROM service routine that another byte can be loaded to ICDB. The ROM service routine sets the TXC flag each time after loading data to the ICDB register. This process is repeated until all registers have been read and output to the host. The host system recognizes the completion of the register read when the status debug-idle is presented. This indicates that the debug engine is ready for another operation.

12.3.7 Single-Step (Trace) Operation

The debug engine supports single-step operation in debug mode by executing a trace command from the host. The debug engine allows the CPU to return to its normal program execution for one cycle and then forces a debug mode re-entry:

- 1) Set status to 10b (debug-busy).
- 2) Pop the return address from the stack.
- 3) Set the IGE bit to logic 1 if debug mode was activated when IGE = 1.
- 4) Supply the CPU with an instruction addressed by the return address.
- 5) Stall the CPU at the end of the instruction execution.
- 6) Block the next instruction fetch from program memory.
- 7) Push the return address onto the stack.
- 8) Set the contents of IP to 8010h.
- 9) Clear the IGE bit to 0 to disable the interrupt handler.
- 10) Halt CPU operation.
- 11) Set the status to debug-idle.

Note that the trace operation uses a return address from the stack as a legitimate address for program fetching. The host must maintain consistency of program flow during the debug process. The instruction pointer is automatically incremented after each trace operation, thus a new return address is pushed onto the stack before returning the control to the debug engine. Also, note that the interrupt handler is an essential part of the CPU and a pending interrupt could be granted during single-step operation since the IGE bit state present on debug mode entry is restored for the single step.

12.3.8 Return

To terminate the debug mode and return the debug engine to background mode, the host must issue a return command to the debug engine. This command causes the following actions:

- 1) Pop the return address from the stack.
- 2) Set the IGE bit to logic 1 if debug mode was activated when IGE = 1.
- 3) Supply the CPU with an instruction addressed by the return address.
- 4) Allow the CPU to execute the normal user program.
- 5) Set the status to 00b (nondebug).

To prevent a possible endless-breakpoint matching loop, no break occurs for a breakpoint match on the first instruction after returning from debug mode to background mode. Returning to background mode also enables all internal timer functions.

12.3.9 Debug Mode Special Considerations

The following are special considerations when using debug mode.

The debug engine cannot be operated reliably when the CPU is configured in the power management mode (divide-by-256 system clock mode). To allow for proper execution of debug mode commands when invoked during PMM, the switchback enable (SWB) bit should be configured to logic 1. With SWB = 1, entering active debug mode (whether by breakpoint match or issuance of the debug command) forces a switchback to the divide-by-1 system clock mode and allows the debug engine to function correctly. This allows user code to configure breakpoints that occur inside PMM, thus providing reliable use of debug commands. However, it does not allow a good means for re-entering PMM.

- Special caution should be exercised when using the write-register command on register bits that globally affect system operation (e.g., IGE, STOP). If the write-register command is used to invoke STOP mode (setting STOP = 1), the RST pin can be asserted to reset the debug engine and return to the background mode of operation.
- Single stepping (trace) through any IGE bit change operation results in the debug engine overriding the bit change since it retains the IGE bit setting captured when active debug mode was entered.
- Single stepping (trace) into an operation that sets STOP = 1 when IGE = 1 effectively allows enabled interrupts normally capable of causing exit from STOP mode to do so.
- Single stepping (trace) through any memory read instruction that reads from the utility ROM (such as "move Acc," @DP[0] with DP[0] set to 8000h) causes the memory read to return an incorrect value.
- Single stepping (trace) cannot be used when executing code from the utility ROM.
- Data memory allocation is important during system development if in-circuit debug is planned. The top 32-byte memory location can be used by the debug service routine during debug mode. The data contents in these locations can be altered and cannot be recovered.
- One available stack location is needed for debug mode. If the stack is full when entering debug mode, the oldest data in the stack will be overwritten.
- The crystal warmup counter is the only counter not disabled when active debug mode is entered. If the crystal warmup counter completes while in active debug mode, a glitchless switch will be made to selected clock source, which was being counted. It is important the user recognize that this action will occur as the TAP clock should be run no faster than 1/8th the system clock frequency.
- Any signal sampling that relies upon the internal system clock (e.g., counter inputs) can be unreliable since the system clock is turned off inside active debug mode between debug mode commands.

12.3.10 Debug Command Operation

The following sections provide specific notes on the MAXQ7667's operation in debugging mode.

12.3.10.1 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 MAXQ7667.

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).

12.3.10.2 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 *Section 2*. Data memory words returned by this command are output LSB first.

12.3.10.3 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 *Section 2*.

12.3.10.4 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 will be used, as words are popped off the stack and returned in descending order. Stack words returned by this command are output LSB first.

12.3.10.5 Read Register Map Command

This command outputs all peripheral registers in the range M0[00h] to M5[0Dh], along with a fixed set of system registers. The following formatting rules apply to the returned data.

- System registers are output as 8-bit or 16-bit, least significant byte first.
- All peripheral registers are output as 16-bit, least significant byte first. The top byte of 8-bit registers is returned as 00h.
- Nonimplemented peripheral registers in the range M0[00h] to M5[0Dh] are returned as 0000h.
- The value of SBUF0, SPIB, C0S, C0DB, C0RMS, C0TMA, and ASR are not read, and this register is returned as 0000h.

The first byte output by this command is the value 192 (C0h), which represents the number of peripheral registers output. Table 12-3 lists the remaining 448 bytes output by this command.

Table 12-3. Output from Read Register Map Command

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	PO0		PO1		00	00	EIF0		EIF1		00	00	00	00	00	00
1x	PI0		PI1		00	00	EIE0		EIE1		00	00	00	00	00	00
2x	PD0		PD1		00	00	EIES0		EIES1		00	00	00	00	00	00
3x	PS0		PS1		00	00	PR0		PR1		00	00	00	00	00	00
4x	MCNT		MA		MB		MC2		MC1		MC0		00	00	SPICN	
5x	SPICF		SPICK		FCNTL*		FDATA*		MCR1		MC0R		SCNT		STIM	
6x	SALM		FPCTL		00	00	00	00	BIASTRM*		BGTRM*		MONTRM*		RCTRM	
7x	SP0TRM*		SP1TRM*		SARLTRM*		SAROTRM*		ID0		ID1		ID2*		ID3*	
8x	T2CNA0		T2H0		T2RH0		T2CH0		T2CNA1		T2H1		T2RH1		T2CH1	
9x	T2CNB0		T2V0		T2R0		T2C0		T2CNB1		T2V1		T2R1		T2C1	
Ax	T2CFG0		T2CFG1		00	00	00	00	00	00	00	00	00	00	00	00
Bx	ICDT0		ICDT1		00	ICDC	00	ICDF	00	ICDB	ICDA		ICDD		TM*	
Cx	T2CNA2		T2H2		T2RH2		T2CH2		00	00	CNT1		SCON		00	00
Dx	T2CNB2		T2V2		T2R2		T2C2		FSTAT		ERRR		CHKSUM		ISVEC	
Ex	T2CFG2		STA0		SMD		FCON		CNT0		CNT2		IDFB		SADDR	
Fx	SADEN		BT		TMR		00	00	00	00	00	00	00	00	00	00
10x	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
11x	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
12x	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
13x	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
14x	BPH		BTRN		SARC		RCVC		PLLF		AIE		CMPC		CMPT	
15x	00	00	SARD		LPFC		OSCC		BPFI		BPFO		LPFD		LPFF	
16x	APE		ATST		FGAIN		B1COEF		B2COEF		B3COEF		A2A		A2B	
17x	00	00	A2D		00	00	A3A		A3B		00	00	A3D		00	00
18x	AP	APC	PSF	IC	IMR	SC	IIR	CKCN	WDCN	00	A[0]		A[1]		A[2]	
19x	A[3]		A[4]		A[5]		A[6]		A[7]		A[8]		A[9]		A[10]	
1Ax	A[11]		A[12]		A[13]		A[14]		A[15]		IP		SP+1		IV	
1Bx	LC[0]		LC[1]		OFFS		DPC		GR		BP		DP[0]		DP[1]	

*Reserved for factory use.

SECTION 13: IN-SYSTEM PROGRAMMING/BOOTLOADER

This section contains the following information:

13.1 Bootstrap-Loader Mode	13-2
13.2 In-System Programming Peripheral Registers	13-3
13.2.1 In-Circuit Debug Flag Register (ICDF)	13-3
13.2.2 System Control Register (SC)	13-4
13.3 Bootloader Protocol	13-4
13.3.1 Family 0 Commands (Not Password Protected)	13-5
13.3.2 Family 1 Commands: Load Variable Length (Password Protected)	13-7
13.3.3 Family 2 Commands: Dump Variable Length (Password Protected)	13-8
13.3.4 Family 3 Commands: CRC Variable Length (Password Protected)	13-8
13.3.5 Family 4 Commands: Verify Variable Length (Password Protected)	13-9
13.3.6 Family 5 Commands: Load and Verify Variable Length (Password Protected)	13-9
13.3.7 Family 6 Commands: Erase Variable Length (Password Protected)	13-9
13.3.8 Family E Commands: Erase Fixed Length (Password Protected)	13-10
13.4 Password-Protected Access	13-11
13.4.1 Entering a Password	13-11
13.5 JTAG Bootloader Operation	13-11
13.6 UART Bootloader Operation	13-13
13.6.1 UART Bootloader Host Routine	13-13

LIST OF FIGURES

Figure 13-1. Host-Side Operation Through UART	13-14
---	-------

LIST OF TABLES

Table 13-1. Programming Source Select Decode	13-3
Table 13-2. Bootloader Status Codes	13-5
Table 13-3. Bootloader Status Flags	13-6
Table 13-4. JTAG Status Decode	13-12

SECTION 13: IN-SYSTEM PROGRAMMING/BOOTLOADER

Note: The reader should be familiar with *Section 11: Test Access Port (TAP)* and *Section 12: In-Circuit Debug Mode* before reading this section.

The MAXQ7667 is equipped with a bootstrap loader as part of the utility ROM firmware. The main function of the bootstrap loader is to provide in-system programming capability to the user application. The MAXQ7667 in-system programming features include:

- Standard JTAG/TAP interface-based communication
- Built-in JTAG bootstrap loader for flash programming and verifying
- UART-based communication
- Built-in UART bootstrap loader for flash programming and verifying
- Password lock protection to prevent access to certain loader operations

13.1 Bootstrap-Loader Mode

The MAXQ7667 allows the user to program its flash through the JTAG or the UART port by allowing access to the ROM-based boot-loader through these ports.

The bootloader is entered in one of three ways: by a JTAG request during the power-up sequence, through a UART request immediately after power up when no password has been set, and by jumping to the bootloader from the application code.

After a reset, the MAXQ7667 instruction pointer jumps to the beginning of ROM code (0x8000). The ROM code does some initial house-keeping and then looks for a request from the JTAG port. If there is a valid request (i.e., SPE = 1, PSS = 00), the processor establishes communication between the ROM bootloader and the JTAG port. If there is no JTAG request and the password has been set (0x0010 to 0x001F is not all 0s or all Fs), the program execution jumps to the application code at address 0x0000. If the password has not been set (0x0010 to 0x001F is all 0s or all Fs), the ROM code monitors the UART for 5 seconds waiting to receive the autobaud character, 0x0D (carriage return). If the autobaud character is not detected within 5 seconds, program execution jumps to the application code at address 0x0000. If the autobaud character is detected during the 5-second window, the UART is established as the bootloader communication port and the MAXQ7667 responds with 0x3E. 0x3E is the loader prompt, which acknowledges that a command has been completed.

Once communication has been established with the loader, the host has access to all the Family 0 Commands regardless of the state of the PWL bit (Password Lock). If PWL = 0, all the loader commands are accessible. Family 0 commands all start with a 0 and provide basic functionality, but do not allow access to information in either program memory or data memory. This prevents unauthorized access of proprietary information.

One of the Family 0 Commands is Password Match (0x03). The host can clear the PWL bit by transmitting this loader command followed by the correct 32-byte password. If the 32 transmitted bytes match the information in flash at addresses 0x0010 to 0x001F, the PWL bit is cleared and all loader commands are available. Executing the Password Match command and sending the wrong password when the PWL bit is cleared sets the PWL bit and limits the loader to Family 0 Commands. If the bootloader is being entered from the application code, the application code may clear the PWL bit before jumping to the loader. If this is done, all loader commands are immediately available to the host. If the application code is used to clear the PWL bit, the application code should also provide security features to prevent unauthorized access to proprietary information.

Another Family 0 Command is "Master Erase" (0x02). Executing this unprotected command erases the entire flash memory. Executing a Master Erase removes the password and allows access to all the loader commands; however, since the proprietary code has been erased, security is maintained. The Master Erase is a useful way to regain control of a chip where the password has been inadvertently set or forgotten.

In applications where the JTAG port is not available, the password is set, and the user intends to flash the part through the UART port, **the user must develop reliable application code for jumping to the loader.** This code must be debugged before setting the password. If the password is set, the JTAG port is not available, and the application code cannot start the loader or control the flash, it is impossible to load a new program into flash.

The host can exit the loader by using the loader command "Exit Loader" (0x01), or a physical reset can be used to exit the loader. If the Exit Loader command is used and a password has been set in flash, then program execution will immediately transfer to flash at address 0x0000. If the password is not set in flash, then after exiting the loader the UART will be monitored for 5 seconds waiting for the autobaud character. If the autobaud character is detected, the loader is reentered. If the autobaud character is not detected, program execution transfers to flash at address 0x0000.

13.2 In-System Programming Peripheral Registers

The MAXQ7667 in-system programming peripheral registers are described here. It is also possible for the MAXQ7667 to bootstrap itself into in-system programming mode by setting the proper bits in the in-circuit debug flag register (ICDF) and invoking a reset. The procedure for invoking in-system programming in this manner must be defined and supported by the application firmware as discussed in *Section 13.5* and *Section 13.6*.

13.2.1 In-Circuit Debug Flag Register (ICDF)

Register Description: **In-Circuit Debug Flag Register**
 Register Name: **ICDF**
 Register Address: **Module 02h, Index 1Bh**

Bit #	7	6	5	4	3	2	1	0
Name	—	—	—	—	PSS1	PSS0	SPE	TXC
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	rw	rw	rw	rw

r = read, w = write

Bits 7 to 4: Reserved.

Bits 3 and 2: Programming Source Select Bits 1 and 0 (PSS[1:0]). These bits are used to select a programming interface during in-system programming when SPE is set to logic 1. Otherwise, the logic values of these bits have no meaning. The logical states of these bits, when read by the CPU, reflect the logical-OR of the PSS bits that are write accessible by the CPU and those in the system programming buffer register (SPB) of the TAP module (which are accessible via JTAG). These bits are read/write accessible by the CPU and are cleared to 0 by a power-on reset or test-logic-reset (see *Section 11*). CPU writes to the PSS bits result in clearing of the JTAG PSS[1:0] bits, as shown in Table 13-1.

Bit 1: System Program Enable (SPE). This bit controls the behavior of the MAXQ7667 following a reset. The SPE bit is used for in-system programming support, and its logical state, when read by the CPU, always reflects the logical-OR of the SPE bit that is write accessible by the CPU and the SPE bit of the SPB register in the TAP module, which is accessible via JTAG. The logical state of this bit determines the program flow after a reset.

- 0 = The MAXQ7667 jumps to application code in flash at 0000h following a reset.
- 1 = The MAXQ7667 executes the in-system programming bootloader following a reset.

This bit allows read/write access by the CPU and is cleared to 0 only on a power-on reset or test-logic-reset (see *Section 11*). The JTAG SPE bit is cleared by hardware when the ROD bit is set. CPU writes to the SPE bit result in clearing of the JTAG PSS[1:0] bits.

Bit 0: Serial Transfer Complete (TXC). See *Section 12* for more information on this bit.

Table 13-1. Programming Source Select Decode

PSS1	PSS0	PROGRAMMING SOURCE
0	0	JTAG
0	1	UART
1	0	Reserved
1	1	Reserved

13.2.2 System Control Register (SC)

Register Description: **System Control Register**
 Register Name: **SC**
 Register Address: **Module 08h, Index 08h**

Bit #	7	6	5	4	3	2	1	0
Name	TAP	—	CDA1	CDA0	UPA	ROD	PWL	—
Reset	1	0	0	0	0	0	1*	0
Access	rw	r	rw	rw	rw	rw	rw	r

r = read, w = write

*This register defaults to 80h on all forms of reset except after power-on reset. After power-on reset, the PWL bit is also set and this register defaults to 82h.

Bit 7: Test Access (JTAG) Port Enable (TAP). This bit controls whether the TAP special function pins are enabled. The TAP defaults to being enabled. See *Section 11* for more information on this bit.

0 = JTAG/TAP functions are disabled and P0.0–P0.3 can be used as general-purpose I/O pins

1 = TAP special function pins P0.0–P0.3 are enabled to act as JTAG inputs and outputs

Bits 6 and 0: Reserved.

Bits 5 and 4: Code Data Access Bits 1 and 0 (CDA[1:0]). See *Section 2* for more information on these bits.

Bit 3: Upper Program Access (UPA). Not used in the MAXQ7667.

Bit 2: ROM Operation Done (ROD). This bit is used to signify completion of a ROM operation sequence. The utility ROM signals that it has completed a requested task by setting the ROD (ROM operation done) bit of the SC register to logic 1. The ROD bit is reset by the debug engine when it recognizes the done condition. Setting the ROD bit to 1 when the SPE bit is also set causes the system to reset.

Bit 1: Password Lock (PWL). This bit defaults to 1 on a power-on reset. When this bit is 1, it requires a 32-byte password to be matched with the password in the program space before allowing access to the password protected in-circuit debug or bootstrap loader ROM routines. Clearing this bit to 0 disables the password protection for these ROM routines.

The password is defined as the 16 words of physical program memory at addresses 0010h to 001Fh. A password value of all ones or all zeros for all 16 words at addresses 0010h–001Fh will also unlock the password lock by setting the PWL bit to 0.

13.3 Bootloader Protocol

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 “>” (3Eh). Completion of exit loader command does not return a character.

Bootloader commands that fail for any reason set the bootloader status byte to an error code value describing the reason for the failure. See Table 13-2. This status byte can be read by means of the Get Status command (04h).

Table 13-2. 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 Password Match 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.
09	No AutoBaud Character. If no valid character (0x0D) has been received during the 5s timeout period.
10	Bad AutoBaud Character. After three retries, if the received character did not have the correct number of transitions.

13.3.1 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, and execution jumps to 8000h in the UROM and from there to the application code at 000h.

I/O	Byte 1
Input	01h
Output	

Command 02h—Master Erase

This command clears (programs to FFFFh) all words in the program flash memory.

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–003Fh. If the value matches, the password lock is cleared.

I/O	Byte 1	32 Bytes
Input	03h	Password value
Output		

Command 04h—Get Status

The status code returned by this command is defined in Table 13-2. The flags byte contains the following bit status flags.

I/O	Byte 1	Byte 2
Input	04h	
Output	Flags	Status Code

Table 13-3. 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.
2	Word/Byte Mode Supported 0 = The bootloader supports byte mode only. 1 = The bootloader supports word mode as well as byte mode.
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.

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. Fixed length is not supported if it returns.

I/O	Byte 1	Byte 2	Byte 3	Byte 4
Input	05h			
Output	SupportL	SupportH	CodeLen	DataLen

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 will be 0000h meaning “unknown amount of memory.”

I/O	Byte 1	Byte 2
Input	06h	
Output	SizeL	SizeH

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 will be 0000h meaning “unknown amount of memory.”

I/O	Byte 1	Byte 2
Input	07h	
Output	SizeL	SizeH

Command 08h—Get Loader Version

I/O	Byte 1	Byte 2
Input	08h	
Output	VersionL	VersionH

Command 09h—Get Utility ROM Version

I/O	Byte 1	Byte 2
Input	09h	
Output	VersionL	VersionH

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. In byte access mode, the range of memory that can be accessed is 0-0xFFFF bytes or 0-0x7FFF words. In word access mode, the range of memory that can be accessed is 0-0x1FFFF bytes or 0-0xFFFF words. Essentially the word access mode allows a larger range of memory access.

I/O	Byte 1	Byte 2
Input	0Ah	Mode
Output		

Command 0Dh—Get ID Information

For the MAXQ7667, the information returned by this command is a zero-terminated ROM banner string.

I/O	Byte 1	(Variable)
Input	0Dh	
Output		Device dependent information

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

Command 10h—Load Code Variable Length

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

- In byte mode, if the starting address is on an odd word boundary (such as 0001), the low bit will be changed to zero to make it an even word address.
- In byte mode, if an odd number of bytes is input, the data will be padded out with a 00 to make it an even number.

I/O	Byte 1	Byte 2	Byte 3	Byte 4	(Length) Bytes/Words
Input	10h	Length	AddressL	AddressH	Data to load
Output					

Command 11h—Load Data Variable Length

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

I/O	Byte 1	Byte 2	Byte 3	Byte 4	(Length) Bytes/Words
Input	11h	Length	AddressL	AddressH	Data to load
Output					

13.3.3 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 application flash/ROM—(LengthL) or (LengthH:LengthL) bytes/words starting at (AddressH:AddressL).

I/O	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Input (to dump < 256 bytes/words)	20h	1	AddressL	AddressH	LengthL	
Input (to dump 256+ bytes/words)	20h	2	AddressL	AddressH	LengthL	LengthH
Output	CodeByte 1	CodeByte 2	• • •		CodeByte N, where N = dump length	

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—(LengthL) or (LengthH:LengthL) bytes/words starting at (AddressH:AddressL).

I/O	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Input (to dump < 256 bytes/words)	21h	1	AddressL	AddressH	LengthL	
Input (to dump 256+ bytes/words)	21h	2	AddressL	AddressH	LengthL	LengthH
Output	DataByte 1	DataByte 2	• • •		DataByte N, where N = dump length	

13.3.4 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 application flash/ROM—(LengthL) or (LengthH:LengthL) bytes/words starting at (AddressH:AddressL).

I/O	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Input (to CRC < 256 bytes/words)	30h	1	AddressL	AddressH	LengthL	
Input (to CRC 256+ bytes/words)	30h	2	AddressL	AddressH	LengthL	LengthH
Output	CrCH	CrCL				

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 – (LengthL) or (LengthH:LengthL) bytes/words starting at (AddressH:AddressL).

I/O	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Input (to CRC < 256 bytes/words)	31h	1	AddressL	AddressH	LengthL	
Input (to CRC 256+ bytes/words)	31h	2	AddressL	AddressH	LengthL	LengthH
Output	CrCH	CrCL				

13.3.5 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 code flash, 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/Words
Input	40h	Length	AddressL	AddressH	Data to verify
Output					

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/Words
Input	41h	Length	AddressL	AddressH	Data to verify
Output					

13.3.6 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/Words
Input	50h	Length	AddressL	AddressH	Data to load/verify
Output					

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/Words
Input	51h	Length	AddressL	AddressH	Data to load/verify
Output					

13.3.7 Family 6 Commands: Erase Variable Length (Password Protected)

Command 60h—Erase Data Variable Length

This command has a slightly different format depending on the length of the erase requested. It clears (LengthL) or (LengthH:LengthL) bytes/words in the data SRAM to zero starting at (AddressH:AddressL).

I/O	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Input (to erase < 256 bytes/words)	60h	1	AddressL	AddressH	LengthL	
Input (to erase 256+ bytes/words)	60h	2	AddressL	AddressH	LengthL	LengthH

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

Command E0h—Erase Code Fixed Length

This command erases (programs to FFFFh) a 64-byte block of the program flash memory. The address given should be located in the block to be erased.

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

Command E1h—Erase Data Fixed Length

This command erases a single word/byte in data RAM to zero at (AddressH:AddressL) to zero.

I/O	Byte 1	Byte 2	Byte 3	Byte 4
Input	E1h	0	AddressL	AddressH
Output				

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. 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).

Due to the buffer between TDI and TDO (when using the JTAG port), there is always an output when data is put in. Therefore, for commands that do not require a response, the data out is not relevant and should be ignored (for example, exit loader, input 01h).

Another important point to note, when using the JTAG port, is that the data is pushed out sequentially. To get a relevant data output a series of no operation (Command 00h) should be pushed in to get the data out. For example, the code command 04h has two outputs: byte 1 is flags and byte 2 is status code. Hence, the data that goes in must have <Command> <No Operation cmd #1, for echo> <No Operation cmd #2, for pushing data out> <No Operation #3, for pushing data out>, or, in other words, 04 00 00 00. The data that comes out is valid after the <No Operation cmd #2> and <No Operation cmd #3>, and will be in the order byte 1 is flags and byte 2 is status code.

13.4 Password-Protected Access

Some applications require preventive measures to protect against simple access and viewing of program code memory. To address this need for code protection, the MAXQ7667 utility ROM that manages in-system programming, in-application programming, or in-circuit debugging grants full access to those utilities only after a password has been supplied. The password is defined as the 16 words (32 bytes) of physical program memory at addresses 0x0010 to 0x001Fh. Note that using these memory locations as a password does not exclude their usage for general code space if a unique password is not needed. A single password-lock bit (PWL) is implemented in the SC register. When the PWL is set to 1 (default at power-up when a password is found at 0010h–001Fh word), a password is required to access the in-circuit debug and in-system programming ROM routines that allow reading or writing of internal memory. When PWL is cleared to 0, all ROM routines are accessible without a password.

The PWL bit defaults to 1 by a power-on reset. To access the ROM utilities, a correct password is needed; otherwise, access to the ROM utilities is denied. Once the user supplies the correct password, the ROM clears the password lock. The PWL remains clear until either a power-on reset occurs or it is set to logic 1 by user software.

For the MAXQ7667, the password is always known for a fully erased device since the unprogrammed state of these memories is all ones. Password data set to all ones or all zeros for all 16 words at addresses 0010h–001Fh will remove the password lock by setting the PWL bit to 0. Once the memory has been programmed, a password is established and can be used for access protection. The utility ROM code denies access to the protected routines when PWL indicates a locked state.

13.4.1 Entering a Password

A password can be entered in two ways:

- Via the UART interface. After a connection is established between the host and the bootloader through the UART, the password command can be executed to allow access to all the family of commands. The unlock password command requires 32 bytes of data to compare with the program memory password.
- Via the TAP interface directly by issuing the Unlock Password debug mode command. The Unlock Password command requires 32 follow-on transfer cycles each containing a byte value to be compared with the program memory password.

13.5 JTAG Bootloader Operation

Maxim's JTAG interface board and most development tools take care of the item covered in this section, therefore this section can be skipped by most users.

The JTAG bootloader can only be entered after a reset. To enter the bootloader, the MAXQ7667 must be held in the reset state by keeping the $\overline{\text{RESET}}$ pin low. While the $\overline{\text{RESET}}$ pin is low the MAXQ7667 core is inactive, but the JTAG/TAP is available and operational. At this time the SPE bit can be set, and, when the $\overline{\text{RESET}}$ pin is released, the MAXQ7667 core becomes operational and the code jumps to the bootloader and eventually looks for the SPE bit state, which if set, establishes the communication between the JTAG and the bootloader. The following explanation shows the steps to set the SPE bit and establish a link to the bootloader. The reader must be familiar with *Section 11: Test Access Port (TAP)* before proceeding further.

To enable the bootstrap loader and establish a desired communication channel, the MAXQ7667 first must be reset and the reset pin must be kept low. Through a host device, the system programming instruction (100b) must be loaded into the TAP instruction register using the IR-scan sequence. Once the instruction is latched in the instruction parallel buffer (IR[2:0]) and is recognized by the TAP controller in the update-IR state, a 3-bit data shift register is activated as the communication channel for DR-scan sequences. The TAP retains the system programming instruction until a new instruction is shifted in or the TAP controller returns to the test-logic-reset state.

This 3-bit shift register formed between the TDI and TDO pins is directly interfaced to the 3-bit Serial Programming Buffer Register (SPB) in the TAP module (which is accessible via JTAG). The SPB register contains three bits with the following functions:

- **SPB.0: System Programming Enable (SPE).** Setting this bit to logic 1 denotes that system programming is desired upon exiting reset. When it is cleared to logic 0, no system programming is needed through JTAG. The reset vector examines the logic state of SPE in the utility ROM to determine the program flow after a reset. When $\text{SPE} = 1$, the bootstrap loader selected by the PSS[1:0] bits are activated to perform a bootstrap-loader function. When $\text{SPE} = 0$, the utility ROM can either transfer execution control to the application code (if $\text{PWL} = 1$) or to the UART bootloader (if $\text{PWL} = 0$) routine (where further testing is done to determine if a request is pending to establish a link between UART and the bootloader).

- **SPB.2 and SPB.1: Programming Source Select (PSS[1:0]).** These bits allow the host to select programming interface sources. PSS[1:0] = 00 for JTAG; PSS[1:0] = 01 for UART.

The DR-scan sequence is used to configure the SPB bits. The data content of the SPB register is reflected in the ICDF register and allows read/write access by the CPU. These bits are cleared by power-on reset or test-logic-reset of the TAP controller.

The MAXQ7667 JTAG bootloader uses the same status bit handshaking hardware as is used for in-circuit debugging. When the SPE bit of the system programming buffer (SPB) is set to 1 and JTAG is selected as the programming source (PSS[1:0] = 00b), the background and active-debug-mode state machines are disabled. Once the host loads the debug instruction into the TAP instruction register (IR[2:0]), the 10-bit shift register interfaces to ICDB and the status bits become available for JTAG-to-ROM bootloader communication. The status bits should be interpreted as noted in Table 13-4 for a JTAG bootloader operation.

When the using the JTAG bootloader option (SPE = 1, PSS[1:0] = 00b), the sole purpose of the debug hardware is to simultaneously transfer the data byte shifted in from the host into the ICDB register and transfer the contents of an internal holding register (loaded by ROM code writes of ICDB) into the shift register for output to the host. This transfer takes place on the falling edge of TCK at the update-DR state. The debug hardware additionally clears the TXC bit at this point in the state diagram. The ROM-loader code controls the status bit output to the host by asserting TXC = 1 when it has valid data to be shifted out. The ROM code can flexibly implement whatever communication protocol and command set it wishes within the data byte portion of the shifted 10-bit word.

Table 13-4. JTAG Status Decode

BITS (1:0)		STATUS	CONDITION
0	0	Reserved	Invalid condition.
0	1	Reserved	Invalid condition.
1	0	Loader-Busy	ROM loader is busy executing code or processing the current command.
1	1	Loader-Valid	ROM loader is supplying valid output data to the host in current shift operation.

13.6 UART Bootloader Operation

The host computer can request the application program to jump to the utility ROM bootloader to access the bootloader function through the UART serial port. Figure 13-1 shows the steps for the MAXQ7667 to evoke the bootloader through the UART.

```

.
.
.
Turn off Timer 0

Configure UART to operate in Mode 1 (see Section 8 for details).
move M2[ 27] , 0x04          // ICDF is set to select the UART as the communication Port
move dpc, #1Ch              // all data pointers in word mode
move A[ 2] , #01            // Prepare accumulator 2 for UART bootloader
move AP, #08d               // Active accumulator is 8
move DP[ 0] , #0800Dh       // This is where the address of the table is stored.
move ACC, @DP[ 0]          // Get the location of the function table.
add #13d                    // Add the index of the bootloader function.
move DP[ 0] , ACC           // Point to where the address of bootloader is stored.
move ACC, @DP[ 0]          // Retrieve the address of the function.
ljump ACC                   // Jump to the bootloader
.
.
Autobaud, establishes communication link between the host and the MAXQ7667

Password matching through UART, if desired, will allow full access to all the bootloader
commands
.
No return, once code exits it takes it to the application code
.

```

13.6.1 UART Bootloader Host Routine

The MAXQ7667 UART bootloader requires the UART to operate in Mode 1; 1 start bit, 8 data bits, and stop bit, no parity bits are used. Refer to *Section 8* for details on the UART.

The flow chart in Figure 13-1 shows the minimum program requirements for a host that is uploading a program to the MAXQ7667 through the UART bootloader. Most programmers will want to add additional features such as timeouts, error routines, program verification, etc. Uploading is done with the MAXQ7667 UART operating in Mode 1; 1 start bit, 8 data bits, and one stop bit. No parity bits are used.

If the password area 10h–1Fh has a valid password (i.e., anything other than all zeros and all ones), then to access the family of commands that are password protected, it is necessary to execute the password match command after the baud-rate detection.

Note: Loading code other than all zeros or all Fs into addresses 10h (word) through 1Fh (word) sets the password. The information in these locations is the password.

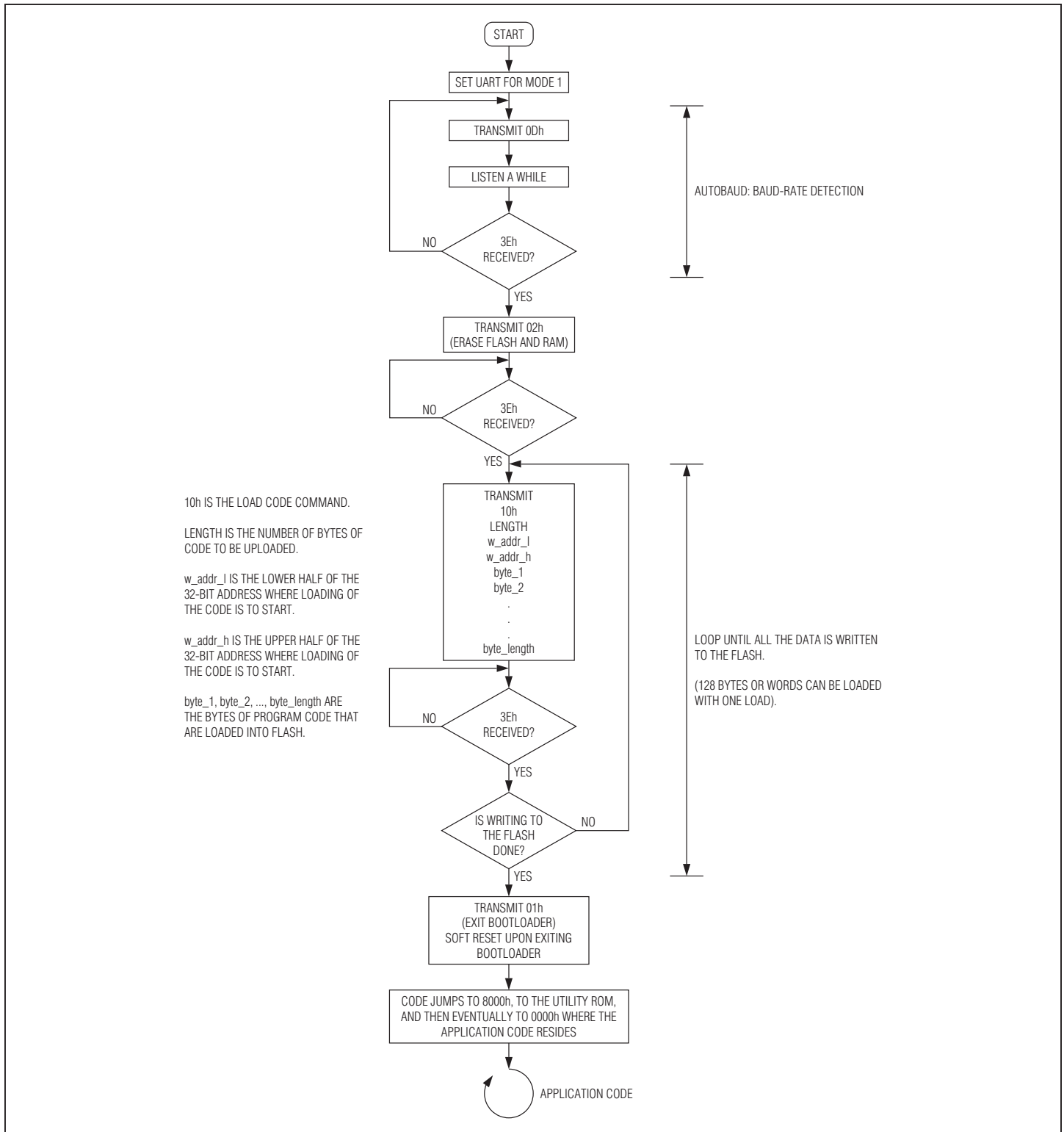


Figure 13-1. Host-Side Operation Through UART

SECTION 14: SAR ADC MODULE

This section contains the following information:

14.1 Architecture	14-3
14.2 SAR ADC Pins	14-4
14.3 SAR ADC Registers	14-5
14.3.1 SAR ADC Control Register (SARC)	14-5
14.3.2 Analog Interrupt Enable Register (AIE)	14-6
14.3.3 Analog Status Register (ASR)	14-7
14.3.4 SAR ADC Output Data Register (SARD)	14-8
14.3.5 Oscillator Control Register (OSCC)	14-8
14.3.6 Analog Power Enable Register (APE)	14-9
14.4 Voltage References	14-10
14.5 Analog-to-Digital Converter (ADC) Port	14-10
14.5.1 Single-Ended/Differential Inputs	14-10
14.5.2 True Differential Analog Input T/H	14-10
14.5.3 Unipolar/Bipolar	14-11
14.5.4 Transfer Function	14-12
14.5.5 Analog Input Protection	14-14
14.5.6 ADC Clock	14-14
14.5.7 Auto Shutdown Mode	14-15
14.5.8 ADC Conversion Start Sources and Timing	14-15
14.5.9 ADC Interrupts	14-19
14.5.10 Using the ADC	14-19
14.5.11 Measuring ADC Reference Voltage Using AVDD as Reference	14-19

LIST OF FIGURES

Figure 14-1. SAR ADC Block Diagram	14-3
Figure 14-2. Equivalent Input Circuit (Track/Acquisition Mode)	14-11
Figure 14-3. Equivalent Input Circuit (Hold/Conversion Mode)	14-11
Figure 14-4. Unipolar Transfer Function	14-12
Figure 14-5. Bipolar Transfer Function	14-13
Figure 14-6. Analog Input Range Measuring a Positive Analog Input Value	14-14
Figure 14-7. Analog Input Range Measuring a Negative Analog Input Value	14-14
Figure 14-8. Single-Edge ADC Conversion Timing; ADC Previously Off	14-17
Figure 14-9. Single-Edge ADC Conversion Timing; ADC Previously On	14-18
Figure 14-10. Dual-Edge ADC Conversion Timing; ADC Previously Off	14-18
Figure 14-11. Setting Up and Using the ADC Flow Chart	14-20

LIST OF TABLES

Table 14-1. SAR ADC Module Pins	14-4
Table 14-2. Unipolar Code Table	14-12
Table 14-3. Bipolar Code Table	14-13
Table 14-4. ADC Conversion Start Source Selection	14-15
Table 14-5. ADC Dual- and Single-Edge Modes	14-16

SECTION 14: SAR ADC MODULE

The MAXQ7667 incorporates a 12-bit SAR ADC with built-in sample-and-hold and a conversion rate up to 125ksps. The ADC allows general-purpose measurements for temperature, supply voltage, diagnostics, or other parameters. Some of the features include the following:

- Low-power, 125ksps, SAR (successive approximation) ADC, available as 12 bits.
- ADC can measure up to five external, single-ended signals or two differential signals.
- ADC has unipolar and bipolar modes with respective input ranges 0V to REF and -REF/2 to REF/2 (where REF is the reference voltage).
- An internal 2.5V bandgap-based voltage reference.
- An external voltage reference between 1V and AVDD.
- ADC can be used to measure the analog supply voltage, AVDD, without external connections.
- ADC can use AVDD as its reference voltage, allowing ratiometric measurements proportional to the AVDD,
- Selectable ADC conversion start source from on-chip timers, ADC conversion pin, and software write.

14.1 Architecture

Figure 14-1 shows the SAR ADC block diagram. The analog module is composed of a 12-bit SAR ADC and voltage reference blocks. The input to the SAR ADC is fed through a multiplexer section. The input to the multiplexer can be set through software to measure five single-ended or two differential inputs. The ADC can be configured to be in unipolar mode or bipolar mode.

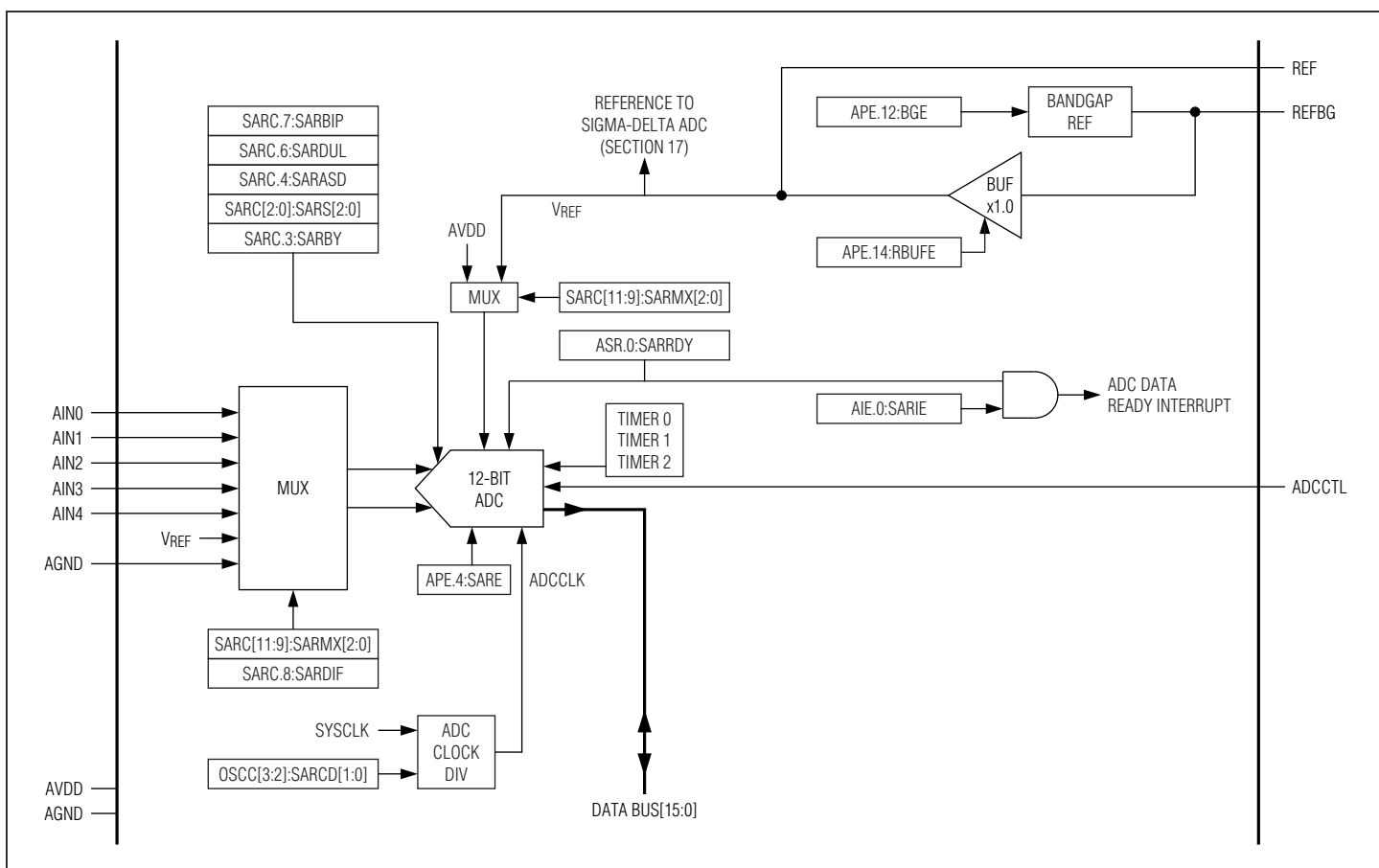


Figure 14-1. SAR ADC Block Diagram

The ADC gets the reference from three sources: external source REF, internal buffered-bandgap reference that provides 2.5V, and AVDD. The internal bandgap and reference buffer can be individually disabled, allowing external reference(s) to drive any of these nodes. All are disabled at power-on to avoid contention if external reference(s) are used. The reference sources are designed to quickly power on and off to minimize power consumption in applications that duty-cycle between run and sleep modes.

14.2 SAR ADC Pins

Table 14-1 shows all the pins in the SAR ADC module.

Table 14-1. SAR ADC Module Pins

NAME	PIN	FUNCTION
AIN0	36	Analog Input 0. This ADC input pin can be used for single-ended measurements relative to AGND or for differential measurement relative to analog input AIN1 (negative side). The active ADC input signal is selectable via the analog multiplexer.
AIN1	37	Analog Input 1. This ADC input pin can be used for single-ended measurements relative to AGND or for differential measurement relative to analog input AIN0 (positive side). The active ADC input signal is selectable via the analog multiplexer.
AIN2	38	Analog Input 2. This ADC input pin can be used for single-ended measurements relative to AGND or for differential measurement relative to analog input AIN3 (negative side). The active ADC input signal is selectable via the analog multiplexer.
AIN3	39	Analog Input 3. This ADC input pin can be used for single-ended measurements relative to AGND or for differential measurement relative to analog input AIN2 (positive side). The active ADC input signal is selectable via the analog multiplexer.
AIN4	40	Analog Input 4. This ADC input pin can be used for single-ended measurements only relative to AGND.
REFBG	35	Internal 2.5V Reference Output. Connect a minimum value of 0.47 μ F bypass capacitor to AGND.
REF	34	ADC Reference Input and Reference Buffer Output. This pin is for the ADC reference input. The buffer connected to the REFBG pin must be disabled to allow the pin to accept an external reference input. Provide a bypass to AGND with a 0.47 μ F capacitor. This pin requires a low ESR, which can be done by using two capacitors in parallel instead of one, e.g., a 1 μ F capacitor in parallel with a 10nF capacitor instead of just a 1 μ F capacitor.
AGND	28, 31, 33	Analog Ground. This pin provides the ground reference for all internal analog circuitry.
AVDD	27, 32	Analog Supply Voltage (+3.3V). This pin provides power to all the analog blocks. Connect all AVDD pins to the same point. Connect to REG3P3 and bypass with a 0.47 μ F capacitor to ground for self-powered operation.
ADCCTL	12	ADC Control Input. Digital GPIO, Timer 0 I/O, and Port 0.3. As ADCCTL this pin is user programmable; rising or falling edge controls the SAR ADC sampling instant and start of conversion. Optionally, the other edge can be used to enable the ADC and begin acquiring prior to sampling. This pin can also be configured as a digital I/O or as a primary timer/PWM I/O.

14.3 SAR ADC Registers

14.3.1 SAR ADC Control Register (SARC)

Register Description: **SAR ADC Control Register**

Register Name: **SARC**

Register Address: **Module 05h, Index 02h**

Bit #	15	14	13	12	11	10	9	8
Name	—	—	—	—	SARMX2	SARMX1	SARMX0	SARDIF
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	SARBIP	SARDUL	SARRSEL	SARASD	SARBY	SARS2	SARS1	SARS0
Reset	0	0	0	0	0	1	1	1
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: SARC is cleared to 0007h on all forms of reset.

Bits 15 to 12: Reserved. Read returns 0.

Bits 11 to 9: Analog Input Mux Control (SARMX[2:0])

For SARDIF = 0:

000 selects AIN0 vs. AGND

001 selects AIN1 vs. AGND

010 selects AIN2 vs. AGND

011 selects AIN3 vs. AGND

100 selects AIN4 vs. AGND

101 measures REF vs. AGND; AVDD is used as the reference by setting SARRSEL (see *Section 14.5.11*)

For SARDIF = 1:

000 selects AIN0 vs. AIN1

001 selects AIN2 vs. AIN3

Bit 8: SAR ADC Differential Input Select (SARDIF). When set to 1, this bit selects differential measurements. When set to 0, this bit selects single-ended measurements relative to AGND.

Bit 7: SAR ADC Bipolar Input Select (SARBIP). When set to 1, this bit selects bipolar mode with differential analog input range of $-V_{REF}/2$ to $+V_{REF}/2$. When set to 0, this bit selects unipolar mode with differential analog input range of 0 to V_{REF} .

Bit 6: Dual-Edge Conversion Start Mode Select (SARDUL). This bit determines the SAR ADC's acquisition timing. When 1, the ADC is operated in dual-edge mode. The rising edge of the ADC conversion control source (specified by SARS) causes the ADC to power-up and begin acquisition, and the falling edge causes the ADC to sample and perform a conversion. When 0, the ADC is operated in single-edge mode. The rising edge of the ADC conversion control source causes the ADC to power-up, acquire, and convert automatically.

Bit 5: SAR ADC Reference Select (SARRSEL). When set to 1, this bit selects AVDD as reference. When set to 0, this bit selects the REF pin voltage as reference.

Bit 4: SAR ADC Auto Shutdown Enable (SARASD). When 1, the SAR ADC shuts down automatically after a conversion is complete. When 0, the SAR ADC is powered up as long as SARE is 1. **Note:** If SAR ADC is in continuous conversion (i.e., SARS = 110), SARASD is ignored.

Bit 3: SAR ADC Start/Busy (SARBY). Setting this bit to 1 starts a conversion if SARS = 111. Reading from this bit reflects the busy status of the SAR ADC. Changing SARBY from 1 to 0 by software is blocked by hardware.

Bits 2 to 0: SAR ADC Conversion Control Source Select (SARS[2:0]). These bits select the source that initiates SAR ADC conversions.

- 000 selects timer 0
- 001 selects timer 1
- 010 selects timer 2
- 011 is reserved
- 100 selects from ADCCTL pin
- 101 selects from ADCCTL pin with inverted data
- 110 selects continuous conversions every 16 clocks
- 111 selects the SARBY bit

14.3.2 Analog Interrupt Enable Register (AIE)

Register Description: **Analog Interrupt Enable Register**
 Register Name: **AIE**
 Register Address: **Module 05h, Index 05h**

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	XTIE	VIBIE	VDBIE	VABIE	CMPIE	LFLIE	LPFIE	SARIE
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: AIE is cleared to 0000h on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0.

Bit 7: Crystal Oscillator Failure (XTIE). See Section 15 for details on this bit.

Bit 6: DVDDIO Brownout Interrupt Enable (VIBIE). See Section 16 for details on this bit.

Bit 5: DVDD Brownout Interrupt Enable (VDBIE). See Section 16 for details on this bit.

Bit 4: AVDD Brownout Interrupt Enable (VABIE). See Section 16 for details on this bit.

Bit 3: Echo Envelope Comparator Interrupt Enable (CMPIE). See Section 17 for details on this bit.

Bit 2: Echo Envelope Lowpass Filter FIFO Full Interrupt Enable (LFLIE). See Section 17 for details on this bit.

Bit 1: Echo Envelope Lowpass Filter Output Data Ready Interrupt Enable (LPFIE). See Section 17 for details on this bit.

Bit 0: SAR ADC Data Ready Interrupt Enable (SARIE). A 1 allows an interrupt request to be generated when ADC completes a conversion and SARRDY bit is 1.

14.3.3 Analog Status Register (ASR)

Register Description: **Analog Status Register**
 Register Name: **ASR**
 Register Address: **Module 05h, Index 08h**

Bit #	15	14	13	12	11	10	9	8
Name	VIOLVL	DVLVL	AVLVL	CMPLVL	—	—	—	XTRDY
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	XTI	VIBI	VDBI	VABI	CMPI	LPFFL	LPFRDY	SARRDY
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: ASR bits 3:0 are cleared to 0h on reset; bits 8:4 clear to 0h on power-on reset.

Bit 15: DVDDIO Brownout Comparator Output Level (VIOLVL). See Section 16 for details on this bit.

Bit 14: DVDD Brownout Comparator Output Level (DVLVL). See Section 16 for details on this bit.

Bit 13: AVDD Brownout Comparator Output Level (AVLVL). See Section 16 for details on this bit.

Bit 12: Echo Envelope Comparator Output Level (CMPLVL). See Section 17 for details on this bit.

Bits 11 to 9: Reserved. Read returns 0.

Bit 8: Crystal Oscillator Ready (XTRDY). See Section 15 for details on this bit.

Bit 7: Crystal Oscillator Failure Interrupt Flag (XTI). See Section 15 for details on this bit.

Bit 6: DVDDIO Brownout Interrupt Flag (VIBI). See Section 16 for details on this bit.

Bit 5: DVDD Brownout Interrupt Flag (VDBI). See Section 16 for details on this bit.

Bit 4: AVDD Brownout Interrupt Flag (VABI). See Section 16 for details on this bit.

Bit 3: Echo Envelope Comparator Interrupt Flag (CMPI). See Section 17 for details on this bit.

Bit 2: Echo Envelope Lowpass Filter FIFO Full Interrupt Flag (LPFFL). See Section 17 for details on this bit.

Bit 1: Echo Envelope Lowpass Filter Output Data Ready Flag (LPFRDY). See Section 17 for details on this bit.

Bit 0: SAR ADC Data Ready Flag (SARRDY). Set to 1 when the SAR ADC completes a conversion.

14.3.4 SAR ADC Output Data Register (SARD)

Register Description: **SAR ADC Output Data Register**
 Register Name: **SARD**
 Register Address: **Module 05h, Index 09h**

Bit #	15	14	13	12	11	10	9	8
Name	—	—	—	—	SARD11	SARD10	SARD9	SARD8
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	SARD7	SARD6	SARD5	SARD4	SARD3	SARD2	SARD1	SARD0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: SARD is cleared to 0000h on all forms of reset.

Bits 15 to 12: Reserved. Read returns 0.

Bits 11 to 0: SAR ADC Output Data 11:0 (SARD[11:0])

14.3.5 Oscillator Control Register (OSCC)

Register Description: **Oscillator Control Register**
 Register Name: **OSCC**
 Register Address: **Module 05h, Index 0Bh**

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	—	—	—	—	SARCD1	SARCD0	XTE	RCE
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read (Note that some clock control bits may have locking mechanisms for write.)

Note: OSCC is cleared to 0000h on power-on reset.

Bits 15 to 4: Reserved. Read returns 0.

Bits 3 and 2: SAR ADC Clock Divider (SARCD[1:0]). Determines the SAR ADC clock frequency, which is divided down from the clock source selected by the XTRC (CKCN.7) bit.

00 divides by 2

01 divides by 4

10 divides by 8

11 divides by 16

Note that the SAR ADC accuracy is not guaranteed for SAR ADC clock frequencies exceeding 2MHz. This divider should be selected with regard to the source clock frequency to ensure this restriction is met.

Bit 1: Crystal Enable (XTE). See *Section 15* for details on this bit.

Bit 0: RC Oscillator Enable (RCE). See *Section 15* for details on this bit.

14.3.6 Analog Power Enable Register (APE)

Register Description: **Analog Power Enable Register**
 Register Name: **APE**
 Register Address: **Module 05h, Index 10h**

Bit #	15	14	13	12	11	10	9	8
Name	—	RBUFE	—	BGE	LRIOPD	LRDPD	LRAPD	VIBE
Reset	0	0	0	0	0	0	0	0
Access	r	rw	r	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	VDPE	VDBE	VABE	SARE	PLLE	MDE	LNAE	BIASE
Reset	1	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: APE is cleared to 0080h on all forms of reset.

Bits 15 and 13: Reserved. Read returns 0.

Bit 14: Reference Buffer Enable (RBUFE). The reference bandgap is enabled when set to 1 and disabled when set to 0.

Bit 12: Bandgap Enable (BGE). The reference bandgap is enabled when set to 1 and disabled when set to 0.

Bit 11: I/O Linear Regulator Power-Down (LRIOPD). See *Section 16* for details on this bit.

Bit 10: Digital Linear Regulator Power-Down (LRDPD). See *Section 16* for details on this bit.

Bit 9: Analog Linear Regulator Power-Down (LRAPD). See *Section 16* for details on this bit.

Bit 8: I/O Voltage Brownout Detection Enable (VIBE). See *Section 16* for details on this bit.

Bit 7: Digital Voltage Reset Enable (VDPE). See *Section 16* for details on this bit.

Bit 6: Digital Voltage Brownout Detection Enable (VDBE). See *Section 16* for details on this bit.

Bit 5: Analog Voltage Brownout Detection Enable (VABE). See *Section 16* for details on this bit.

Bit 4: SAR Enable (SARE). When set to 1, this bit enables SAR ADC. When set to 0, this bit disables SAR ADC and places it in a leakage-only state.

Bit 3: PLL Enable (PLLE). See *Section 17* for details on this bit.

Bit 2: Sigma-Delta Modulator Enable (MDE). See *Section 17* for details on this bit.

Bit 1: LNA Enable (LNAE). See *Section 17* for more information on this bit.

Bit 0: Bias Enable (BIASE). When set to 1, this bit enables current bias generator. When set to 0, this bit disables current bias generator and places it in a leakage-only state. BIASE is automatically set to 1 whenever LNAE, MDE, PLLE, SARE, BGE, or RBUFE are set to 1.

14.4 Voltage References

The voltage reference for the SAR ADC can either be an internal bandgap reference or an external reference. The internal, 2.5V bandgap reference can be activated by setting the BGE bit (APE.12) and the RBUFE bit (APE.14) to 1. The BGE bit enables the bandgap reference, while the RBUFE bit (APE.14) enables the buffer at the output of the bandgap. A bypass capacitor of 0.47 μ F should be placed between REF_{BG} and AGND pin.

To provide the reference from an external source, the bandgap buffer must be turned off by setting RBUFE bit (APE.14) to 0, and an external reference voltage between 1V and AVDD can be connected to the REF pin. A bypass capacitor of 0.47 μ F should be placed between REF and AGND pin.

14.5 Analog-to-Digital Converter (ADC) Port

The MAXQ7667 contains a low-power, high-precision, 12-bit, 125ksps successive approximation ADC and five single-ended or two differential input channel multiplexer.

14.5.1 Single-Ended/Differential Inputs

The MAXQ7667 ADC uses a fully differential SAR conversion technique and an on-chip track-and-hold (T/H) block to convert voltage signals into a 12-bit digital result. Both single-ended and differential configurations are supported using an analog input channel multiplexer that supports five single-ended or two differential channels.

In single-ended mode, the positive analog input (AIN+) for the ADC is connected to the selected input channel and the negative input (AIN-) is connected to AGND. In differential input configuration, analog inputs AIN+ and AIN- are selected from the following pairs: AIN0/AIN1, AIN2/AIN3. The differential input configuration references all input signals to the complementary multiplexer channel input, eliminating common-mode DC offsets and noise. The analog inputs can be configured for either single-ended or differential conversion by writing to the SARDIF (SARC.8) control bit, while analog input channel selection is controlled by the SARMX[2:0] (SARC.11:9) control field in the SARC peripheral register.

14.5.2 True Differential Analog Input T/H

The equivalent input circuit of the MAXQ7667's analog input architecture is shown in Figure 14-2 and Figure 14-3. In track (acquisition) mode, a positive input capacitor is connected to AIN0–AIN4 in single-ended mode or AIN0, AIN2 in differential mode. A negative input capacitor is connected to AGND in single-ended mode or AIN1, AIN3 in differential mode. T/H timing is controlled by the ADC source select (SARS[2:0]) and ADC dual mode select SARDUL (SARC.6). SARS[2:0] selects an ADC conversion start source, which could be one of the timers, ADC conversion pin, or software writes to the ADC start bit. All three conversion start sources support single-edge or dual-edge modes of operation, which is determined by the SARDUL bit. When SARDUL is set to 1, the ADC operates in dual-edge mode. The rising edge of the selected conversion start source causes the ADC to power-up and begin acquisition; the falling edge causes it to sample and perform a conversion. When SARDUL is 0, the ADC operates in single-edge mode. The rising edge controls the entire conversion, i.e., power-up, acquisition, and conversion sequence if the ADC was off; if the ADC was on, the falling edge starts the acquisition and then starts conversion. Once a conversion has been initiated, the T/H enters acquisition mode for the next conversion on the 13th falling edge of ADCCLK, if auto shutdown (SARASD = 0 in the SARC register) is disabled. See *Section 14.5.8: ADC Conversion Start Sources and Timing* for details.

The time required for the T/H to acquire an input signal is determined by how quickly its input capacitance is charged. If the input signal's source impedance is high, the acquisition time lengthens. The acquisition time, t_{ACQ} , is the minimum time needed for the signal to be acquired. It is calculated by the following equation:

$$t_{ACQ} \geq k \times (R_{SOURCE} + R_{IN}) \times C_{IN}$$

where

$$k = 9 \approx \ln(2 \times 2^{12})$$

The constant, k , is the number of RC time constants required so that the voltage on the internal sampling capacitor reaches 12-bit accuracy, i.e., so that the difference between the input voltage and the sampling capacitor voltage is equal to 0.5 LSB.

R_{SOURCE} is the source impedance of the input signal. $R_{IN} = 800\Omega$ is the equivalent differential analog input resistance, and $C_{IN} = 14\text{pF}$ is the equivalent differential analog input capacitance. t_{ACQ} is never less than 1.5 μ s (three ADCCLK periods at 2MHz), and any source impedance less than 4k Ω does not significantly affect the ADC's AC performance. For higher source impedance, a longer acquisition time is required.

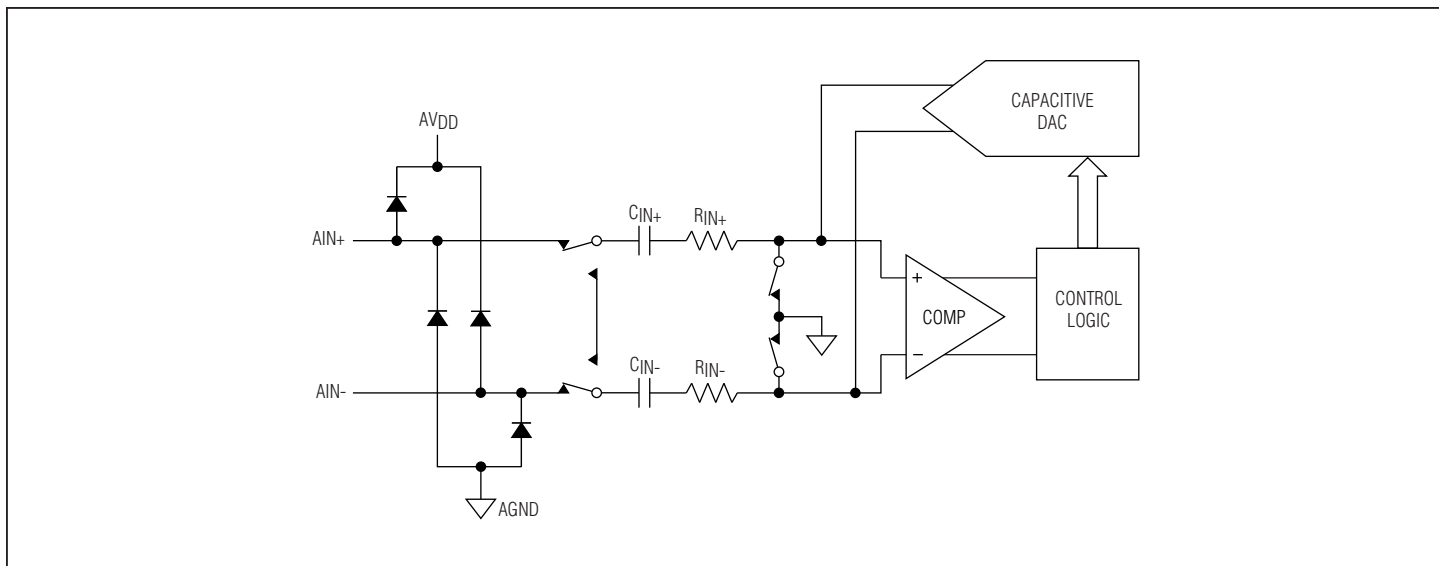


Figure 14-2. Equivalent Input Circuit (Track/Acquisition Mode)

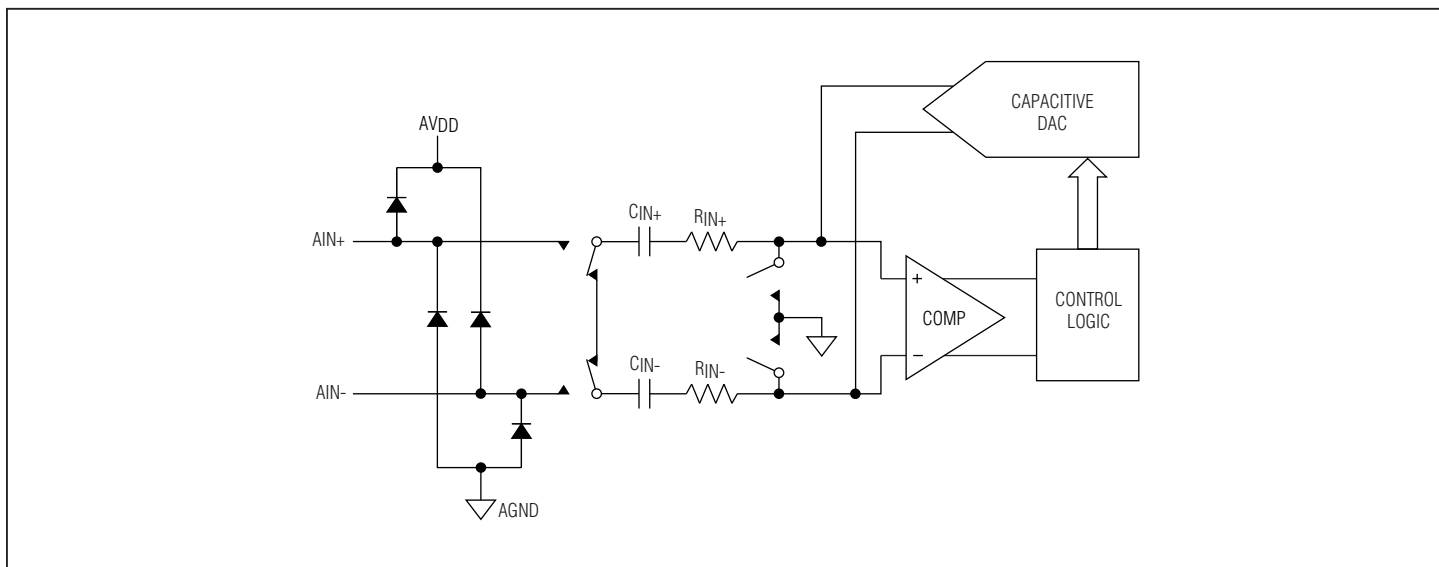


Figure 14-3. Equivalent Input Circuit (Hold/Conversion Mode)

14.5.3 Unipolar/Bipolar

The MAXQ7667 ADC produces a digital output that corresponds to the differential analog input voltage as long as the differential analog inputs are within the specified range. The analog inputs are configured for differential conversion when the SARDIF (SARC.8) control bit is set. When performing differential conversions, the control bit SARBP (SARC.7) selects between unipolar and bipolar operation modes. Unipolar mode sets the differential input range from 0 to REF. A negative differential analog input in unipolar mode causes the digital output code to be zero. Selecting bipolar mode sets the differential input range to $-REF/2$ to $+REF/2$. The digital output code is straight binary in unipolar mode and two's complement in bipolar mode.

In single-ended mode (SARDIF set to 0), the MAXQ7667 ADC analog inputs are internally referenced to AGND. Selecting unipolar mode sets the full-scale input range to 0 to REF. Bipolar mode sets the full-scale input range to 0 to $REF/2$. As above, the digital output code is straight binary in unipolar mode and two's complement in bipolar mode.

14.5.4 Transfer Function

The MAXQ7667 ADC output is straight binary in unipolar mode. Figure 14-4 shows the MAXQ7667 ADC unipolar transfer function. Table 14-2 shows the unipolar relationship between the differential analog input voltage and the digital output code.

Table 14-2. Unipolar Code Table

BINARY DIGITAL OUTPUT CODE D11–D0	HEXADECIMAL EQUIVALENT OF D11–D0	DECIMAL EQUIVALENT OF D11–D0 (CODE ₁₂)	DIFFERENTIAL INPUT VOLTAGE (V) (EXTERNAL, REF = 2.5V)
1111 1111 1111	0xFFF	4095	+2.49939 ± 0.5 LSB
1111 1111 1110	0xFFE	4094	+2.498779 ± 0.5 LSB
1000 0000 0001	0x801	2049	+1.25061 ± 0.5 LSB
1000 0000 0000	0x800	2048	+1.25 ± 0.5 LSB
0111 1111 1111	0x7FF	2047	+1.24939 ± 0.5 LSB
0000 0000 0001	0x001	1	+0.0061 ± 0.5 LSB
0000 0000 0000	0x000	0	+0.000 ± 0.5 LSB

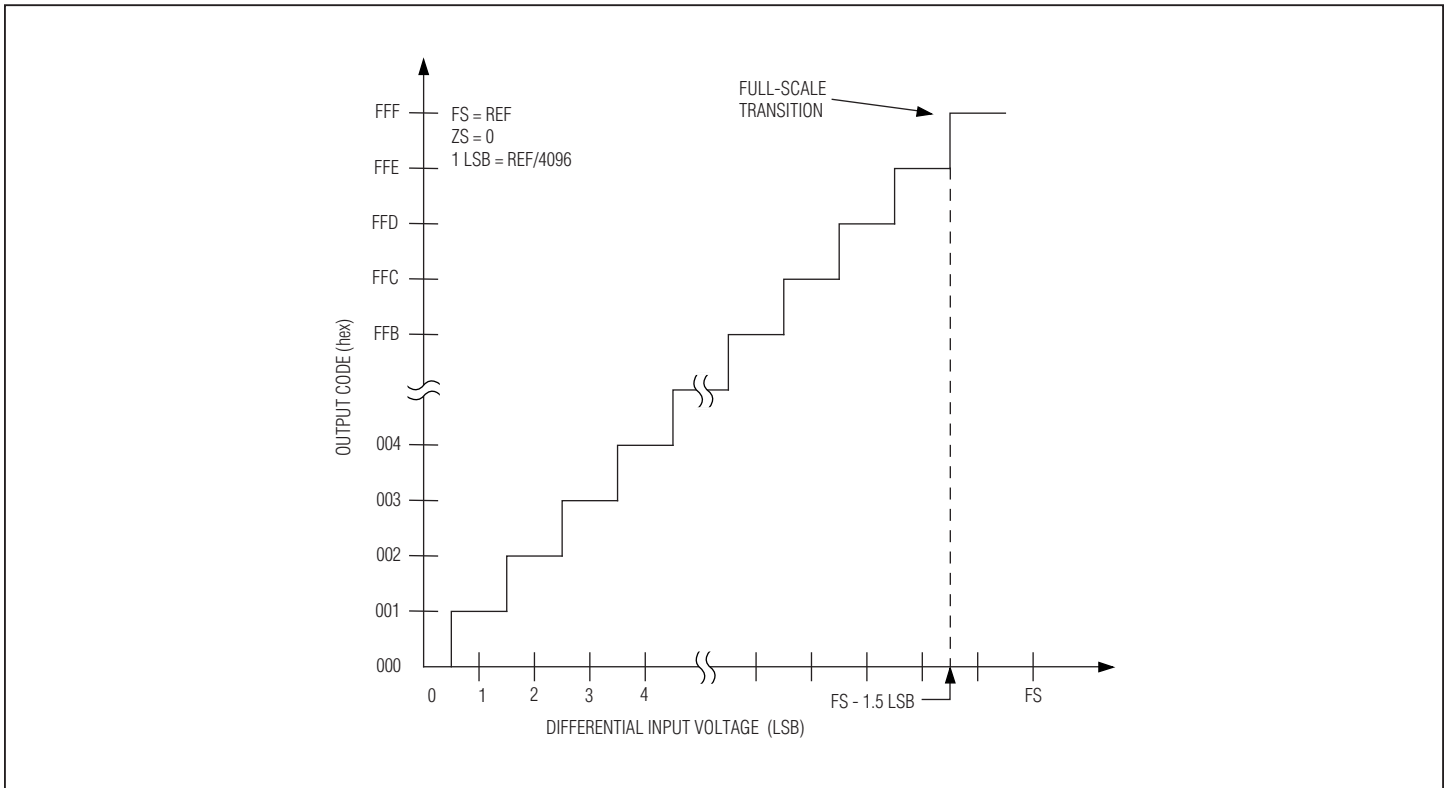


Figure 14-4. Unipolar Transfer Function

The MAXQ7667 ADC output is two's complement in bipolar mode. Figure 14-5 shows the MAXQ7667 ADC bipolar transfer function. Table 14-3 shows the bipolar relationship between the differential analog input voltage and the digital output code. In bipolar mode, the inputs are measured in a truly differential fashion where either input can exceed the other by up to REF/2.

Table 14-3. Bipolar Code Table

BINARY DIGITAL OUTPUT CODE D11-D0	HEXADECIMAL EQUIVALENT OF D11-D0	DECIMAL EQUIVALENT OF D11-D0 (CODE ₁₂)	DIFFERENTIAL INPUT VOLTAGE (V) (EXTERNAL, REF = 2.5V)
0111 1111 1111	0x7FF	+2047	+1.24939 ± 0.5 LSB
0111 1111 1110	0x7FE	+2046	+1.248779 ± 0.5 LSB
0000 0000 0001	0x001	+1	+0.00061 ± 0.5 LSB
0000 0000 0000	0x000	0	0.000 ± 0.5 LSB
1111 1111 1111	0xFFF	-1	-0.00061 ± 0.5 LSB
1000 0000 0001	0x801	-2047	-1.24939 ± 0.5 LSB
1000 0000 0000	0x800	-2048	-1.25 ± 0.5 LSB

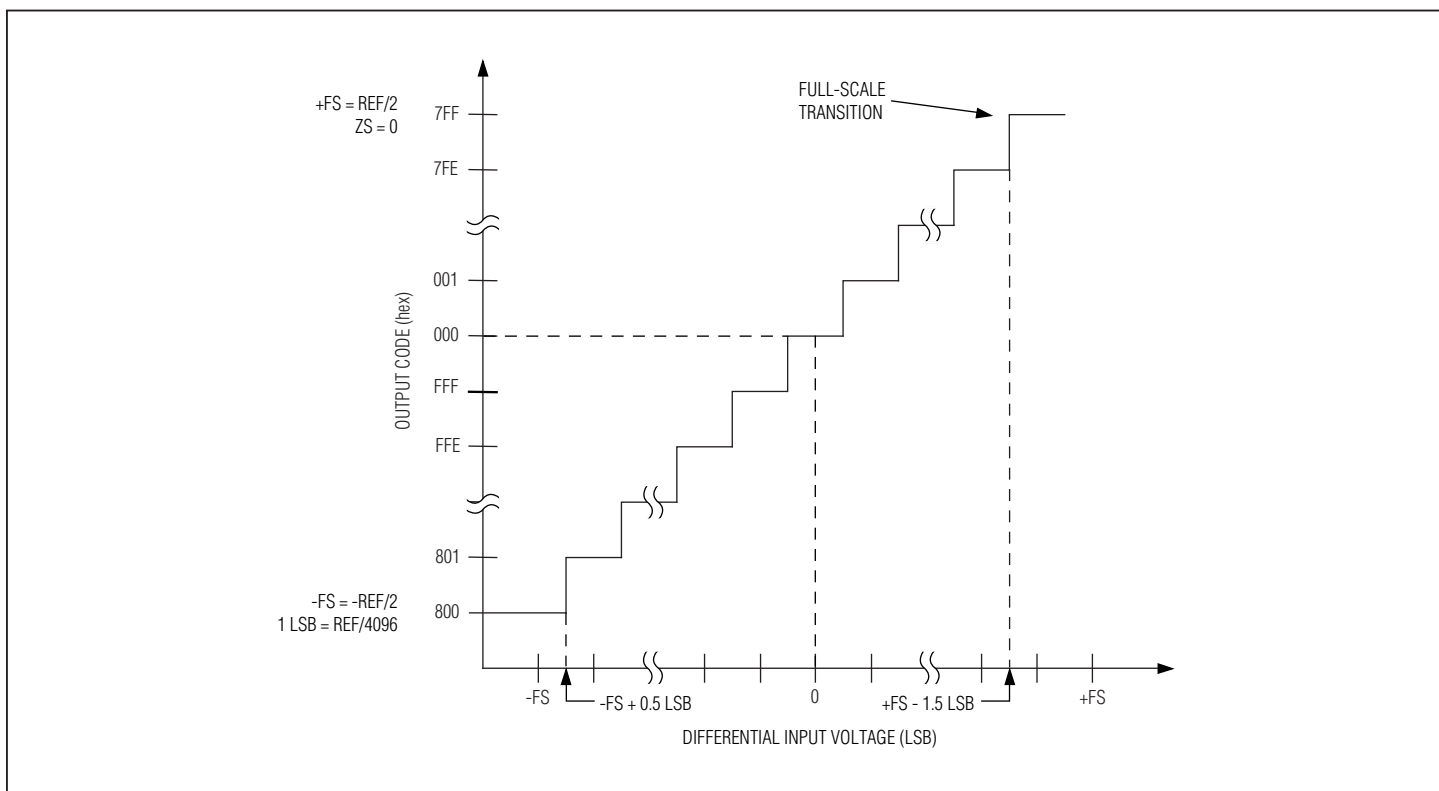


Figure 14-5. Bipolar Transfer Function

14.5.5 Analog Input Protection

Internal ESD protection diodes limit all analog inputs to AVDD and AGND, allowing the inputs to swing from (AGND - 0.3V) to (AVDD + 0.3V) without damage. However, for accurate conversions near full scale, the inputs should not exceed AVDD by more than +50mV or be lower than AGND by -50mV. Input voltages beyond AGND - 0.3V and AVDD + 0.3V forward bias the internal protection diodes. In this situation, limit the forward diode current to 50mA to avoid damaging the MAXQ7667.

The common-mode analog input range or absolute analog input range for the MAXQ7667 is specified from AGND to AVDD. Signals may run outside of that range but will be interpreted as an overrange (ADC data output set to 0xFFFF in unipolar mode and to 0x7FF in bipolar mode) or an underrange condition (ADC data output set to 0x000 in unipolar mode and to 0x800 in bipolar mode). Analog input signals cannot go outside of the ABS max input signal ratings of +0.3V above AVDD or -0.3V below AGND.

Figure 14-6 shows the common-mode or absolute analog input range as specified with a selected REFADC value. In unipolar input configuration (SARBIP = 0 in the SARC register) the output data coding of the ADC is straight binary. It is recommended that the unipolar mode be used with a single-ended input configuration (SARDIF = 0 in the SARC register) where analog input signals are referenced to the AGND pin.

In bipolar input configuration (SARBIP = 1, register bit location SARC.7) the output data coding of the ADC is two's complement. Bipolar mode is commonly used with a differential analog input configuration (SARDIF = 1, register bit location SARC.8) where the analog input signals are referenced to their complementary analog input (AIN+/-) pin. Figure 14-7 shows how a negative ADC value is created. The absolute voltage of each analog input signal must be within the MAXQ7667 supply range so as to satisfy the critical absolute maximum tolerance ratings of the analog inputs, and must also be within the REF range to produce useful information from the ADC.

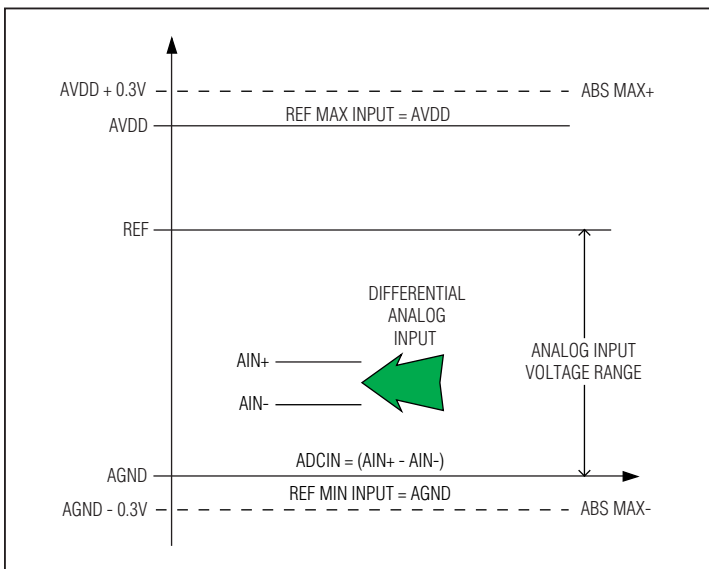


Figure 14-6. Analog Input Range Measuring a Positive Analog Input Value

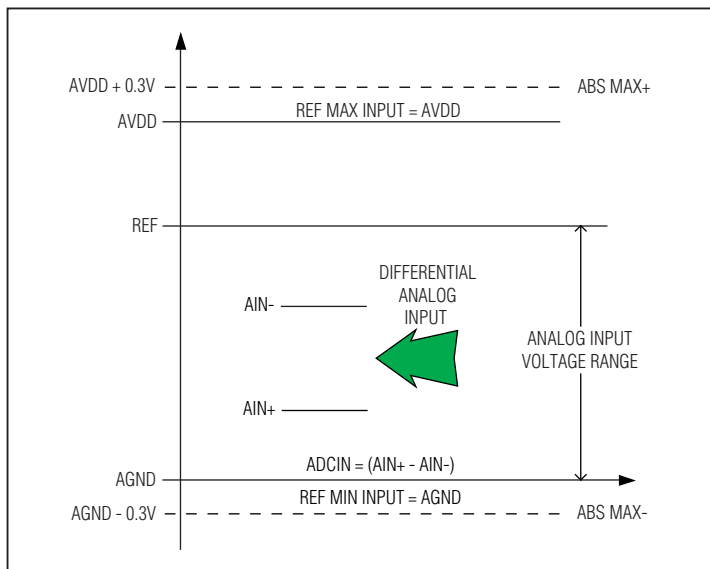


Figure 14-7. Analog Input Range Measuring a Negative Analog Input Value

14.5.6 ADC Clock

The MAXQ7667 ADC clock frequency is controlled by the SARCD[1:0] bits in the OSCC control register and the system clock speed. These bits determine the ADC clock frequency (ADCCLK) that is divided down from the system clock. See *Section 15* for further details on the system clock. The MAXQ7667 ADC uses the divided system clock to clock the multiplexer front-end selection, track and hold acquisition, and each step of the successive approximation conversion.

14.5.7 Auto Shutdown Mode

Power consumption is reduced significantly by placing the MAXQ7667 ADC in auto shutdown mode after a conversion. Auto shutdown is ideal for infrequent data sampling and fast wake-up time applications. Auto shutdown is controlled by the SARASD bit (SARC.4). If the SARASD bit is set, the ADC automatically shuts down when the ADC data ready (SARRDY) bit in the ASR register is set at the end of a conversion. If the SARASD is not set, the ADC returns to acquisition mode after a conversion. Auto shutdown reduces the ADC supply current (refer to the MAXQ7667 data sheet for exact current saving), but there is a power-up (1 μ s) after an auto shutdown.

Note that auto shutdown is different from a full power-down state. The ADC is disabled and fully powered down if the SARE bit in the APE register is cleared. Full power-down reduces ADC supply current (refer to the MAXQ7667 data sheet for exact current saving) and is ideal for infrequent data sampling. The SARE bit is the master control for ADC operation and, unless set, no ADC conversion is possible. From full power-down state (SARE = 0) the ADC requires approximately 1.0 μ s to power-up.

Data in the ADC SFR registers is not lost when the ADC is in auto shutdown or full power-down state.

14.5.8 ADC Conversion Start Sources and Timing

The MAXQ7667 ADC supports three different conversion start sources: timers, ADC convert pin, and software writes. The conversion start source provides the input trigger for the ADC to start acquisition and conversion. The ADC enable bit (SARE) in the APE register must be set so that the ADC block is enabled for operation. The ADC source select field (SARS[2:0]) in the SARC register selects the ADC conversion start source, as shown in Table 14-4.

Table 14-4. ADC Conversion Start Source Selection

ADC SOURCE SELECT (SARS2:SARS0)	ADC CONVERSION START SOURCE	DESCRIPTION
000	Timer 0	<ul style="list-style-type: none"> • Timer output is internally connected to ADC to act as the ADC conversion trigger control. • Configure timer for 8-bit or 16-bit operation.
001	Timer 1	
010	Timer 2	
011	Reserved	
100	ADC Conversion Pin	This configures ADCCTL pin as ADC conversion trigger control input pin.
101	ADC Conversion Pin with Inverted Data	This configures ADCCTL pin as ADC conversion trigger control input pin. ADCCTL pin input is inverted and used as ADC conversion trigger control.
110	Continuous Conversion	Writing 110 to SARS[2:0] triggers conversion. Once started ADC continuously performs a conversion every 16 ADC clock cycles.
111	SARBY (SARC.3) Bit	Write to start/busy bit triggers conversion.

All three conversion start sources support single-edge or dual-edge modes of operation, which is determined by the SARDUL (SARC.6) bit. When SARDUL is set to 1, the ADC operates in dual-edge mode. The rising edge of the selected conversion start source causes the ADC to power-up and begin acquisition (track); the falling edge causes it to sample and perform a conversion (hold). An ADC power-up delay is required only if ADC is in auto shutdown from a prior conversion, otherwise, there is no ADC power-up delay. When SARDUL is 0, the ADC operates in single-edge mode. The rising edge of the selected conversion start source controls the entire conversion, i.e., power-up, acquisition, and conversion if the ADC was off (auto shutdown); if the ADC was on, it stays in acquisition mode until the rising edge and then starts conversion. Table 14-5 summarizes ADC operation in dual- and single-edge modes.

Table 14-5. ADC Dual- and Single-Edge Modes





ADC DUAL-MODE (SARDUL)	ADC CONVERSION SOURCE (SARS[2:0])	ADC CONVERSION TRIGGER	ADC CONVERSION DESCRIPTION
1 (Dual-Edge Mode)	000 (Timer 0) 001 (Timer 1) 010 (Timer 2) 100 (ADCCTL)		<p>Rising Edge of Conversion Source</p> <ul style="list-style-type: none"> • Sets T/H into track (acquisition) mode. • Track duration is under user control. • If ADC is in auto shutdown, a minimum of 2.5µs (1µs for power-up and 1.5µs for acquisition) is required. • If ADC is not in auto shutdown, a minimum of 1.5µs acquisition delay is required. <p>Falling Edge of ADCCTL</p> <ul style="list-style-type: none"> • Sets T/H into hold (conversion) mode. • Then SAR conversion executes (13 ADC clock cycles).
	101 (Inverted ADCCTL)		<p>Falling Edge of ADCCTL</p> <ul style="list-style-type: none"> • Sets T/H into track mode. • Track duration is under user control. • If ADC is in auto shutdown, a minimum of 2.5µs (1µs for power-up and 1.5µs for acquisition) is required. • If ADC is not in auto shutdown, a minimum of 1.5µs acquisition delay is required. <p>Rising Edge of ADCCTL</p> <ul style="list-style-type: none"> • Sets T/H into hold (conversion) mode. • Then SAR conversion executes (13 ADC clock cycles).
	110 (Continuous)	Write 110 to SARS[2:0]	<p>Write 110 to SARS[2:0]</p> <ul style="list-style-type: none"> • If in auto shutdown, logic requires 8 cycles to power up. • Sets T/H into track mode. • ADC control logic provides the required track duration. • T/H placed in hold after 3 clock cycles. • Then SAR conversion executes (13 ADC clock cycles). <p>Conversion continuously repeated every 16 ADC clock cycles.</p>
	111 (Start/Busy Bit) (This mode works exactly as the single-edge mode.)	Write 1 to SARBY (Start/Busy Bit)	<p>Write 1 to SARBY (Start/Busy Bit)</p> <ul style="list-style-type: none"> • Sets T/H into track mode. • ADC control logic provides the required track duration composed of power-up delay (10 cycles), acquisition delay (3 cycles), and settling delay. • If ADC is in auto shutdown, T/H placed in hold after 11 clock cycles. • If ADC is not in auto shutdown, T/H placed in hold immediately. • Then SAR conversion executes (13 ADC clock cycles).
0 (Single-Edge Mode)	000 (Timer 0) 001 (Timer 1) 010 (Timer 2) 100 (ADCCTL)		<p>Falling Edge of Conversion Source</p> <ul style="list-style-type: none"> • Sets T/H into track mode. • ADC control logic provides the required track duration composed of power-up delay (10 cycles), acquisition delay (3 cycles), and settling delay. • If ADC is in auto shutdown, T/H placed in hold after 11 clock cycles. • If ADC is not in auto shutdown, T/H placed in hold immediately. • Then SAR conversion executes (13 ADC clock cycles).
	101 (Inverted ADCCTL)		<p>Rising Edge of ADCCTL</p> <ul style="list-style-type: none"> • Sets T/H into track mode. • ADC control logic provides the required track duration composed of power-up delay (10 cycles), acquisition delay (3 cycles), and settling delay. • If ADC is in auto shutdown, T/H placed in hold after 11 clock cycles. • If ADC is not in auto shutdown, T/H placed in hold immediately. • Then SAR conversion executes (13 ADC clock cycles).

Table 14-5. ADC Dual- and Single-Edge Modes (continued)

ADC DUAL-MODE (SARDUL)	ADC CONVERSION SOURCE (SARS[2:0])	ADC CONVERSION TRIGGER	ADC CONVERSION DESCRIPTION
0 (Single-Edge Mode)	110 (Continuous)	Write 110 to SARS[2:0]	Write 110 to SARS[2:0] <ul style="list-style-type: none"> • If in auto shutdown, logic requires 8 cycles to power up. • Sets T/H into track mode. • ADC control logic provides the required track duration. • T/H placed in hold after 3 clock cycles. • Then SAR conversion executes (13 ADC clock cycles). Conversion continuously repeated every 16 ADC clock cycles.
	111 (Start/Busy bit)	Write 1 to SARBY (Start/Busy Bit)	Write 1 to Start/Busy Bit <ul style="list-style-type: none"> • Sets T/H into track mode. • ADC control logic provides the required track duration composed of power-up delay (10 cycles), acquisition delay (3 cycles), and settling delay. • If ADC is in auto shutdown, T/H placed in hold after 11 clock cycles. • If ADC is not in auto shutdown, T/H placed in hold immediately. • Then SAR conversion executes (13 ADC clock cycles).

Figure 14-8 shows single-edge-controlled ADC conversion timing when the ADC is in auto shutdown state. The power-up and acquisition is triggered by the falling edge of the ADC conversion start source signal ADC_CNVST. ADC_CNVST is an internal signal generated from a combination of all the three conversion start sources previously described.

In single-edged conversions, the ADC control logic provides the necessary power-up, acquisition, and conversion delay.

- a) If ADC is in auto shutdown state, it takes a total of 27 ADC clock cycles before the 12-bit result is available.
- b) If ADC is not in auto shutdown state, it takes a total of 17 ADC clock cycles before the 12-bit result is available.

Figure 14-9 shows single-edge-controlled ADC conversion when the ADC is not in auto shutdown state.

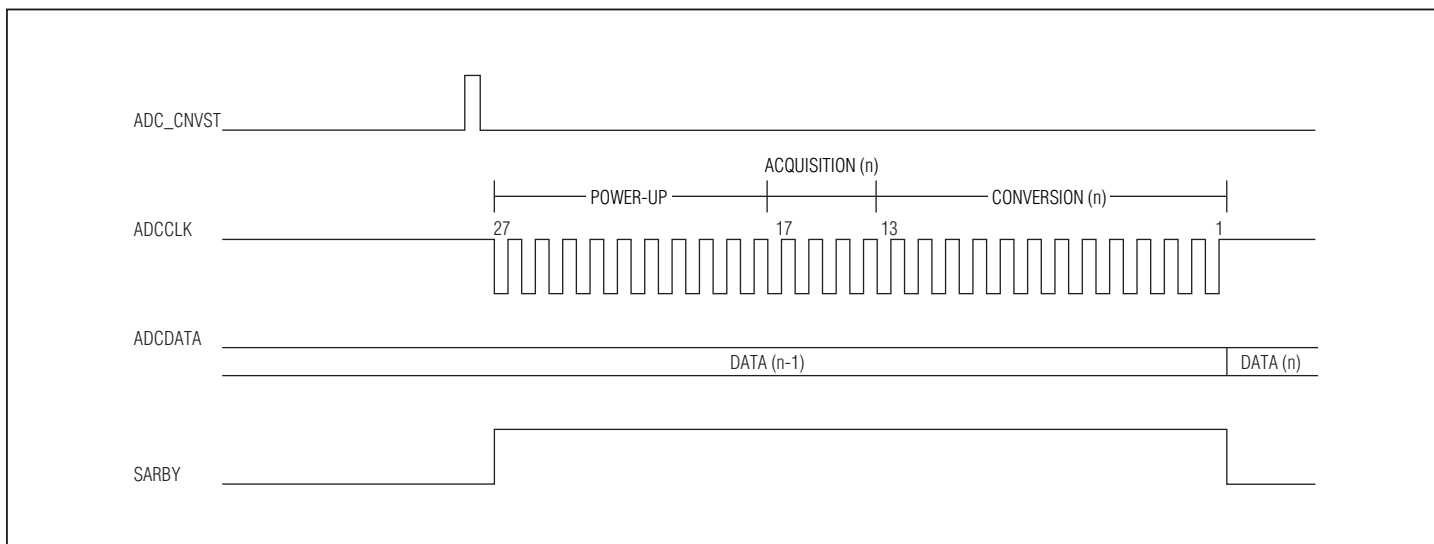


Figure 14-8. Single-Edge ADC Conversion Timing; ADC Previously Off

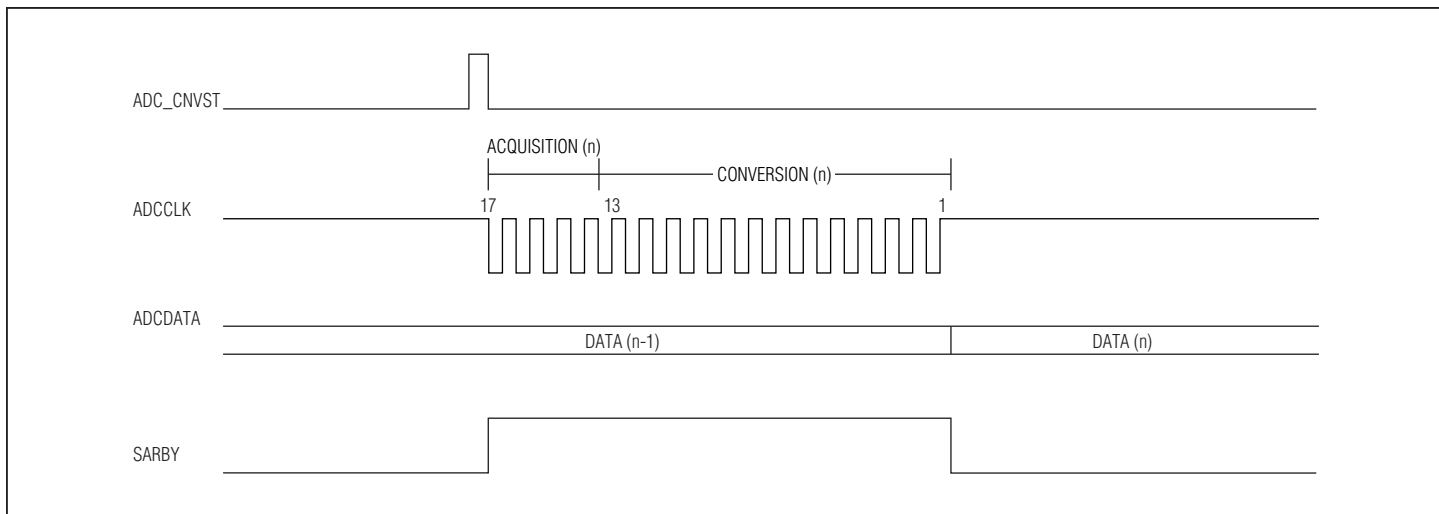


Figure 14-9. Single-Edge ADC Conversion Timing; ADC Previously On

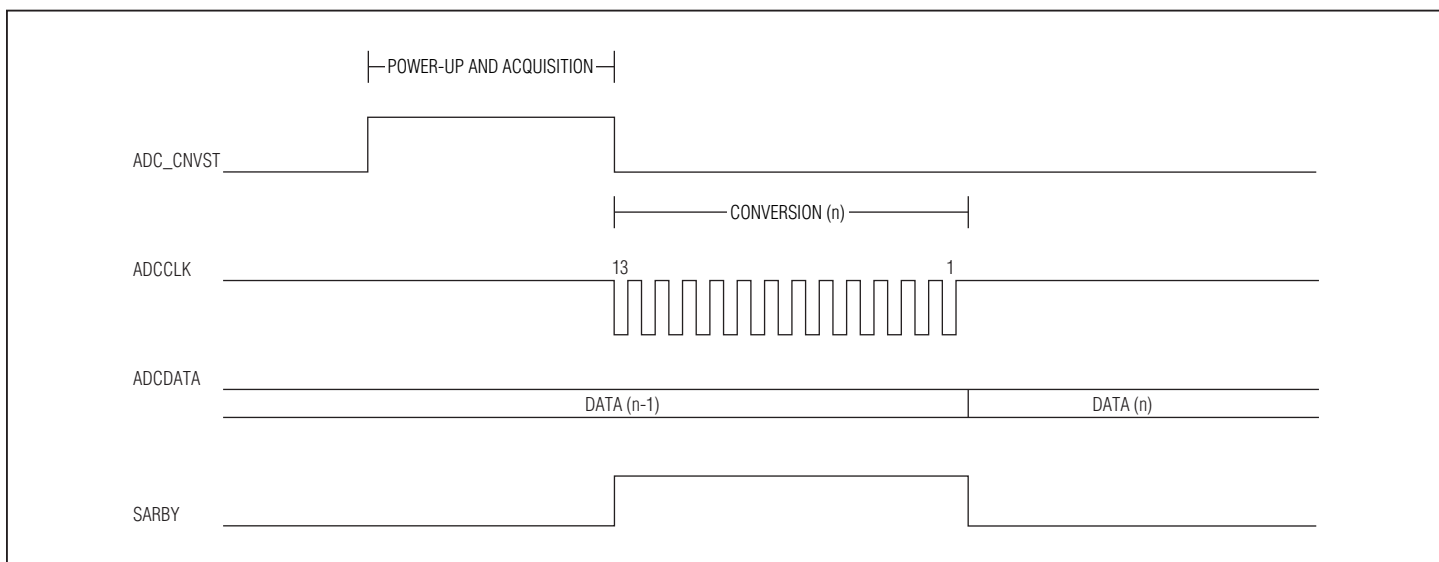


Figure 14-10. Dual-Edge ADC Conversion Timing; ADC Previously Off

In dual-edged conversions, it is up to the user to provide the required power-up and acquisition delay as explained in Table 14-5.

- a) If ADC is in auto shutdown state, a minimum of 2.5 μ s power-up and acquisition delay is required followed by 13 ADC clock cycles, for conversion, before the 12-bit result is available.
- b) If ADC is not in auto shutdown state, a minimum of 1.5 μ s acquisition delay is required followed by 13 ADC clock cycles, for conversion, before the 12-bit result is available.

Figure 14-10 shows dual-edge-controlled ADC conversion when the ADC is in auto shutdown state.

In dual-edge conversion, the power-up and acquisition is triggered by the rising edge of the ADC conversion start source signal ADC_CNVSST (ADC_CNVSST is an internal signal generated from a combination of all the three conversion start sources described earlier in this subsection). At the falling edge, the ADC starts conversion and a 12-bit result is written to the ADC result register in 13 ADC clock cycles. The advantage of dual-edge mode is, depending on the analog input signal's source impedance, the user can provide additional acquisition time if required.

14.5.9 ADC Interrupts

The MAXQ7667 ADC can generate an interrupt when ADC data is ready.

The ADC data ready interrupt is generated when the conversion on a channel is complete and a 12-bit result is written into the SARD register. The SAR ADC data ready (SARRDY) flag in the ASR register (ASR.0) is also set when a conversion is complete. The SARIE bit in the AIE register (AIE.0) must be set for the interrupt to be generated. Otherwise, only the SARRDY status flag is set and the interrupt is not generated.

Note: It is also required that the interrupt global enable bit (IGE) in the Interrupt and Control Register (IC) and the interrupt mask bit 5 (IM5) in the Interrupt Mask Register (IMR) are enabled for the interrupt to occur.

14.5.10 Using the ADC

The flow chart in Figure 14-11 highlights all the steps required for initializing and using the ADC.

14.5.11 Measuring ADC Reference Voltage Using AVDD as Reference

The analog ADC reference voltage can be measured by setting the ADC input multiplexer bits SARMX[2:0] to 101. This setting measures the REF (internal or external) and AVDD should be used as the reference. The AVDD can be selected as the reference by setting the SARRSEL bit (SARC.5).

Under normal circumstances, a 12-bit ADC would normally work as shown by the following equation. (Note: ADC-count is rounded to the nearest integer.)

$$\text{ADC-count (SARD Register)} = 4096 \times \text{input volt/reference voltage}$$

However, by setting the SARMX and the SARRSEL bits as mentioned above, the input volt becomes the REF and the reference voltage becomes the AVDD. Therefore,

$$\text{ADC-count} = 4096 \times \text{REF/AVDD}$$

Hence, the ADC-count from the SARD register is the digitized REF value.

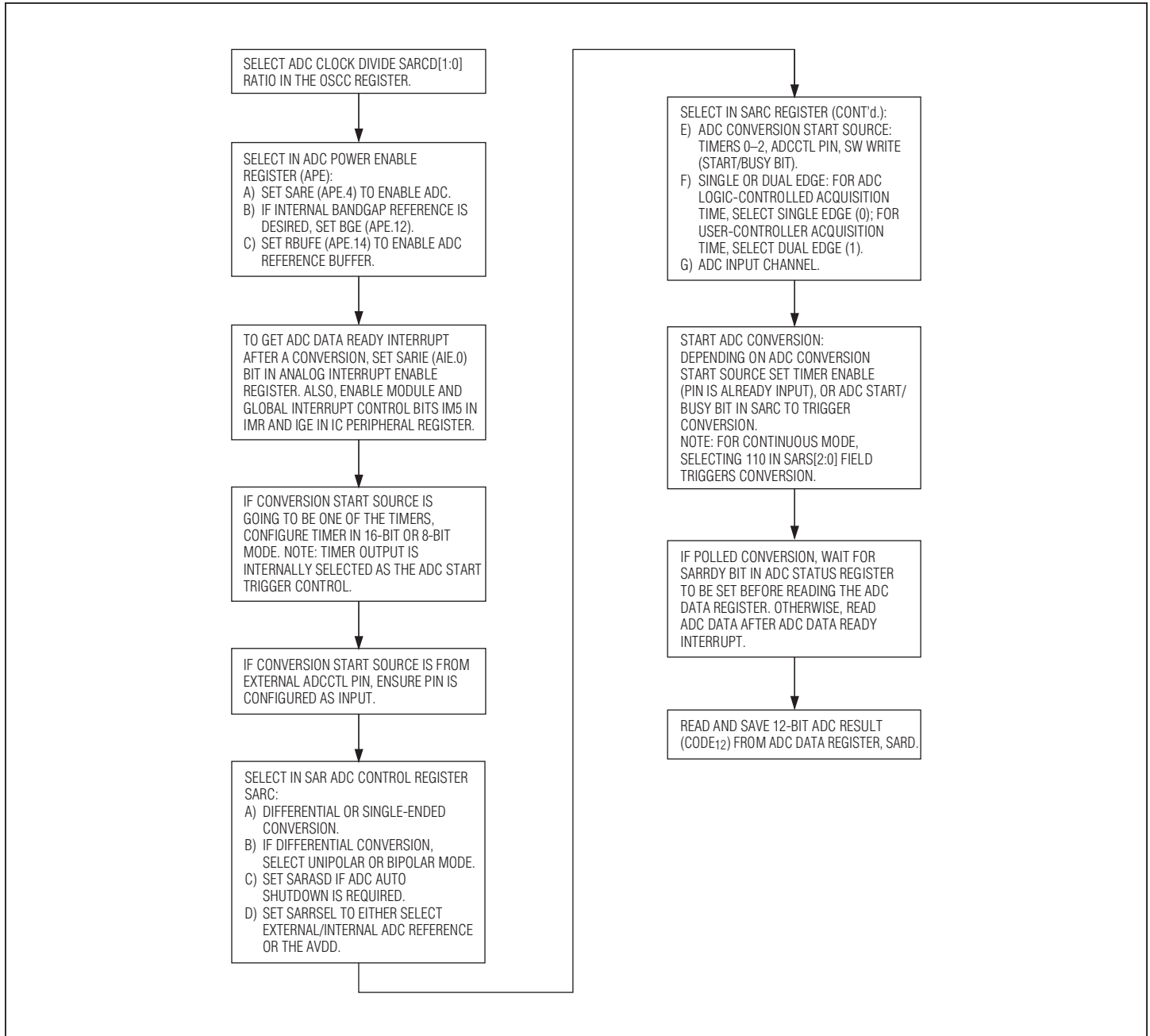


Figure 14-11. Setting Up and Using the ADC Flow Chart

SECTION 15: OSCILLATOR/CLOCK GENERATION

This section contains the following information:

15.1 Architecture	15-2
15.2 Oscillator/Clock Status and Control Registers	15-2
15.2.1 Analog Interrupt Enable Register (AIE)	15-4
15.2.2 Analog Status Register (ASR)	15-5
15.2.3 Oscillator Control Register (OSCC)	15-6
15.2.4 System Clock Control Register (CKCN)	15-6
15.2.5 Watchdog Timer Control Register (WDCN)	15-8
15.2.6 RC Oscillator Trim Register (RCTRM)	15-9
15.3 System Clock Generation	15-9
15.3.1 Internal RC Oscillator	15-9
15.3.1.1 Recalibrating the RC Oscillator	15-10
15.3.2 External Oscillator and Crystal	15-11
15.3.3 Internal System Clock Generation (SYSCLK)	15-11
15.4 External Oscillator Clock/Crystal Failure Switchover	15-12
15.5 Watchdog Timer	15-12
15.6 Oscillator/Clock Power-Saving Management Modes	15-13
15.6.1 Stop Mode	15-13
15.6.1.1 Resuming from Stop Mode	15-13
15.6.2 Switchback Mode	15-14

LIST OF FIGURES

Figure 15-1. Oscillator/Clock Module Block Diagram	15-3
Figure 15-2. Watchdog Timer Function	15-12

LIST OF TABLES

Table 15-1. Watchdog Timeout Settings	15-8
Table 15-2. Divide Ratio of the System Clock from a Clock Source	15-11
Table 15-3. Watchdog Timeout Values (in Number of Source Clock Cycles)	15-13

SECTION 15: OSCILLATOR/CLOCK GENERATION

The MAXQ7667 oscillator/clock generation module is the clock generator that supplies the system clock for the microcontroller core and all the peripheral modules.

The oscillator is designed to allow flexibility for selecting a timing source for the MAXQ7667 microcontroller. The MAXQ7667 clock source can be derived from the following:

- Internal RC oscillator with a maximum of 16MHz frequency (factory default setting is 13.5MHz; no external crystal or oscillator source required)
- Internal high-frequency oscillator driving an external 2.0MHz to 16.0MHz crystal or ceramic resonator
- External high-frequency 2.0MHz to 16.0MHz oscillator input
- External oscillator/crystal source failure detection and automatic switchover to the internal RC oscillator
- Crystal warmup timer to assure that the internal digital core voltage (DVDD) has reached appropriate level
- Watchdog timer
- Low-power stop state

15.1 Architecture

The MAXQ7667 is designed to operate from 2MHz up to 16MHz of external high-frequency source. Because of its RISC-based design, the MAXQ7667 executes most instructions in a single system clock (SYSCLK) period. All functional modules (SPI, timer/counter, SAR ADC, etc.) are synchronized to this single system clock. This system clock is derived from one of the following possible sources:

- Internal RC oscillator running at a maximum of 16MHz with a default value of approximately 13.5MHz. This frequency could be trimmed by the user, but it is subject to minor variation due to voltage and temperature.
- External high-frequency oscillator clock
- External high-frequency crystal (with an internal oscillator)

15.2 Oscillator/Clock Status and Control Registers

Six registers control the oscillator/clock configuration, report status, and control operation in the MAXQ7667. Three of the registers are found in the peripheral register bank in Module 05h, indexes 05h, 08h, 0Bh; two registers are found in the system register bank in Module 08h, indexes 0Eh and 0Fh; and one register is found in Module 01h, index 17h. The registers are:

Peripheral Registers

- Analog Interrupt Enable Register (AIE): Module 05h, Index 05h
- Analog Status Register (ASR): Module 05h, Index 08h
- Oscillator Control Register (OSCC): Module 05h, Index 0Bh

System Registers

- System Clock Control Register (CKCN): Module 08h, Index 0Eh
- Watchdog Timer Control Register (WDCN): Module 08h, Index 0Fh
- RC Oscillator Trim Register (RCTRM): Module 01h, Index 17h

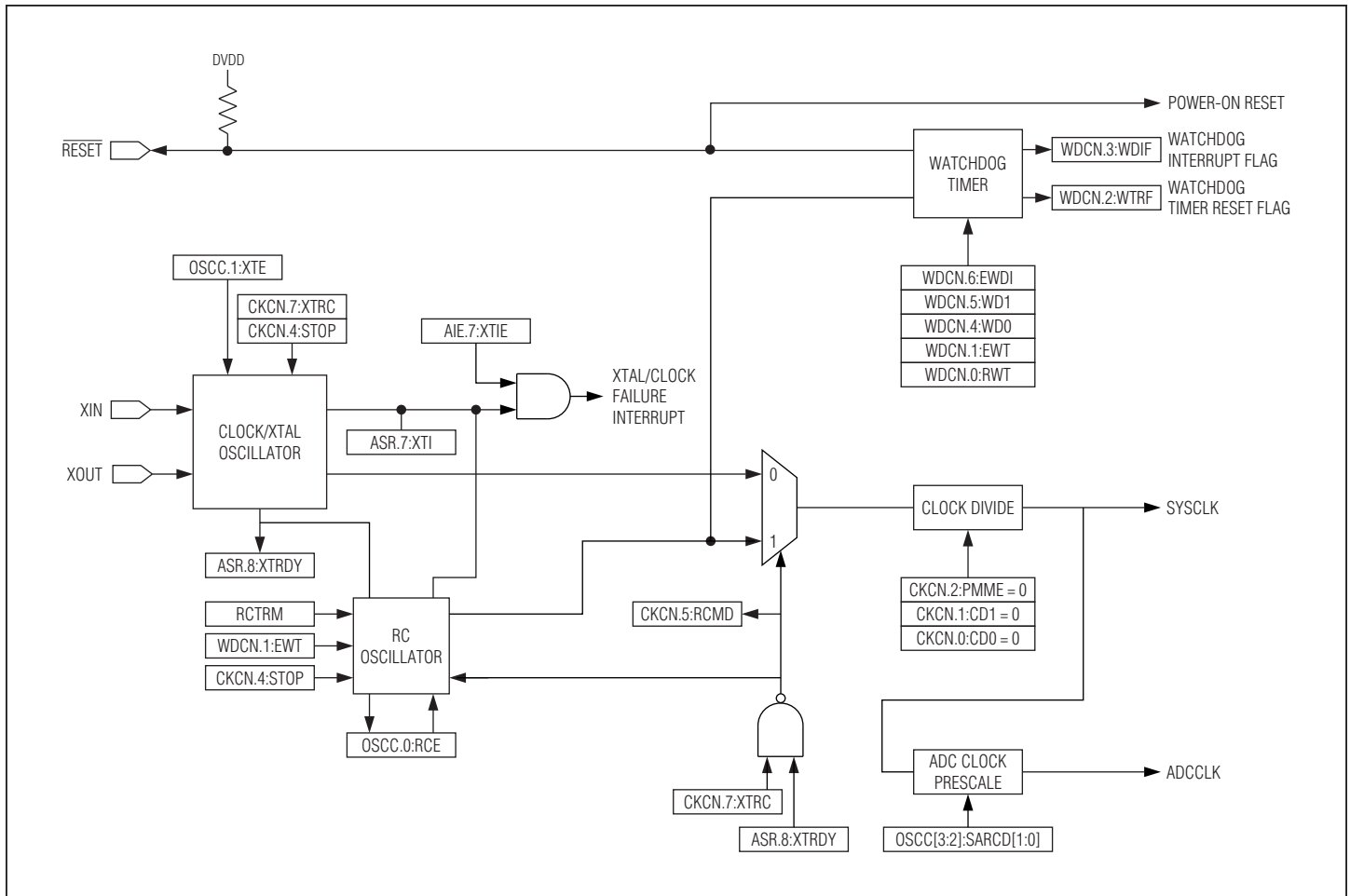


Figure 15-1. Oscillator/Clock Module Block Diagram

15.2.1 Analog Interrupt Enable Register (AIE)

Register Description: **Analog Interrupt Enable Register**
 Register Name: **AIE**
 Register Address: **Module 05h, Index 05h**

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	XTIE	VIBIE	VDBIE	VABIE	CMPIE	LFLIE	LPFIE	SARIE
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: AIE is cleared to 0000h on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0.

Bit 7: Crystal Oscillator Failure (XTIE). Set to 1 to enable the interrupt when the crystal oscillator failure interrupt flag goes (XTI) high, indicating a high clock source failure. Setting this bit to 0 disables the interrupt.

Bit 6: DVDDIO Brownout Interrupt Enable (VIBIE). See *Section 16* for details on this bit.

Bit 5: DVDD Brownout Interrupt Enable (VDBIE). See *Section 16* for details on this bit.

Bit 4: AVDD Brownout Interrupt Enable (VABIE). See *Section 16* for details on this bit.

Bit 3: Echo Envelope Comparator Interrupt Enable (CMPIE). See *Section 17* for details on this bit.

Bit 2: Echo Envelope Lowpass Filter FIFO Full Interrupt Enable (LFLIE). See *Section 17* for details on this bit.

Bit 1: Echo Envelope Lowpass Filter Output Data Ready Interrupt Enable (LPFIE). See *Section 17* for details on this bit.

Bit 0: SAR ADC Data Ready Interrupt Enable (SARIE). See *Section 14* for details on this bit.

15.2.2 Analog Status Register (ASR)

Register Description: **Analog Status Register**
 Register Name: **ASR**
 Register Address: **Module 05h, Index 08h**

Bit #	15	14	13	12	11	10	9	8
Name	VIOLVL	DVLVL	AVLVL	CMPLVL	—	—	—	XTRDY
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	XTI	VIBI	VDBI	VABI	CMPI	LPFFL	LPFRDY	SARRDY
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: ASR bits 3:0 are cleared to 0h on reset; bits 8:4 clear to 0h on power-on reset.

Bit 15: DVDDIO Brownout Comparator Output Level (VIOLVL). See Section 16 for details on this bit.

Bit 14: DVDD Brownout Comparator Output Level (DVLVL). See Section 16 for details on this bit.

Bit 13: AVDD Brownout Comparator Output Level (AVLVL). See Section 16 for details on this bit.

Bit 12: Echo Envelope Comparator Output Level (CMPLVL). See Section 17 for details on this bit.

Bits 11 to 9: Reserved. Read returns 0.

Bit 8: Crystal Oscillator Ready (XTRDY). This bit is set to 1 after the crystal oscillator has warmed up and stabilized. It is cleared when the crystal oscillator fails or is shut down.

Bit 7: Crystal Oscillator Failure Interrupt Flag (XTI). This bit is set to 1 if the previously stable clock source (XTRDY = 1) is sourced as the system clock (XTRC (CKCN.7) = 1) and is detected failing (XTRDY = 0). This condition causes the clock to switch over by forcing RCE = 1 and XTRC (CKCN.7) = 0.

Bit 6: DVDDIO Brownout Interrupt Flag (VIBI). See Section 16 for details on this bit.

Bit 5: DVDD Brownout Interrupt Flag (VDBI). See Section 16 for details on this bit.

Bit 4: AVDD Brownout Interrupt Flag (VABI). See Section 16 for details on this bit.

Bit 3: Echo Envelope Comparator Interrupt Flag (CMPI). See Section 17 for details on this bit.

Bit 2: Echo Envelope Lowpass Filter FIFO Full Interrupt Flag (LPFFL). See Section 17 for details on this bit.

Bit 1: Echo Envelope Lowpass Filter Output Data Ready Flag (LPFRDY). See Section 17 for details on this bit.

Bit 0: SAR ADC Data Ready Flag (SARRDY). See Section 14 for details on this bit.

15.2.3 Oscillator Control Register (OSCC)

Register Description: **Oscillator Control Register**
 Register Name: **OSCC**
 Register Address: **Module 05h, Index 0Bh**

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	—	—	—	—	SARCD1	SARCD0	XTE	RCE
Reset	0	0	0	0	0	0	0	1
Access	r	r	r	r	r	r	r	r

r = read. (Note that some clock control bits may have locking mechanisms for write.)

Note: OSCC is cleared to 0001h on reset.

Bits 15 to 4: Reserved. Read returns 0.

Bits 3 and 2: SAR ADC Clock Divider (SARCD[1:0]). See Section 14 for details on this bit.

Bit 1: Crystal Enable (XTE). Forces the XTAL oscillator to be enabled. When this bit is set to 0, the XTAL oscillator powers on and off as required by the clock selection XTRC (CKCN.7). Setting this bit forces the XTAL oscillator to remain on even when it is not being used. By setting this bit, the XTAL remains warm, and source clock switches via XTRC (CKCN.7) will not require a crystal warmup time.

Bit 0: RC Oscillator Enable (RCE). Forces the RC oscillator to be enabled. When this bit is set to 0, the RC oscillator powers on and off as required by the clock selection XTRC (CKCN.7), use of the watchdog, or flash program/erase cycles. Setting this bit to 1 forces the RC to remain on at all times.

15.2.4 System Clock Control Register (CKCN)

Register Description: **System Clock Control Register**
 Register Name: **CKCN**
 Register Address: **Module 08h, Index 0Eh**

Bit #	7	6	5	4	3	2	1	0
Name	XTRC	—	RCMD	STOP	SWB	PMME	CD1	CD0
Reset	0	0	0	0	0	0	0	0
Access	rw	r	r	rw	rw	rw	rw	rw

r = read, w = write (There is a locking mechanism for the PMME, CD1, and CD0 bits when changing their bit values.)

Note: Bits 5:0 are cleared to 0 on all forms of reset. Bits 6 and 7 are cleared to 0 by a power-on reset only.

Bit 7: External Crystal Select (XTRC). This bit selects the external oscillator/crystal or internal RC oscillator as the desired clock source. The XTRC bit will be inverse of RCMD except during the crystal warmup period when resuming from the stop mode through the RC oscillator. This bit is cleared to 0 after a power-on/brownout reset, which selects the internal RC oscillator as the clock source; otherwise, it is unchanged by other forms of reset. Changing the XTRC bit from 0 to 1 causes the system clock source to swap from RC to crystal oscillator. This change occurs automatically within a few clock cycles if sufficient crystal warmup time has elapsed since the XTE bit of the OSCC register was set. If the crystal has not finished warming up when XTRC is set to 1, the crystal oscillator continues to warm up and the clock source swaps to crystal when this is complete. The XTRDY bit of the ASR register indicates when the crystal oscillator circuit is ready (XTRDY = 1), and the clock source can swap from RC oscillator to crystal.

Changing XTRC from 1 to 0 selects the internal RC oscillator as the system clock source. If the RC oscillator is already turned on, then the switch to the RC clock is instantaneous. If the RC oscillator is turned off, then it takes four RC clock periods to turn on the oscillator and make the switch.

Bit 6: Reserved. Read returns 0.

Bit 5: RC Oscillator Mode (RCMD). This read-only bit reflects the selection of clock source. RCMD = 1 indicates the RC oscillator is providing the system clock. RCMD = 0 indicates the external crystal is providing the system clock. Note that although RCMD is cleared to 0 on all forms of reset, the RC clock is the default clock source so RCMD will normally be set to 1 shortly after the design starts to clock.

Bit 4: Stop Mode Select (STOP). Setting this bit to 1 stops program execution and commences low-power operation. This bit is cleared by a reset or any of the enabled external interrupts. Setting and resetting the STOP bit does not change the system clock-divide ratio.

Bit 3: Switchback Enable (SWB). When set to 1, SWB allows mask-enabled external interrupts as well as enabled serial port receive functions to reset the PMME bit from 1 to 0, and allows the processor to switch back to the original system frequency as selected by the CD1 and CD0 bits. When SWB is cleared to 0, switchback mode is disabled.

Conditions that trigger switchback include the following:

- The detection of a selected edge transaction on any of the external interrupts when the respective pin has been programmed and enabled to issue an interrupt. Note that the switchback interrupt relationship requires that the respective external interrupt source be allowed to actually generate an interrupt, before the switchback will actually occur.
- Activity on the SPI interface (when enabled). The switchback is independent of the SPI interrupt relationship.
- Activity on the UART interface (when enabled). The switchback is independent of the UART interrupt relationship.
- Entry into active debug mode either by a breakpoint match or issuance of the debug command.

Bit 2: Power Management Mode Enable (PMME). The PMME and CD[1:0] control bits select the divide ratio of the system clock from a clock source.

PMME	CD1	CD0	CLOCK-DIVIDE RATIO
0	0	0	Divide by 1 (default)
0	0	1	Divide by 2 (unavailable)
0	1	0	Divide by 4 (unavailable)
0	1	1	Divide by 8 (unavailable)
1	0	0	Divide by 256, Switchback to Divide by 1 (unavailable)
1	0	1	Divide by 256, Switchback to Divide by 2 (unavailable)
1	1	0	Divide by 256, Switchback to Divide by 4 (unavailable)
1	1	1	Divide by 256, Switchback to Divide by 8 (unavailable)

Bits 1 and 0: Clock Divide Control Bits 1 and 0 (CD[1:0])

15.2.5 Watchdog Timer Control Register (WDCN)

Register Description: **Watchdog Timer Control Register**
 Register Name: **WDCN**
 Register Address: **Module 08h, Index 0Fh**

Bit #	7	6	5	4	3	2	1	0
Name	POR	EWDI	WD1	WD0	WDIF	WTRF	EWT	RWT
Reset	0	0	0	0	0	0	0	0
Access	r	rw	rw	rw	r	r	rw	rw

r = read, w = write

Note 1: The watchdog timer always uses the RC oscillator as the clock source.

Note 2: Bits 5, 4, 3, and 0 are cleared to 0 on all forms of reset. See description for other bits.

Bit 7: Power-On Reset Flag (POR). See Section 16 for details on this bit.

Bit 6: Watchdog Interrupt Enable (EWDI). Setting this bit to 1 enables interrupt requests generated by the watchdog timer. Clearing this bit to 0 disables the interrupt requests. This bit is cleared following a power-on reset and is unaffected by all other resets.

Bits 5 and 4: Watchdog Timer Mode Select Bit 1 and 0 (WD[1:0]). These bits are used to provide a user selection of watchdog timer interrupt periods that determine the watchdog timer interrupt timeout when the watchdog timer is enabled. All watchdog timer reset timeouts follow the programmed interrupt timeouts by 512 times the clock-divide ratio oscillator cycles. Table 15-1 summarizes the watchdog timer mode select bits settings and the timeout values. Changing the WD[1:0] bit settings will reset the watchdog timer unless the 512 RC clock reset counter has already started, in which case, changing the WD[1:0] bits will not affect the watchdog timer or reset counter.

Bit 3: Watchdog Interrupt Flag (WDIF). This bit is set to 1 by a watchdog timeout, which indicates that a watchdog timer event has occurred if EWT and/or EWDI are set. When the WDIF is set, EWT and EWDI determine the action to be taken. Setting this bit from 0 to 1 also activates the reset counter for the watchdog-reset timeout, which allows 512 RC cycles for the system to reset the watchdog timer via the RWT bit. Setting this bit in software generates a watchdog interrupt if enabled and triggers the reset counter. This bit must be cleared in software before exiting the interrupt service routine or another interrupt will be generated. The reset counter must be cleared by RWT once started. See Table 15-1.

Table 15-1. Watchdog Timeout Settings

EWT	EWDI	WDIF	ACTIONS
X	X	0	No interrupt has occurred.
0	0	X	Watchdog disable, clock is gated off.
0	1	1	Watchdog interrupt has occurred.
1	0	1	No interrupt has been generated. Watchdog reset occurs in 512 RC clock cycles if RWT is not set.
1	1	1	Watchdog interrupt has occurred. Watchdog reset occurs in 512 RC clock cycles if RWT is not set.

Bit 2: Watchdog Timer Reset Flag (WTRF). When set, this bit indicates that a watchdog timer reset has occurred. It is typically interrogated to determine if a reset was caused by the watchdog timer. It is cleared by power-on reset but otherwise must be cleared by software before the next reset of any kind to allow software to work correctly. Setting this bit by software does not generate a watchdog timer reset. If the EWT bit is cleared, the watchdog timer has no effect on this bit.

Bit 1: Enable Watchdog Timer Reset (EWT). Setting this bit to 1 enables the watchdog timer to reset the device; clearing this bit to 0 disables the watchdog timer reset. It has no effect on the timer itself and its ability to generate a watchdog interrupt. This bit is cleared following a power-on reset and unaffected by all other resets.

Bit 0: Reset Watchdog Timer (RWT). Setting this bit resets the watchdog timer count. This bit must be set before the watchdog timer expires, or a watchdog timer reset and/or interrupt will be generated if enabled. The timeout period is defined by WD1 and WD0. This bit is always 0 when read.

15.2.6 RC Oscillator Trim Register (RCTRM)

Register Description: **RC Oscillator Trim Register**
 Register Name: **RCTRM**
 Register Address: **Module 01h, Index 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	RCTRM7	RCTRM6	RCTRM5	RCTRM4	RCTRM3	RCTRM2	RCTRM1	RCTRM0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read (Writeable only when TME = 1).

Note: RCTRM is cleared to 00h on power-on reset, then initialized to a value stored within the flash information block.

Bits 15 to 8: Reserved

Bits 7 to 0: Trim Codes for the BIAS Block (RCTRM[7:0])

15.3 System Clock Generation

15.3.1 Internal RC Oscillator

The internal RC oscillator is the default system clock source for the MAXQ7667, since the XTRC defaults to 0 on a power-on/brownout reset. No external clock or crystal is required for the RC oscillator.

The RC oscillator provides a low-cost system solution that eliminates the need for an external high-frequency clock source for high-speed operation.

The power-on/brownout reset circuitry holds the device in reset at power-up, while the DVDD supply crosses above the reset threshold voltage, and the internal free-running RC oscillator starts running at a default value of approximately 13.5MHz, and a 16-bit power-up timer is enabled. After the power-on/brownout circuitry releases the reset, the microcontroller jumps to the program in the utility ROM at location 8000h.

The RCMD (CKCN.5) bit set to 1 indicates that the RC oscillator is providing the system clock. Since the default clock source at power-up is the RC clock, RCMD is set to 1. (RCMD = 0 indicates that it is the external crystal or the clock providing the system clock).

15.3.1.1 Recalibrating the RC Oscillator

The MAXQ7667 has an internal RC oscillator that is factory calibrated to approximately 13.5MHz. Although the oscillator is factory calibrated, its frequency drifts with temperature and supply voltage. If precision timing is required, the MAXQ7667's precision crystal-controlled oscillator must be used, or the RC oscillator must be recalibrated to account for any frequency drift.

The frequency of the RC oscillator is controlled by the 9-bit, two's complement value stored in the RCTRM register. This register allows the oscillator frequency to be changed by approximately $\pm 25\%$, with a resolution of 0.2% ($1/512 = 0.2\%$). The large adjustment range is needed to accommodate process variations that affect the oscillator frequency. As part of the manufacturing process, the trim value needed to set the RC oscillator to 16MHz is determined and stored in nonvolatile memory. During the power-on-reset process, this value is read from nonvolatile memory and written to the RCTRM register. If an accurate external time reference is available, it is possible for the application code to recalibrate the RC oscillator by changing the value of the RCTRM register and storing it in the flash somewhere. During power-up, the factory default is loaded in the RCTRM register; therefore, it is up to the user application to load the new value during power-up in the RCTRM register. A precisely tuned RC oscillator can eliminate the need for a precision resonator or crystal.

The first step in recalibration is to determine the amount of frequency error in the RC oscillator. This is done by using the MAXQ7667 to measure the duration of an external event that has an accurately known duration. The external event with a known time duration can take several forms. It can be a single pulse applied to a timer control pin, or it can be two commands on an I/O port with a fixed interval between them. It could even come from timestamped commands on an I/O port, which allows a time interval to be accurately calculated. Regardless of the method used, the goal is the same: to determine the amount of error in the RC clock.

When measuring the reference time, there is a nominal number of system clock cycles that would occur during the reference time if the system clock were operating at exactly 16MHz. Let C_N equal this nominal number of system clock cycles, and let C_A equal the number of system clock cycles that are actually measured during the reference time. The percent error in the oscillator frequency OE can then be calculated by using Equation 1.

Equation 1:
$$OE = [(C_A/C_N) - 1] \times 100$$

Since the adjustment resolution is 0.2%, the value of RCTRM needs to change by five counts for every percent error in the oscillator frequency.

$$\Delta RCTRM = -5 \times OE$$

Because the initial RCTRM value varies significantly from part to part, it is important that new RCTRM values are always based on the existing RCTRM value and the calculated change. All these concepts and equations are brought together in Equation 2.

Equation 2:
$$\text{newRCTRM} = \text{oldRCTRM} - [(C_A/C_N) - 1] \times 500$$

The new value of RCTRM remains in effect until there is a power-on reset. At that time the original factory calibration value will be loaded into RCTRM. RCTRM is not changed by a software reset.

Recalibration is needed whenever the voltage and temperature conditions change enough to cause a "significant" frequency error or timing error. What is considered "significant" must be decided by the individual user, as this varies depending on the application.

RC Oscillator Recalibration Example:

Assume that a reference pulse of exactly 25ms is measured using a timer with the prescaler set to 16. If the RC oscillator were running at exactly 16MHz, then each clock cycle would be 0.0625 μ s long. Because the prescaler on the timer is set to 16, each count of the timer will be equal to 16 system clocks or exactly 1 μ s. Over the time of the 25ms reference pulse, the timer should nominally accumulate 25,000 of the 1 μ s clocks, so $C_N = 25,000$.

Now assume that the actual number of clock cycles (C_A) counted by the timer is 25,355 and that the value read from RCTRM is -125. By plugging this data into Equation 2, the correct value for RCTRM can be found:

$$-125 - (25,355/25,000 - 1) \times 500 = -132$$

The correct new value for RCTRM is -132 (or 0xFF7C in two's complement).

It is recommended that a reference pulse be measured again after calibration to verify the calibration.

15.3.2 External Oscillator and Crystal

The external clock source can be an external oscillator, a quartz crystal, or ceramic resonator. The core is designed to work at a maximum frequency of 16MHz.

If an external oscillator is used, it can be connected to XIN (pin 20), and XOUT (pin 21) can be left floating. However, a crystal should be connected from XIN to XOUT. (Refer to the crystal manufacturer's data sheet for more details.)

Typically, a switch to the external crystal/oscillator source is made after the crystal has been oscillating for long enough for it to "warm up," stabilizing its frequency. The crystal warmup timer provides this delay, which is a total of 65,536 crystal oscillations.

There are two ways to switch from the internal RC oscillator to the external crystal/oscillator source.

- **Automatic Switch:** This does not have precise control on the switch over time. Set XTRC (CKCN.7) and monitor RCMD (CKCN.5) to detect when the switchover happens.
- **Manual Switch:** This method precisely controls the switch over time. Set XTE (OSCC.1) and poll XTRDY (ASR.8) (or use interrupts) to wait until the crystal is ready. When the crystal is ready, set XTRC (CKCN.7); switchover to the external crystal/clock happens immediately.

A switch from the external crystal/oscillator source to the internal RC oscillator requires setting XTRC = 0. The clock switches the clock source to internal within a few RC oscillator cycles, so this switch happens immediately. Note that if the XTE bit is left set to 1, the external crystal oscillator does not turn off, and therefore, the switch back to the crystal oscillator is immediate because the warmup time for the crystal oscillator/clock is eliminated.

15.3.3 Internal System Clock Generation (SYSCLK)

The Internal system clock shown in Figure 15-1 as SYSCLK is derived from one of the selected system clock sources discussed in *Section 15.3.1* and *Section 15.3.2*. SYSCLK is the clock source to all the functional blocks (SPI, UART, timer/counter, etc.) within the MAXQ7667.

The selected clock source can be manipulated as shown in Table 15-2 to produce the desired system clock.

Table 15-2. Divide Ratio of the System Clock from a Clock Source

PMME	CD1	CD0	CLOCK-DIVIDE RATIO
0	0	0	Divide by 1 (default)
0	0	1	Divide by 2 (unavailable)
0	1	0	Divide by 4 (unavailable)
0	1	1	Divide by 8 (unavailable)
1	0	0	Divide by 256, Switchback to Divide by 1 (unavailable)
1	0	1	Divide by 256, Switchback to Divide by 2 (unavailable)
1	1	0	Divide by 256, Switchback to Divide by 4 (unavailable)
1	1	1	Divide by 256, Switchback to Divide by 8 (unavailable)

15.4 External Oscillator Clock/Crystal Failure Switchover

The external oscillator clock/crystal failure can occur during normal operation and is detected when the XTRDY bit switches over to 0, indicating that the external source has failed. The hardware forces a switch to the internal RC oscillator to keep the MAXQ7667 processor operational.

The following must be true to detect the failure on the high-frequency input:

- XTRC must be set to logic 1, which enables the external clock source and selects external clock/crystal as the active clock source.
- XTRDY bit is set to 1, which indicates external clock is ready.
- A falling edge is detected on the XTRDY bit, which signals a crystal/clock source failure.

When the above conditions have been detected, the clock circuitry will:

- Enable the RC oscillator immediately, regardless of the original state of the RCE bit.
- Switch the clock source from external to internal by forcing XTRC to 0.
- Make the system clock ready after a few clock cycles delay.
- Set the high-frequency crystal failure interrupt XTI (AIE.7) flag to 1; an interrupt is generated if the interrupt was enabled, XTIE = 1 (ASR.7).

This switchover remains until software reconfigures the clock structure.

15.5 Watchdog Timer

Generally, the primary function of the watchdog timer is to provide system control in case a software problem occurs; the watchdog timer performs a controlled system restart when the selected time interval expires.

The watchdog timer is driven by the internal RC clock, as illustrated in Figure 15-2. If the watchdog is enabled, the RC clock remains turned on, even after the system clock has been switched over to the external source (XTRC = 1).

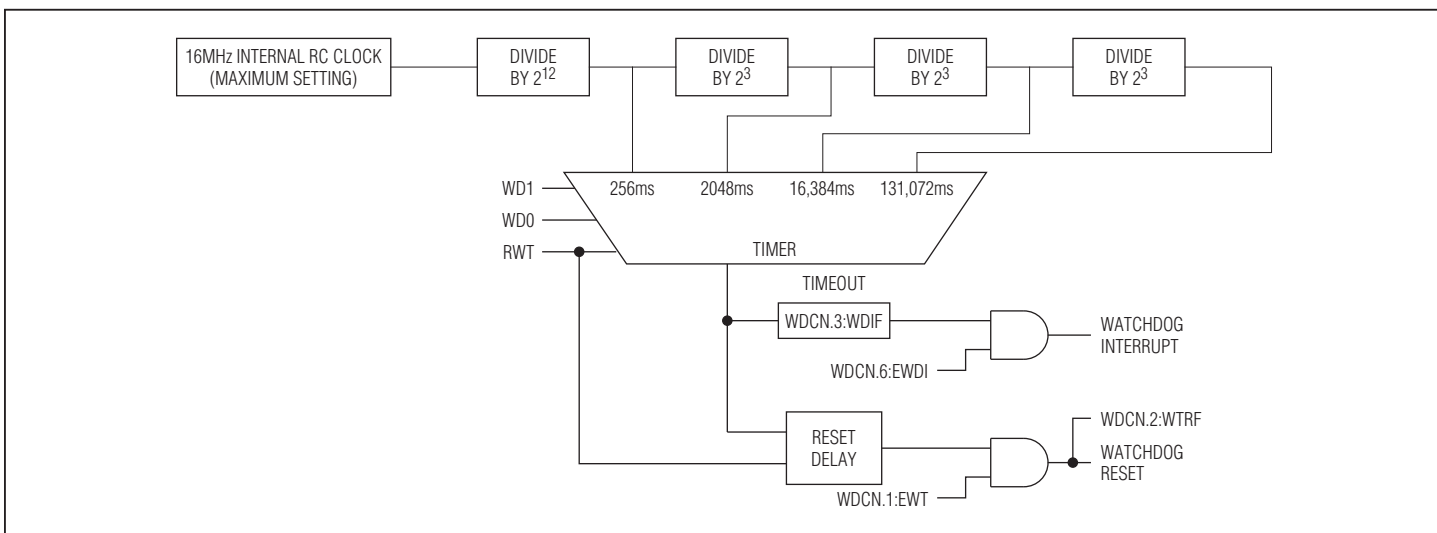


Figure 15-2. Watchdog Timer Function

As shown in Figure 15-2, the timer is driven by the internal RC clock (set at the maximum 16MHz frequency) through a series of dividers. The divider output is selectable and determines the interval between timeouts. When the watchdog interrupt timeout is reached, the interrupt flag WDIF (WDCN.3) is set and generates an interrupt if the interrupt enable bit EWDI (WDCN.6) bit is set. Typically, after the WDIF gets set the user code resets the watchdog timer RWT (WDCN.0). In the case when this bit is not cleared before the watchdog reset timeout, the MAXQ7667 undergoes a soft reset.

The watchdog timer functions as the source of both the watchdog interrupt and the watchdog reset. At normal clock frequency, the interrupt timeout has a default divide ratio of 2^{12} the source clock, with the watchdog reset set to time out 2^9 clock cycles later. This results in a 16MHz internal RC clock producing an interrupt timeout every 0.256ms ($2^{12}/16\text{MHz}$), followed by a watchdog reset, 32.0 μs ($2^9/16\text{MHz}$) later. The watchdog timer is reset to the default divide ratio (2^{12}) following any reset. Using the WD0 and WD1 bits in the WDCN register, other divide ratios can be selected for longer watchdog interrupt periods. If the WD[1:0] bits are changed before the watchdog interrupt timeout occurs (i.e., before the watchdog reset counter begins), the watchdog timer count is reset. All watchdog timer reset timeouts follow the programmed interrupt timeouts by 512 (or 2^9) times the divider ratio of the source clock cycles. Table 15-3 summarizes the watchdog bit settings and the timeout values.

Table 15-3. Watchdog Timeout Values (in Number of Source Clock Cycles)

WD[1:0]	DIVIDE RATIO	WATCHDOG INTERRUPT TIMEOUT = DIVIDE RATIO/16MHz (ms)	WATCHDOG RESET TIMEOUT = DIVIDE RATIO/16MHz + $2^9/16\text{MHz}$ (ms)
00 (default)	2^{12}	0.256	0.256 + 0.032
01	2^{15}	2.048	2.048 + 0.032
10	2^{18}	16.384	16.384 + 0.032
11	2^{21}	131.072	131.072 + 0.032

15.6 Oscillator/Clock Power-Saving Management Modes

15.6.1 Stop Mode

The stop mode disables all circuits within the processor. All on-chip clocks, timers, and serial port communication are stopped, and no processing is possible. Once in stop mode, the device is in a static state; its power consumption is mostly limited by the leakage current.

Stop mode is invoked by setting the STOP bit (CKCN.4) to logic 1. The processor enters stop mode on the instruction that sets the STOP bit. 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 the stop mode removal.

The processor can exit the stop mode:

- By using any of the external interrupts that are enabled
- By external reset via the $\overline{\text{RESET}}$ pin

When the stop mode is removed, the processor resumes its normal execution:

- After a four-cycle delay:
 - If XTRC is 1, the RC oscillator is used until XTRDY is set to logic 1 (indicating external clock is stable) before switching back to external clock source.
 - If XTRC is 0, the internal RC oscillator is the clock source.

15.6.1.1 Resuming from Stop Mode

When the system is in stop mode while the system clock is sourced from the external high-frequency source, the internal RC oscillator is disabled. When stop mode is removed, the crystal oscillator requires a long period of time to start up and stabilize. To allow the system to begin quick execution of software following the removal of stop mode, the RC oscillator can be used to supply a system clock until the crystal startup time is satisfied. Once this time has passed, the internal system clock is switched over to the external crystal oscillator. To allow the processor to know when it is being clocked by the RC oscillator or the crystal oscillator, the RCMD bit indicates which clock source is being used.

15.6.2 Switchback Mode

The system clock is used to provide standard baud-rate generation for the external interface. However, the clock speed choice affects all functional logic including timers and the baud-rate generator in the serial port module.

The switchback feature allows low-power operation and quick response to events that require full processing capacity. The switchback function is enabled by setting the SWB bit to logic 1. The switchback occurs whenever one of the following conditions occurs:

- Detection of a selected edge transaction on any of the external interrupts when the respective pin has been programmed and allowed to issue an interrupt.
- SPI activity:
 - SPIB is written in master mode (STBY = 1).
 - The SS signal is asserted in slave mode.
- Wake-up alarm from the system timer when enabled.
- Entry into active debug mode either by a breakpoint match or issuance of the debug command.

The switchback interrupt relationship requires that the respective external interrupt source be allowed to actually generate an interrupt before the switchback will actually occur. An interrupt by SPI is not required, nor is the setting of the serial peripheral enable. Disabling external interrupts and serial devices' receive/transmission mode disables the automatic switchback mode. Clearing the SWB bit also disables the switchback.

SECTION 16: POWER-SUPPLY/SUPERVISORY MONITORING

This section contains the following information:

16.1 Power-Supply/Supervisory Module Pins	16-3
16.2 Power-Supply/Supervisory Monitoring Registers	16-5
16.2.1 Analog Interrupt Enable Register (AIE)	16-5
16.2.2 Analog Status Register (ASR)	16-6
16.2.3 Analog Power Enable Register (APE)	16-7
16.2.4 Watchdog Timer Control Register (WDCN)	16-8
16.3 Supply Configuration	16-8
16.3.1 5V Regulator with External Pass Transistor	16-10
16.3.1.1 Depletion Mode FET as the Pass Device	16-10
16.3.1.2 Darlington Bipolar Transistor as the Pass Device	16-11
16.3.1.3 Enhancement Mode FET as the Pass Device	16-11
16.4 Power-On Reset	16-12
16.4.1 Reset Output	16-13
16.5 Power-Supply Voltage Monitors	16-13
16.5.1 Digital Core Supply (DVDD) Monitor	16-13
16.5.2 Digital I/O Supply (DVDDIO) Monitor	16-14
16.5.3 Analog Supply Voltage (AVDD) Monitor	16-14
16.6 Reset Mode	16-15
16.6.1 Watchdog Timer Reset	16-15
16.6.2 External Reset	16-15
16.6.3 Internal System Reset	16-15

LIST OF FIGURES

Figure 16-1. MAXQ7667 Power-Supply Block Diagram	16-4
Figure 16-2. Supply Configuration Using All External Power Supply	16-9
Figure 16-3. Supply Configuration Using External Power Supply and Internal Regulators	16-9
Figure 16-4. DVDDIO Using a Depletion Mode Pass Transistor	16-10
Figure 16-5. DVDDIO Using a Darlington Pass Transistor	16-11
Figure 16-6. MAXQ7667 Power-On Reset	16-12
Figure 16-7. DVDD Brownout Interrupt Threshold Detection	16-13
Figure 16-8. DVDDIO Brownout Interrupt Threshold Detection	16-14
Figure 16-9. AVDD Brownout Interrupt Threshold Detection	16-14
Figure 16-10. MAXQ7667 External Reset	16-16

LIST OF TABLES

Table 16-1. MAXQ7667 Power-Supply Module Pins	16-3
---	------

SECTION 16: POWER-SUPPLY/SUPERVISORY MONITORING

The MAXQ7667 power-supply/supervisory monitoring module supports dedicated supply pins to independently power analog, digital I/O, and digital core functions. The analog functions, digital core, and the digital I/O can be powered from an internal regulator.

The MAXQ7667 power-supply/supervisory monitoring module includes the following features:

- Dedicated analog supply (+3.3V) pins
- Dedicated digital I/O supply (+5.0V) pins
- Dedicated digital core supply (+2.5V) pins
- On-chip +2.5V regulator with a dedicated output pin
- On-chip +3.3V regulator with a dedicated output pin
- On-chip +5.0V regulator (needs an external pass transistor for operation)
- Digital core voltage monitor
- Digital I/O voltage monitor
- Analog voltage monitor
- Preset threshold voltage level for power-on reset
- Preset threshold voltage level for digital core interrupt generation
- Preset threshold voltage level for digital I/O interrupt generation
- Preset threshold voltage level for analog interrupt generation
- Four reset sources: power-on (brownout), external, watchdog timer, and internal system

16.1 Power-Supply/Supervisory Module Pins

Table 16-1 shows the MAXQ7667 power-supply module signals.

Table 16-1. MAXQ7667 Power-Supply Module Pins

POWER-SUPPLY SIGNAL	PIN	FUNCTION
AVDD	27, 32	Analog Supply (+3.3V). AVDD is the power-supply pin for all analog functions. Connect to REG3P3 and bypass with a 0.1 μ F capacitor to ground for self-powered operation.
AGND	28, 31, 33	Analog Ground
DVDD	6, 19, 42	Digital Supply Voltage (+2.5V). Connect all DVDD pins to the same point. Connect to REG2P5 and bypass with a 0.1 μ F capacitor to ground for self-powered operation.
DVDDIO	8, 17, 44	I/O Supply Voltage (+5.0V). DVDDIO is the power-supply pin for all digital input/output pins. Connect all DVDDIO pins to the same point.
DGND	7, 18, 43	Digital Ground
$\overline{\text{RESET}}$	25	Active-Low Reset Input/Output. Open-drain signal with internal pullup resistor to DVDD that remains low until DVDD rises above the brownout-reset (BOR) threshold and a timeout period expires. $\overline{\text{RESET}}$ is pulled low by internal circuitry if DVDD falls below its brownout reset value. It can also be pulled low by the internal watchdog timer, or externally by the user.
GATE5	24	DVDDIO Pass Transistor Voltage Regulation Control. GATE5 connects directly to the gate of a depletion mode transistor or through level-shifting diodes to an enhancement mode or npn Darlington transistor that supplies 5V to DVDDIO.
REG2P5	22	+2.5V Regulator Output. For self-powered operation, connect to DVDD and bypass to DGND with a 0.1 μ F capacitor.
REG3P3	23	+3.3V Regulator Output. For self-powered operation, connect to AVDD and bypass to AGND with a 0.1 μ F capacitor.

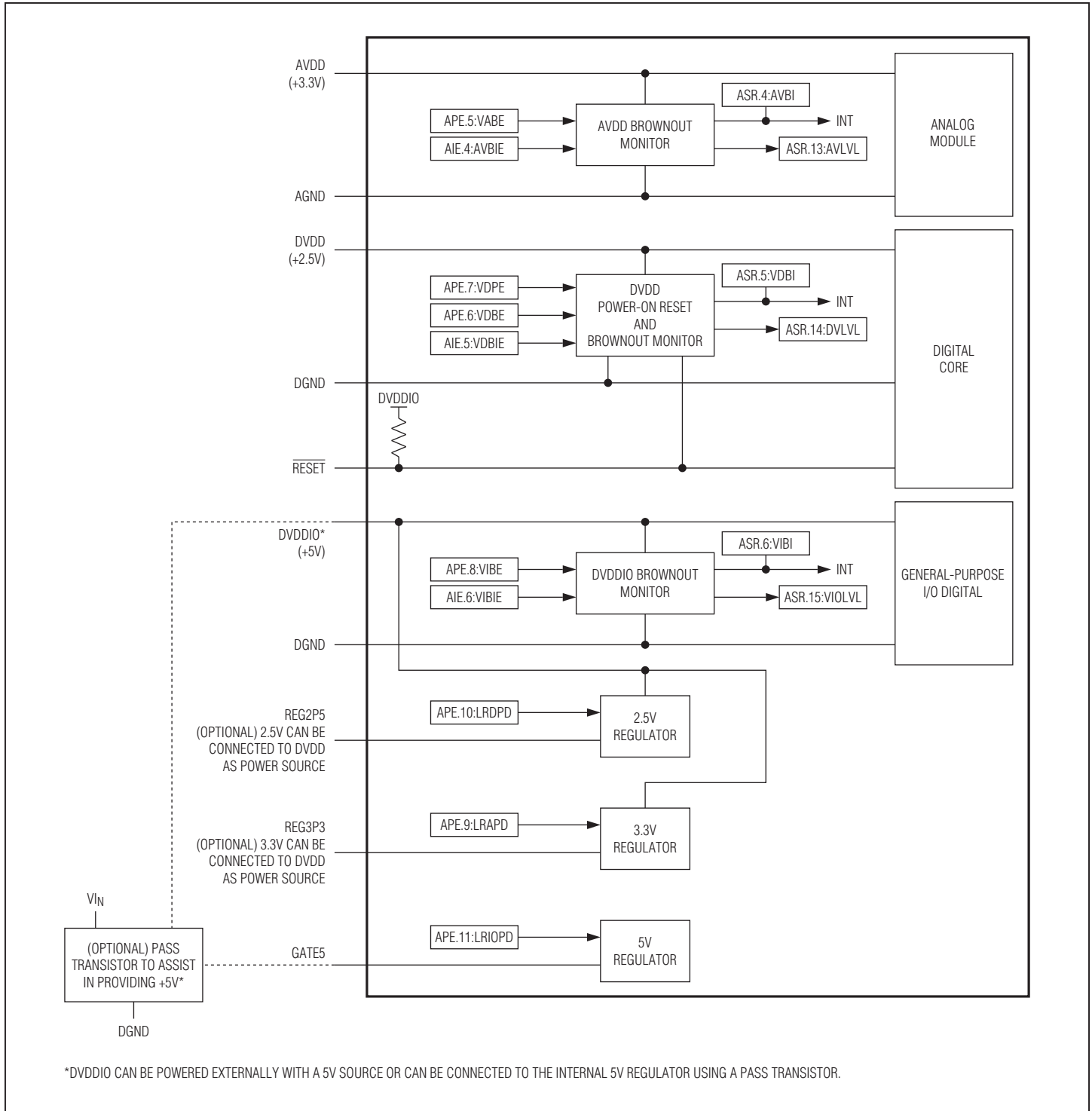


Figure 16-1. MAXQ7667 Power-Supply Block Diagram

16.2 Power-Supply/Supervisory Monitoring Registers

16.2.1 Analog Interrupt Enable Register (AIE)

Register Description: **Analog Interrupt Enable Register**

Register Name: **AIE**

Register Address: **Module 05h, Index 05h**

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	XTIE	VIBIE	VDBIE	VABIE	CMPIE	LFLIE	LPFIE	SARIE
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: AIE is cleared to 0000h on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0.

Bit 7: Crystal Oscillator Failure (XTIE). See *Section 15* for details on this bit.

Bit 6: DVDDIO Brownout Interrupt Enable (VIBIE). When set to 1, this bit allows an interrupt request to be generated when DVDDIO falls below the specified threshold and VIBE is 1.

Bit 5: DVDD Brownout Interrupt Enable (VDBIE). When set to 1, this bit allows an interrupt request to be generated when DVDD falls below the specified threshold and VDBE is 1.

Bit 4: AVDD Brownout Interrupt Enable (VABIE). When set to 1, this bit allows an interrupt request to be generated when AVDD falls below the specified threshold and VABE is 1.

Bit 3: Echo Envelope Comparator Interrupt Enable (CMPIE). See *Section 17* for details on this bit.

Bit 2: Echo Envelope Lowpass Filter FIFO Full Interrupt Enable (LFLIE). See *Section 17* for details on this bit.

Bit 1: Echo Envelope Lowpass Filter Output Data Ready Interrupt Enable (LPFIE). See *Section 17* for details on this bit.

Bit 0: SAR ADC Data Ready Interrupt Enable (SARIE). See *Section 14* for details on this bit.

16.2.2 Analog Status Register (ASR)

Register Description: **Analog Status Register**
 Register Name: **ASR**
 Register Address: **Module 05h, Index 08h**

Bit #	15	14	13	12	11	10	9	8
Name	VIOLVL	DVLVL	AVLVL	CMPLVL	—	—	—	XTRDY
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	XTI	VIBI	VDBI	VABI	CMPI	LPFFL	LPFRDY	SARRDY
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: ASR bits 3:0 are cleared to 0h on reset; bits 8:4 clear to 0h on power-on reset.

Bit 15: DVDDIO Brownout Comparator Output Level (VIOLVL). This bit always reflects the DVDDIO brownout comparator's current state when read.

Bit 14: DVDD Brownout Comparator Output Level (DVLVL). This bit always reflects the DVDD brownout comparator's current state when read.

Bit 13: AVDD Brownout Comparator Output Level (AVLVL). This bit always reflects the AVDD brownout comparator's current state when read.

Bit 12: Echo Envelope Comparator Output Level (CMPLVL). See Section 17 for details on this bit.

Bits 11 to 9: Reserved. Read returns 0.

Bit 8: Crystal Oscillator Ready (XTRDY). See Section 15 for details on this bit.

Bit 7: Crystal Oscillator Failure Interrupt Flag (XTI). See Section 15 for details on this bit.

Bit 6: DVDDIO Brownout Interrupt Flag (VIBI). Set to 1 when DVDDIO brownout is detected.

Bit 5: DVDD Brownout Interrupt Flag (VDBI). Set to 1 when DVDD brownout is detected.

Bit 4: AVDD Brownout Interrupt Flag (VABI). Set to 1 when AVDD brownout is detected.

Bit 3: Echo Envelope Comparator Interrupt Flag (CMPI). See Section 17 for details on this bit.

Bit 2: Echo Envelope Lowpass Filter FIFO Full Interrupt Flag (LPFFL). See Section 17 for details on this bit.

Bit 1: Echo Envelope Lowpass Filter Output Data Ready Flag (LPFRDY). See Section 17 for details on this bit.

Bit 0: SAR ADC Data Ready Flag (SARRDY). See Section 14 for details on this bit.

16.2.3 Analog Power Enable Register (APE)

Register Description: **Analog Power Enable Register**
 Register Name: **APE**
 Register Address: **Module 05h, Index 10h**

Bit #	15	14	13	12	11	10	9	8
Name	—	RBUFE	—	BGE	LRIOPD	LRDPD	LRAPD	VIBE
Reset	0	0	0	0	0	0	0	0
Access	r	rw	r	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	VDPE	VDBE	VABE	SARE	PLLE	MDE	LNAE	BIASE
Reset	1	0	0	0	0	0	0	0
Access	rw	rw	rw	r	rw	rw	rw	rw

r = read, w = write

Note: APE is cleared to 0080h on all forms of reset.

Bits 15 and 13: Reserved. Read returns 0.

Bit 14: Reference Buffer Enable (RBUFE). Set to 1 to enable; set to 0 to disable.

Bit 12: Bandgap Enable (BGE). Set to 1 to enable; set to 0 to disable.

Bit 11: I/O Linear Regulator Power-Down (LRIOPD). When set to 1, this bit disables the DVDDIO linear regulator and places the GATE5 pin into high-impedance mode. When set to 0, this bit powers up the regulator.

Bit 10: Digital Linear Regulator Power-Down (LRDPD). When set to 1, this bit disables the DVDD linear regulator and places it in a leakage-only state. When set to 0, this bit powers up the regulator.

Bit 9: Analog Linear Regulator Power-Down (LRAPD). When set to 1, this bit disables the AVDD linear regulator and places it in a leakage-only state. When set to 0, this bit powers up the regulator.

Bit 8: I/O Voltage Brownout Detection Enable (VIBE). When set to 1, this bit enables the DVDDIO monitor to generate an interrupt if the VIBIE bit is also set and DVDDIO falls below the specified threshold. When set to 0, this bit disables the DVDDIO monitor and places it in a leakage-only state.

Bit 7: Digital Voltage Reset Enable (VDPE). When set to 0, this bit disables the DVDD reset supervisor. This bit defaults to 1 at reset.

Bit 6: Digital Voltage Brownout Detection Enable (VDBE). When set to 1, this bit enables the DVDD monitor to generate an interrupt if the VDBIE bit is also set and DVDD falls below the specified threshold. When set to 0, this bit disables the DVDD monitor and places it in a leakage-only state.

Bit 5: Analog Voltage Brownout Detection Enable (VABE). When set to 1, this bit enables the AVDD monitor to generate an interrupt if the VABIE bit is also set and AVDD falls below the specified threshold. When set to 0, this bit disables the AVDD monitor and places it in a leakage-only state.

Bit 4: SAR Enable (SARE). See *Section 14* for details on this bit.

Bit 3: PLL Enable (PLLE). See *Section 17* for details on this bit.

Bit 2: Sigma-Delta Modulator Enable (MDE). See *Section 17* for details on this bit.

Bit 1: LNA Enable (LNAE). See *Section 17* for more information on this bit.

Bit 0: Bias Enable (BIASE). See *Section 14* and *Section 17* for more information on this bit.

16.2.4 Watchdog Timer Control Register (WDCN)

Register Description: **Watchdog Timer Control Register**
 Register Name: **WDCN**
 Register Address: **Module 08h, Index 0Fh**

Bit #	7	6	5	4	3	2	1	0
Name	POR	EWDI	WD1	WDO	WDIF	WTRF	EWT	RWT
Reset	0	0	0	0	0	0	0	0
Access	r	rw	rw	r	r	r	rw	rw

r = read, w = write

Note 1: The watchdog timer always uses the RC oscillator as the clock source.

Note 2: Bits 5, 4, 3, and 0 are cleared to 0 on all forms of reset. See description for other bits.

Bit 7: Power-On Reset Flag (POR). This bit indicates whether the last reset was a power-on/brownout reset. This bit is typically interrogated following a reset. It must be cleared before the next reset of any kind for software to work correctly. This bit is set following a power-on/brownout reset and is unaffected by all other resets.

Bit 6: Watchdog Interrupt Enable (EWDI). See Section 15 for details on this bit.

Bits 5 and 4: Watchdog Timer Mode Select Bit 1 and 0 (WD[1:0]). See Section 15 for details on this bit.

Bit 3: Watchdog Interrupt Flag (WDIF). See Section 15 for details on this bit.

Bit 2: Watchdog Timer Reset Flag (WTRF). See Section 15 for details on this bit.

Bit 1: Enable Watchdog Timer Reset (EWT). See Section 15 for details on this bit.

Bit 0: Reset Watchdog Timer (RWT). See Section 15 for details on this bit.

16.3 Supply Configuration

The MAXQ7667 uses three supplies to power the internal analog, digital core, and digital I/O circuits:

- AVDD = +3.3V (with internal linear regulator enabled or through an external supply)
- DVDD = +2.5V (with internal linear regulator enabled or through an external supply)
- DVDDIO = +5V

Figure 16-2 shows the recommended supply configurations when only external power supplies are used. Bypass capacitors should be mounted as close as possible to the MAXQ7667 to reduce noise. Figure 16-3 shows the recommended supply configuration when the internal regulators are used to provide power to the chip. Bypass capacitors should be mounted as close as possible to the MAXQ7667 to reduce noise.

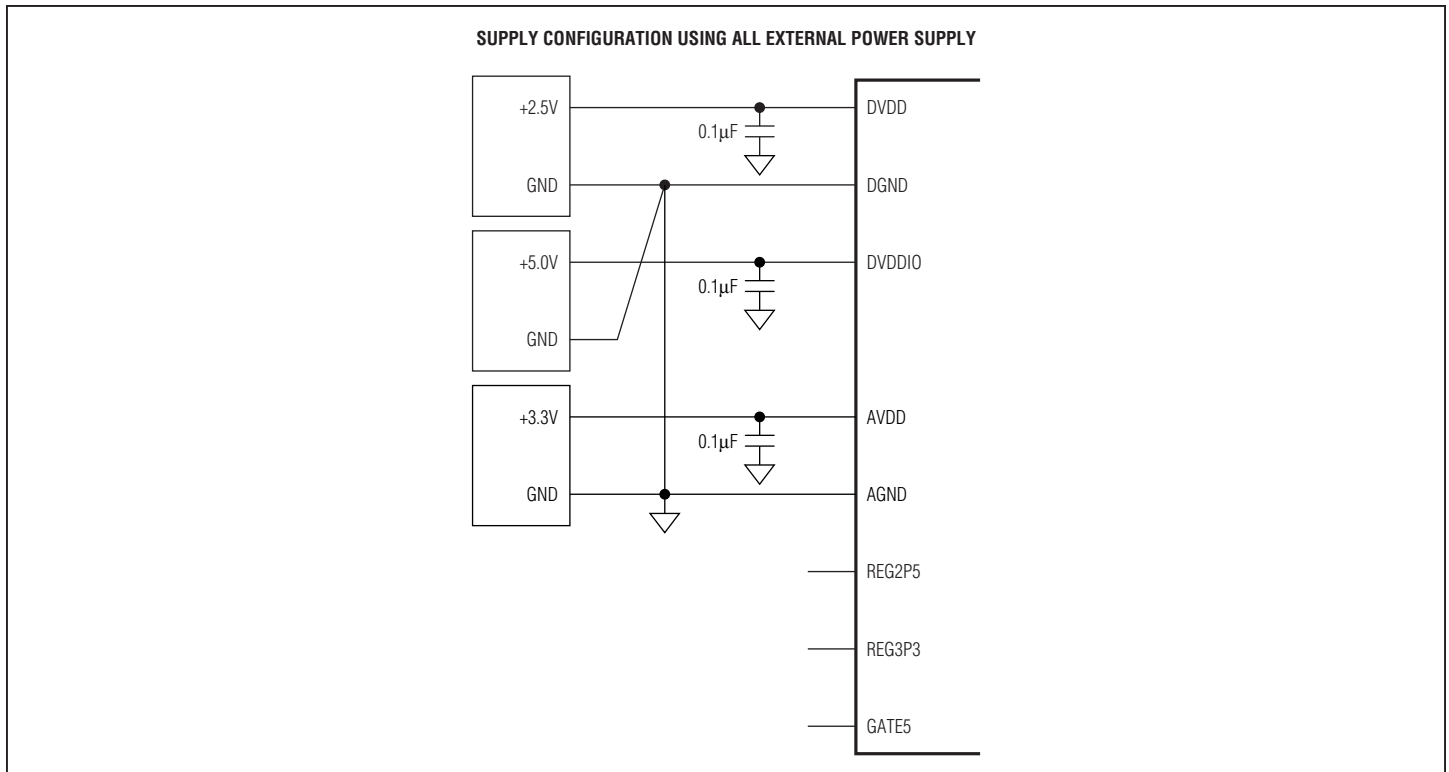


Figure 16-2. Supply Configuration Using All External Power Supply

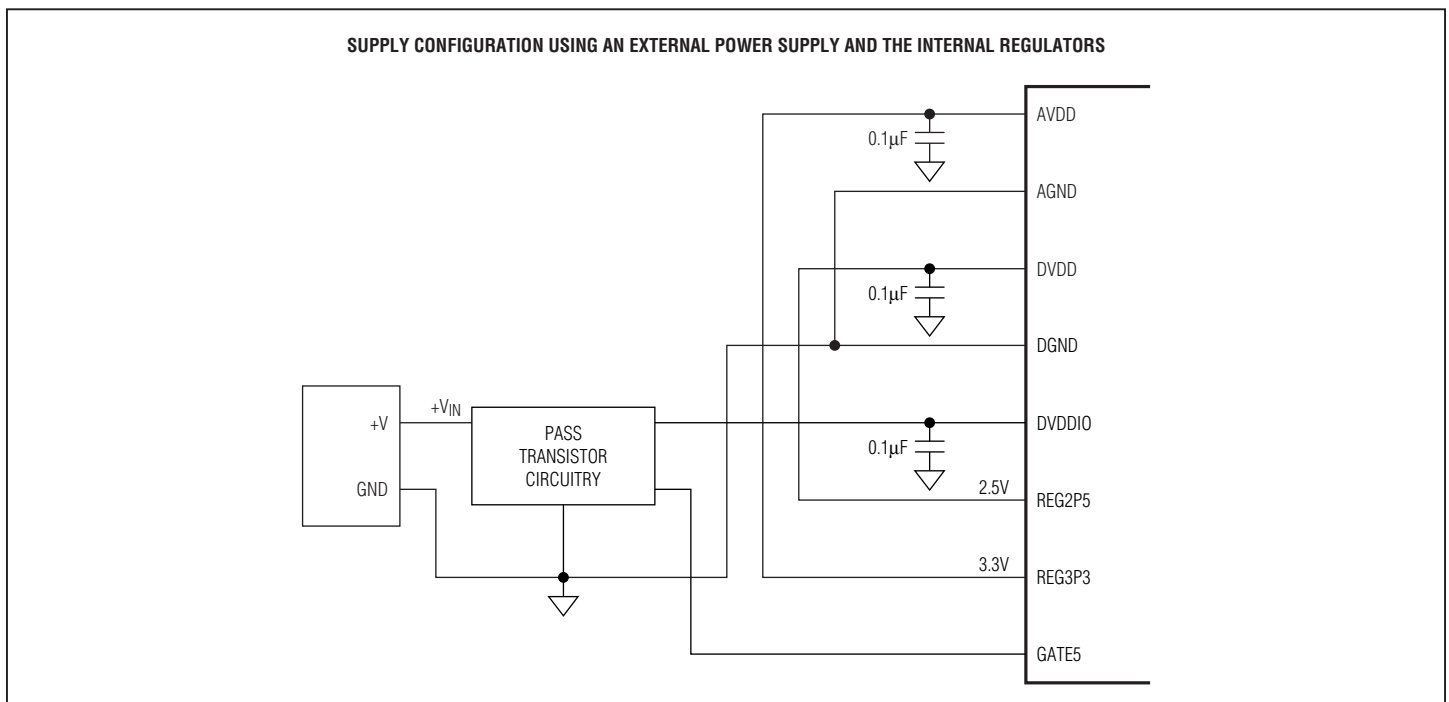


Figure 16-3. Supply Configuration Using External Power Supply and Internal Regulators

16.3.1 5V Regulator with External Pass Transistor

The 5V supply (DVDDIO) for the MAXQ7667 can be economically obtained from a higher voltage supply through the use of a few external components. The MAXQ7667 has an internal error amplifier that compares DVDDIO to a fixed voltage reference. The output of the error amplifier can be used to control an external pass transistor and thus create a regulated 5V supply. A capacitor from DVDDIO to ground supplies current for fast transient loads. The largest capacitor that can be used is $0.47\mu\text{F}$. Larger capacitors degrade the loop stability. This capacitor and the pass transistor should be placed as close as practical to the DVDDIO pin.

The output of the error amplifier has a voltage range of 2.0V to $\text{DVDDIO} - 0.1\text{V}$. Depending on the control voltage needed at the base or gate of the pass transistor, an external level shifter may be necessary. Level shifting is easily accomplished with a resistor and a few diodes. Figure 16-4 and Figure 16-5 show the use of three types of pass transistors: a depletion mode FET, a bipolar Darlington transistor, and an enhancement mode FET.

All three figures contain an optional RC filter ($R2$ and $C2$) between V_{IN} and the pass transistor. This filter is not necessary if V_{IN} is a "clean" voltage. However, adding this filter is recommended if there are transients or RF noise on V_{IN} . $R2$ has a suggested value of 10Ω , but a lower value may be used if a smaller voltage drop across $R2$ is needed.

16.3.1.1 Depletion Mode FET as the Pass Device

The depletion mode FET is the simplest configuration. The device is normally on, which allows the MAXQ7667 to power-up and go into regulation mode. The depletion mode FET provides a built-in current-limiting mechanism, and also offers low dropout since the gate voltage does not limit the dropout voltage.

There are only three restrictions on the choice of a depletion mode FET as the series pass device. The first restriction is that the device be able to supply the desired maximum load current when the GATE5 output is at its maximum. Note that I_{DSS} is usually specified at $V_{\text{GS}} = 0\text{V}$ instead of $V_{\text{GS}} = -0.1\text{V}$, which is the maximum (most on) condition realized with the MAXQ7667. Temperature effects also must be considered. The transistor must be able to supply enough current over the entire temperature range when $V_{\text{GS}} = -0.1\text{V}$.

The second restriction is that the GATE5 voltage must be able to turn the FET off. When at $V_{\text{GS}} = -3\text{V}$, I_{D} must be less than the minimum possible supply current. This can be as low as a few μA if the MAXQ7667 is placed in a standby configuration.

The third restriction is that the pole formed by the C_{ISS} of the FET and the output resistance of the error amp should be greater than 10 times the error amplifier's gain bandwidth product. This third criterion is not a problem for most transistors.

A depletion mode FET that works well with the MAXQ7667 is the BSP129.

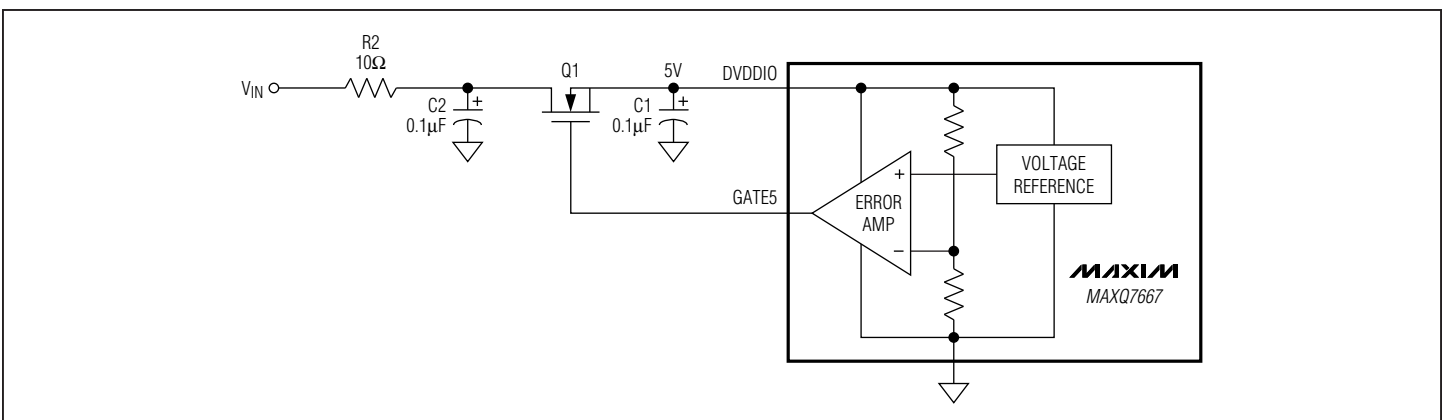


Figure 16-4. DVDDIO Using a Depletion Mode Pass Transistor

16.3.1.2 Darlington Bipolar Transistor as the Pass Device

Bipolar transistors may also be used for the pass transistor. In Figure 16-5 a Darlington pair is used to minimize the amount of base current that is required. Pullup resistor R1 is required to start the circuit and as part of the level shifter. All the base current for Q1 comes from R1. R1 must be sized so that it can supply the required base current when V_{IN} is at its minimum and Q1 has the lowest beta. Excess current from R1 flows through the level-shifting diodes and into the error amp. This excess current, which will be greatest when V_{IN} is at its maximum and Q1 has a high beta, must not exceed the current sink capability of the error amp (500 μ A).

Diodes D1 to D4 are required for level shifting the control voltage. To a first approximation, the voltage across D1 and D2 will match the base-emitter voltage of the Darlington. This makes the voltage at the cathode of D2 roughly equal to DVDDIO. Adding a third diode places the output of the error amp one diode drop, or about 600mV, below DVDDIO. While this voltage is acceptable, it is a condition that exists at room temperature. At elevated temperatures the diode voltage is reduced and the voltage available at the base of the transistor becomes marginal. For a robust design it is best to use a minimum of four level-shifting diodes with a Darlington transistor.

The dropout voltage for a bipolar pass transistor is the sum of the base-emitter voltage(s) and the voltage required across R1 to supply the necessary base current.

16.3.1.3 Enhancement Mode FET as the Pass Device

An enhancement mode FET may also be used for a pass transistor. R1 biases the gate during startup and then becomes part of the level-shifting circuit. The maximum V_{IN} and the current sinking capability of the error amp determine the minimum value of R1. The maximum value for R1 should be limited to 100k Ω for stability reasons. R1 values between 25k Ω and 100k Ω work in nearly all applications.

The number of diodes in the level-shifting string varies with the choice of FET and the temperature range. At all temperatures ($V_{GS(TH)} - N V_F$) and ($V_{GS(TH)} + I/S - N V_F$) must both be between -0.1V and -3.0V, where V_F is the forward voltage of the diode, N is the number of diodes, $V_{GS(TH)}$ is the threshold voltage of the FET, S is the transconductance, and I is the supply current for the MAXQ7667 and any other devices powered from DVDDIO.

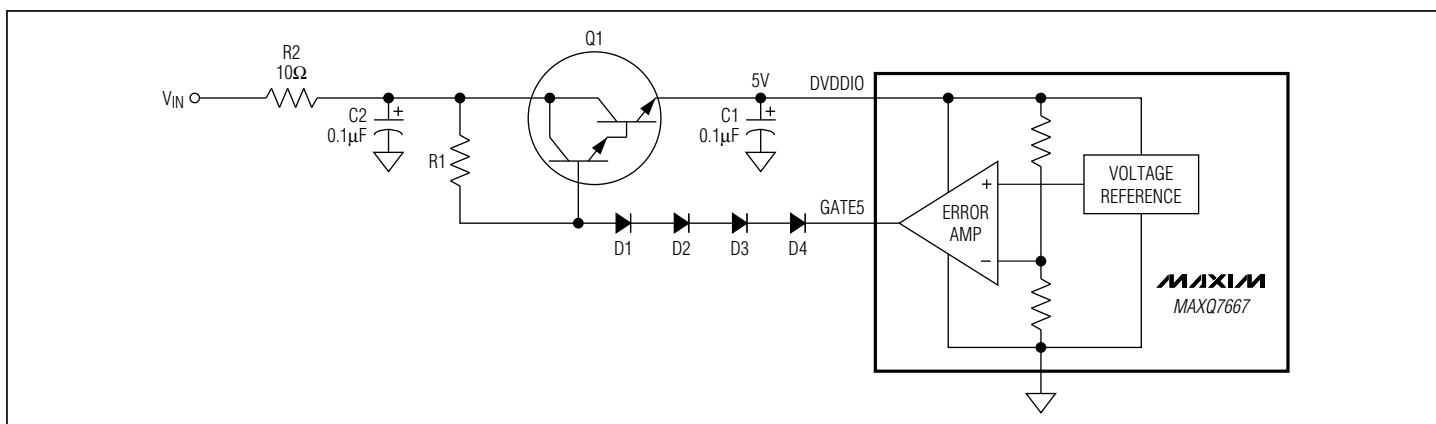


Figure 16-5. DVDDIO Using a Darlington Pass Transistor

16.4 Power-On Reset

On power-up, the VDPE bit (APE.7) is automatically set to 1, enabling the DVDD reset supervisor; consequently, the internal circuitry pulls the $\overline{\text{RESET}}$ pin low and all the internal system and peripheral registers are reset. As DVDD rises, it crosses the power-on-reset voltage threshold level, which can be between 2.10V and 2.25V, causing the crystal warmup counter to start. The $\overline{\text{RESET}}$ pin is held low until the completion of the counter.

Once the crystal warmup period has elapsed, the internal circuit releases the $\overline{\text{RESET}}$ pin. If no external circuitry is holding $\overline{\text{RESET}}$ low, the voltage on $\overline{\text{RESET}}$ rises and software execution begins at the reset vector location 8000h (in the utility ROM). Software can determine whether a reset was caused by a power-on/brownout reset or by other forms of reset by checking the POR flag in the WDCN register (WDCN.7). This flag is set to 1 following a power-on/brownout reset, and should be cleared by software after it has been checked.

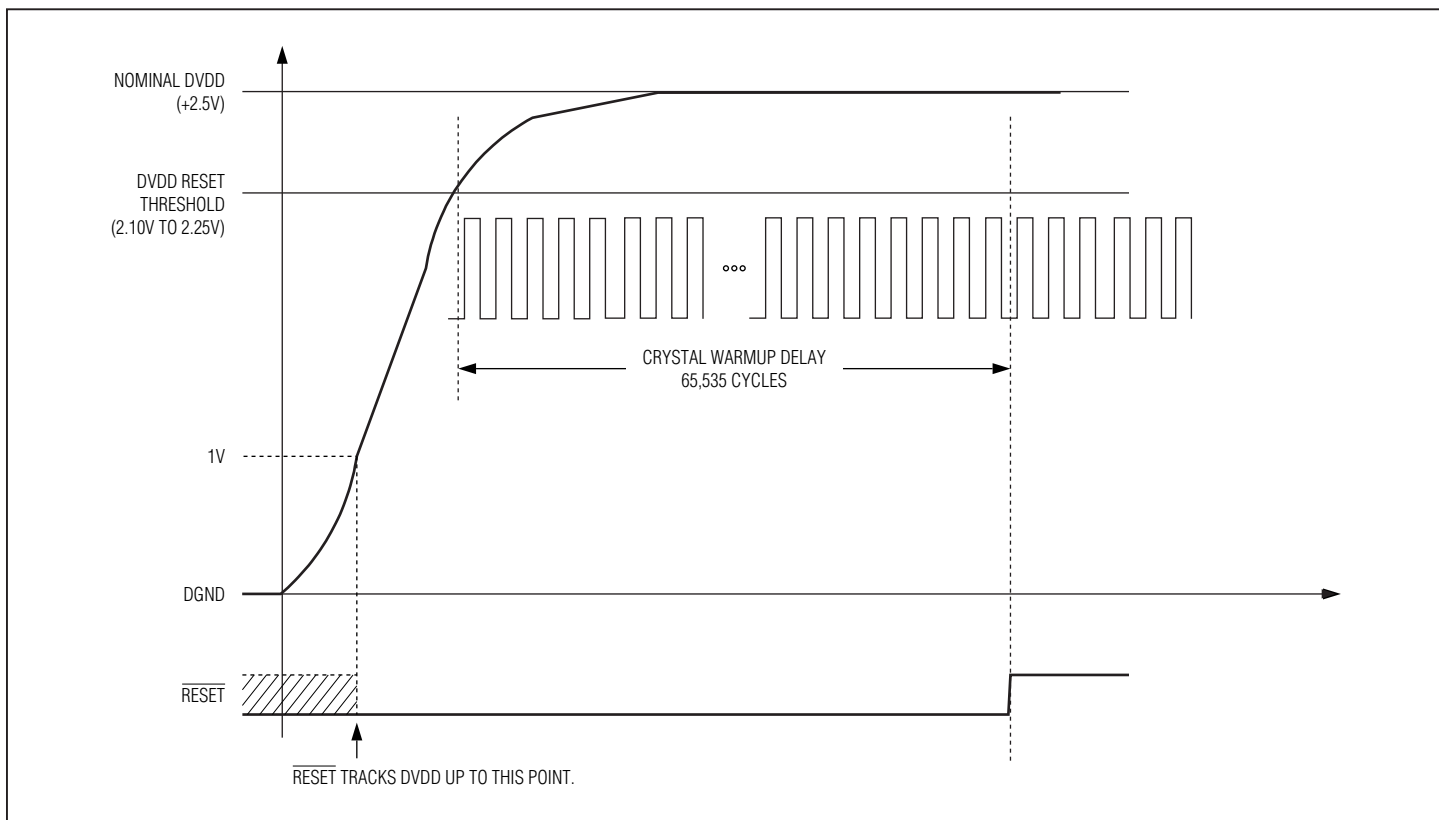


Figure 16-6. MAXQ7667 Power-On Reset

16.4.1 Reset Output

The MAXQ7667 asserts the $\overline{\text{RESET}}$ signal during power-up. On power-up, once DVDD exceeds 1V $\overline{\text{RESET}}$ is asserted to be logic-low. As DVDD rises, $\overline{\text{RESET}}$ remains low. When DVDD exceeds the DVDD POR threshold, $\overline{\text{RESET}}$ is kept low until the internal RC oscillator becomes stable and has completed 65,536 cycles. After this period, if DVDD remains above the DVDD POR threshold, $\overline{\text{RESET}}$ goes high. If a brownout reset condition occurs, $\overline{\text{RESET}}$ is asserted low. Each time a DVDD BOR reset is triggered, it stays low until DVDD exceeds the BOR reset threshold.

16.5 Power-Supply Voltage Monitors

The MAXQ7667 contains three power-supply voltage monitors that can be used to continually monitor AVDD, DVDD, and DVDDIO supply voltages. All three power-supply voltage monitors can be monitored for brownout condition and initiate interrupt requests if enabled. The voltage monitors can be independently activated by programming the corresponding enable bits in the Analog Power Enable register (APE), such as VDBE (digital voltage brownout detection enable) for DVDD, VIBE (I/O voltage brownout detection enable) for DVDDIO, and VABE (analog voltage brownout detection enable) for AVDD.

16.5.1 Digital Core Supply (DVDD) Monitor

The digital core supply monitor detects a brownout condition on the +2.5V DVDD supply. The DVDD supply monitor can be independently activated by programming the corresponding enable bit (VDBE) in the APE register. A brownout is detected when the DVDD supply voltage drops below the brownout interrupt threshold (Figure 16-7). If enabled, a DVDD brownout interrupt can be generated that would give the application code ample time to react appropriately. A DVDD brownout interrupt is generated only if the interrupt enable bit (VDBIE) in the Analog Interrupt Enable (AIE) register is set. Also, global interrupt mask bits IM5 (in the IMR register) and IGE (in the IC register) must be enabled.

If the DVDD supply falls further, the brownout reset threshold is tripped, terminating program operation and holding the MAXQ7667 to the power-on reset state. The MAXQ7667 remains in the power-on reset state until the supply rises above the reset threshold. See Section 16.4 for details.

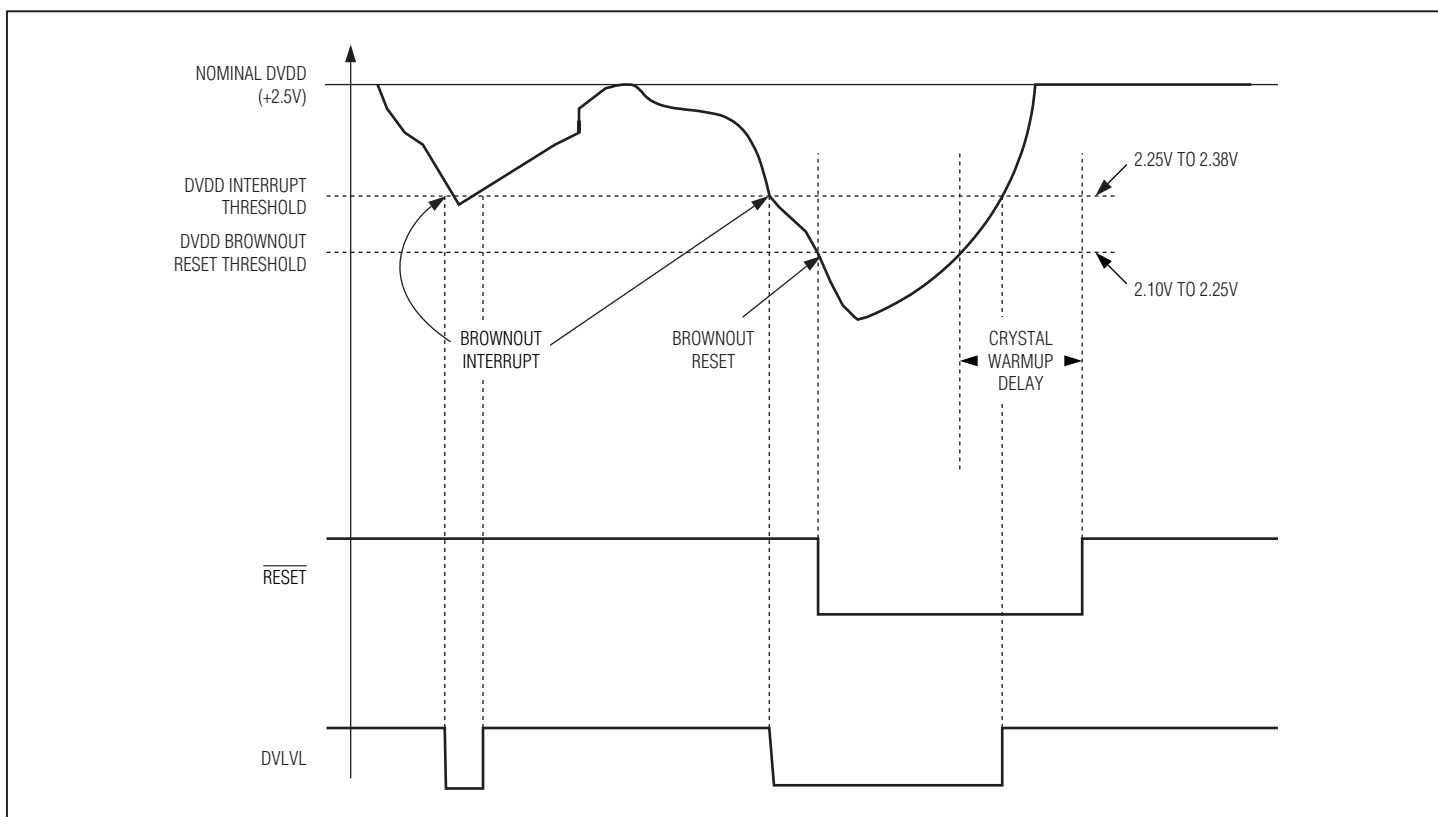


Figure 16-7. DVDD Brownout Interrupt Threshold Detection

16.5.2 Digital I/O Supply (DVDDIO) Monitor

The DVDDIO monitor detects a brownout condition on the +5V digital I/O supply. The DVDDIO supply monitor can be independently activated by setting to 1 the VIBE bit in the APE register. A brownout is detected when the DVDDIO supply voltage falls below the programmed DVDDIO brownout detection threshold (see Figure 16-8). When using the internal voltage regulators, the DVDDIO brownout interrupt provides the earliest warning of a loss of power event. If enabled, a DVDDIO brownout interrupt can be generated. When using the internal voltage regulators, the DVDDIO brownout interrupt provides the earliest warning of a loss of power event. A DVDDIO brownout interrupt is generated only if the interrupt enable bit (VIBIE) in the AIE register is set. Also, global interrupt mask bits IM5 (in the IMR register) and IGE (in the IC register) must be enabled.

16.5.3 Analog Supply Voltage (AVDD) Monitor

The AVDD monitor detects a brownout condition on the +3.3V analog I/O supply. The AVDD supply monitor can be independently activated by setting the VABE bit in the APE register. A brownout is detected when the AVDD supply voltage falls in the programmed AVDD brownout detection threshold (Figure 16-9). An AVDD brownout interrupt is generated only if the interrupt enable bit (VABIE) in the AIE register is set. Also, global interrupt mask bits IM5 (in the IMR register) and IGE (in the IC register) must be enabled.

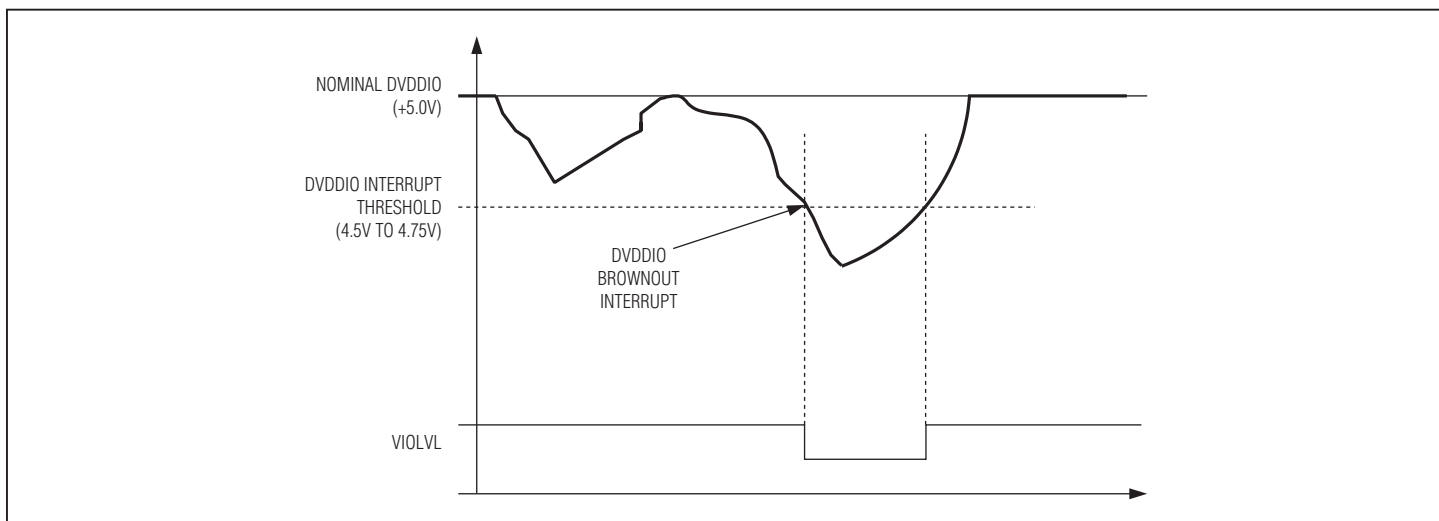


Figure 16-8. DVDDIO Brownout Interrupt Threshold Detection

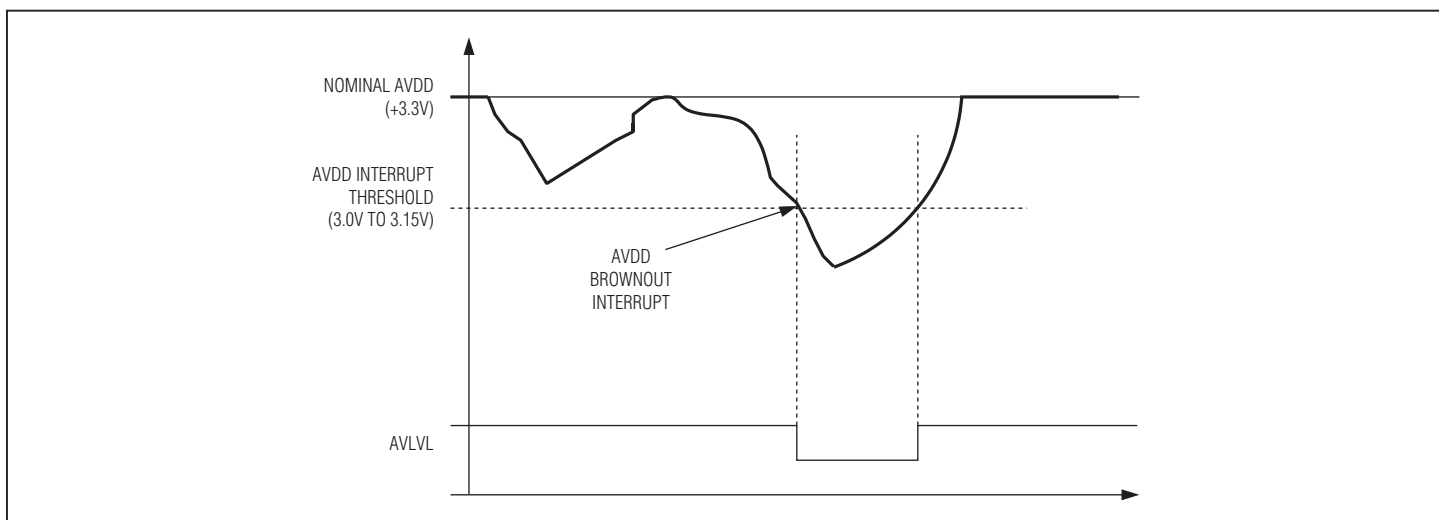


Figure 16-9. AVDD Brownout Interrupt Threshold Detection

16.6 Reset Mode

When the MAXQ7667 is in reset mode, the enabled system clock oscillator continues running, but no instruction execution or other system or peripheral operations occur, and all input/output pins return to default states. Once the condition that caused the reset (whether internal or external) is removed, code execution resumes at address 8000h for all reset types.

There are four different sources that can cause the MAXQ7667 to enter reset mode. See *Sections 16.4* and *16.5.1* for information on power-on and brownout reset.

- Power-On Reset (Brownout Reset)
- Watchdog Timer Reset
- External Reset
- Internal System Reset

16.6.1 Watchdog Timer Reset

The MAXQ7667 watchdog timer is described in *Section 15: Oscillator Clock Generation*. The watchdog timer is a programmable hardware timer that can be configured to reset the MAXQ7667 in the case of a software lockup or other unrecoverable error. Once the watchdog is enabled in this manner, the processor must reset the watchdog periodically to avoid a reset. If the processor does not reset the watchdog timer before it elapses, the watchdog initiates a reset state.

If the watchdog resets the MAXQ7667, it remains in reset, and holds the $\overline{\text{RESET}}$ pin low for four clock cycles. Once the reset condition is removed, the processor begins executing program code at address 8000h. When a reset occurs due to a watchdog timeout, the watchdog timer reset flag, WTRF (WDCN.2), is set to 1 and can only be cleared by software. User software can examine this bit following a reset to determine if that reset was caused by a watchdog timeout.

A watchdog reset does not affect the XTRDY bit (ASR.8) or the XTE bit (OSCC.1), so after a watchdog reset the MAXQ7667 reboots with the same oscillator it was using before the reset.

16.6.2 External Reset

For the MAXQ7667 the $\overline{\text{RESET}}$ pin is an output as well as an input. If a reset condition is caused by an internal source (such as a POR, watchdog, or internal reset), an output reset pulse (low level) is generated at the $\overline{\text{RESET}}$ pin.

During normal operation, the MAXQ7667 device is placed into an external reset mode by holding the $\overline{\text{RESET}}$ pin low for at least four clock cycles. If the MAXQ7667 is in the low-power stop mode (i.e., system clock is not active), the $\overline{\text{RESET}}$ pin becomes an asynchronous source, forcing the reset state immediately after being taken to logic 0. Once the MAXQ7667 enters reset mode, it remains in reset as long as the $\overline{\text{RESET}}$ pin is held at logic 0. After the $\overline{\text{RESET}}$ pin returns to logic 1, the processor starts the internal RC oscillator if necessary and exits the reset state within four clock cycles (Figure 16-10) and begins program execution at address 8000h.

16.6.3 Internal System Reset

The MAXQ7667 supports internal system reset capability from in-system programming mode. An internal system reset is generated when the ROD bit (SC.2) in the SC register is set when the SPE bit (ICDF.2) in the ICDF register is also set. The bootloader software can use this capability to initiate an internal system reset when the flash loader completes its operation. See *Section 13: In-System Programming/Bootloader* for more details on in-system programming.

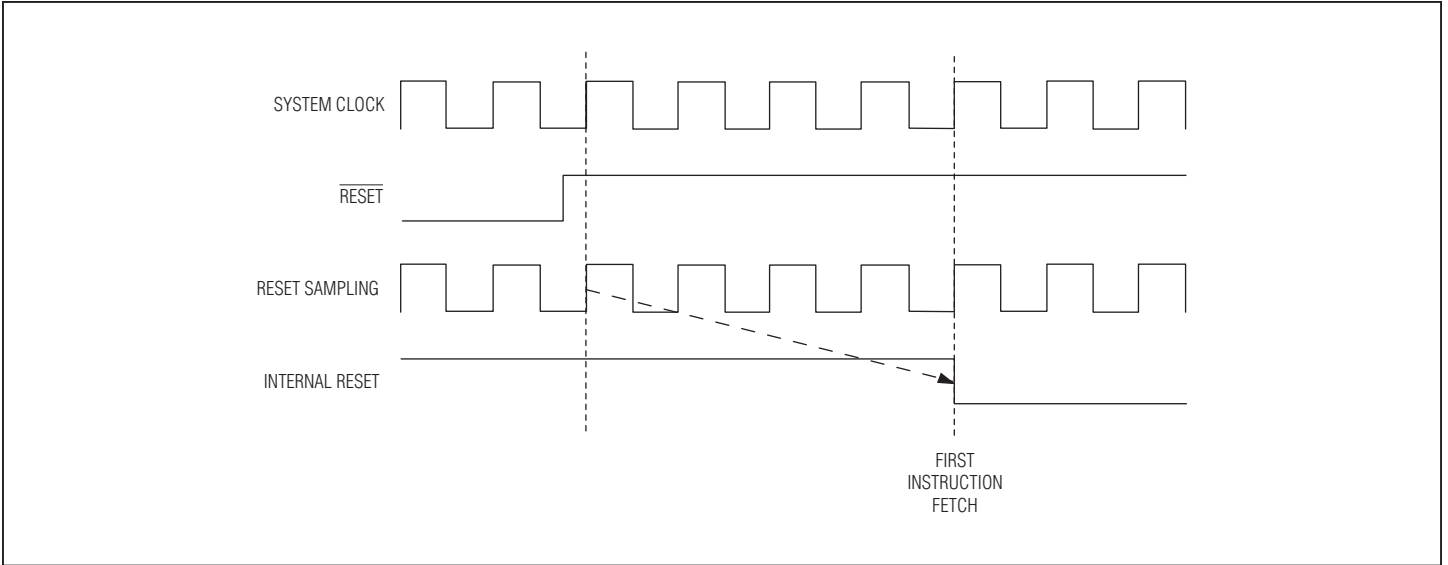


Figure 16-10. MAXQ7667 External Reset

SECTION 17: ULTRASONIC DISTANCE MEASUREMENT MODULE—BURST TRANSMISSION AND ECHO RECEPTION

This section contains the following information:

17.1 Architecture	17-4
17.1.1 Burst Transmission Stage	17-4
17.1.2 Echo Reception	17-4
17.2 Burst Transmission and Echo Reception Pinouts	17-6
17.3 Echo Burst Transmission and Burst Echo Reception Registers	17-6
17.3.1 Burst Pulse-Width High Control Register (BPH)	17-6
17.3.2 Burst Transmit Control Register (BTRN)	17-7
17.3.3 Echo Path Control Register (RCVC)	17-8
17.3.4 PLL-Based Programmable Oscillator Control Register (PLLFC)	17-9
17.3.5 Analog Interrupt Enable Register (AIE)	17-10
17.3.6 Echo Envelope Comparator Control Register (CMPC)	17-11
17.3.7 Echo Envelope Comparator Threshold Register (CMPT)	17-11
17.3.8 Analog Status Register (ASR)	17-12
17.3.9 Echo Envelope Lowpass Filter FIFO Control Register (LPFC)	17-13
17.3.10 Echo Envelope Bandpass Filter Input Data Register (BPFI)	17-14
17.3.11 Echo Envelope Bandpass Filter Output Data Register (BPFO)	17-14
17.3.12 Echo Envelope Lowpass Filter Output Data Register (LPFD)	17-15
17.3.13 Echo Envelope Lowpass Filter FIFO Output Register (LPFF)	17-15
17.3.14 Analog Power Enable Register (APE)	17-16
17.4 Burst Signal Generation	17-17
17.4.1 Setting Up the Hardware	17-17
17.4.1.1 External RC Filter (FILT)	17-17
17.4.1.2 Ultrasonic Transducer	17-17
17.4.2 Setting Up the Registers in Software	17-17
17.4.2.1 Turning On the PLL and the Clock Sections	17-17
17.4.2.2 Configuring the Burst Variables	17-18
17.4.2.3 Triggering the Burst Signal	17-20
17.4.3 Burst Generation Example	17-20

17.5 Echo Reception	17-21
17.5.1 Hardware Setup	17-21
17.5.1.1 Receiver Clock Frequency	17-21
17.5.1.2 Voltage Reference for the Sigma-Delta ADC	17-21
17.5.1.3 External Input Coupling Capacitors	17-21
17.5.2 Software Setup	17-21
17.5.2.1 Powering Up Receiver Hardware	17-21
17.5.2.2 Echo Receive Register Configuration	17-22
17.5.3 Echo Receive Data Register	17-24

LIST OF FIGURES

Figure 17-1. Burst Transmission Stage	17-4
Figure 17-2. Echo Reception Stage	17-5
Figure 17-3. Circuit for Ultrasonic Distance Measurement	17-17
Figure 17-4. Burst Pulse Example	17-20
Figure 17-5. Comparator Hysteresis	17-23

LIST OF TABLES

Table 17-1. Burst Transmission Pinout	17-6
Table 17-2. Echo Reception Pinout	17-6
Table 17-3. Voltage Reference Pinout	17-6
Table 17-4. PLLC Settings to Identify the Frequency of the Crystal to the PLL Stage	17-18
Table 17-5. Integer Divisor Values	17-19
Table 17-6. Echo Reception Input Mux Selection	17-22
Table 17-7. Effective Gain for the Echo Reception Stage	17-22
Table 17-8. FIFO Load Source Selection	17-23
Table 17-9. Echo Receive Path Interrupts	17-24

SECTION 17: ULTRASONIC DISTANCE MEASUREMENT MODULE—BURST TRANSMISSION AND ECHO RECEPTION

The MAXQ7667 was developed for ultrasonic time-of-flight distance measurement. To accomplish this task several hardware blocks were added to the basic MAXQ. These blocks can be divided into two functioning units: burst transmission and echo reception.

Typically, the burst transmission stage sends out an ultrasonic sound wave to a surface, from which the distance is to be measured, and the echo reception stage listens to the reflected burst frequency (echo). The echo reception stage is synchronized to the burst frequency and only processes the echo signals that are of the same frequency as the burst signal.

The burst transmission stage has the following features:

- The burst frequency signal can be generated using the system clock (SYSCLK) or the phase-locked loop (PLL) oscillator.
- The PLL is programmable and allows any frequency between 18.8kHz and 1.67MHz (with a worst-case resolution of 0.13%).
- Selectable polarity of the output pulses.
- Programmable duty cycle.
- Programmable number of pulses (1 to 255) in a single burst.
- The burst output pin can be set to a three-state mode.

The echo reception stage has the following features:

- The circuit is synchronized with the burst transmission unit to receive echoes at the burst frequency for burst frequencies in the range of 25kHz to 100kHz.
- Detects echoes down to 10 μ V_{p-p}.
- Programmable amplification with 23.5dB range.
- 16-bit sigma-delta ADC to digitize the echo signal.
- Bandpass filter (BPF) to reduce noise.
- Full-wave detector and lowpass filter (LPF) (with several stages) produce a clean echo envelope.
- FIFO stores up to 8 LPF readings.
- FIFO can be loaded by software, LPF data ready, or a timer.
- Digital comparator with interrupt to indicate when output of the LPF exceeds a programmable threshold.
- Simulated echo signal (derived from burst signal) for testing and debugging.

17.1 Architecture

17.1.1 Burst Transmission Stage

The MAXQ7667's burst output excites the transducer when transmitting a burst of ultrasonic sound. The burst output is typically used to switch an external transistor that drives a high voltage transformer, which excites the transducer.

The burst signal can be derived from either the system clock or from a programmable oscillator (PLL) that is phase locked to the system clock. Deriving the burst signal from the system clock limits the burst frequency to one of 16 choices. These 16 frequencies are generated by integer division of the system clock.

Using the PLL allows a fractional division of the system clock. Fractional division allows the PLL to be programmed for any frequency within its range with a worst-case programming resolution of 0.13%.

Figure 17-1 shows a block diagram of the echo receive path.

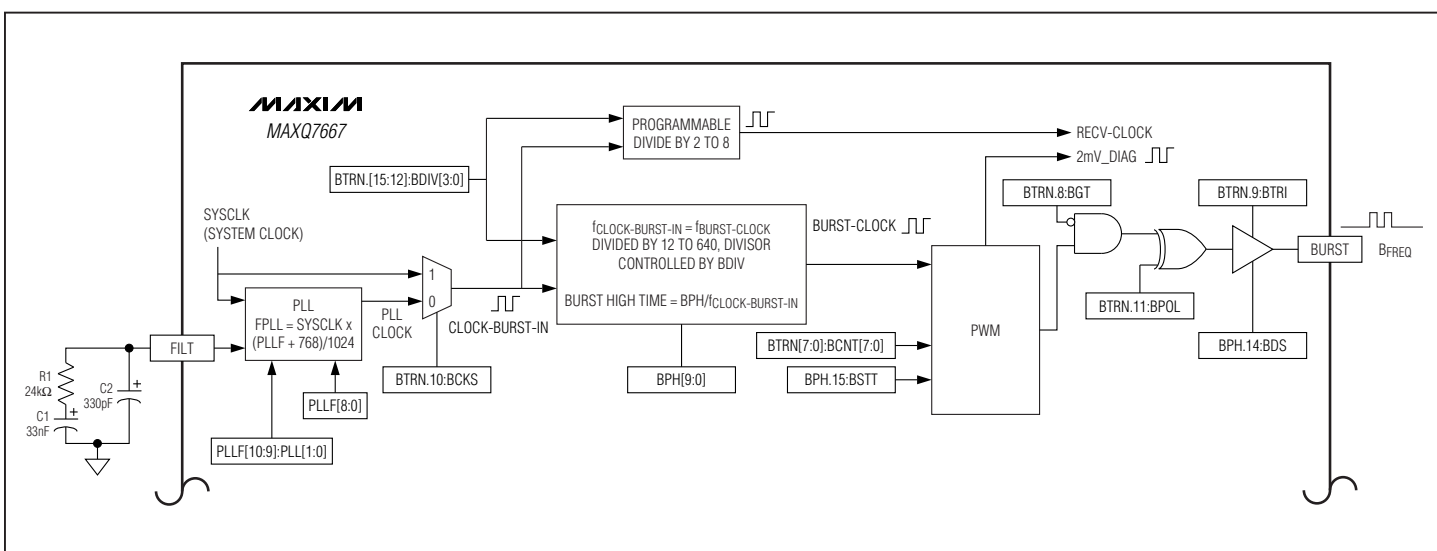


Figure 17-1. Burst Transmission Stage

17.1.2 Echo Reception

The echo receive path along with the control and data registers are shown in Figure 17-2. The circuit is designed to receive echoes at the burst frequency, with amplitudes ranging from 10μV_{P-P} to 100mV_{P-P}. Echo signals between 100mV and 2V are clipped and do not saturate the receiver. This allows very large echoes to be detected even though information about their magnitude will be lost. To optimize echo reception, the clock used for processing the echo is locked to the burst frequency (BFREQ). This guarantees that the center frequency of the BPF matches the transducer excitation frequency.

At the beginning of the signal chain is a wideband low-noise amplifier (LNA) with a fixed gain of 40V/V. For debugging purposes the output of the LNA's differential output can be monitored with an oscilloscope by connecting it to pins AIN0 and AIN1. For more details on this capability, see *Section 17.5.2.2*.

The total gain of the echo receive path is programmable over a 23.5dB range with an average gain step of 0.8dB. Changes in gain are achieved through a combination of analog and digital techniques. All gain changes settle within one ADC conversion, and any switching glitches are removed by the LPF. This allows for the implementation of a virtual time variable gain amplifier since gain changes can be made on the fly.

The 16-bit sigma-delta ADC digitizes the output of the LNA. The ADC's input sample rate is 80 x BFREQ and the output data rate is 10 x BFREQ. The output of the ADC's SINC filter is fed directly to the input of the BPF and can be read from the Echo Envelope Bandpass Filter Input Data register (BPIF). The reference voltage for the sigma-delta ADC can be provided by two sources: an internal bandgap reference or an external reference.

The BPF center frequency tracks BFREQ. The bandpass width is 14% of the center frequency ($Q = 7$). The 16-bit Echo Envelope Bandpass Filter Input Data register (BPIF) and Echo Envelope Bandpass Filter Output Data register (BPFO) data are available in two's complement format at a data rate equal to $10 \times \text{BFREQ}$. The BPIF and BPFO registers are typically used only for debugging purposes. They may not be synchronized to the CPU system clock and can only be reliably observed when the system clock is used as the receive path clock.

A full-wave detector and LPF convert the AC output of the BPF to a DC value that represents the envelope of the echo waveform. The output of the LPF in the Echo Envelope Lowpass Filter Output Data register (LPFD) is in straight binary format and is updated at a rate equal to $5 \times \text{BFREQ}$. The output of the LPF can be read directly or it can be loaded into an 8-words deep FIFO.

The output of the LPF is automatically sent to the input of a 16-bit digital comparator. The comparator has a programmable threshold level and can be used to detect the time at which the echo amplitude (LPF output) crosses a given threshold. This technique allows for simple echo detection that can automatically stop a timer and generate an interrupt.

Figure 17-2 shows a block diagram of the echo receive path.

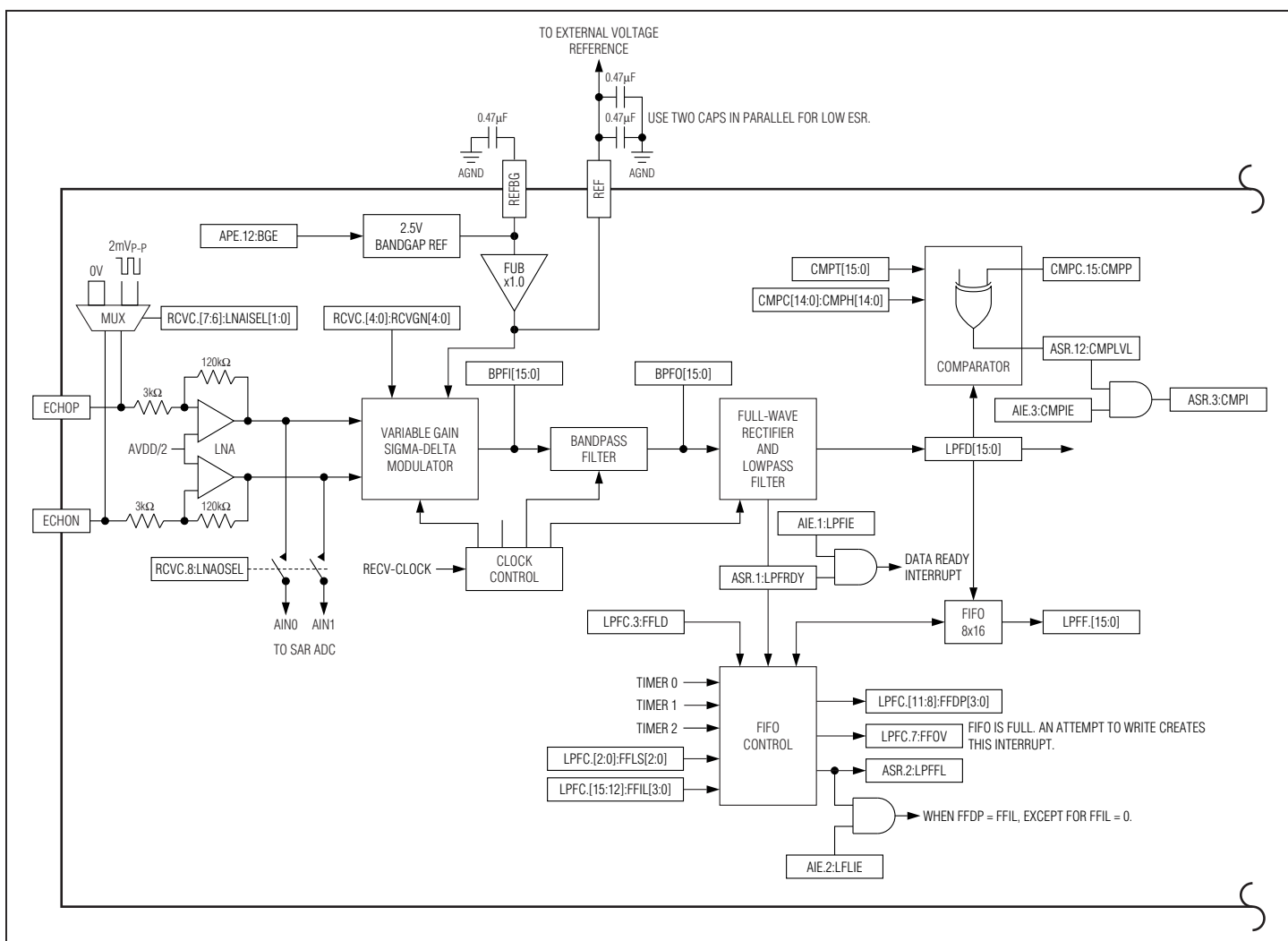


Figure 17-2. Echo Reception Stage

17.2 Burst Transmission and Echo Reception Pinouts

Table 17-1. Burst Transmission Pinout

BURST TRANSMISSION SIGNAL	PIN	FUNCTION
BURST	45	Burst Output. Provides transmit pulses to ultrasonic transducer. Output polarity is programmable. Powers up in three-state mode.
FILT	26	Filter pin for burst frequency phase-locked loop (PLL).

Table 17-2. Echo Reception Pinout

ECHO RECEPTION SIGNAL	PIN	FUNCTION
ECHOP	30	Positive Echo Input. Capacitively coupled from ultrasonic transducer.
ECHON	29	Negative Echo Input. Capacitively coupled from ultrasonic transducer.

Table 17-3. Voltage Reference Pinout

VOLTAGE REFERENCE	PIN	FUNCTION
REFBG	35	Internal 2.5V Reference Output. Connect a minimum value of 0.47 μ F bypass capacitor to AGND.
REF	34	ADC Reference Input and Reference Buffer Output. This pin is for the ADC reference input. The buffer connected to the REFBG pin must be disabled to allow the pin to accept an external reference input. Provide a bypass to AGND with a 0.47 μ F capacitor. This pin requires a low ESR. This can be done by using two capacitors in parallel instead of one, e.g., a 1 μ F capacitor in parallel with a 10nF capacitor instead of just a 1 μ F capacitor.

17.3 Echo Burst Transmission and Burst Echo Reception Registers

17.3.1 Burst Pulse-Width High Control Register (BPH)

Register Description: **Burst Pulse-Width High Control Register**
 Register Name: **BPH**
 Register Address: **Module 05h, Index 00h**

Bit #	15	14	13	12	11	10	9	8
Name	BSTT	BDS	—	—	—	—	BPH9	BPH8
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	r	r	r	r	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	BPH7	BPH6	BPH5	BPH4	BPH3	BPH2	BPH1	BPH0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: BPH is cleared to 0000h on all forms of reset.

Bit 15: Burst Start (BSTT). Setting this bit to 1 starts a burst transmission with frequency, duty cycle, and number of pulses determined by BDIV, BPH, and BCNT. This bit remains 1 until the burst transmission has completed, at which point it is automatically reset to 0. Changing BSTT to 0 in software is blocked by hardware.

Bit 14: Burst Drive Strength (BDS). This bit selects the drive strength on the BURST pin. When the BDS bit is set to 0, the output driver is lower; when set to 1, the output driver is higher.

Bits 13 to 10: Reserved. Read returns 0.

Bits 9 to 0: Burst Pulse-Width High (BPH[9:0]). Defines the number of PLL clock cycles (BCKS = 0), or system clock cycles (BCKS = 1) in each burst high pulse. When BPH is zero or one, the burst pulse-width high is 1 cycle. When BPH is equal to or greater than the burst clock-cycle width, the burst pulse-width high is one cycle less than the clock cycle width (i.e., pulse-width low is one cycle).

17.3.2 Burst Transmit Control Register (BTRN)

Register Description: **Burst Transmit Control Register**
 Register Name: **BTRN**
 Register Address: **Module 05h, Index 01h**

Bit #	15	14	13	12	11	10	9	8
Name	BDIV3	BDIV2	BDIV1	BDIV0	BPOL	BCKS	BTRI	BGT
Reset	1	0	0	1	0	1	1	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	BCNT7	BCNT6	BCNT5	BCNT4	BCNT3	BCNT2	BCNT1	BCNT0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: BTRN is cleared to 9600h on all forms of reset.

Bits 15 to 12: Burst Clock-Divide Select (BDIV[3:0]). Selects the divide ratio between the clock source specified by BCKS and the burst and receive path clocks. Note that the receive path clock is not active for burst divide ratios of less than 160. Defaults to 1001 upon reset.

BDIV	PLL CLOCK CYCLES	
	PER RECEIVE CLOCK CYCLE (Rdivisor)	PER BURST CLOCK CYCLE (Bdivisor)
0000 (0h)	—	12
0001 (1h)	—	16
0010 (2h)	—	20
0011 (3h)	—	28
0100 (4h)	—	36
0101 (5h)	—	48
0110 (6h)	—	64
0111 (7h)	—	90
1000 (8h)	—	120
1001 (9h)	2	160
1010 (Ah)	3	240
1011 (Bh)	4	320
1100 (Ch)	5	400
1101 (Dh)	6	480
1110 (Eh)	7	560
1111 (Fh)	8	640

Bit 11: Burst Polarity Control (BPOL). Setting this bit to 0 causes a low idle state and a high pulse width of BPH. Set BPOL 1 to invert the BURST output, idling high with BPH specifying the low period within the period.

Bit 10: Burst Clock Source Select (BCKS). Setting this bit to 0 selects the PLL output as the clock source for burst transmission and for the echo receive path. Setting this bit to 1 selects the system clock as the clock source for burst transmission and for the echo receive path. The reset state sets this bit to a 1, selecting the system clock.

Bit 9: Burst Three-State (BTRI). When set to 1, this bit places the burst output pad in a high-impedance state to allow internal transmit/receive path testing.

Bit 8: Burst Gate (BGT). When set to 1, this bit immediately gates the output of the burst pulse, driving the output to its nonasserted state (dependent on the polarity selection BPOL).

Bits 7 to 0: Burst Pulse Count (BCNT[7:0]). Defines the number of burst pulses (at least 1) transmitted each time BSTT is set to 1. When BCNT is 0, there is still one pulse to be transmitted.

17.3.3 Echo Path Control Register (RCVC)

Register Description: **Echo Path Control Register**
 Register Name: **RCVC**
 Register Address: **Module 05h, Index 03h**

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

Bit #	7	6	5	4	3	2	1	0
Name	LNAISEL1	LNAISEL0	—	RCVGN4	RCVGN3	RCVGN2	RCVGN1	RCVGN0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	r	rw	rw	rw	rw	rw

r = read, w = write

Note: RCVC is cleared to 0000h on all forms of reset.

Bits 15 to 9 and 5: Reserved. Read returns 0.

Bit 8: LNA Output Mux Select (LNAOSEL). When set to 1, this bit routes LNA outputs to AIN0 and AIN1.

Bits 7 and 6: LNA Input Mux Select (LNAISEL[1:0]). Determines LNA input connections.

- 00 selects ECHOP and ECHON
- 01 selects AGND vs. AGND
- 10 selects 2mVP-P vs. AGND
- 11 is reserved

Bits 4 to 0: Echo Receive Path Gain Control (RCVGN[4:0])

- 00000 selects gain of 32dB
- 11111 selects gain of 50dB

17.3.4 PLL-Based Programmable Oscillator Control Register (PLLF)

Register Description: **PLL-Based Programmable Oscillator Control Register**

Register Name: **PLLF**

Register Address: **Module 05h, Index 04h**

Bit #	15	14	13	12	11	10	9	8
Name	—	—	—	—	—	PLLC1	PLLC0	PLLF8
Reset	0	0	0	0	0	0	0	1
Access	r	r	r	r	r	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	PLLF7	PLLF6	PLLF5	PLLF4	PLLF3	PLLF2	PLLF1	PLLF0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: PLLF is cleared to 0100h on all forms of reset.

Bits 15 to 11: Reserved. Read returns 0.

Bits 10 and 9: PLL Reference Clock Divide Control (PLLC[1:0]). Selects reference clock-divide ratio.

00 selects 1024 to support a 16MHz system clock

01 selects 512 to support an 8MHz system clock

10 selects 256 to support a 4MHz system clock

11 selects 128 to support a 2MHz system clock

Bits 8 to 0: PLL Fine Frequency Control (PLLF[8:0]). PLL output clock frequency is $(768 + PLLF) \times (SYSCLK)/1024$, where SYSCLK is the system clock frequency and PLLF represents a value from 0 to 511. PLLF defaults to 256 upon reset.

17.3.5 Analog Interrupt Enable Register (AIE)

Register Description: **Analog Interrupt Enable Register**
 Register Name: **AIE**
 Register Address: **Module 05h, Index 05h**

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	XTIE	VIBIE	VDBIE	VABIE	CMPIE	LFLIE	LPFIE	SARIE
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: AIE is cleared to 0000h on all forms of reset.

Bits 15 to 8: Reserved. Read returns 0.

Bit 7: Crystal Oscillator Failure (XTIE). See Section 15 for details on this bit.

Bit 6: DVDDIO Brownout Interrupt Enable (VIBIE). See Section 16 for details on this bit.

Bit 5: DVDD Brownout Interrupt Enable (VDBIE). See Section 16 for details on this bit.

Bit 4: AVDD Brownout Interrupt Enable (VABIE). See Section 16 for details on this bit.

Bit 3: Echo Envelope Comparator Interrupt Enable (CMPIE). When set to 1, this bit allows an interrupt request to be generated when the echo envelope comparator crosses the threshold. See Figure 17-5 and the description of the CMPP bit (CMPC.15) for further details. The interrupt, when enabled, is triggered by changes in the CMPLVL signal (ASR.12).

Bit 2: Echo Envelope Lowpass Filter FIFO Full Interrupt Enable (LFLIE). When set to 1, this bit allows an interrupt request to be generated when the echo receive path FIFO is full or has overflowed. The interrupt, when enabled, is triggered by the FIFO full flag, LPFFL (ASR.2), and an internal overflow flag.

Bit 1: Echo Envelope Lowpass Filter Output Data Ready Interrupt Enable (LPFIE). When set to 1, this bit allows an interrupt request to be generated when the echo receive path LPFD register receives a new value. The interrupt, when enabled, is triggered by changes in the LPFRDY signal (ASR.1).

Bit 0: SAR ADC Data Ready Interrupt Enable (SARIE). See Section 14 for details on this bit.

17.3.6 Echo Envelope Comparator Control Register (CMPC)

Register Description: **Echo Envelope Comparator Control Register**
 Register Name: **CMPC**
 Register Address: **Module 05h, Index 06h**

Bit #	15	14	13	12	11	10	9	8
Name	CMPP	CMPH14	CMPH13	CMPH12	CMPH11	CMPH10	CMPH9	CMPH8
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	CMPH7	CMPH6	CMPH5	CMPH4	CMPH3	CMPH2	CMPH1	CMPH0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: CMPC is cleared to 0000h on all forms of reset.

Bit 15: Comparator Output Polarity (CMPP). Set CMPP = 0 to generate an interrupt on the rising edge of the CMPLVL signal. Set CMPP = 1 to generate an interrupt on the falling edge of the CMPLVL signal.

Bits 14 to 0: Comparator Hysteresis (CMPH[14:0]). This register holds the user-adjustable comparator hysteresis. CMPLVL is reset to 0 when LPFD is less than CMPT minus CMPH.

17.3.7 Echo Envelope Comparator Threshold Register (CMPT)

Register Description: **Echo Envelope Comparator Threshold Register**
 Register Name: **CMPT**
 Register Address: **Module 05h, Index 07h**

Bit #	15	14	13	12	11	10	9	8
Name	CMPT15	CMPT14	CMPT13	CMPT12	CMPT11	CMPT10	CMPT9	CMPT8
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	CMPT7	CMPT6	CMPT5	CMPT4	CMPT3	CMPT2	CMPT1	CMPT0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Note: CMPT is cleared to 0000h on all forms of reset.

Bits 15 to 0: Comparator Threshold (CMPT[15:0]). This register holds the comparator threshold. CMPLVL is set to 1 when LPFD exceeds CMPT.

17.3.8 Analog Status Register (ASR)

Register Description: **Analog Status Register**
 Register Name: **ASR**
 Register Address: **Module 05h, Index 08h**

Bit #	15	14	13	12	11	10	9	8
Name	VIOLVL	DVLVL	AVLVL	CMPLVL	—	—	—	XTRDY
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	XTI	VIBI	VDBI	VABI	CMPI	LPFFL	LPFRDY	SARRDY
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: ASR bits 3:0 are cleared to 0h on reset; bits 8:4 clear to 0h on power-on reset.

Bit 15: DVDDIO Brownout Comparator Output Level (VIOLVL). See Section 16 for details on this bit.

Bit 14: DVDD Brownout Comparator Output Level (DVLVL). See Section 16 for details on this bit.

Bit 13: AVDD Brownout Comparator Output Level (AVLVL). See Section 16 for details on this bit.

Bit 12: Echo Envelope Comparator Output Level (CMPLVL). This bit always reflects the echo envelope comparator's current state when read.

Bits 11 to 9: Reserved. Read returns 0.

Bit 8: Crystal Oscillator Ready (XTRDY). See Section 15 for details on this bit.

Bit 7: Crystal Oscillator Failure Interrupt Flag (XTI). See Section 15 for details on this bit.

Bit 6: DVDDIO Brownout Interrupt Flag (VIBI). See Section 16 for details on this bit.

Bit 5: DVDD Brownout Interrupt Flag (VDBI). See Section 16 for details on this bit.

Bit 4: AVDD Brownout Interrupt Flag (VABI). See Section 16 for details on this bit.

Bit 3: Echo Envelope Comparator Interrupt Flag (CMPI). Set to 1 when envelope comparator output is 1.

Bit 2: Echo Envelope Lowpass Filter FIFO Full Interrupt Flag (LPFFL). Set to 1 when the FIFO is full or has overflowed.

Bit 1: Echo Envelope Lowpass Filter Output Data Ready Flag (LPFRDY). Set to 1 when the echo envelope lowpass filter completes a calculation.

Bit 0: SAR ADC Data Ready Flag (SARRDY). See Section 14 for details on this bit.

17.3.9 Echo Envelope Lowpass Filter FIFO Control Register (LPFC)

Register Description: **Echo Envelope Lowpass Filter FIFO Control Register**

Register Name: **LPFC**

Register Address: **Module 05h, Index 0Ah**

Bit #	15	14	13	12	11	10	9	8
Name	FFIL3	FFIL2	FFIL1	FFIL0	FFPD3	FFPD2	FFPD1	FFDP0
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	r	r	r	r

Bit #	7	6	5	4	3	2	1	0
Name	FFOV	—	—	—	FFLD	FFLS2	FFLS1	FFLS0
Reset	0	0	0	0	0	1	1	1
Access	r	r	r	r	rw	rw	rw	rw

r = read, w = write

Note: LPFC is cleared to 0007h on all forms of reset.

Bits 15 to 12: Lowpass Filter FIFO Interrupt Level (FFIL[3:0]). An interrupt is issued when the FIFO depth indicator FFDP reaches the level set by FFIL[3:0] (excluding 000, which disables interrupt generation due to fill level, leaving only overflow interrupts to be generated).

Bits 11 to 8: Lowpass Filter FIFO Depth (FFDP[3:0]). Indicates the number of samples in the FIFO (0 to 8). The FIFO is full when it has eight data items in it (i.e., the 11th bit denotes FIFO full).

Bit 7: Lowpass Filter FIFO Overflow Indicator (FFOV). This is a sticky bit that indicates that a FIFO write has been attempted while the FIFO was full (that data was blocked and is lost). An interrupt is generated on an overflow if the interrupt has been enabled.

Bits 6 to 4: Reserved. Read returns 0.

Bit 3: Lowpass Filter FIFO Load (FFLD). Writing a 1 to this bit loads LPFD into the LPFF FIFO if FFLS is set to 111. Writing a 0 has no effect. This bit always reads 0.

Bits 2 to 0: Lowpass Filter FIFO Load Control Source Select (FFLS[2:0]). These bits select the source that initiates a FIFO load.

000 selects timer 0

001 selects timer 1

010 selects timer 2

011, 100, and 101 are reserved

110 selects continuous loading at each LPFRDY condition

111 selects FFLD bit

17.3.10 Echo Envelope Bandpass Filter Input Data Register (BPF1)

Register Description: **Echo Envelope Bandpass Filter Input Data Register**
 Register Name: **BPF1**
 Register Address: **Module 05h, Index 0Ch**

Bit #	15	14	13	12	11	10	9	8
Name	BPF15	BPF14	BPF13	BPF12	BPF11	BPF10	BPF9	BPF8
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	BPF7	BPF6	BPF5	BPF4	BPF3	BPF2	BPF1	BPF0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: BPF1 is cleared to 0000h on all forms of reset.

Bits 15 to 0: Bandpass Filter Input Data (BPF1[15:0]). Automatically loaded with echo-receive path ADC output data.

17.3.11 Echo Envelope Bandpass Filter Output Data Register (BPFO)

Register Description: **Echo Envelope Bandpass Filter Output Data Register**
 Register Name: **BPFO**
 Register Address: **Module 05h, Index 0Dh**

Bit #	15	14	13	12	11	10	9	8
Name	BPFO15	BPFO14	BPFO13	BPFO12	BPFO11	BPFO10	BPFO9	BPFO8
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	BPFO7	BPFO6	BPFO5	BPFO4	BPFO3	BPFO2	BPFO1	BPFO0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: BPFO is cleared to 0000h on all forms of reset.

Bits 15 to 0: Bandpass Filter Output Data (BPFO[15:0])

17.3.12 Echo Envelope Lowpass Filter Output Data Register (LPFD)

Register Description: **Echo Envelope Lowpass Filter Output Data Register**

Register Name: **LPFD**

Register Address: **Module 05h, Index 0Eh**

Bit #	15	14	13	12	11	10	9	8
Name	LPFD15	LPFD14	LPFD13	LPFD12	LPFD11	LPFD10	LPFD9	LPFD8
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	LPFD7	LPFD6	LPFD5	LPFD4	LPFD3	LPFD2	LPFD1	LPFD0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: LPFD is cleared to 0000h on all forms of reset.

Bits 15 to 0: Lowpass Filter Data Output (LPFD[15:0])

17.3.13 Echo Envelope Lowpass Filter FIFO Output Register (LPFF)

Register Description: **Echo Envelope Lowpass Filter FIFO Output Register**

Register Name: **LPFF**

Register Address: **Module 05h, Index 0Fh**

Bit #	15	14	13	12	11	10	9	8
Name	LPFF15	LPFF14	LPFF13	LPFF12	LPFF11	LPFF10	LPFF9	LPFF8
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	LPFF7	LPFF6	LPFF5	LPFF4	LPFF3	LPFF2	LPFF1	LPFF0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Note: LPFF is cleared to 0000h on all forms of reset.

Bits 15 to 0: Lowpass Filter FIFO Data Output (LPFF[15:0]). The FIFO holds up to eight samples.

17.3.14 Analog Power Enable Register (APE)

Register Description: **Analog Power Enable Register**
 Register Name: **APE**
 Register Address: **Module 05h, Index 10h**

Bit #	15	14	13	12	11	10	9	8
Name	—	RBUFE	—	BGE	LRIOPD	LRDPD	LRAPD	VIBE
Reset	0	0	0	0	0	0	0	0
Access	r	rw	r	rw	rw	rw	rw	rw

Bit #	7	6	5	4	3	2	1	0
Name	VDPE	VDBE	VABE	SARE	PLLE	MDE	LNAE	BIASE
Reset	1	0	0	0	0	0	0	0
Access	rw	rw	rw	r	rw	rw	rw	rw

r = read, w = write

Note: APE is cleared to 0080h on all forms of reset.

Bits 15 and 13: Reserved. Read returns 0.

Bit 14: Reference Buffer Enable (RBUFE). The reference buffer is enabled when set to 1 and disabled when set to 0.

Bit 12: Bandgap Enable (BGE). The reference buffer is enabled when set to 1 and disabled when set to 0.

Bit 11: I/O Linear Regulator Power-Down (LRIOPD). See *Section 16* for details on this bit.

Bit 10: Digital Linear Regulator Power-Down (LRDPD). See *Section 16* for details on this bit.

Bit 9: Analog Linear Regulator Power-Down (LRAPD). See *Section 16* for details on this bit.

Bit 8: I/O Voltage Brownout Detection Enable (VIBE). See *Section 16* for details on this bit.

Bit 7: Digital Voltage Reset Enable (VDPE). See *Section 16* for details on this bit.

Bit 6: Digital Voltage Brownout Detection Enable (VDBE). See *Section 16* for details on this bit.

Bit 5: Analog Voltage Brownout Detection Enable (VABE). See *Section 16* for details on this bit.

Bit 4: SAR Enable (SARE). See *Section 14* for details on this bit.

Bit 3: PLL Enable (PLLE). When set to 1, this bit enables PLL. When set to 0, this bit disables PLL and places it in a leakage-only state.

Bit 2: Sigma-Delta Modulator Enable (MDE). When set to 1, this bit enables the sigma-delta modulator. When set to 0, this bit disables the modulator and places it in a leakage-only state.

Bit 1: LNA Enable (LNAE). When set to 1, this bit enables LNA. When set to 0, this bit disables LNA and places it in a leakage-only state.

Bit 0: Bias Enable (BIASE). When set to 1, this bit enables the current bias generator. When set to 0, this bit disables the current bias generator and places it in a leakage-only state. BIASE is automatically set to 1 whenever LNAE, MDE, PLLE, SARE, BGE or RBUFE is set to 1.

17.4 Burst Signal Generation

To facilitate implementation the following items must be addressed:

- **Hardware Setup:**
 - External RC Filter (FILT)
 - Ultrasonic Transducer
- **Software Setup:**
 - Turning On the PLL and the Clock Sections
 - Configuring the Burst Registers
 - Triggering the Burst Signal

17.4.1 Setting Up the Hardware

17.4.1.1 External RC Filter (FILT)

An external filter must be connected to the FILT pin if the PLL is used. This filter consists of two capacitors and one resistor ($R1 = 24k\Omega$, $C1 = 0.033\mu F$, and $C2 = 330pF$) as shown in Figure 17-1. These values are appropriate for the entire PLL frequency range. These components are used to filter the analog voltage that controls the voltage-controlled oscillator, VCO, in the PLL. The FILT pin can be monitored with an oscilloscope to gauge the settling time of a change in the PLL frequency setting.

17.4.1.2 Ultrasonic Transducer

There are many ways to connect the ultrasonic transducer to the BURST pin. Figure 17-3 shows one of the possibilities.

17.4.2 Setting Up the Registers in Software

17.4.2.1 Turning On the PLL and the Clock Sections

To reduce power consumption, many sections of the MAXQ7667 are not powered up unless needed. To turn on the PLL, set the PLLE bit (APE.3) to 1. The MAXQ7667 operates from the internal RC oscillator at power-up and only switches to the crystal-controlled clock if instructed to by the software. To turn on the precision system clock that is controlled by the external crystal or resonator, set the XTE bit (OSCC.1) to 1 (see Section 15).

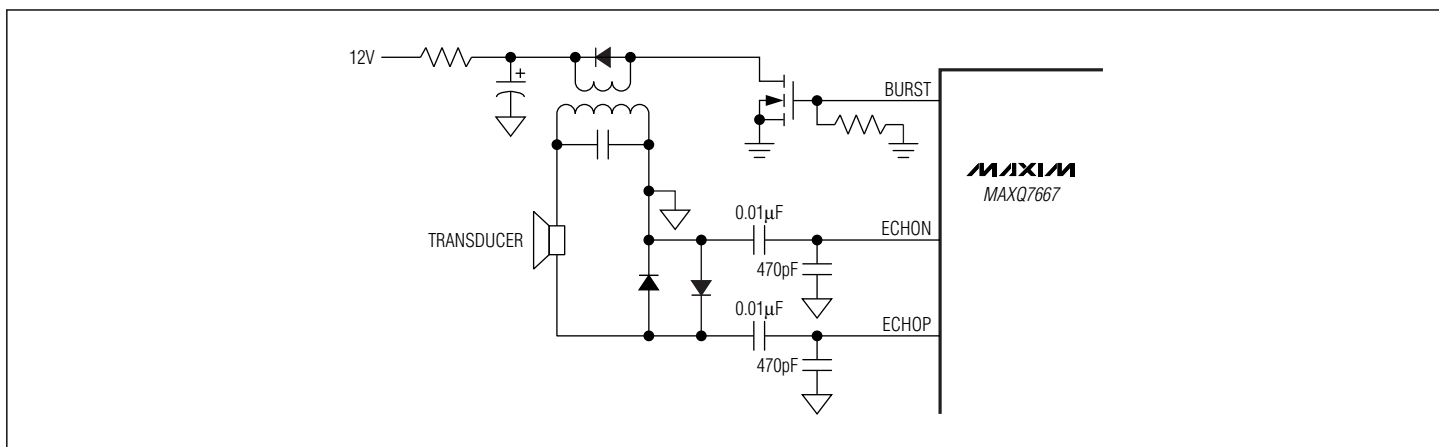


Figure 17-3. Circuit for Ultrasonic Distance Measurement

17.4.2.2 Configuring the Burst Variables

The Clock-Burst-In (Figure 17-1) is selected by the burst clock select bit, BCKS (BTRN.10). The Clock-Burst-In is the clock used to generate the burst frequency (BFREQ) and the Recv-Clock that feeds the echo reception section. If the BCKS bit is set to 1, the system clock is selected; otherwise, if BCKS = 0 the programmable PLL oscillator is selected. The reset state sets BCKS to a 1, selecting the system clock as the default.

If PLL is the source for the clock input, the PLL should be configured appropriately. The PLL clock-divide control bits, PLLC[1:0] (PLLF.10:9), identify to the PLL the frequency of the crystal or resonator being used for the system clock. Table 17-4 shows the PLLC settings and the corresponding SYSCLK.

Table 17-4. PLLC Settings to Identify the Frequency of the Crystal to the PLL Stage

PLLC[1:0]	SYSCLK (XTAL/RESONATOR FREQUENCY) (MHz)
00 (default)	16
01	8
10	4

While the MAXQ7667 processor can operate from a wide range of crystal frequencies, the PLL will not. If crystal/resonator frequencies other than those specified in Table 17-4 are used, the PLL can attempt to lock to a frequency outside its designed range and may not function correctly. If the system clock is the internal RC oscillator, make sure to set the RC oscillator to 16MHz; then setting the PLLC bits to 0b00 selects this value correctly.

The PLL clock frequency is dictated by the equation:

$$PLL_{FREQ} = SYSCLK \times [(PLL_{F} + 768)/1024]$$

where PLL_{FREQ} is the output frequency and PLL_{F} is a control value between 0 and 511, set in the PLLF.[8:0] register. This allows for an adjustment of $\pm 25\%$ around the center frequency with a worst-case resolution of 1 part in 1024 or 0.1%.

The PLL clock undergoes further division (Figure 17-1), controlled by the burst divider bits BDIV[3:0] (BTRN.15:12). The BDIV bits are not the actual divisor. See Table 17-5 for actual divisor values. The BDIV bits also divide the receive clock in the echo reception stage. In Table 17-5, the Bdivisor is the actual divisor for the burst transmission stage, while the Rdivisor is the divisor for the receiver clock for the echo reception stage. The Rdivisor value is transparent to the user.

Hence, if the PLL is used as the clock source, the frequency of the burst signal is given by:

$$BFREQ = [SYSCLK \times (PLL_{F} + 768)] / (1024 \times Bdivisor)$$

If system clock, SYSCLK, is chosen as the clock source instead of the PLL, the SYSCLK is divided by the Bdivisor rather than the PLL. Therefore, the frequency of the burst signal is given by:

$$BFREQ = (SYSCLK) / Bdivisor$$

The duty cycle of the burst signal is set by the burst pulse-width high bits, BPH[9:0]. The duty cycle is given by:

$$Duty\ Cycle = 100 \times BPH / Bdivisor \text{ (see Table 17-5 for Bdivisor)}$$

However, if $Bdivisor \leq BPH$:

$$Duty\ Cycle = (Bdivisor - 1) / Bdivisor$$

A single burst has a number of whole cycles that can be programmed through the burst pulse count bits, BCNT.[7:0] (BTRN.7:0). BCNT can be set to any value from 0 to 255, however one cycle is the smallest burst. The burst contains one cycle if $BCNT = 0$ or 1. If $BCNT = 255$, the burst contains 255 cycles.

Table 17-5. Integer Divisor Values

BDIV		Rdivisor	Bdivisor	NOMINAL BURST FREQUENCY IF f _{Clock-Burst-In} = 16MHz (kHz)
BINARY	HEX			
0000	00	—	12	1333
0001	01	—	16	1000
0010	02	—	20	800
0011	03	—	28	571
0100	04	—	36	444
0101	05	—	48	333
0110	06	—	64	250
0111	07	—	90	178
1000	08	—	120	133
1001 (default)	09 (default)	2	160	100
1010	0A	3	240	66.7
1011	0B	4	320	50
1100	0C	5	400	40
1101	0D	6	480	33.3
1110	0E	7	560	28.6
1111	0F	8	640	25

Note: The burst clock-divide select bit selects the divide ratio between the clock source specified by BCKS and the burst and receive path clocks. Note that the receive path clock is not active for burst-divide ratios of less than 160. Defaults to 1001 upon reset.

The burst clock generated can be controlled and manipulated, in a few ways, before it reaches the output pin BURST.

The polarity of the output signal is controlled by the burst polarity bit, BPOL (BTRN.11). Setting it to 0 would keep the signal normally low with high going pulses. Setting BPOL to 1 keeps the signal high with low going pulses.

The drive strength of the burst signal is controlled by the burst drive strength bit, BDS (BPH.14). This bit selects the drive strength on the BURST pin. When the BDS bit is set to 0, the output driver is lower (refer to the datasheet for this value); when set to 1, the output driver is higher (refer to the data sheet for this value).

The burst signal can be stopped from reaching the BURST pin by setting the burst gate bit, BGT (BTRN.8), to 1. Internally the burst signal is still available for diagnostic purposes. Setting BGT to 0 allows the burst signal to pass through to the output pin.

The BURST output pin can be set to three-state mode by setting the burst three-state bit, BTRI (BTRN.9) to 1. At power-up, BTRI is set to 1. This places the BURST output pin in a three-state mode, which prevents large amounts of current from flowing through the external transformer while the MAXQ7667 is being initialized. During the initialization process BPOL and the other burst parameters should be set before BTRI is set to 0, which enables the BURST output pin. A pullup or pulldown resistor should be used to hold the external transistor in the desired state while the MAXQ7667 is being initialized.

17.4.2.3 Triggering the Burst Signal

Once all the register bits have been set to generate the frequency, duty cycle, and number of pulses, and enable the BURST pin, the burst signal can be transmitted out through the BURST output pin by setting the burst start bit, BSTT (BPH.15) to 1. Once a burst sequence is initiated, the sequence progresses to completion, always producing the exact number of whole pulses specified. BSTT remains 1 during this time and automatically resets to 0 at the end of the last whole pulse. Changing BSTT to 0 in software is blocked by the hardware.

17.4.3 Burst Generation Example

The following example shows a burst signal with a frequency of 48kHz and a 45% duty cycle. The crystal frequency is 16MHz.

The register bits are set as follows:

BCKS = 0: Clock source is the PLL.

PLLC = 0b00: 16MHz crystal.

PLL F = 215: Sets the PLL to 15.36MHz, $16\text{MHz} \times (215 + 768)/1024 = 15.36\text{MHz}$.

BDIV = 0b1011: Divides the clock by 320; $\text{BFREQ} = 15.36\text{MHz}/320 = 48\text{kHz}$.

BPH = 144: 45% duty cycle, $144/320 = 0.45$.

BPOL = 0: Output idles low with high going pulses.

BCNT = 3: One burst contains three pulses.

BGT = 0: Allow signal on the BURST pin.

BTRI = 0: Normal state for the BURST pin.

Burst Duration = $3/48\text{kHz} = 62.5\mu\text{s}$

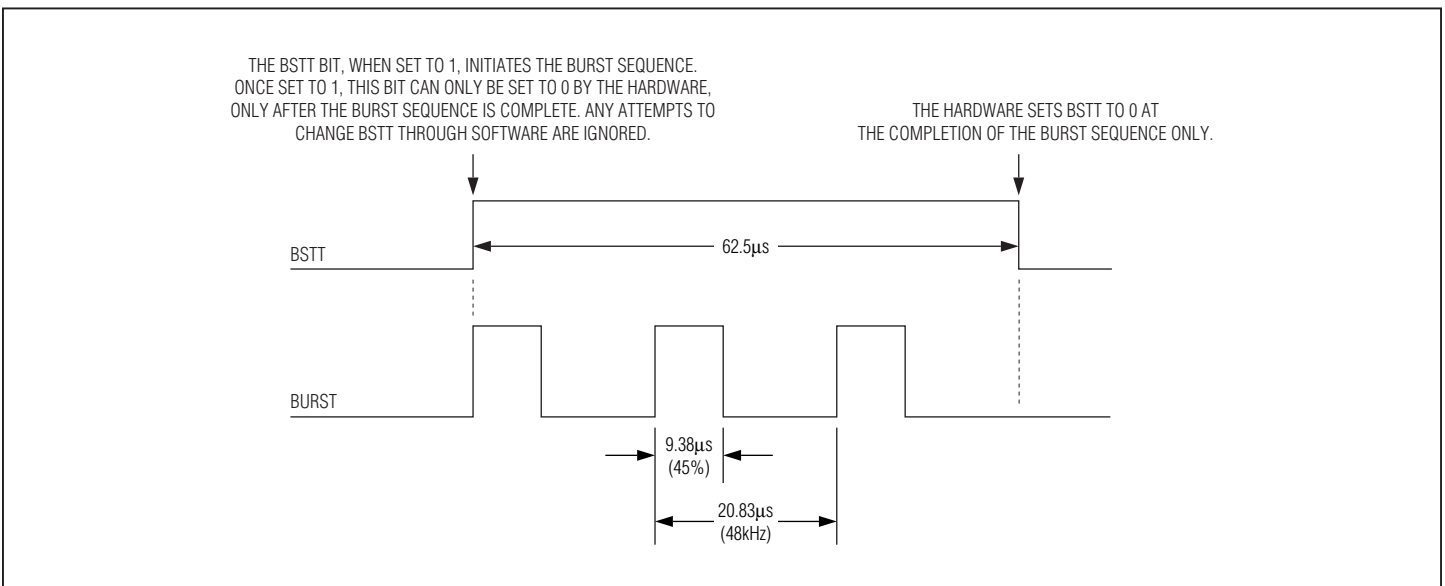


Figure 17-4. Burst Pulse Example

17.5 Echo Reception

To facilitate implementation the following items must be addressed:

- **Hardware Setup:**
 - Receiver Clock Frequency
 - Voltage Reference for the Sigma-Delta ADC
 - External Input Coupling Capacitors
- **Software Setup:**
 - Powering Up Receiver Hardware
 - Echo Reception Register Configuration
 - Echo Reception Interrupts

17.5.1 Hardware Setup

17.5.1.1 Receiver Clock Frequency

The center frequency of the BPF tracks the burst frequency. Therefore, the burst output must be configured even if the BURST pin is not being used. See *Section 17.4: Burst Signal Generation* for proper configuration of the burst output.

The highest usable frequency for the echo receive path is 100kHz (due to hardware limitations). The MAXQ7667 burst generator is capable of higher frequencies, but for frequencies above 100kHz the echo receive path must be implemented externally. The echo envelope can then be measured using the SAR ADC (see *Section 14*). Also note, from Table 17-5, that the BDIV value must be in the 09h–0Fh range because the divisor values less than 09h are not supported by the echo reception stage as indicated by the Rdivisor.

17.5.1.2 Voltage Reference for the Sigma-Delta ADC

The voltage reference for the sigma-delta ADC in the echo reception stage can either be an internal bandgap reference or an external reference. The internal 2.5V bandgap reference can be activated by setting the BGE bit (APE.12) and the RBUFE bit (APE.14) to 1. The BGE bit enables the bandgap reference, while the RBUFE bit enables the buffer at the output of the bandgap. Bypass capacitors of 0.47 μ F should be placed from REF_{BG} to AGND and from REF to AGND. A low ESR is required on the REF pin, therefore, it is advised to put two capacitors in parallel.

To provide the reference from an external reference, the buffer at the output of the bandgap must be turned off, and an external reference voltage ranging between 1V and AVDD volts (analog supply voltage) can be connected to the REF pin. A bypass capacitor of 0.47 μ F should be placed between REF and AGND.

17.5.1.3 External Input Coupling Capacitors

For proper input biasing, the ECHOP and ECHON inputs must be capacitively coupled to the signal source (see Figure 17-2 and Figure 17-3). If the signal source is single-ended, ECHON should be connected through a capacitor to the transducer ground as close to the transducer as possible. Circuit traces and wires from ECHON and ECHOP should be routed next to each other as much as possible. This allows extraneous signals picked up by the traces to be rejected by the MAXQ7667's differential input. The input coupling capacitors should be sized so they are large enough to pass the desired signal with little attenuation but small enough to settle quickly at power-up and to aid in rejecting low-frequency noise. Capacitors of 10nF meet these requirements over the entire 25kHz to 100kHz input range. It is also a good practice to place 470pF capacitors close to the MAXQ7667 from ECHOP to GND and ECHON to GND. This attenuates high-frequency noise that might be present at the echo inputs, and prevents it from aliasing down to the base band. The LNA also prevents aliasing by acting as a first-order LPF with its -3dB corner near 150kHz.

Proper layout techniques, including a ground plane, should be used to minimize the reception of unwanted signals on the echo inputs.

17.5.2 Software Setup

17.5.2.1 Powering Up Receiver Hardware

In addition to powering up the components necessary for the burst clock, it is also necessary to set LNAE (APE.1) and MDE (APE.2) to 1. These bits turn on the LNA, sigma-delta modulator, sigma-delta reference buffer, and the bandgap voltage reference, respectively.

17.5.2.2 Echo Receive Register Configuration

The input of echo reception stage can be configured to accept external (echo) signal on the ECHOP and ECHON pins or internally generated diagnostic signals can be added to the echo signal (see Figure 17-2). Selection of the signal is done by setting the LNA input mux select bits, LNAISEL[1:0] (RCVC.7:6). Table 17-6 gives a description for the different settings of the LNAISEL bits.

Table 17-6. Echo Reception Input Mux Selection

LNAISEL[1:0]	DESCRIPTION
00	Mux open. Only echo signal is connected to the LNA.
01	0V channel selected on the mux. This places a short of approximately 50Ω between ECHOP and ECHON, which reduces the input signal to near 0V.
10	2mV _{PP} channel is selected on the mux. This connects the 2mV _{PP} square wave to ECHOP and ECHON. The 2mV square wave matches the frequency and duty cycle of the burst signal. This feature allows an echo to be simulated that is processed by the entire signal chain. The 2mV square wave is derived from the burst signal but is not affected by BGT or BTRI.
11	Reserved (do not use).

When selecting a new channel on the input mux, both the BPF and the LPF must be allowed to settle before the LPF output provides a valid representation of the input signal. To reduce electrical noise, the echo signal does not pass through the mux and is always connected to the LNA.

The output of the LNA can be monitored by setting the LNA output mux select bit, LNAOSEL (RCVC.8) to 1. The differential output of the LNA gets connected to AIN0 and AIN1 of the SAR ADC (see *Section 14*). This action does not disconnect the LNA from the sigma-delta ADC. When LNAOSEL is set to 1, the output of the LNA can be monitored by connecting an oscilloscope to pins AIN0 and AIN1. Avoid loading AIN0 and AIN1 or injecting signals on them when LNAOSEL = 1, as this will affect the signal going to the sigma-delta ADC. For lowest noise operation, set LNAOSEL to 0, keeping the differential output of the LNA disconnected from AIN0 and AIN1.

The receive gain bits, RCVGN4:0 (RCVC.[4:0]), set the gain of the echo receive path. The amount of amplification is adjustable over a 23.5dB range with an average gain step of 0.8dB. Changes in gain are achieved through a combination of analog and digital techniques. Gain changes settle within one ADC conversion, and any switching glitches are removed by the LPF. This rapid settling time allows a virtual time variable gain amplifier to be created by the software. In a typical application the software sets the gain to a low value when the burst is first sent and then increases the gain (dynamically) with passing time. This allows strong echoes from nearby objects to be processed without clipping, while small signals from distant objects are processed with the maximum gain. All the gain takes place before the BPF. Therefore, ambient acoustic noise and electrical interference are amplified along with the echo signal. To prevent clipping, the entire input signal must be considered when selecting the gain. The available gains and their corresponding RCVGN settings are shown in Table 17-7.

Table 17-7. Effective Gain for the Echo Reception Stage

ECHO RECEIVE PATH GAIN FROM THE ECHO INPUTS TO THE LOWPASS FILTER OUTPUT (GAIN IS CONTROLLED BY THE VALUE IN RCVGN BIT)					
RCVGN	FULL SCALE (mV _{PP})	LPFD RESOLUTION (μV _{PP} /LSB)	RCVGN	FULL SCALE (mV _{PP})	LPFD RESOLUTION (μV _{PP} /LSB)
0b00000	101.58	1.55	0b10000	25.56	0.39
0b00001	90.44	1.38	0b10001	22.94	0.35
0b00010	81.26	1.24	0b10010	20.32	0.31
0b00011	74.05	1.13	0b10011	18.35	0.28
0b00100	68.16	1.04	0b10100	17.04	0.26
0b00101	62.91	0.96	0b10101	15.73	0.24
0b00110	58.33	0.89	0b10110	14.42	0.22
0b00111	54.39	0.83	0b10111	13.76	0.21
0b01000	51.12	0.78	0b11000	13.11	0.2
0b01001	45.22	0.69	0b11001	11.14	0.17
0b01010	40.63	0.62	0b11010	10.49	0.16

Table 17-7. Effective Gain for the Echo Reception Stage (continued)

ECHO RECEIVE PATH GAIN FROM THE ECHO INPUTS TO THE LOWPASS FILTER OUTPUT (GAIN IS CONTROLLED BY THE VALUE IN RCVGN BIT)					
RCVGN	FULL SCALE (mV _{p-p})	LPFD RESOLUTION (μV _{p-p} /LSB)	RCVGN	FULL SCALE (mV _{p-p})	LPFD RESOLUTION (μV _{p-p} /LSB)
0b01011	37.35	0.57	0b11011	9.17	0.14
0b01100	34.08	0.52	0b11100	8.52	0.13
0b01101	31.46	0.48	0b11101	7.86	0.12
0b01110	29.49	0.45	0b11110	7.21	0.11
0b01111	27.52	0.42	0b11111	6.55	0.10

The amplified echo signal passes through a BPF, a full-wave rectifier, and an LPF.

After the echo signal goes through these filtering and signal processing stages, the LPF data is available from the LPFD register. This data is automatically updated at a rate equal to 5 x BFREQ. LPFD can be read directly by the processor or it can be loaded into an 8-words deep FIFO. If the FIFO is used, the source that initiates a FIFO load is selected by setting the FIFO load source bits, FFLS.[2:0] (LPFC[2:0]) as shown in Table 17-8.

Table 17-8. FIFO Load Source Selection

FFLS[2:0]	FIFO LOAD SOURCE
000	Timer 0
001	Timer 1
010	Timer 2
011, 100, 101	Reserved
110	Continuous loading whenever data is ready at the LPF indicated by the LPFRDY bit.
111	Software loaded. Selects the FIFO load control bit, FFLD (LPFC.3). When software sets FFLD = 1, the output of the lowpass filter (LPFD register) is put into the FIFO. FFLD = 0 has no effect. FFLD is automatically reset to 0 after the FIFO is loaded.

The output of the LPF (LPFD register) is automatically sent to a digital comparator where the LPFD register is compared to the Echo Envelope Comparator Threshold register (CMPT[15:0]). This happens even if LPFD is not read or loaded into the FIFO.

The output of the comparator, CMPLVL, depends on the values of the following: CMPT register, CMPLVL register, CMPH[14:0] bits (CMPC.14:0) (comparator hysteresis), and CMPP bit (comparator polarity) (CMPC.15). CMPP selects the edge of the CMPLVL that causes a comparator interrupt. Figure 17-5 represents the link between the registers and CMPLVL output.

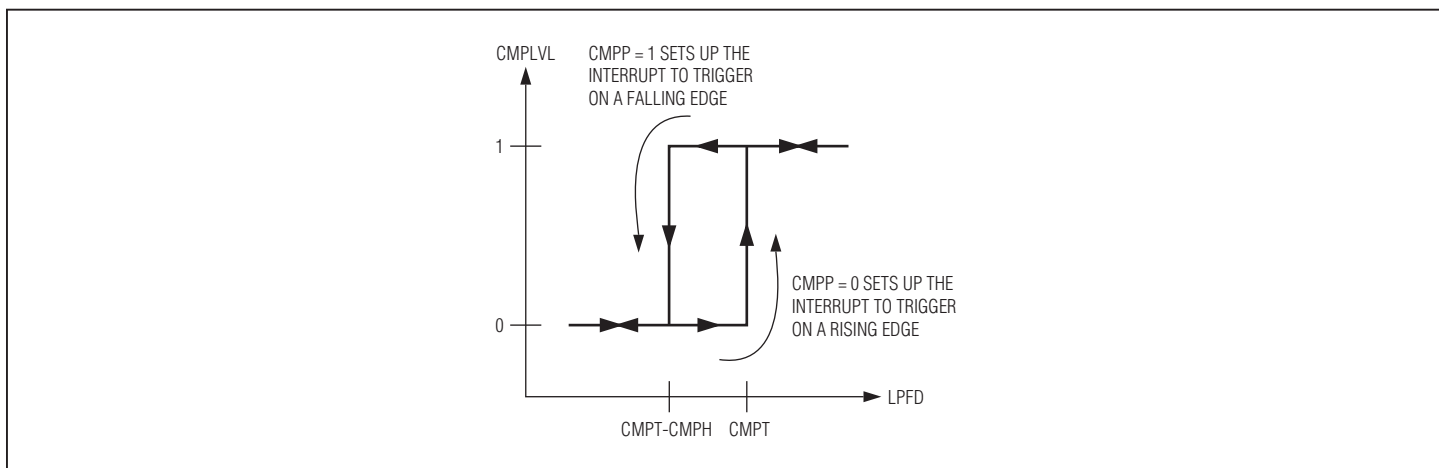


Figure 17-5. Comparator Hysteresis

17.5.3 Echo Receive Data Register

The LPFD register contains the most recent filtered and processed data. Reading LPFD does not clear it. The data in LPFD is updated as soon as a new value is available. The LPFD data is in straight binary format.

When a new LPF value is ready, the data ready flag, LPFRDY (ASR.1), is set to 1. If the echo envelope LPF output data ready interrupt enable bit, LPFIE (AIE.2), is set to 1, an interrupt is generated by the LPFRDY flag.

The FIFO has a data pointer, FFDP[3:0] (LPFC.11:8), that indicates space taken up in the FIFO. A value of 0 indicates that the FIFO is empty and a value of 8 indicates that it is full.

The FIFO holds a maximum of eight filter readings and this data can be read from the Echo Envelope Lowpass Filter FIFO Output register (LPFF). When LPFF is read, the read data is automatically popped off the stack and the next reading is available. **Note:** The data read from an empty FIFO (FFDP = 0) should be considered invalid. However, due to the FIFO's implementation, it is most likely to be a previous sample value and may appear valid. So the depth value (FFDP) should always be tested before reading the FIFO.

Four FIFO interrupt level bits, FFIL[3:0] (LPFC.15:12), in the Echo Envelope Lowpass Filter FIFO Control register (LPFC) can be set to generate an interrupt whenever FFDP matches FFIL. This interrupt is enabled only when LFLIE (AIE.2) is set to 1. An exception to this is when FFIL = 0; setting FFIL to 0 disables FFIL interrupts (even with LFLIE = 1).

Another interrupt occurs due to the overflowing of the FIFO. This is indicated by the FIFO overflow flag, FFOV (LPFC.7). This bit is set whenever the FIFO is already full and an attempt has been made to write to the FIFO. The data that would have been written to the FIFO is lost, and an interrupt may be generated.

Table 17-9 summarizes the interrupts that can be generated by the echo receive path, the data they are associated with, and how they are enabled.

All the above-mentioned interrupts are affected by the setting of the MAXQ global interrupt enable (IGE) and the module interrupt enable for Module 5. To enable interrupts, both the global and module interrupts must be enabled.

Table 17-9. Echo Receive Path Interrupts

INTERRUPT CONDITION	STATUS BIT SET BY INTERRUPT	INTERRUPT ENABLED BY
LPF data ready.	LPFRDY (ASR.1)	LPFIE = 1 (AIE.1)
LPF FIFO depth is equal to the lowpass interrupt level, FFDP = FFIL. (No interrupts are generated for FFIL = 0.)	LPFFL (ASR.2)	LFLIE = 1 (AIE.2)
LPF FIFO overflow (an attempt was made to put data in the FIFO, but it was already full).	FFOV (LPFC.7)	May be enabled
Comparator output is high.	CMPI (ASR.3)	CMPIE (AIE.3)

SECTION 18: UTILITY ROM

This section contains the following information:

18.1 In-Application Programming Functions	18-3
18.2 Data Transfer Functions	18-4
18.3 ROM Example 1: Calling A MAXQ7667 Utility ROM Function Directly	18-7
18.4 ROM Example 2: Calling A MAXQ7667 Utility ROM Function Indirectly	18-8

LIST OF TABLES

Table 18-1. Utility ROM User Functions (for Utility ROM Version 1.01)	18-2
---	------

SECTION 18: UTILITY ROM

The ROM in the MAXQ7667 provides routines for handling the bootloader and application code, the debug engine, and some utility function for the user. The focus of this section is the utility function. The area of the UROM where the utility functions reside is known as the utility ROM.

The MAXQ7667 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 in flash memory

To provide backward 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 in Table 18-1 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 **flashEraseAll** function can be accessed and called by the following procedure.

```

get_urom_table_entry:
    move dpc, #1Ch           //all data pointers in word mode
    move dp[ 0], #0800Dh    //initialize dp[ 0]
    move bp, @dp[ 0]        //load function address table location in bp
    move offs, #2           //load function number in offs
    call @bp[ offs]        //call flashEraseAll
    
```

It is also possible to call utility ROM functions directly, using the memory entry points of the function given in Table 18-1. Standard include files are provided for this purpose with the MAXQ7667 development tool set. 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 18-1. Utility ROM User Functions (for Utility ROM Version 1.01)

FUNCTION NUMBER	FUNCTION NAME	FUNCTION ADDRESS TABLE OFFSET	MEMORY ENTRY POINT	SUMMARY
0	—	—	—	Reserved.
1	flashWrite	0	085C0h	Writes a word to the flash memory.
2	flashErasePage	1	085C1h	Erases 1 page (256 words) in flash memory.
3	flashEraseAll	2	08563h	Erases all flash memory.
4	moveDP0	3	08572h	Reads a byte/word at DP[0] in flash memory.
5	moveDP0inc	4	08575h	Reads a byte/word at DP[0] in flash memory, then increments DP[0].
6	moveDP0dec	5	08578h	Reads a byte/word at DP[0] in flash memory, then decrements DP[0].
7	moveDP1	6	0857Bh	Reads a byte/word at DP[1] in flash memory.
8	moveDP1inc	7	0857Eh	Reads a byte/word at DP[1] in flash memory, then increments DP[1].
9	moveDP1dec	8	08581h	Reads a byte/word at DP[1] in flash memory, then decrements DP[1].
10	moveFP	9	08584h	Reads a byte/word at BP[OFFS] in flash memory.
11	moveFPinc	10	08587h	Reads a byte/word at BP[OFFS] in flash memory, then increments OFFS.
12	moveFPdec	11	0858Ah	Reads a byte/word at BP[OFFS] in flash memory, then decrements OFFS.
13	copyBuffer	12	0858Dh	Copies LC[0] values from DP[0] to BP[OFFS].
14	UARTloader*	13	0859Eh	This is the entry point to call the UART bootloader from an application.
15	AutoBaud*	14	085CEh	Waits for a 0x0D and measures the bit time for it, so that a value for the baud clock can be calculated.

*See Section 8 for UART with LIN communication and Section 13 for in-system programming through UART (and JTAG).

18.1 In-Application Programming Functions

Function: flashWrite
Summary: Write a word to the flash block (program memory).
Inputs: A[0]: Word address in main flash block to write to.
 A[1]: Word value to write to flash block.
Outputs: Carry: Set on error and cleared on success. If set, then A[0] contains one of the following error codes:
 1: Failure due to software timeout.
 2: Failure reported by hardware (DQ5).
Destroys: PSF, LC[1]

Notes:

- 1) If the watchdog reset is enabled, user code should disable it before calling this function. Also, disable interrupts globally (IGE = 0).
- 2) If the flash location has already been programmed to a non-FFFF (hex) value, this function returns with an error (Carry set). To reprogram a flash sector, it must first be erased by calling flashErasePage or flashEraseAll.

Function: flashErasePage
Summary: Erases one page, 256 words, in the flash block (program memory).
Inputs: A[0]: Word address located in the page to be erased.
Outputs: Carry: Set on error and cleared on success. If set, then A[0] contains one of the following error codes:
 1: Failure due to software timeout.
 2: Failure reported by hardware (DQ5).
 3: Failure due to trying to erase current page.
Destroys: PSF, LC[1], GR, AP, APC (AP = APC = 0)

Notes:

- 1) If the watchdog reset is enabled, user code should disable it before calling this function. Also, disable interrupts globally (IGE = 0).

Function: flashEraseAll
Summary: Erases the entire flash block.
Inputs: None.
Outputs: Carry: Set on error and cleared on success.
Destroys: PSF, GR, LC[0], LC[1], APC, AP, A[0] (AP, APC set to 0)

Notes:

- 1) If the watchdog reset is enabled, user code should disable it before calling this function. Also, disable interrupts globally (IGE = 0).
- 2) This function can only be called by code running from the RAM. An attempt to call this function while running from the flash results in an error.

Function: AutoBaud*
Summary: Waits for a 0x0D and measures the bit time for it and a reasonable value for BT is calculated (baud clock).
Input: A[15]: Number of retries when bad character received.
Output: PR set and return code in A[7]. Return codes are one of ERROR_NONE, ERROR_NO_AUTOBAUD_CHARACTER, or ERROR_BAD_AUTOBAUD_CHARACTER.
Destroys: Accumulators (pretty much all of them), AP, PSF, Timer 0

Notes:

- 1) Initializes the UART to mode 1 with baud rate quadrupler on.

*Refer to Section 8 for UART with LIN communication and Section 13 for in-system programming through UART (and JTAG).

Function: UARTloader*

Summary: This is an entry point for the customer to call the UART bootloader from their application code. The UART performs an autobaud and if that succeeds, the loader will be launched and this function will never return. If the autobauding fails, the routine returns an error code in A[7].

Input: Clear Password lock before calling (if desired).

Output: A[7] contains exit code of ERROR_NO_AUTOBAUD_CHARACTER or ERROR_BAD_AUTOBAUD_CHARACTER if the function returns.

Notes:

- 1) This routine only returns on autobaud failure. It destroys nearly all accumulator, AP, GR, etc. values. Assume nothing survives and only A[7] is valid on return.

18.2 Data Transfer Functions

Function: 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: Selects DP[0] in DPC.

Notes:

- 1) Before calling this function, DPC should be set appropriately to configure DP[0] for byte or word mode.
- 2) The address passed to this function should be based on the data memory mapping for the utility ROM, as explained in *Section 2*.
- 3) This function automatically refreshes the data pointer before reading the byte/word value.

Function: 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: Selects DP[0] in DPC.

Notes:

- 1) Before calling this function, DPC should be set appropriately to configure DP[0] for byte or word mode.
- 2) The address passed to this function should be based on the data memory mapping for the utility ROM, as explained in *Section 2*.
- 3) This function automatically selects DP[0] as the data pointer before reading the byte/word value.

Function: 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: Selects DP[0] in DPC.

Notes:

- 1) Before calling this function, DPC should be set appropriately to configure DP[0] for byte or word mode.
- 2) The address passed to this function should be based on the data memory mapping for the utility ROM, as explained in *Section 2*.
- 3) This function automatically refreshes the data pointer before reading the byte/word value.

*Refer to Section 8 for UART with LIN communication and Section 13 for in-system programming through UART (and JTAG).

Function: 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: Selects DP[1] in DPC.

Notes:

- 1) Before calling this function, DPC should be set appropriately to configure DP[1] for byte or word mode.
- 2) The address passed to this function should be based on the data memory mapping for the utility ROM, as explained in *Section 2*.
- 3) This function automatically selects DP[1] as the data pointer before reading the byte/word value.

Function: 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: Selects DP[1] in DPC.

Notes:

- 1) Before calling this function, DPC should be set appropriately to configure DP[1] for byte or word mode.
- 2) The address passed to this function should be based on the data memory mapping for the utility ROM, as explained in *Section 2*.
- 3) This function automatically selects DP[1] as the data pointer before reading the byte/word value.

Function: 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: Selects DP[1] in DPC.

Notes:

- 1) Before calling this function, DPC should be set appropriately to configure DP[1] for byte or word mode.
- 2) The address passed to this function should be based on the data memory mapping for the utility ROM, as explained in *Section 2*.
- 3) This function automatically selects DP[1] as the data pointer before reading the byte/word value.

Function: moveFP
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: Selects BP in DPC.

Notes:

- 1) Before calling this function, DPC should be set appropriately to configure BP[OFFS] for byte or word mode.
- 2) The address passed to this function should be based on the data memory mapping for the utility ROM, as explained in *Section 2*.
- 3) This function automatically selects BP[OFFS] as the data pointer before reading the byte/word value.

Function: moveFPinc
Summary: Reads the byte/word value pointed to by BP[OFFS] then increments OFFS.
Inputs: BP[OFFS]: Address to read from.
Outputs: GR: Data byte/word read.
OFFS is incremented.
Destroys: Selects BP in DPC.

Notes:

- 1) Before calling this function, DPC should be set appropriately to configure BP[OFFS] for byte or word mode.
- 2) The address passed to this function should be based on the data memory mapping for the utility ROM, as explained in *Section 2*.
- 3) This function automatically selects BP[OFFS] as the data pointer before reading the byte/word value.

Function: moveFPdec
Summary: Reads the byte/word value pointed to by BP[OFFS] then decrements OFFS.
Inputs: BP[OFFS]: Address to read from.
Outputs: GR: Data byte/word read.
OFFS is decremented.
Destroys: Selects BP in DPC.

Notes:

- 1) Before calling this function, DPC should be set appropriately to configure BP[OFFS] for byte or word mode.
- 2) The address passed to this function should be based on the data memory mapping for the utility ROM, as explained in *Section 2*.
- 3) This function automatically selects BP[OFFS] as the data pointer before reading the byte/word value.

Function: copyBuffer
Summary: Copies LC[0] bytes/words 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:

- 1) 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.
- 2) The addresses passed to this function should be based on the data memory mapping for the utility ROM, as explained in *Section 2*.
- 3) This function automatically selects DP[0] as the data pointer before reading the byte/word value.

18.3 ROM Example 1: Calling A MAXQ7667 Utility ROM Function Directly

This example shows the direct addressing method for calling MAXQ7667 utility functions, using the function **moveDP1inc** to read a static string from code space. Note the equate **UROM_MOVEDP1INC**.

```
UROM_MOVEDP1INC EQU 0857Eh
```

```
Text:
  DB  "Hello World!",0          ; Define a string in code space.

  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;; Function:      PrintText
  ;; Description:   Prints the string stored at the "Text" label.
  ;; Returns:      N/A
  ;; Destroys:     ACC, DP[ 1], DP[ 0], and GR.
  ;; Notes:        This function assumes that DP[ 0] is set to word mode, and
  ;;              DP[ 1] is in byte mode.
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
PrintText:
  move  DP[ 1], #Text          ; Point to the string to display.
  move  ACC, DP[ 1]           ; "Text" is a word address and we need a
  sla   ; byte address, so shift left 1 bit.
  or    #08000h              ; Code space is mapped to 8000h when running
  move  DP[ 1], ACC          ; from the ROM, so the address must be masked.
```

```
PrintText_Loop:
  call  UROM_MOVEDP1INC      ; Fetch the byte from code space.
  move  ACC, GR
  jump  Z, PrintText_Done   ; Reached the null terminator.
  call  PrintChar           ; Call a routine to output the char in ACC
  jump  PrintText_Loop     ; Process the next byte.
```

```
PrintText_Done:
  ret
```

SECTION 19: INSTRUCTION SET SUMMARY

This section contains the following information:

ADD/ADDC src	19-5
AND src	19-6
AND Acc.	19-6
{L/S}CALL	19-7
CMP src	19-8
CPL	19-8
CPL C	19-9
{L/S}DJNZ LC[n], src	19-9
{L/S}JUMP src	19-10
{L/S}JUMP C/{L/S}JUMP NC, src, L/S}JUMP Z/{L/S}JUMP NZ, src, {L/S}JUMP E/{L/S}JUMP NE, src, {L/S}JUMP S, src	19-11
MOVE dst, src	19-13
MOVE Acc., C	19-15
MOVE C, Acc.	19-16
MOVE C, src.	19-16
MOVE C, #0	19-16
MOVE C, #1	19-17
MOVE dst., #0	19-17
MOVE dst., #1	19-17
NEG	19-18
OR src	19-18
OR Acc.	19-19
POP dst	19-19
POPI dst	19-20
PUSH src	19-20
RET	19-21
RET C/RET NC, RET Z/RET NZ, RET S	19-21
RETI	19-23

RETI C/RETI NC, RETI Z/RETI NZ, RETI S	19-23
RL/RLC	19-25
RR/RRC	19-26
SLA/SLA2/SLA4	19-27
SR/SRA/SRA2/SRA4	19-28
SUB/SUBB src	19-30
XCH	19-31
XCHN	19-31
XOR src	19-32
XOR Acc.	19-32

LIST OF TABLES

Table 19-1. MAXQ7667 Instruction Set Summary	19-3
Table 19-2. Source Specifier Codes	19-13
Table 19-3. Destination Specifier Codes	19-14

SECTION 19: INSTRUCTION SET SUMMARY

Table 19-1. MAXQ7667 Instruction Set Summary

	MNEMONIC	DESCRIPTION	16-BIT INSTRUCTION WORD	STATUS BITS AFFECTED	AP INC/DEC	NOTES
LOGICAL OPERATIONS	AND src	$Acc \leftarrow Acc \text{ AND } src$	f001 1010 ssss ssss	S, Z	Y	1
	OR src	$Acc \leftarrow Acc \text{ OR } src$	f010 1010 ssss ssss	S, Z	Y	1
	XOR src	$Acc \leftarrow Acc \text{ XOR } src$	f011 1010 ssss ssss	S, Z	Y	1
	CPL	$Acc \leftarrow \sim Acc$	1000 1010 0001 1010	S, Z	Y	
	NEG	$Acc \leftarrow \sim Acc + 1$	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 \rightarrow 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 (through C)	1000 1010 1101 1010	C, S, Z	Y	
BIT OPERATIONS	MOVE C, Acc.	$C \leftarrow Acc.$	1110 1010 bbbb 1010	C		
	MOVE C, #0	$C \leftarrow 0$	1101 1010 0000 1010	C		
	MOVE C, #1	$C \leftarrow 1$	1101 1010 0001 1010	C		
	CPL C	$C \leftarrow \sim C$	1101 1010 0010 1010	C		
	MOVE Acc., C	$Acc. \leftarrow C$	1111 1010 bbbb 1010	S, Z		
	AND Acc.	$C \leftarrow C \text{ AND } Acc.$	1001 1010 bbbb 1010	C		
	OR Acc.	$C \leftarrow C \text{ OR } Acc.$	1010 1010 bbbb 1010	C		
	XOR Acc.	$C \leftarrow C \text{ XOR } Acc.$	1011 1010 bbbb 1010	C		
	MOVE dst., #1	$dst. \leftarrow 1$	1ddd dddd 1bbb 0111	C, S, E, Z		2
	MOVE dst., #0	$dst. \leftarrow 0$	1ddd dddd 0bbb 0111	C, S, E, Z		2
MOVE C, src.	$C \leftarrow src.$	fbbb 0111 ssss ssss	C			
MATH	ADD src	$Acc \leftarrow Acc + src$	f100 1010 ssss ssss	C, S, Z, OV	Y	1
	ADDC src	$Acc \leftarrow Acc + (src + C)$	f110 1010 ssss ssss	C, S, Z, OV	Y	1
	SUB src	$Acc \leftarrow Acc - src$	f101 1010 ssss ssss	C, S, Z, OV	Y	1
	SUBB src	$Acc \leftarrow Acc - (src + C)$	f111 1010 ssss ssss	C, S, Z, OV	Y	1

Table 19-1. MAXQ7667 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	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 0-5 (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 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 will be 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 will not be transferred to AP).

ADD/ADDC src

Add/Add with Carry

Description: The ADD instruction sums the active accumulator (Acc or A[AP]) and the specified src data and stores the result back to the active accumulator. The ADDC instruction additionally includes the Carry (C) Status Flag in the summation. For the complete list of src specifiers, reference the MOVE instruction. The MAXQ7667 may use the PFX[n] register to supply the high byte of data for 8-bit sources.

Status Flags: C, S, Z, OV

ADD Operation: $Acc \leftarrow Acc + src$

Encoding: 15 0

f100	1010	ssss	ssss
------	------	------	------

Example(s): ;Acc = 2345h for each example

```

ADD A[3] ; A[3]=FF0Fh
          ; → Acc =2254h,C=1, Z=0, S=0, OV=0
ADD #0C0h ; → Acc =2405h,C=0, Z=0, S=0, OV=0
ADD A[4] ; A[4]=C000h
          ; → Acc = E345h, C=0, Z=0, S=1, OV=0
ADD A[5] ; A[5]=6789h
          ; → Acc = 8ACEh, C=0, Z=0, S=1, OV=0
    
```

ADDC Operation: $Acc \leftarrow Acc + C + src$

Encoding: 15 0

f110	1010	ssss	ssss
------	------	------	------

Example(s): ; Acc = 2345h for each example

```

ADDC A[3] ; A[3] = DCBAh, C=1
           ; → Acc = 0000h, C=1, Z=1, S=0, OV=0
ADDC @DP[0]-- ; @DP[0] = 00EEh, C=1
              ; → Acc = 2434h, C=0, Z=0, S=0, OV=0
    
```

Special Notes: The active accumulator (Acc) is not allowed as the src for these operations.

AND src

Logical AND

Description: Performs a logical-AND between the active accumulator (Acc) and the specified src data. For the complete list of src specifiers, reference the MOVE instruction. The MAXQ7667 may use the PFX[n] register to supply the high byte of data for 8-bit sources.

Status Flags: S, Z

Operation: Acc ← Acc AND src

Encoding: 15 0

f001	1010	ssss	ssss
------	------	------	------

Example(s):

```

; Acc = 2345h for each example
AND A[3]      ; A[3]=0F0Fh
               ; → Acc = 0305h, S=0, Z=0
AND #33h     ; → Acc = 0001h
AND #2233h   ; generates object code below
               ; MOVE PFX[0], #22h (smart-prefixing)
               ; AND #33h
               ; → Acc = 2201h
MOVE PFX[0], #0Fh
AND M0[8]    ; M0[8]=0Fh (assume M0[8] is an 8-bit register)
               ; → Acc = 0305h
    
```

Special Notes: The active accumulator (Acc) is not allowed as the src for this operation.

AND Acc.

Logical AND Carry Flag with Accumulator Bit

Description: Performs a logical-AND between the Carry (C) status flag and a specified bit of the active accumulator (Acc.) and returns the result to the Carry.

Status Flags: C

Operation: C ← C AND Acc.

Encoding: 15 0

1001	1010	bbbb	1010
------	------	------	------

Example(s):

```

; Acc = 2345h, C=1 at start
AND Acc.0    ; Acc.0=1 → C=1
AND Acc.1    ; Acc.1=0 → C=0
AND C, Acc.8 ; Acc.8=1 → C=0
    
```


CMP src	Compare Accumulator
----------------	----------------------------

Description: Compare for equality between the active accumulator and the least significant byte of the specified src. The MAXQ7667 may use the PFX[n] register to supply the high byte of data for 8-bit sources.

Status Flags: E

Operation: Acc = src: E ← 1
 Acc <> src: E ← 0

Encoding: 15 0

f111	1000	ssss	ssss
------	------	------	------

Example(s): CMP #45h ; Acc = 0145h, E=0
 CMP #145h ; PFX[0] register used
 ; MOVE PFX[0], #01h (smart-prefixing)
 ; CMP #45h E=1

CPL	Complement Acc
------------	-----------------------

Description: Performs a logical bitwise complement (1's complement) on the active accumulator (Acc or A[AP]) and returns the result to the active accumulator.

Status Flags: S, Z

Operation: Acc ← ~Acc

Encoding: 15 0

1000	1010	0001	1010
------	------	------	------

Example(s): ; Acc = FFFFh, S=1, Z=0
 CPL ; Acc ← 0000h, S=0, Z=1
 ; Acc = 0990h, S=0, Z=0
 CPL ; Acc ← F66Fh, S=1, Z=0

CPL C

Complement Carry Flag

Description: Logically complements the Carry (C) Flag.

Status Flags: C

Operation: $C \leftarrow \sim C$

Encoding: 15 0

1101	1010	0010	1010
------	------	------	------

Example(s): ; C = 0

CPL C ; C ← 1

{L/S}DJNZ LC[n], src

Decrement Counter, {Long/Short} Jump Not Zero

Description: The DJNZ LC[n], src instruction performs a conditional branch based upon the associated Loop Counter (LC[n]) register. The DJNZ LC[n], src instruction decrements the LC[n] loop counter and branches to the address defined by src if the decremented counter has not reached 0000h. Program branches can be relative or absolute depending upon the src specifier and may be qualified by using the 'L' or 'S' prefixes as documented in the JUMP src op code.

Status Flags: None

Operation: $LC[n] \leftarrow LC[n] - 1$

$LC[n] <> 0$: $IP \leftarrow IP + src$ (relative) -or- src (absolute)

$LC[n] = 0$: $IP \leftarrow IP + 1$

Encoding: 15 0

f10n	1101	ssss	ssss
------	------	------	------

Example(s): MOVE LC[1], #10h ; counter = 10h

Loop:

ADD @DP[0]++ ; add data memory contents to Acc, post-inc DP[0]

DJNZ LC[1], Loop ; 16 times before falling through

{L/S}JUMP C/{L/S}JUMP NC, src, Conditional {Long/Short} Jump on Status Flag
L/S}JUMP Z/{L/S}JUMP NZ, src,
{{L/S}JUMP E/{L/S}JUMP NE, src,
{L/S}JUMP S, src

Description: Performs conditional branching based upon the state of a specific processor status flag. JUMP C results in a branch if the Carry flag is set while JUMP NC branches if the Carry flag is clear. JUMP Z results in a branch if the Zero flag is set while JUMP NZ branches if the Zero flag is clear. JUMP E results in a branch if the Equal flag is set while JUMP NE branches if the Equal flag is clear. JUMP S results in a branch if the Sign flag is set. Program branches can be relative or absolute depending upon the src specifier and may be qualified by using the 'L' or 'S' prefixes as documented in the JUMP src op code. Special src restrictions apply to JUMP E and JUMP NE.

Status Flags: None

JUMP C C=1: IP ← IP + src (relative) -or- src (absolute)

Operation: C=0: IP ← IP + 1

Encoding: 15 0

f010	1100	ssss	ssss
------	------	------	------

Example(s): JUMP C, label1 ; C=0, branch not taken

JUMP NC C=0: IP ← IP + src (relative) -or- src (absolute)

Operation: C=1: IP ← IP + 1

Encoding: 15 0

f110	1100	ssss	ssss
------	------	------	------

Example(s): JUMP NC, label1 ; C=0, branch taken

JUMP Z Z=1: IP ← IP + src

Operation: Z=0: IP ← IP + 1

Encoding: 15 0

f001	1100	ssss	ssss
------	------	------	------

Example(s): JUMP Z, label1 ; Z=1, branch taken

MOVE dst, src **Move Data**

Description: Moves data from a specified source (src) to a specified destination (dst). A list of defined source, destination specifiers is given in the table below. Also, since src can be either 8-bit (byte) or 16-bit (word) data, the rules governing data transfer are also explained below in the encoding section.

Status Flags: S, Z (if dst is Acc or AP or APC)
C, E (if dst is PSF)

Operation: dst ← src

Encoding: 15 0

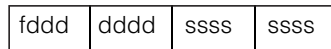


Table 19-2. Source Specifier Codes

src	src BIT ENCODING (f ssssssss)	WIDTH (16 or 8)	DESCRIPTION
#k	0 kkkk kkkk	8	kkkkkkkk = Immediate (Literal) Data
MN[n]	1 nnnn 0NNN	8/16	nnnn Selects One of First 16 Registers in Module NNN; where NNN = 0 to 5. Access to Second 16 Using PFX[n].
AP	1 0000 1000	8	Accumulator Pointer
APC	1 0001 1000	8	Accumulator Pointer Control
PSF	1 0100 1000	8	Processor Status Flag Register
IC	1 0101 1000	8	Interrupt and Control Register
IMR	1 0110 1000	8	Interrupt Mask Register
SC	1 1000 1000	8	System Control Register
IIR	1 1011 1000	8	Interrupt Identification Register
CKCN	1 1110 1000	8	Clock Control Register
WDCN	1 1111 1000	8	Watchdog Control Register
A[n]	1 nnnn 1001	8/16	nnnn Selects One of 16 Accumulators
Acc	1 0000 1010	8/16	Active Accumulator = A[AP]. Update AP per APC
A[AP]	1 0001 1010	8/16	Active Accumulator = A[AP]. No change to AP
IP	1 0000 1100	16	Instruction Pointer
@SP--	1 0000 1101	16	16-Bit Word @SP, Post-Decrement SP
SP	1 0001 1101	16	Stack Pointer
IV	1 0010 1101	16	Interrupt Vector
LC[n]	1 011n 1101	16	n Selects 1 of 2 Loop Counter Registers
@SPL--	1 1000 1101	16	16-bit word @SP, Post-Decrement SP, INS = 0
@BP[OFFS]	1 0000 1110	8/16	Data Memory @BP[OFFS]
@BP[OFFS++]	1 0001 1110	8/16	Data memory @BP[OFFS]; Post-Increment OFFS
@BP[OFFS--]	1 0010 1110	8/16	Data Memory @BP[OFFS]; Post-Decrement OFFS
OFFS	1 0011 1110	8	Frame Pointer Offset from Base Pointer (BP)
DPC	1 0100 1110	16	Data Pointer Control Register
GR	1 0101 1110	16	General Register
GRL	1 0110 1110	8	Low Byte of GR Register
BP	1 0111 1110	16	Frame Pointer Base Pointer (BP)
GRS	1 1000 1110	16	Byte-Swapped GR Register
GRH	1 1001 1110	8	High Byte of GR Register
GRXL	1 1010 1110	16	Sign Extended Low Byte of GR Register
FP	1 1011 1110	16	Frame Pointer (BP[OFFS])
@DP[n]	1 0n00 1111	8/16	Data Memory @DP[n]
@DP[n]++	1 0n01 1111	8/16	Data Memory @DP[n], Post-Increment DP[n]
@DP[n]--	1 0n10 1111	8/16	Data Memory @DP[n], Post-Decrement DP[n]
DP[n]	1 0n11 1111	16	n Selects 1 of 2 Data Pointers

MOVE dst, src

Move Data

Table 19-3. Destination Specifier Codes

dst	dst BIT ENCODING (ddd dddd)	WIDTH (16 OR 8)	DESCRIPTION
NUL	111 0110	8/16	Null (Virtual) Destination. Intended as a bit bucket to assist software with pointer increments/decrements.
MN[n]	nnn 0NNN	8/16	nnn Selects One of First 8 Registers in Module NNN; where NNN = 0 to 5. Access to Next 24 Using PFX[n].
AP	000 1000	8	Accumulator Pointer
APC	001 1000	8	Accumulator Pointer Control
PSF	100 1000	8	Processor Status Flag Register
IC	101 1000	8	Interrupt and Control Register
IMR	110 1000	8	Interrupt Mask Register
A[n]	nnn 1001	8/16	nnn Selects 1 of First 8 Accumulators: A[0]..A[7]
Acc	000 1010	8/16	Active Accumulator = A[AP]
PFX[n]	nnn 1011	8	nnn Selects One of 8 Prefix Registers
@++SP	000 1101	16	16-Bit Word @SP, Pre-Increment SP
SP	001 1101	16	Stack Pointer
IV	010 1101	16	Interrupt Vector
LC[n]	11n 1101	16	n Selects 1 of 2 Loop Counter Registers
@BP[OFFS]	000 1110	8/16	Data Memory @BP[OFFS]
@BP[++OFFS]	001 1110	8/16	Data Memory @BP[OFFS]; Preincrement OFFS
@BP[--OFFS]	010 1110	8/16	Data Memory @BP[OFFS]; Predecrement OFFS
OFFS	011 1110	8	Frame Pointer Offset from Base Pointer (BP)
DPC	100 1110	16	Data Pointer Control Register
GR	101 1110	16	General Register
GRL	110 1110	8	Low Byte of GR Register
BP	111 1110	16	Frame Pointer Base Pointer (BP)
@DP[n]	n00 1111	8/16	Data Memory @DP[n]
@++DP[n]	n01 1111	8/16	Data Memory @DP[n], Preincrement DP[n]
@--DP[n]	n10 1111	8/16	Data Memory @DP[n], Predecrement DP[n]
DP[n]	n11 1111	16	n Selects 1 of 2 Data Pointers
2-CYCLE DESTINATION ACCESS USING PFX[n] REGISTER (See Special Notes)			
SC	000 1000	8	System Control Register
CKCN	110 1000	8	Clock Control Register
WDCN	111 1000	8	Watchdog Control Register
A[n]	nnn 1001	16	nnn Selects 1 of Second 8 Accumulators A[8]...A[15]
GRH	001 1110	8	High Byte of GR Register

Data Transfer Rules

dst (16-bit) ← src (16-bit):	dst[15:0] ← src[15:0]
dst (8-bit) ← src (8-bit):	dst[7:0] ← src[7:0]
dst (16-bit) ← src (8-bit):	dst[15:8] ← 00h *
	dst[7:0] ← src[7:0]
dst (8-bit) ← src (16-bit):	dst[7:0] ← src[7:0]

* **Note:** The PFX[0] register may be used to supply a separate high-order data byte for this type of transfer.

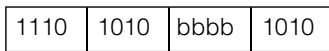
MOVE C, Acc. **Move Accumulator Bit to Carry Flag**

Description: Replaces the Carry (C) status flag with the specified active accumulator bit.

Status Flags: C

Operation: C ← Acc.

Encoding: 15 0



Example(s): `MOVE C, Acc.0` ; Acc = 01C0h, C=0
`MOVE C, Acc.8` ; C = 1

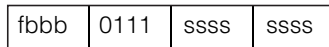
MOVE C, src. **Move Bit to Carry Flag**

Description: Replaces the Carry (C) status flag with the specified source bit src..

Status Flags: C

Operation: C ← src.

Encoding: 15 0



Example(s): `MOVE C, M0[0].0` ; M0[0] = FEh; C=1 (assume M0[0] is an 8-bit register)
`MOVE C, M0[0].0` ; C=0

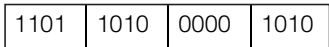
MOVE C, #0 **Clear Carry Flag**

Description: Clears the Carry (C) processor status flag.

Status Flag: C ← 0

Operation: C ← 0

Encoding: 15 0



Example(s): `MOVE C, #0` ; C = 1
`MOVE C, #0` ; C ← 0

MOVE C, #1 Set Carry Flag

Description: Sets the Carry (C) processor status flag.

Status Flag: C ← 1

Operation: C ← 1

Encoding: 15 0

1101	1010	0001	1010
------	------	------	------

Example(s): ; C = 0
 MOVE C, #1 ; C ← 1

MOVE dst., #0 Clear Bit

Description: Clears the bit specified by dst..

Status Flags: C, E (if dst is PSF), S, Z

Operation: dst. ← 0

Encoding: 15 0

1ddd	dddd	0bbb	0111
------	------	------	------

Example(s): ; M0[0] = FEh
 MOVE M0[0].1, #0 ; M0[0] = FCh
 MOVE M0[0].7, #0 ; M0[0] = 7Ch

Special Notes: Only system module 8 and peripheral modules (0-5) are supported by MOVE dst., #0.

MOVE dst., #1 Set Bit

Description: Sets the bit specified by dst..

Status Flags: C, E (if dst is PSF), S, Z

Operation: dst. ← 1

Encoding: 15 0

1ddd	dddd	1bbb	0111
------	------	------	------

Example(s): ; M0[0] = 00h
 MOVE M0[0].1, #1 ; M0[0] = 02h
 MOVE M0[0].7, #1 ; M0[0] = 82h

Special Notes: Only system module 8 and peripheral modules (0-5) are supported by MOVE dst., #1.

NEG **Negate Accumulator**

Description: Performs a negation (two's complement) of the active accumulator and returns the result back to the active accumulator.

Status Flags: S, Z

Operation: $Acc \leftarrow \sim Acc + 1$

Encoding: 15 0

1000	1010	1001	1010
------	------	------	------

Example(s): ; Acc = FEEDh, S=1, Z=0
 NEG ; Acc = 0113h, S=0, Z=0

OR src **Logical OR**

Description: Performs a logical-OR between the active accumulator (Acc or A[AP]) and the specified src data. For the complete list of src specifiers, reference the MOVE instruction. The MAXQ7667 may use the PFX[n] register to supply the high byte of data for 8-bit sources.

Status Flags: S, Z

Operation: $Acc \leftarrow Acc \text{ OR } src$

Encoding: 15 0

f010	1010	ssss	ssss
------	------	------	------

Example(s): ; Acc = 2345h for each example
 OR A[3] ; A[3]= 0F0Fh → Acc = 2F4Fh
 OR #1133h ; MOVE PFX[0], #11h (smart-prefixing)
; OR #33h → Acc = 3377h

Special Notes: The active accumulator (Acc) is not allowed as the src for this operation.

OR Acc. Logical OR Carry Flag with Accumulator Bit

Description: Performs a logical-OR between the Carry (C) status flag and a specified bit of the active accumulator (Acc.) and returns the result to the Carry.

Status Flags: C

Operation: $C \leftarrow C \text{ OR } \text{Acc.}\langle b \rangle$

Encoding: 15 0

1010	1010	bbbb	1010
------	------	------	------

Example(s): ; Acc = 2345h, C=0 at start
 OR Acc.1 ; Acc.1=0 → C=0
 OR Acc.2 ; Acc.2=1 → C=1

POP dst Pop Word from the Stack

Description: Pops a single word from the stack (@SP) to the specified dst and decrements the stack pointer (SP).

Status Flags: S, Z (if dst = Acc or AP or APC)
 C, E (if dst = PSF)

Operation: $\text{dst} \leftarrow @ \text{SP}--$

Encoding: 15 0

1ddd	dddd	0000	1101
------	------	------	------

Example(s): ; GR ← 1234h
 POP GR ; @DP[0] ← 76h (WBS0=0)
 POP @DP[0] ; @DP[0] ← 0876h (WBS0=1)

Stack Data:

xxxxh	
1234h	← SP (initial)
0876h	← SP (after POP GR)
xxxxh	← SP (after POP @DP[0])
xxxxh	

POPI dst	Pop Word from the Stack Enable Interrupts
-----------------	--

Description: Pops a single word from the stack (@SP) to the specified dst and decrements the stack pointer (SP). Additionally, POPI returns the interrupt logic to a state in which it can acknowledge additional interrupts.

Status Flags: S, Z (if dst = Acc or AP or APC)
C, E (if dst = PSF)

Operation: dst ← @ SP--
INS ← 0

Encoding: 15 0

1ddd	dddd	1000	1101
------	------	------	------

Example(s): See POP

PUSH src	Push Word to the Stack
-----------------	-------------------------------

Description: Increments the stack pointer (SP) and pushes a single word specified by src to the stack (@SP).

Status Flags: None

Operation: SP ← ++SP

Encoding: 15 0

f000	1101	ssss	ssss
------	------	------	------

Example(s): PUSH GR ; GR=0F3Fh
PUSH #40h

Stack Data:

xxxxh	
0040h	← SP (after PUSH #40h)
0F3Fh	← SP (after PUSH GR)
xxxxh	← SP (initial)
xxxxh	

RET **Return from Subroutine**

Description: RET pops a single word from the stack (@SP) into the Instruction Pointer (IP) and decrements the stack pointer (SP). The decremented SP is saved as the new stack pointer (SP).

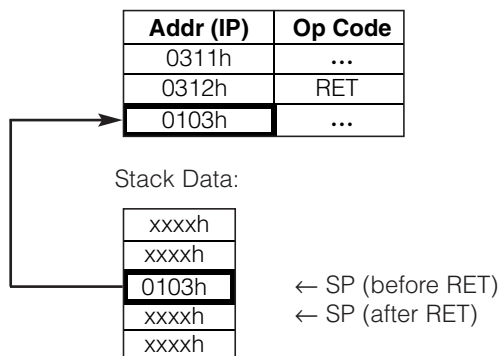
Status Flags: None

Operation: IP ← @ SP--

Encoding: 15 0

1000	1100	0000	1101
------	------	------	------

Example(s): RET
Code Execution:



RET C/RET NC, RET Z/RET NZ, RET S **Conditional Return on Status Flag**

Description: Performs conditional return (RET) based upon the state of a specific processor status flag. RET C returns if the Carry flag is set while RET NC returns if the Carry flag is clear. RET Z returns if the Zero flag is set while RET NZ returns if the Zero flag is clear. RET S returns if the Sign flag is set. See RET for additional information on the return operation.

Status Flags: None

RET C C=1: IP ← @SP--

Operation: C=0: IP ← IP + 1

Encoding: 15 0

1010	1100	0000	1101
------	------	------	------

Example(s): RET C ; C=1, return (RET) is performed

RET NC

Operation: C=0: IP ← @SP--
C=1: IP ← IP +1

Encoding: 15 0

1110	1100	0000	1101
------	------	------	------

Example(s): RET NC ; C=1, return (RET) does not occur

RET Z

Operation: Z=1: IP ← @SP--
Z=0: IP ← IP + 1

Encoding: 15 0

1001	1100	0000	1101
------	------	------	------

Example(s): RET Z ; Z=0, return (RET) does not occur

RET NZ

Operation: Z=0: IP ← @SP--
Z=1: IP ← IP +1

Encoding: 15 0

1101	1100	0000	1101
------	------	------	------

Example(s): RET NZ ; Z=0, return (RET) is performed

RET S

Operation: S=1: IP ← @SP--
S=0: IP ← IP + 1

Encoding: 15 0

1100	1100	0000	1101
------	------	------	------

Example(s): RET S ; S=0, return (RET) does not occur

RETJ Return from Interrupt

Description: RETJ pops a single word from the stack (@SP) into the Instruction Pointer (IP) and decrements the stack pointer (SP). Additionally, RETJ returns the interrupt logic to a state in which it can acknowledge additional interrupts.

Status Flags: None

Operation: IP ← @SP--
INS ← 0

Encoding: 15 0

1000	1100	1000	1101
------	------	------	------

Example(s): See RETJ

RETJ C/RETJ NC, RETJ Z/RETJ NZ, RETJ S Conditional Return from Interrupt on Status Flag

Description: Performs conditional return (RETJ) based upon the state of a specific processor status flag. RETJ C returns if the Carry flag is set while RETJ NC returns if the Carry flag is clear. RETJ Z returns if the Zero flag is set while RETJ NZ returns if the Zero flag is clear. RETJ S returns if the Sign flag is set. See RETJ for additional information on the return operation.

Status Flags: None

RETJ C

Operation: C=1: IP ← @SP--
INS ← 0
C=0: IP ← IP + 1

Encoding: 15 0

1010	1100	1000	1101
------	------	------	------

Example(s): RETJ C ; C=1, return from interrupt (RETJ) is performed

RETJ NC

Operation: C=0: IP ← @SP--
INS ← 0
C=1: IP ← IP + 1

Encoding: 15 0

1110	1100	1000	1101
------	------	------	------

Example(s): RETJ NC ; C=1, return from interrupt (RETJ) does not occur

RETI Z

Operation: Z=1: IP ← @SP--
INS ← 0
Z=0: IP ← IP + 1

Encoding: 15 0

1001	1100	1000	1101
------	------	------	------

Example(s): RETI Z ; Z=0, return from interrupt (RETI) does not occur

RETI NZ

Operation: Z=0: IP ← @SP--
INS ← 0
Z=1: IP ← IP + 1

Encoding: 15 0

1101	1100	1000	1101
------	------	------	------

Example(s): RETI NZ ; Z=0, return from interrupt (RETI) is performed

RETI S

Operation: S=1: IP ← @SP--
INS ← 0
S=0: IP ← IP + 1

Encoding: 15 0

1100	1100	1000	1101
------	------	------	------

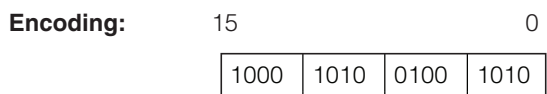
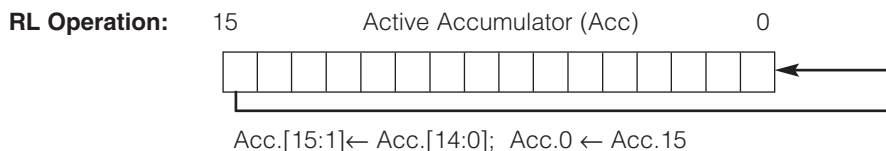
Example(s): RETI S ; S=0, return from interrupt (RETI) does not occur

RL/RLC

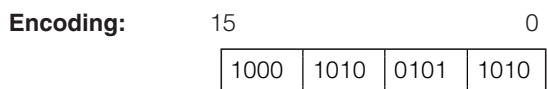
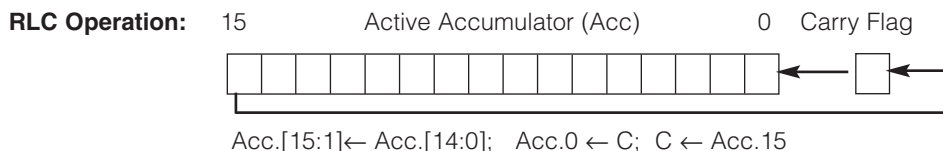
Rotate Left Accumulator Carry Flag (Ex/In)clusive

Description: Rotates the active accumulator left by a single bit position. The RL instruction circulates the msb of the accumulator (bit 15) back to the lsb (bit 0) while the RLC instruction includes the Carry (C) flag in the circular left shift.

Status Flags: C (for RLC only), S, Z (for RLC only)



Example(s): ; Acc = A345h, S=1, Z=0
 RL ; Acc = 468Bh, S=0, Z=0
 RL ; Acc = 8D16h, S=1, Z=0



Example(s): ; Acc = A345h, C=1, S=1, Z=0
 RLC ; Acc = 468Bh, C=1, S=0, Z=0
 RLC ; Acc = 8D17h, C=0, S=1, Z=0

RR/RRC

Rotate Right Accumulator Carry Flag (Ex/In)clusive

Description: Rotates the active accumulator right by a single bit position. The RR instruction circulates the lsb of the accumulator (bit 0) back to the msb (bit 15) while the RRC instruction includes the Carry (C) flag in the circular right shift.

Status Flags: C (for RRC only), S, Z (for RRC only)

RR Operation: 15 Active Accumulator (Acc) 0



$Acc.[14:0] \leftarrow Acc.[15:1]; Acc.15 \leftarrow Acc.0$

Encoding: 15 0

1000	1010	1101	1010
------	------	------	------

Example(s): ;Acc = A345h, S=1, Z=0
 RR ; Acc = D1A2h, S=1, Z=0
 RR ; Acc = 68D1h, S=0, Z=0

RRC Operation: 15 Active Acc (Acc) 0 Carry Flag



$Acc.[14:0] \leftarrow Acc.[15:1]; Acc.15 \leftarrow C; C \leftarrow Acc.0$

Encoding: 15 0

1000	1010	1101	1010
------	------	------	------

Example(s): ; Acc = A345h, C=1, S=1, Z=0
 RRC ; Acc = D1A2h, C=1, S=1, Z=0
 RRC ; Acc = E8D1h, C=0, S=1, Z=0

SLA/SLA2/SLA4

Shift Accumulator Left Arithmetically One, Two, or Four Times

Description: Shifts the active accumulator left once, twice, or four times respectively for SLA, SLA2, and SLA4. For each shift iteration, a 0 is shifted into the lsb, and the msb is shifted into the Carry (C) flag. For signed data, this shifting process effectively retains the sign orientation of the data to the point at which overflow/underflow would occur.

Status Flags: C, S, Z

SLA Operation: Carry Flag 15 Active Accumulator (Acc) 0



$C \leftarrow \text{Acc}.15$; $\text{Acc}.[15:1] \leftarrow \text{Acc}.[14:0]$; $\text{Acc}.0 \leftarrow 0$

Encoding: 15 0

1000	1010	0010	1010
------	------	------	------

Example(s): ; Acc = E345h, C=0, S=1, Z=0
 SLA ; Acc = C68h, C=1, S=1, Z=0
 SLA ; Acc = 8D14h, C=1, S=1, Z=0

SLA2 Operation: Carry Flag 15 Active Accumulator (Acc) 0



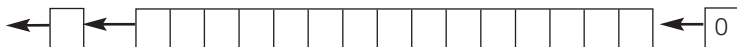
$C \leftarrow \text{Acc}.14$; $\text{Acc}.[15:2] \leftarrow \text{Acc}.[13:0]$; $\text{Acc}.[1:0] \leftarrow 0$

Encoding: 15 0

1000	1010	0011	1010
------	------	------	------

Example(s): ; Acc = E345h, C=0, S=1, Z=0
 SLA2 ; Acc = 8D14h, C=1, S=1, Z=0

SLA4 Operation: Carry Flag 15 Active Accumulator (Acc) 0



$C \leftarrow \text{Acc}.12$; $\text{Acc}.[15:4] \leftarrow \text{Acc}.[11:0]$; $\text{Acc}.[3:0] \leftarrow 0$

Encoding: 15 0

1000	1010	1011	1010
------	------	------	------

Example(s): ; Acc = E345h, C=0, S=1, Z=0
 SLA4 ; Acc = 3450h, C=0, S=0, Z=0

SR/SRA/SRA2/SRA4

Shift Accumulator Right/ Shift Accumulator Right Arithmetically One, Two, or Four Times

Description: Shifts the active accumulator right once for the SR, SRA instructions and 2 or 4 times, respectively, for the SRA2, SRA4 instructions. The SR instruction shifts a 0 into the accumulator msb while the SRA, SRA2, and SRA4 instructions effectively shift a copy of the current msb into the accumulator, thereby preserving any sign orientation. For each shift iteration, the accumulator lsb is shifted into the Carry (C) flag.

Status Flags: C, S (changes for SR only), Z

SR Operation: 15 Active Accumulator (Acc) 0 Carry Flag



$Acc.15 \leftarrow 0$; $Acc.[14:0] \leftarrow Acc.[15:1]$; $C \leftarrow Acc.0$

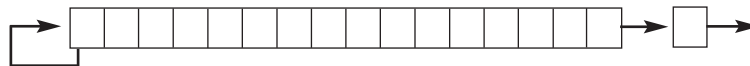
Encoding: 15 0

1000	1010	1010	1010
------	------	------	------

Example(s):

	; Acc = A345h, C=1, S=1, Z=0
SR	; Acc = 51A2h, C=1, S=0, Z=0
SR	; Acc = 28D1h, C=0, S=0, Z=0

SRA Operation: 15 Active Accumulator (Acc) 0 Carry Flag



$Acc.[14:0] \leftarrow Acc.[15:1]$

$Acc.15 \leftarrow Acc.15$

$C \leftarrow Acc.0$

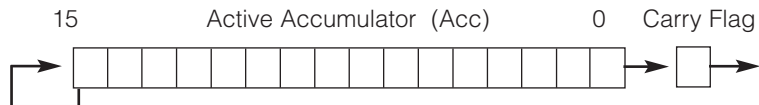
Encoding: 15 0

1000	1010	1111	1010
------	------	------	------

Example(s):

	; Acc = 0003h, C=0, Z=0
SRA	; Acc = 0001h, C=1, Z=0
SRA	; Acc = 0000h, C=1, Z=1

SRA2 Operation:

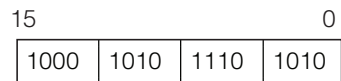


Acc.[13:0] ← Acc.[15:2]

Acc.[15:14] ← Acc.15

C ← Acc.1

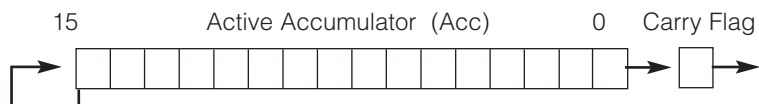
Encoding:



Example(s):

SRA2 ; Acc = 0003h, C=0, Z=0
 SRA2 ; Acc = 0000h, C=1, Z=1

SRA4 Operation:

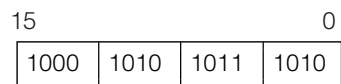


Acc.[11:0] ← Acc.[15:4]

Acc.[15:12] ← Acc.15

C ← Acc.3

Encoding:



Example(s):

SRA4 ; Acc = 9878h, C=0, Z=0
 SRA4 ; Acc = F987h, C=1, Z=0
 SRA4 ; Acc = FF98h, C=0, Z=0

SUB/SUBB src

Subtract/Subtract with Borrow

Description: Subtracts the specified src from the active accumulator (Acc) and returns the result back to the active accumulator. The SUBB additionally subtracts the borrow (Carry Flag), which may have resulted from previous subtraction. For the complete list of src specifiers, reference the MOVE instruction. The MAXQ7667 may use the PFX[n] register to supply the high byte of data for 8-bit sources.

Status Flags: C, S, Z, OV

SUB Operation: $Acc \leftarrow Acc - src$

Encoding: 15 0

f101	1010	ssss	ssss
------	------	------	------

Example(s):

```

; Acc = 2345h to start, A[1]= 1250h
SUB  A[1]      ; Acc = 10F5h, C=0, S=0, Z=0, OV=0
SUB  A[1]      ; Acc = FEA5h, C=1, S=1, Z=0, OV=0
SUB  A[2]      ; A[2] =7FFFh
; → Acc = 7EA6h; C=0, S=0, Z=0, OV=1
    
```

SUBB Operation: $Acc \leftarrow Acc - (src + C)$

Encoding: 15 0

f111	1010	ssss	ssss
------	------	------	------

Example(s):

```

; Acc = 2345h, A[1]= 1250h, C=1
SUBB A[1]      ; Acc = 10F4h, C=0, S=0, Z=0
SUBB A[1]      ; Acc = FEA4h, C=1, S=1, Z=0
    
```

Special Notes: The active accumulator (Acc) is not allowed as the src for these operations.

XCH	Exchange Accumulator Bytes
------------	-----------------------------------

Description: Exchanges the upper and lower bytes of the active accumulator.

Status Flags: S

Operation: Acc.[15:8] ← Acc.[7:0]
Acc.[7:0] ← Acc.[15:8]

Encoding: 15 0

1000	1010	1000	1010
------	------	------	------

Example(s): ; Acc = 2345h
XCHN ; Acc = 4523h

XCHN	Exchange Accumulator Nibbles
-------------	-------------------------------------

Description: Exchanges the upper and lower nibbles in the active accumulator byte(s).

Status Flags: S

Operation: Acc.[7:4] ← Acc.[3:0]
Acc.[3:0] ← Acc.[7:4]
Acc.[15:12] ← Acc.[11:8]
Acc.[11:8] ← Acc.[15:12]

Encoding: 15 0

1000	1010	0111	1010
------	------	------	------

Example(s): ; Acc = 2345h
XCHN ; Acc = 3254h

XOR src	Logical XOR				
<p>Description: Performs a logical-XOR between the active accumulator (Acc or A[AP]) and the specified src data. For the complete list of src specifiers, reference the MOVE instruction. The MAXQ7667 may use the PFX[n] register to supply the high byte of data for 8-bit sources.</p>					
<p>Status Flags: S, Z</p>					
<p>Operation: Acc ← Acc XOR src</p>					
<p>Encoding: 15 0</p> <table border="1" style="margin-left: 20px;"> <tr> <td>f011</td> <td>1010</td> <td>ssss</td> <td>ssss</td> </tr> </table>		f011	1010	ssss	ssss
f011	1010	ssss	ssss		
<p>Example(s): ; Acc = 2345h XOR A[2] ; A[2]=0F0Fh; Acc ← 2C4Ah</p>					
<p>Special Notes: The active accumulator (Acc) is not allowed as the src for this operation.</p>					

XOR Acc.	Logical XOR Carry Flag with Accumulator Bit				
<p>Description: Performs a logical-XOR between the Carry (C) status flag and a specified bit of the active accumulator (Acc.) and returns the result to the Carry.</p>					
<p>Status Flags: C</p>					
<p>Operation: C ← C XOR Acc.</p>					
<p>Encoding: 15 0</p> <table border="1" style="margin-left: 20px;"> <tr> <td>1011</td> <td>1010</td> <td>bbbb</td> <td>1010</td> </tr> </table>		1011	1010	bbbb	1010
1011	1010	bbbb	1010		
<p>Example(s): ; Acc = 2345h, C=1 at start XOR Acc.1 ; Acc.1=0 → C=1 XOR Acc.2 ; Acc.2=1 → C=0</p>					

REVISION HISTORY

REVISION NUMBER	REVISION DATE	SECTION NUMBER	DESCRIPTION	PAGES CHANGED
0	4/09	—	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.

Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA 94086 408-737-7600 _____ **R1**