Analog Devices Inc./Maxim Integrated - MAXQ2010-RFX+ Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	-
Core Size	16-Bit
Speed	10MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LCD, POR, PWM, WDT
Number of I/O	55
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 8x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	100-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/analog-devices/maxq2010-rfx

TABLE OF CONTENTS

SECTION 2: ARCHITECTURE

The MAXQ 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 2-1 illustrates the modular architecture and the basic transport possibilities.

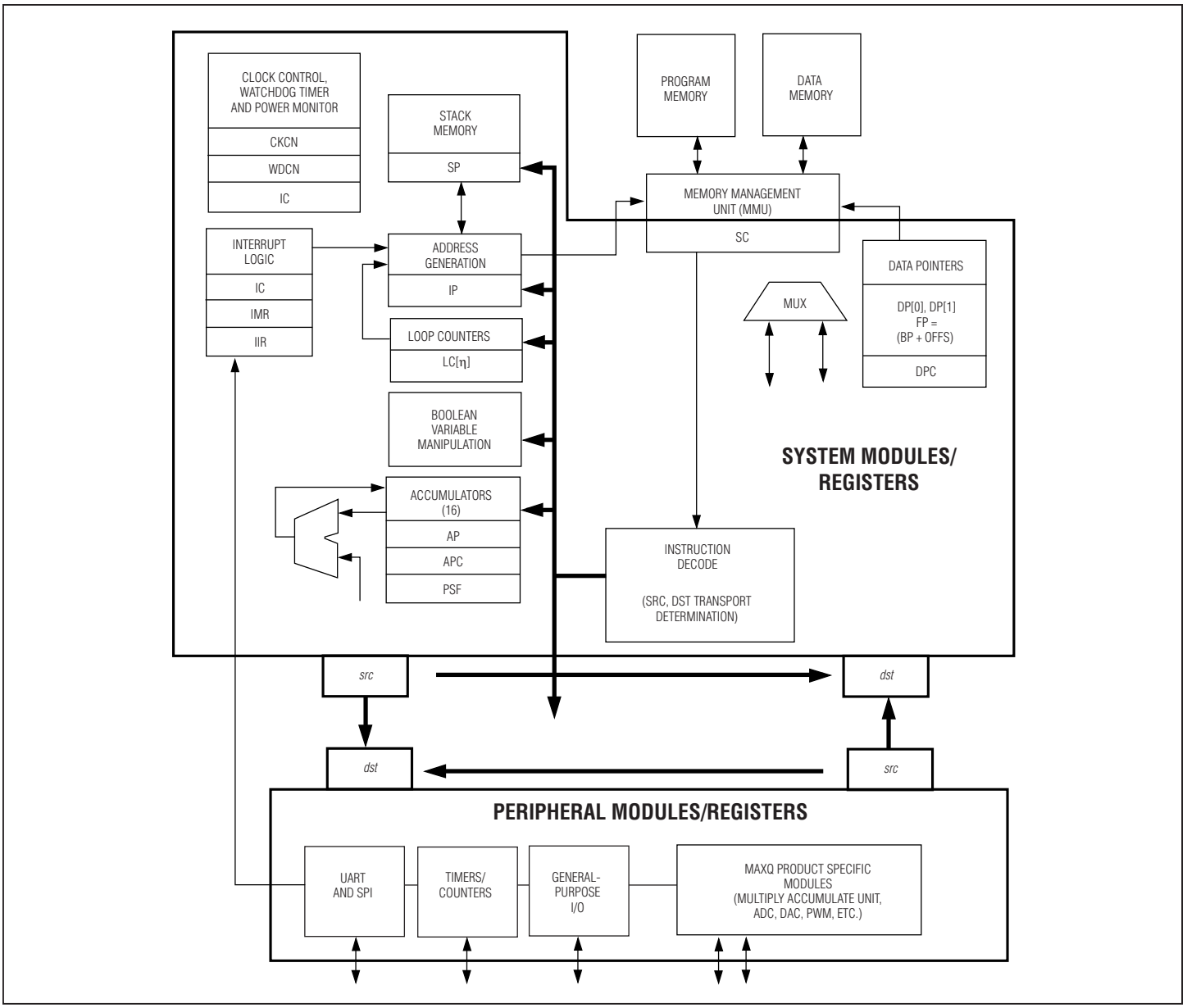


Figure 2-1. MAXQ Transport-Triggered Architecture

resistor to ensure a satisfactory logic level for active clock pulses. To minimize system noise on the clock circuitry, the external clock source must meet the maximum rise and fall times and the minimum high and low times specified for the clock source. The external noise can affect clock generation circuit if these parameters do not meet the specification.

2.7.4 External RC

For timing-insensitive applications, the external RC option offers additional cost savings. The RC oscillator frequency is a function of the supply voltage, external resistor (Rext) and capacitor (Cext) values and tolerances, and the operating temperature. In addition to this, the oscillator frequency varies from unit to unit due to normal process parameter variation. Figure 2-9 shows how the external RC combination is connected to the MAXQ microcontroller.

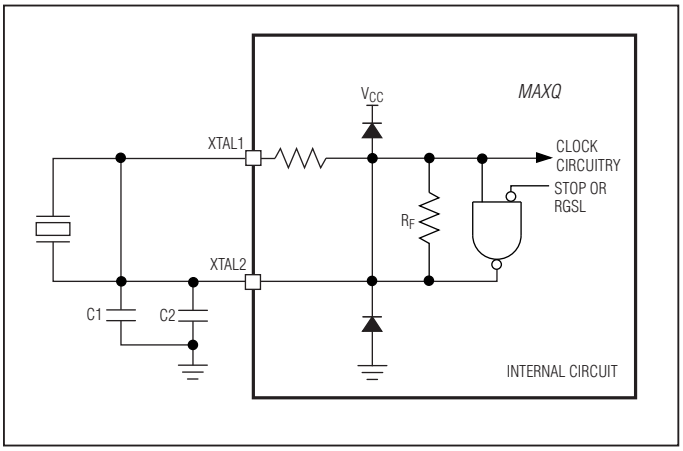


Figure 2-8. On-Chip Crystal Oscillator

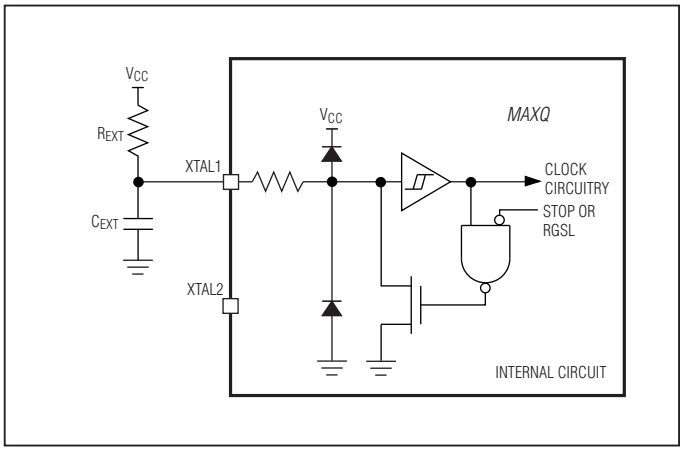


Figure 2-9. RC Relaxation Oscillator

2.7.5 Internal System Clock Generation

The internal system clock is derived from the currently selected oscillator input.

By default, one system clock cycle is generated per oscillator cycle, but the number of oscillator cycles per system clock can also be increased by setting the Power Management Mode Enable (PMME) bit and the Clock Divide Control (CD[1:0]) register bits per Table 2-2.

Table 2-2. System Clock Rate Control Settings

PMME	CD[1:0]	CYCLES PER CLOCK
0	00	1 (default)
0	01	2
0	10	4
0	11	8
1	xx	256

2.8 Interrupts

The MAXQ 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 MAXQ 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.

2.8.1 Servicing Interrupts

For the MAXQ 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 2-10.

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.

2.8.2 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 MAXQ 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.

- if the system clock divide ratio is 2, the interrupt request is recognized after 2 system clock;
- if the system clock divide ratio is 4 or greater, the interrupt request is recognized after 1 system clock;

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.

2.8.4 Interrupt Prioritization by Software

All interrupt sources of the MAXQ 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 the *Handling Interrupts* section of *Section 3: Programming*.

2.8.5 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 MAXQ 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.

2.9 Operating Modes

In addition to the standard program execution mode, there are three other operating modes for the MAXQ. During Reset Mode, the processor is temporarily halted by an external or internal reset source. During Power Management Mode, the processor executes instructions at a reduced clock rate to decrease power consumption. Stop Mode halts execution and all internal clocks to save power until an external stimulus indicates that processing should be resumed.

2.9.1 Reset Mode

When the MAXQ microcontroller is in Reset Mode, 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, the processor begins executing code at address 8000h.

There are four different sources that can cause the MAXQ to enter Reset Mode:

- Power-On/Brownout Reset
- External Reset
- Watchdog Timer Reset
- Internal System Reset

2.9.1.1 Power-On/Brownout Reset

An on-chip power-on reset (POR) circuit is provided to ensure proper initialization on internal device states. The power-on reset circuit provides a minimum power-on-reset delay sufficient to accomplish this initialization. For fast V_{DD} supply rise times, the MAXQ device will, at a minimum, be held in reset for the power-on reset delay when initially powered up. For slow V_{DD} supply rise times, the MAXQ device will be held in reset until V_{DD} is above the power-on-reset voltage threshold. The minimum POR delay and POR voltage threshold can differ depending upon MAXQ device. Refer to the device data sheet(s) for specifics.

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 0's 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 MAXQ provides either an 8-bit (MAXQ10) or 16-bit (MAXQ20) ALU, which allows operations to be performed between the active accumulator and any other register. The default ALU configuration provides eight accumulator registers that are also either 8-bit (MAXQ10) or 16-bit (MAXQ20) wide, of which any one may be selected as the active accumulator. Many MAXQ devices will be equipped with 16 working accumulators.

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 Auto-Increment and Auto-Decrement

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 auto-increment/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, 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.<b> (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.<b>
- OR Acc.<b>
- XOR Acc.<b>
- MOVE C, src.<b> (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.<b>, C (Set selected active accumulator bit to Carry)
- AND Acc.<b> (Carry = Carry AND selected active accumulator bit)
- OR Acc.<b> (Carry = Carry OR selected active accumulator bit)
- XOR Acc.<b> (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 MAXQ 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.

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++]
```

3.11 Using the Watchdog Timer

The Watchdog Timer is a user-programmable clock counter that can serve as a time-base generator, an event timer, or a system supervisor. As can be seen in the diagram below, the timer is driven by the main system clock and is supplied to a series of dividers. If the watchdog interrupt and the watchdog reset are disabled (EWDI = 0 and EWT = 0), the watchdog timer and its input clock are disabled. Whenever the watchdog timer is disabled, the watchdog interval timer (per WD1:0 bits) and 512 clock reset counter will be reset if either the interrupt or reset function is enabled. When the watchdog timer is initially enabled, there will be a 1-clock to 3-clock cycle delay before it starts. The divider output is selectable, and determines the interval between timeouts. When the timeout is reached, an interrupt flag is set, and if enabled, an interrupt occurs. A watchdog-reset function is also provided in addition to the watchdog interrupt. The reset and interrupt are completely discrete functions that can be acknowledged or ignored, together or separately for various applications.

Table 3-2. Watchdog Timer Register Control Bits

BIT NAME	DESCRIPTION	REGISTER LOCATION	BIT POSITION
EWDI	Enable Watchdog Timer Interrupt	WDCN (Fh, 8h)	WDCN.6
WD1, WD0	Watchdog Interval Control Bits		WDCN.5,4
WDIF	Watchdog Interrupt Flag		WDCN.3
WTRF	Watchdog Timer Reset Flag		WDCN.2
EWT	Enable Watchdog Timer Reset		WDCN.1
RWT	Reset Watchdog Timer		WDCN.0

The Watchdog Timer Reset function works as follows. After initializing the correct timeout interval (discussed below), software can enable, if desired, the reset function by setting the Enable Watchdog Timer Reset (EWT = WDCN.1) bit. Setting the EWT bit will reset/restart the Watchdog timer if the Watchdog interrupt is not already enabled. At any time prior to reaching its user selected terminal value, software can set the Reset Watchdog Timer (RWT = WDCN.0) bit. If the Watchdog Timer is reset (RWT bit written to a logic 1) before the timeout period expires, the timer will start over. Hardware will automatically clear RWT after software sets it.

(T1CN.2) must also be set to logic 1 to enable the timer. The DCEN bit has no effect in this mode. This mode produces a 50% duty cycle square-wave output. The frequency of the square wave is given by the formula in Figure 8-4. Each timer overflow causes an edge transition on the pin, i.e., the state of the pin toggles. Note that the timer itself does not generate an interrupt, but if needed, the Timer 1 external interrupt is still available for use when enabled (EXEN1 = 1).

8.2 Timer/Counter 1 Peripheral Registers

8.2.1 Timer/Counter 1 Control Register (T1CN)

Bit #	7	6	5	4	3	2	1	0
Name	TF1	EXF1	T1OE	DCEN	EXEN1	TR1	C/$\overline{T1}$	CP/$\overline{RL1}$
Reset	0	0	0	0	0	0	0	0
Access	rw	rw	rw	rw	rw	rw	rw	rw

r = read, w = write

Bit 7: Timer 1 Overflow Flag (TF1). This bit is set when Timer 1 overflows from FFFFh or the count is equal to the capture register in down count mode. It must be cleared by software.

Bit 6: External Timer 1 Trigger Flag (EXF1). A negative transition on the T1EX causes this flag to be set if (CP/$\overline{RL1}$ = EXEN1 = 1) or (CP/$\overline{RL1}$ = DCEN = 0 and EXEN1 = 1). When CP/$\overline{RL1}$ = 0 and DCEN = 1, EXF1 toggles whenever Timer 1 underflows or overflows. In this mode, EXF1 can be used as the 17th Timer bit and will not cause an interrupt. If set by a negative transition, this flag must be cleared by software. Setting this bit to 1 forces a Timer interrupt if enabled.

Bit 5: Timer 1 Output Enable (T1OE). Setting this bit to 1 enables the clock output function of T1 pin if C/$\overline{T1}$ = 0. Timer 1 rollovers will not cause interrupts. Clearing this bit to 0 causes the T1 pin to function as either a standard port pin or a counter input for Timer 1.

Bit 4: Down Count Enable (DCEN). This bit, in conjunction with the T1EX pin, controls the direction that Timer 1 counts in 16-bit auto-reload mode. Clearing this bit to 0 causes Timer 1 to count up. Setting this bit to 1 causes Timer 1 to count up if the T1EX pin is 1 and to count down if the T1EX pin is 0.

Bit 3: Timer 1 External Enable (EXEN1). Setting this bit to 1 enables the capture/reload function on the T1EX pin for a negative transition. Clearing this bit to 0 causes Timer 1 to ignore all external events on T1EX pin.

Bit 2: Timer 1 Run Control (TR1). Setting this bit enables Timer 1. Clearing this bit halts the Timer 1.

Bit 1: Counter/Timer Select (C/$\overline{T1}$). This bit determines whether Timer 1 functions as a Timer or counter. Setting this bit to 1 causes Timer 1 to count negative transitions on the T1 pin. Clearing this bit to 0 causes Timer 1 to function as a Timer. The speed of Timer 1 is determined by the T1M bit, either divide by 1 or divide by 12 of the system clock, including the clock output mode.

Bit 0: Capture/Reload Select (CP/$\overline{RL1}$). This bit determines whether the capture or reload function is used for Timer 1. Timer 1 functions in an auto-reload mode following each overflow. Setting this bit to 1 causes a Timer 1 capture to occur when a falling edge is detected on T1EX if EXEN1 = 1. Clearing this bit to 0 causes an auto-reload to occur when Timer 1 overflow or a falling edge is detected on T1EX if EXEN1 = 1.

SECTION 9: TIMER/COUNTER 2 MODULE

The Timer/Counter 2 Module provides a 16-bit programmable timer/counter with pulse-width modulation capability. Whether and how many Timer 2 Modules are implemented in a given MAXQ-based microcontroller is product dependent.

Timer 2 is an auto-reload, 16-bit timer/counter offering the following functions:

- 8-bit/16-bit timer/counter
- up/down auto-reload
- counter function of external pulse
- capture
- compare
- input/output enhancements

9.1 Timer 2

The 16-bit Timer 2 value is contained in the T2V register. The upper byte is always accessible through the T2H 8-bit register. When Timer 2 is operated in the dual 8-bit mode of operation, the high byte of T2V always reads x00h and is not write accessible. The low byte of the T2V will often be referenced as T2L. Similar registers and nomenclature are used for the Timer 2 auto-reload value resides in the T2R register. A separate 8-bit T2RH register allows read/write access to the high byte and the low byte of T2R is often referred to as T2RL. The Capture/Compare functionality is supported by Timer 2 through the 16-bit T2C capture register and the 8-bit T2CH high-byte access register. Timer 2 normally requires two pins to support the enhanced input/output functionality. Throughout this section the primary input/output pin will be referred to as T2P and the secondary pin, which may or may not be present for a given device, will be referred to as T2PB. Decision whether and/or where to implement the T2PB pin functionality is product and application dependent. Table 9-1 summarizes the modes supported by Timer 2 and the peripheral register bits associated with those modes.

Table 9-1. Timer/Counter 2 Functions and Control

MODE	T2MD	C/$\overline{T2}$	CCF[1:0]	CONTROL BITS
16-Bit Auto-Reload/Compare Timer	0	0	00	T2OE[1:0]—output enables (PWM out)
				T2POL[1:0]—input/output polarity select
				SS2—single-shot pulse control
				G2EN—gated PWM output
16-Bit Capture (CCF[1:0] bits define capture edge)	0	0	01, 10, or 11	T2OE[0] = 0
				T2POL[0]—gate level/reload edge select
				SS2—single-shot capture
				G2EN—gate timer clock (or gate reload)
				CPRL2—reload enable
16-Bit Counter (CCF[1:0] bits define count edge)	0	1	01, 10, or 11	T2OE[0] = 0
				T2OE[1]—pulse counter output
				T2POL[1]—output polarity select
Dual 8-Bit Auto-Reload Timers	1	0	00	T2OE[1:0]—output enables (PWM out)
				T2POL[1:0]—output polarity select
				T2H Only:
				SS2—single-shot pulse control
8-Bit Capture and 8-Bit Timer/PWM	1	0	01, 10, or 11	T2L Only:
				T2OE[1]—output enable
				T2POL[1]—output polarity select
				T2H Only:
				T2OE[0] = 0
				T2POL[0]—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	T2L Only:
				T2OE[1]—output enable
				T2POL[1]—output polarity select
				T2H Only:
				T2OE[0] = 0

9.2.1 16-Bit Timer: Auto-Reload/Compare

The 16-bit auto-reload/compare mode for Timer 2 is in effect when the Timer 2 mode select bit (T2MD) is cleared and the capture/compare function definition bits are both cleared (CCF[1:0] = 00b). The Timer 2 value is contained in the T2V register. The Timer 2 run control bit (TR2) starts and stops the 16-bit Timer. The input clock for 16-bit Timer 2 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 T2L:T2H register pair until overflowing. When an overflow occurs, the reload value (T2RH:T2RL) is reloaded instead of the x0000h state. The Timer 2 overflow flag (TF2) is set every time that an overflow condition (T2V = 0xFFFFh) is detected. If Timer 2 interrupts have been enabled (ET2 = 1), the TF2 flag can generate an interrupt request. When operating in compare mode, the capture/compare registers (T2CH:T2CL) are compared versus the Timer 2 value registers. Whenever a compare match occurs, the capture/compare status flag (TCC2) is set. If Timer 2 interrupts have been enabled (ET2 = 1), this event is capable of generating an interrupt request. If the capture/compare register is set to a value outside the Timer 2 counting range, a compare match is not signaled and the TCC2 flag is not set. Internally, a Timer 2 output clock is generated, which 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 will occur. This clock may be sourced by certain peripherals and/or may be output on one or more pins as permitted by the microcontroller.

9.2.1.1 Output Enable (PWM Out)

The Output Enable bits (T2OE[1:0]) enable the Timer 2 output clock to be presented on the pins associated with the respective bits. If Timer 2 has a single I/O pin, the T2OE[0] bit is associated with the T2P pin and the T2OE[1] bit is not implemented (as it would serve no purpose).

9.2.1.2 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 2 output signal, therefore setting a given T2POL[x] bit will result in a high starting state. The T2POL[n] bit can be changed at any time, however the assigned T2POL[n] state will take effect on the external pin only when the corresponding T2OE[n] bit is changed from 0 to 1. When generating PWM output, please note that changing the compare match register can result in a perceived duty cycle inversion if a compare match is missed or multiple compare matches occur during the reload to overflow counting.

9.2.1.3 Gated

To use the T2P pin as a timer-input clock gate, the T2OE[0] bit must be cleared to 0 and the G2EN bit must be set to 1. When T2OE[0] = 1, the G2EN bit setting has no effect. When T2OE[0] 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 2 input clock is gated anytime that the external signal matches the state of the T2POL[0] bit. This means that the default clock gating condition for the T2P pin is logic low (since T2POL[0] = 0 default). Setting T2POL[0] = 1 results in the Timer 2 input clock being gated when the T2P pin is high. Note if multiple pins are allocated for Timer 2 (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 T2OE[1] = 1).

9.2.1.4 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 2 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 1, the specified PWM output continues even after SS2 becomes clear.

If two pins are available for the Timer 2 implementation, an additional mode is supported: single-shot gated. Single-shot gated requires that the T2P pin be used as an input (T2OE[0] = 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 for the single-shot enabled output to occur on the pin. When the clock gate is removed, the single-shot output occurs. Just as 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/non-gated intervals. Once the SS2 = 1 conditions completes, if TR2 = 1, the gated PWM mode is in effect. Otherwise (TR2 = 0), Timer 2 is stopped.

SECTION 11: SERIAL PERIPHERAL INTERFACE (SPI) MODULE

This section contains the following information:

LIST OF FIGURES

11.3 SPI Transfer Baud Rates

When operating as a slave device, an external master drives the SPI serial clock. For proper slave operation, the serial clock provided by the external master should not exceed the system clock frequency divided by 8.

When operating in the master mode, the SPI serial clock is sourced to the external slave device(s). The serial clock baud rate is determined by the clock-divide ratio specified in the SPI Clock Divider Ratio (SPICK) register. The SPI module supports 256 different clock-divide ratio selections for serial clock generation. The SPICK clock rate is determined by the following formula:

SPI Baud Rate = System Clock Frequency / (2 x Clock Divider Ratio)

where Clock Divider Ratio = (SPICK.7:0) + 1

Since the SPI baud rate is a function of the System Clock Frequency, using any of the system clock divide modes (including Power Management Mode) alters the baud rate. Attempts to invoke the Power Management Mode while an SPI transfer in is progress (STBY = 1) are ignored.

Note, however, that once in Power Management Mode (PMME = 1), writes to SPIB in master mode and assertion of the $\overline{SSEL}$ pin in slave mode both qualify as switchback sources if enabled (SWB = 1). The SPI module clocks are halted if the device is placed into Stop mode.

11.4 SPI System Errors

The SPI module can detect three types of SPI system errors. A mode fault error arises in a multiple master system when more than one SPI device simultaneously tries to be a master. A receive overrun error occurs when an SPI transfer completes before the previous character has been read from the receive-holding buffer. The third kind of error, write collision, indicates that an attempted write to SPIB was detected while a transfer was in progress (STBY = 1).

11.4.1 Mode Fault

When a SPI device is configured as a master and its Mode Fault Enable bit (SPICN.2: MODFE) is also set, a mode fault error occurs if $\overline{SSEL}$ input signal is driven low by an external device. This error is typically caused when a second SPI device attempts to function as a master in the system. In the condition where more than one device is configured as master concurrently, there is possibility of bus contention that can cause permanent damage to push-pull CMOS drivers. The mode fault error detection is to provide protection from such damage by disabling the bus drivers. When a mode fault is detected, the following actions are taken immediately:

1) The MSTM bit is forced to logic 0 to reconfigure the SPI device as a slave.

2) The SPIEN bit is forced to logic 0 to disable the SPI module.

3) The Mode Fault (SPICN.3: MODF) status flag is set. Setting the MODF bit can generate an interrupt if it is enabled.

The application software must correct the system conflict before resuming its normal operation. The MODF flag is set automatically by hardware but must be cleared by software or a reset once set. Setting the MODF bit to logic 1 by software causes an interrupt if enabled.

Mode fault detection is optional and can be disabled by clearing the MODFE bit to logic 0. Disabling the mode fault detection disables the function of the $\overline{SSEL}$ signal during master mode operation, allowing the associated port pin to be used as a general-purpose I/O.

Note that the mode fault mechanism does not provide full protection from bus contention in multiple master, multiple slave systems. For example, if two devices are configured as master at the same time, the mode fault-detect circuitry offers protection only when one of them selects the other as slave by asserting its $\overline{SSEL}$ signal. Also, if a master accidentally activates more than one slave and those devices try to simultaneously drive their output pins, bus contention can occur without and a mode fault error being generated.

11.4.2 Receive Overrun

Since the receive direction of SPI is double buffered, there is no overrun condition as long as the received character in the read buffer is read before the next character in the shift register ready to be transferred to the read buffer. However, if previous data in the read buffer has not been read out when a transfer cycle is completed and the new character is loaded into the read buffer, a receive overrun occurs and the Receive Overrun flag (SPICN.5: ROVR) is set. Setting the ROVR flag indicates that the oldest received character has been overwritten and is lost. Setting the ROVR bit to logic 1 causes an interrupt if enabled. Once set, the ROVR bit is cleared only by software or a reset.

12.5.3 Multiplier Operand B Register (MB)

Bit #	15	14	13	12	11	10	9	8
Name	MB.15	MB.14	MB.13	MB.12	MB.11	MB.10	MB.9	MB.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	MB.7	MB.6	MB.5	MB.4	MB.3	MB.2	MB.1	MB.0
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: Multiplier Operand B Register (MB.[15:0]). This operand B register is used by the application code to load 16-bit values for multiplier operations.

12.5.4 Multiplier Accumulator 2 Register (MC2)

Bit #	15	14	13	12	11	10	9	8
Name	MC2.15	MC2.14	MC2.13	MC2.12	MC2.11	MC2.10	MC2.9	MC2.8
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	MC2.7	MC2.6	MC2.5	MC2.4	MC2.3	MC2.2	MC2.1	MC2.0
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: Multiplier Accumulator 2 Register (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. The MC2 register width is dependent upon the hardware multiplier accumulator width of the given MAXQ device. For a MAXQ device having only a 32-bit accumulator, the MC2 register will not be present.

12.5.7 Multiplier Read Register 1 (MC1R)

Bit #	15	14	13	12	11	10	9	8
Name	MC1R.15	MC1R.14	MC1R.13	MC1R.12	MC1R.11	MC1R.10	MC1R.9	MC1R.8
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	MC1R.7	MC1R.6	MC1R.5	MC1R.4	MC1R.3	MC1R.2	MC1R.1	MC1R.0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Bits 15 to 0: Multiplier Read Register 1 (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 content of this register may change if MCNT, MA, MB, or MC[2:0] is changed.

12.5.8 Multiplier Read Register 0 (MC0R)

Bit #	15	14	13	12	11	10	9	8
Name	MC0R.15	MC0R.14	MC0R.13	MC0R.12	MC0R.11	MC0R.10	MC0R.9	MC0R.8
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	MC0R.7	MC0R.6	MC0R.5	MC0R.4	MC0R.3	MC0R.2	MC0R.1	MC0R.0
Reset	0	0	0	0	0	0	0	0
Access	r	r	r	r	r	r	r	r

r = read

Bits 15 to 0: Multiplier Read Register 0 (MC0R.[15:0]). The MC1R 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 content of this register may change if MCNT, MA, MB or MC[2:0] is changed.

Table 13-2. ROM ID Read Time Slot Possibilities

READ TIME SLOT 1 (SLAVE)	READ TIME SLOT 2 (SLAVE)	WRITE TIME SLOT (MASTER)	DESCRIPTION
0	1	0	All slave devices remaining in the selection process have a 0 in this ROM ID bit position.
1	0	1	All slave devices remaining in the selection process have a 1 in this ROM ID bit position.
0	0	0 or 1	ID Discrepancy—Slave devices remaining in the selection process have both 0 and 1 in this ROM ID bit position. The Bus Master write time slot dictates which devices remain in the selection process.
1	1	1	Error—No slave devices responded during the read time slots.

The general principle of this search process is to deselect slave devices at every conflicting bit position. At the end of each ROM Search process, the master has learned another ROM ID. A pass of search process takes 64 reading/selection cycles for the master to learn one device's ROM ID. Each reading/selection cycle, as noted above, consists of two Read time slots and a Write time slot. Subsequent search passes are performed identically to the last up until the point of the last decision. For details of Search ROM algorithm in the 1-Wire system, refer to the *Book of iButton Standards*.

To speed up this ROM ID Search process, the 1-Wire Bus Master incorporates a Search ROM Accelerator. To enable the Search ROM accelerator, the SRA bit in the Command Register must be set immediately following the Reset sequence and issuance of the Search ROM command. Note that the receive buffer must empty before invoking SRA mode. After the bus master is placed in Search ROM Accelerator mode, each byte loaded into the transmit buffer contains one nibble (4 bits) worth of discrepancy decision data. The two slave read time slots are automatically generated by the bus master as a part of the transmit sequence. After four reading/selection cycles, the receive buffer data will reflect four newly acquired bits of the ROM ID and four corresponding bits flagging whether a discrepancy existed in a given bit position. Table 13-3 details the format for the transmit and receive data (when in Search ROM Accelerator mode).

Table 13-3. Search ROM Accelerator Transmit/Receive Byte Sequence

BYTE SEQUENCE	BUFFER	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Byte 1	Transmit	r_3	x	r_2	x	r_1	x	r_0	x
	Receive	ID_3	d_3	ID_2	d_2	ID_1	d_1	ID_0	d_0
Byte 2	Transmit	r_7	x	r_6	x	r_5	x	r_4	x
	Receive	ID_7	d_7	ID_6	d_6	ID_5	d_5	ID_4	d_4
• • •									
Byte 16	Transmit	r_{63}	x	r_{62}	x	r_{61}	x	r_{60}	x
	Receive	ID_{63}	d_{63}	ID_{62}	d_{62}	ID_{61}	d_{61}	ID_{60}	d_{60}

r_n = decision discrepancy data (write time slot selection data if ID discrepancy)
ID_n = selected ROM ID bit (r_n if discrepancy occurred, otherwise read time slot 1)
d_n = discrepancy detected flag (ID discrepancy or no response)
x = don't care data

The CPU must send and receive 16 bytes of data to complete a single Search ROM pass on the 1-Wire bus. To perform a Search ROM sequence one starts with all decision discrepancy bits (r_n) being 0. In case of bus error, all subsequent response bits IDn are 1s until the Search Accelerator is deactivated by clearing the SRA bit in the Command Register. Thus if ID_{63} and d_{63} are both 1, an error has occurred during the search process and the last sequence has to be repeated. Otherwise, $ID_{63:0}$ is the ROM code of the device that has been found and addressed. For the next Search ROM sequence one reuses the previous set r_n (for n = 0:63), changing to 1 only that bit position where the highest discrepancy was detected (d_n flags). This process is repeated until the highest discrepancy occurs in the same bit position for two passes, then the next lower discrepancy flag is used for next search. When the Search ROM process is completed, the SRA bit should be cleared in order to release the 1-Wire Master from Search ROM Accelerator mode.

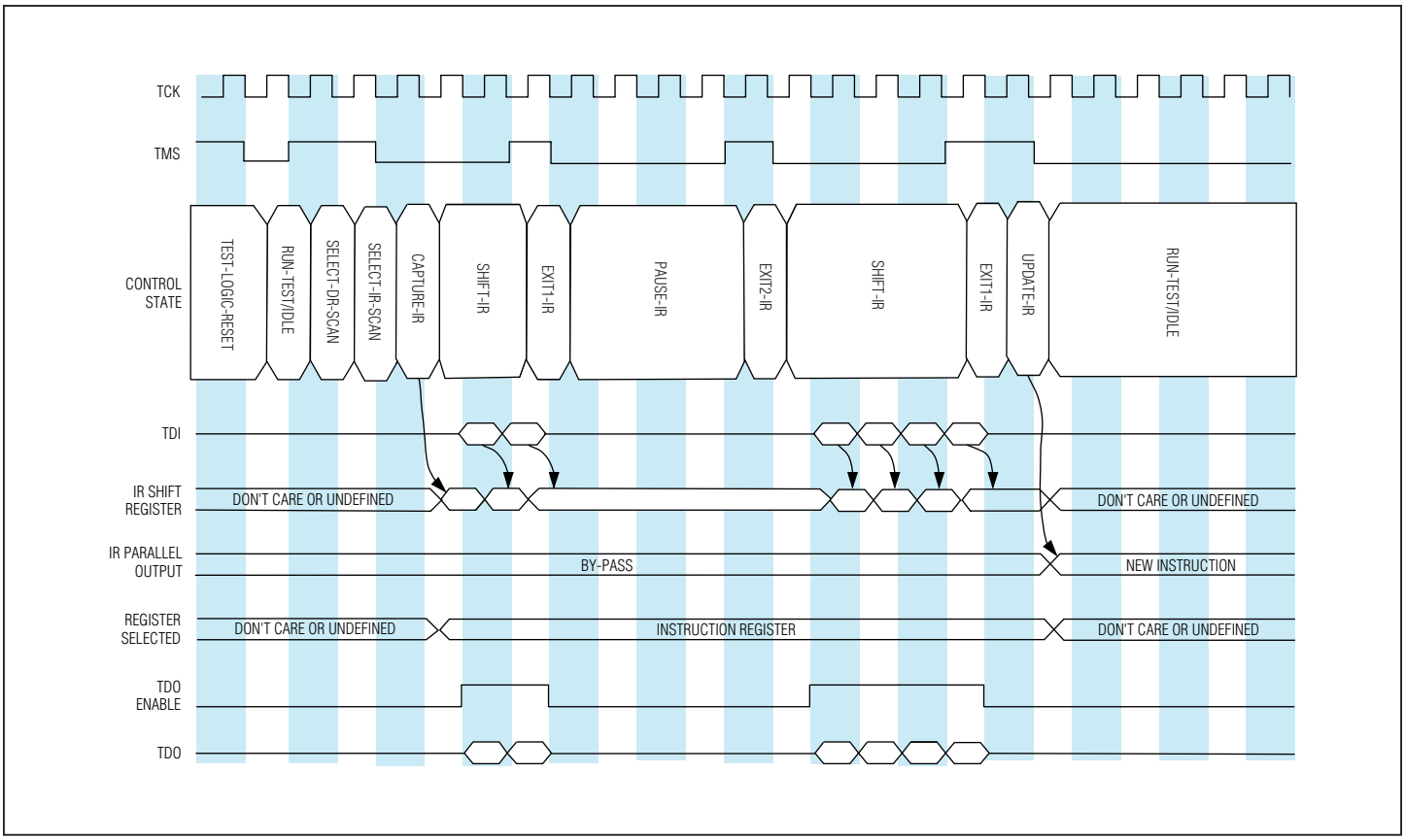


Figure 15-3. TAP Controller Debug Mode IR-Scan Example

16.1.1.3 Breakpoint 2 Register (BP2)

Bit #	15	14	13	12	11	10	9	8
Name	BP2.15	BP2.14	BP2.13	BP2.12	BP2.11	BP2.10	BP2.9	BP2.8
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	BP2.7	BP2.6	BP2.5	BP2.4	BP2.3	BP2.2	BP2.1	BP2.0
Reset	1	1	1	1	1	1	1	1
Access	s	s	s	s	s	s	s	S

s = special

Bits 15 to 0: Breakpoint 2 (BP2.[15:0]). This register is accessible only via background mode read/write commands. Breakpoint registers BP0, BP1, BP2, and BP3 serve as program memory address breakpoints. When DME bit is set in background mode, the debug engine monitors the program-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.

16.1.1.4 Breakpoint 3 Register (BP3)

Bit #	15	14	13	12	11	10	9	8
Name	BP3.15	BP3.14	BP3.13	BP3.12	BP3.11	BP3.10	BP3.9	BP3.8
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	BP3.7	BP3.6	BP3.5	BP3.4	BP3.3	BP3.2	BP3.1	BP3.0
Reset	1	1	1	1	1	1	1	1
Access	s	s	s	s	s	s	s	S

s = special

Bits 15 to 0: Breakpoint 3 (BP3.[15:0]). This register is accessible only via background mode read/write commands. Breakpoint registers BP0, BP1, BP2, and BP3 serve as program memory address breakpoints. When DME bit is set in background mode, the debug engine monitors the program-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.

16.3.4 In-Circuit Debug Flag Register (ICDF)

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 to 2: Programming Source Select Bits 1:0 (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 (SPB) register of the TAP module (which are accessible via JTAG). These bits are read/write accessible for the CPU and are cleared to 0 by a power-on reset or Test-Logic-Reset. CPU writes to the PSS bits result in clearing of the JTAG PSS[1:0] bits.

PSS1	PSS0	SOURCE SELECTION
0	0	JTAG
0	1	UART
1	0	SPI
1	1	Reserved

Bit 1: System Program Enable (SPE). The SPE bit is used for in-system programming support and its logical state, when by the CPU, always reflects the logical-OR of the SPE bit that is write accessible by the CPU and SPR bit of the System Programming Buffer (SPB) Register in the TAP Module (which is accessible via JTAG.) The logical state of this bit determines the program flow after a reset. When it is set to logic 1, in-system programming is executed by the Utility ROM. When it is cleared to 0, execution is transferred to user code. This but allows read/write access by the SPU and is cleared to 0 only on a power-on reset or Test-Logic-Reset. The JTAG SPE bit is cleared by hardware when the ROD bit is set. CPU writes to the SPE bit result in clearing the JTAG PSS[1:0] bits.

Bit 0: Serial Transfer Complete (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. The TXC bit is cleared by the debug engine once set. CPU writes to the TXC bit results in clearing of the JTAG PSS[1:0] bits.

16.3.5 In-Circuit Debug Buffer Register (ICDB)

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

r = read, w = write

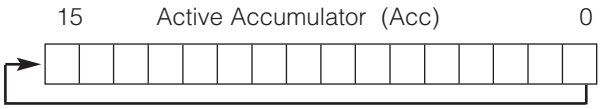
Bits 7 to 0: In-Circuit Debug Buffer Register (ICDB.[7:0]). This register serves as the parallel holding buffer for the debug shift register of the TAP. Data is read from or written to ICDB for serial communication between the debug routines and the external host.

(MAXQ20 Version) **Rotate Right Accumulator**
RR/RRC **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.

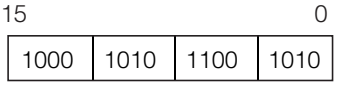
Status Flags: C (for RRC only), S, Z (for RRC only)

RR Operation:



Acc.[14:0]← Acc.[15:1]; Acc.15 ← Acc.0

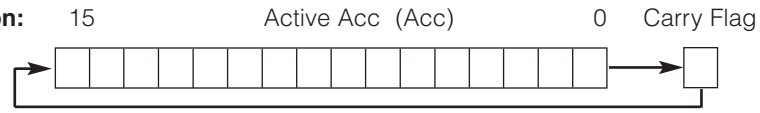
Encoding:



Example(s):

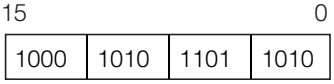
```
                    ;Acc = A345h, S=1, Z=0
RR                  ; Acc = D1A2h, S=1, Z=0
RR                  ; Acc = 68D1h, S=0, Z=0
```

RRC Operation:



Acc.[14:0]← Acc.[15:1]; Acc.15 ← C; C ← Acc.0

Encoding:



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
```
