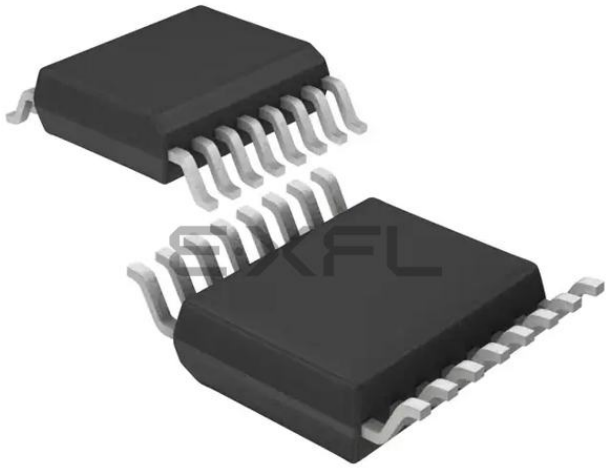


Welcome to E-XFL.COM

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	HC08
Core Size	8-Bit
Speed	8MHz
Connectivity	-
Peripherals	LVD, POR, PWM
Number of I/O	13
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	16-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	16-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc908qy4acdte

This page intentionally blank.

Chapter 1
General Description

1.1 Introduction

The MC68HC908QY4A is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

Table 1-1. Summary of Device Variations

Device	FLASH Memory Size	ADC	Pin Count
MC68HC908QT1A	1536 bytes	—	8 pins
MC68HC908QT2A	1536 bytes	6 channel, 10 bit	8 pins
MC68HC908QT4A	4096 bytes	6 channel, 10 bit	8 pins
MC68HC908QY1A	1536 bytes	—	16 pins
MC68HC908QY2A	1536 bytes	6 channel, 10 bit	16 pins
MC68HC908QY4A	4096 bytes	6 channel, 10 bit	16 pins

1.2 Features

Features include:

- High-performance M68HC08 CPU core
- Fully upward-compatible object code with M68HC05 Family
- 5-V and 3-V operating voltages (V_{DD})
- 8-MHz internal bus operation at 5 V, 4-MHz at 3 V
- Trimmable internal oscillator
  - Software selectable 1 MHz, 2 MHz, or 3.2 MHz internal bus operation
  - 8-bit trim capability
  - ±25% untrimmed
  - Trimmable to approximately 0.4%(1)
- Software selectable crystal oscillator range, 32–100 kHz, 1–8 MHz and 8–32 MHz
- Software configurable input clock from either internal or external source
- Auto wakeup from STOP capability using dedicated internal 32-kHz RC or bus clock source
- On-chip in-application programmable FLASH memory
  - Internal program/erase voltage generation
  - Monitor ROM containing user callable program/erase routines
  - FLASH security(2)

1. See 16.11 Oscillator Characteristics for internal oscillator specifications
2. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the FLASH difficult for unauthorized users.

1.5 Pin Functions

Table 1-2 provides a description of the pin functions.

Table 1-2. Pin Functions

Pin Name	Description	Input/Output
V_{DD}	Power supply	Power
V_{SS}	Power supply ground	Power
PTA0	PTA0 — General purpose I/O port	Input/Output
	TCH0 — Timer Channel 0 I/O	Input/Output
	AD0 — A/D channel 0 input	Input
	KBI0 — Keyboard interrupt input 0	Input
PTA1	PTA1 — General purpose I/O port	Input/Output
	TCH1 — Timer Channel 1 I/O	Input/Output
	AD1 — A/D channel 1 input	Input
	KBI1 — Keyboard interrupt input 1	Input
PTA2	PTA2 — General purpose input-only port	Input
	$\overline{IRQ}$ — External interrupt with programmable pullup and Schmitt trigger input	Input
	KBI2 — Keyboard interrupt input 2	Input
	TCLK — Timer clock input	Input
PTA3	PTA3 — General purpose I/O port	Input/Output
	$\overline{RST}$ — Reset input, active low with internal pullup and Schmitt trigger	Input
	KBI3 — Keyboard interrupt input 3	Input
PTA4	PTA4 — General purpose I/O port	Input/Output
	OSC2 —XTAL oscillator output (XTAL option only) RC or internal oscillator output (OSC2EN = 1 in PTAPUE register)	Output Output
	AD2 — A/D channel 2 input	Input
	KBI4 — Keyboard interrupt input 4	Input
PTA5	PTA5 — General purpose I/O port	Input/Output
	OSC1 — XTAL, RC, or external oscillator input	Input
	AD3 — A/D channel 3 input	Input
	KBI5 — Keyboard interrupt input 5	Input
PTB0(1)	PTB0 — General-purpose I/O port	Input/Output
	AD4 — A/D channel 4 input	Input
PTB1(1)	PTB1 — General-purpose I/O port	Input/Output
	AD5 — A/D channel 5 input	Input
PTB2-PTB7(1)	6 General-purpose I/O port	Input/Output

1. The PTB pins are not available on the 8-pin packages.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE0C	LVI Status Register (LVISR) See page 91.	Read:	LVIOUT	0	0	0	0	0	0	R
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0D ↓ $FE0F	Reserved									
$FFBE	FLASH Block Protect Register (FLBPR) See page 34.	Read:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
		Write:								
		Reset:	Unaffected by reset							
$FFBF	Reserved									
$FFC0	Internal Oscillator Trim (Factory Programmed VDD = 3.0 V)	Read:	TRIM7	TRIM6	TRIM5	TRIM4	TRIM3	TRIM2	TRIM1	TRIM0
		Write:								
		Reset:	Unaffected by reset							
$FFC1	Internal Oscillator Trim (Factory Programmed VDD = 5.0 V)	Read:	TRIM7	TRIM6	TRIM5	TRIM4	TRIM3	TRIM2	TRIM1	TRIM0
		Write:								
		Reset:	Unaffected by reset							
$FFFF	COP Control Register (COPCTL) See page 63.	Read:	LOW BYTE OF RESET VECTOR							
		Write:	WRITING CLEARS COP COUNTER (ANY VALUE)							
		Reset:	Unaffected by reset							

= Unimplemented R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 5 of 5)

2.6.7 EEPROM Memory Emulation Using FLASH Memory

In some applications, the user may want to repeatedly store and read a set of data from an area of nonvolatile memory. This is easily implemented in EEPROM memory because single byte erase is allowed in EEPROM.

When using FLASH memory, the minimum erase size is a page. However, the FLASH can be used as EEPROM memory. This technique is called “EEPROM emulation”.

The basic concept of EEPROM emulation using FLASH is that a page is continuously programmed with a new data set without erasing the previously programmed locations. Once the whole page is completely programmed or the page does not have enough bytes to program a new data set, the user software automatically erases the page and then programs a new data set in the erased page.

In EEPROM emulation when data is read from the page, the user software must find the latest data set in the page since the previous data still remains in the same page. There are many ways to monitor the page erase timing and the latest data set. One example is unprogrammed FLASH bytes are detected by checking programmed bytes (non-$FF value) in a page. In this way, the end of the data set will contain unprogrammed data ($FF value).

A couple of application notes, describing how to emulate EEPROM using FLASH, are available on our web site. Titles and order numbers for these application notes are given at the end of this subsection.

For EEPROM emulation software to work successfully, the following items must be taken care of in the user software:

1. Each FLASH byte in a page must be programmed only one time until the page is erased.
2. A page must be erased before the FLASH cumulative program HV period (t_{HV}) is beyond the maximum t_{HV}. t_{HV} is defined as the cumulative high-voltage programming time to the same row before the next erase. For more detailed information, refer to 16.15 Memory Characteristics.
3. FLASH row erase and program cycles should not exceed 10,000 cycles, respectively.

The above EEPROM emulation software can be easily developed by using the on-chip FLASH routines implemented in the MCU. These routines are located in the ROM memory and support FLASH program and erase operations. Proper utilization of the on-chip FLASH routines guarantee conformance to the FLASH specifications.

In the on-chip FLASH programming routine called PRGRNGE, the high-voltage programming time is enabled for less than 125 μs when programming a single byte at any operating bus frequency between 1.0 MHz and 8.4 MHz. Therefore, even when a row is programmed by 32 separate single-byte programming operations, t_{HV} is less than the maximum t_{HV}. Hence, item 2 listed above is already taken care of by using this routine.

A page erased operation is provided in the FLASH erase routine called ERARNGE.

Application note AN2635 (*On-Chip FLASH Programming Routines*) describes how to use these routines.

The following application notes, available at www.freescale.com, describe how EERPOM emulation is implemented using FLASH:

AN2183 — *Using FLASH as EEPROM on the MC68HC908GP32*

AN2346 — *EEPROM Emulation Using FLASH in MC68HC908QY/QT MCUs*

AN2690 — *Low Frequency EEPROM Emulation on the MC68HC908QY4*

An EEPROM emulation driver, available at www.freescale.com, has been developed and qualified:

AN3040 — *M68HC08 EEPROM Emulation Driver*

Chapter 3
Analog-to-Digital Converter (ADC10) Module

3.1 Introduction

This section describes the 10-bit successive approximation analog-to-digital converter (ADC10).

The ADC10 module shares its pins with general-purpose input/output (I/O) port pins. See Figure 3-1 for port location of these shared pins. The ADC10 on this MCU uses V_{DD} and V_{SS} as its supply and reference pins. This MCU uses BUSCLKX4 as its alternate clock source for the ADC. This MCU does not have a hardware conversion trigger.

3.2 Features

Features of the ADC10 module include:

- Linear successive approximation algorithm with 10-bit resolution
- Output formatted in 10- or 8-bit right-justified format
- Single or continuous conversion (automatic power-down in single conversion mode)
- Configurable sample time and conversion speed (to save power)
- Conversion complete flag and interrupt
- Input clock selectable from up to three sources
- Operation in wait and stop modes for lower noise operation
- Selectable asynchronous hardware conversion trigger

3.3 Functional Description

The ADC10 uses successive approximation to convert the input sample taken from ADVIN to a digital representation. The approximation is taken and then rounded to the nearest 10- or 8-bit value to provide greater accuracy and to provide a more robust mechanism for achieving the ideal code-transition voltage.

Figure 3-2 shows a block diagram of the ADC10

For proper conversion, the voltage on ADVIN must fall between V_{REFH} and V_{REFL}. If ADVIN is equal to or exceeds V_{REFH}, the converter circuit converts the signal to $3FF for a 10-bit representation or $FF for a 8-bit representation. If ADVIN is equal to or less than V_{REFL}, the converter circuit converts it to $000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions.

NOTE

Input voltage must not exceed the analog supply voltages.

ADIV[1:0] — ADC10 Clock Divider Bits

ADIV1 and ADIV0 select the divide ratio used by the ADC10 to generate the internal clock ADCK. Table 3-3 shows the available clock configurations.

Table 3-3. ADC10 Clock Divide Ratio

ADIV1	ADIV0	Divide Ratio (ADIV)	Clock Rate
0	0	1	Input clock ÷ 1
0	1	2	Input clock ÷ 2
1	0	4	Input clock ÷ 4
1	1	8	Input clock ÷ 8

ADICLK — Input Clock Select Bit

If ACLKEN is clear, ADICLK selects either the bus clock or an alternate clock source as the input clock source to generate the internal clock ADCK. If the alternate clock source is less than the minimum clock speed, use the internally-generated bus clock as the clock source. As long as the internal clock ADCK, which is equal to the selected input clock divided by ADIV, is at a frequency (f_{ADCK}) between the minimum and maximum clock speeds (considering ALPC), correct operation can be guaranteed.

1 = The internal bus clock is selected as the input clock source
0 = The alternate clock source IS SELECTED

MODE[1:0] — 10- or 8-Bit or Hardware Triggered Mode Selection

These bits select 10- or 8-bit operation. The successive approximation converter generates a result that is rounded to 8- or 10-bit value based on the mode selection. This rounding process sets the transfer function to transition at the midpoint between the ideal code voltages, causing a quantization error of ± 1/2LSB.

Reset returns 8-bit mode.

00 = 8-bit, right-justified, ADSCR software triggered mode enabled
01 = 10-bit, right-justified, ADSCR software triggered mode enabled
10 = Reserved
11 = 10-bit, right-justified, hardware triggered mode enabled

ADLSMP — Long Sample Time Configuration

This bit configures the sample time of the ADC10 to either 3.5 or 23.5 ADCK clock cycles. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption in continuous conversion mode if high conversion rates are not required.

1 = Long sample time (23.5 cycles)
0 = Short sample time (3.5 cycles)

ACLKEN — Asynchronous Clock Source Enable

This bit enables the asynchronous clock source as the input clock to generate the internal clock ADCK, and allows operation in stop mode. The asynchronous clock source will operate between 1 MHz and 2 MHz if ADLPC is clear, and between 0.5 MHz and 1 MHz if ADLPC is set.

1 = The asynchronous clock is selected as the input clock source (the clock generator is only enabled during the conversion)
0 = ADICLK specifies the input clock source and conversions will not continue in stop mode

LVIPWRD — LVI Power Disable Bit

LVIPWRD disables the LVI module.

1 = LVI module power disabled
0 = LVI module power enabled

LVITRIP — LVI Trip Point Selection Bit

LVITRIP selects the voltage operating mode of the LVI module. The voltage mode selected for the LVI should match the operating V_{DD} for the LVI's voltage trip points for each of the modes.

1 = LVI operates for a 5-V protection
0 = LVI operates for a 3-V protection

NOTE

The LVITRIP bit is cleared by a power-on reset (POR) only. Other resets will leave this bit unaffected.

SSREC — Short Stop Recovery Bit

SSREC enables the CPU to exit stop mode with a delay of 32 BUSCLKX4 cycles instead of a 4096 BUSCLKX4 cycle delay.

1 = Stop mode recovery after 32 BUSCLKX4 cycles
0 = Stop mode recovery after 4096 BUSCLKX4 cycles

NOTE

Exiting stop mode by an LVI reset will result in the long stop recovery.

When using the LVI during normal operation but disabling during stop mode, the LVI will have an enable time of t_{EN}. The system stabilization time for power-on reset and long stop recovery (both 4096 BUSCLKX4 cycles) gives a delay longer than the LVI enable time for these startup scenarios. There is no period where the MCU is not protected from a low-power condition. However, when using the short stop recovery configuration option, the 32 BUSCLKX4 delay must be greater than the LVI's turn on time to avoid a period in startup where the LVI is not protecting the MCU.

STOP — STOP Instruction Enable Bit

STOP enables the STOP instruction.

1 = STOP instruction enabled
0 = STOP instruction treated as illegal opcode

COPD — COP Disable Bit

COPD disables the COP module.

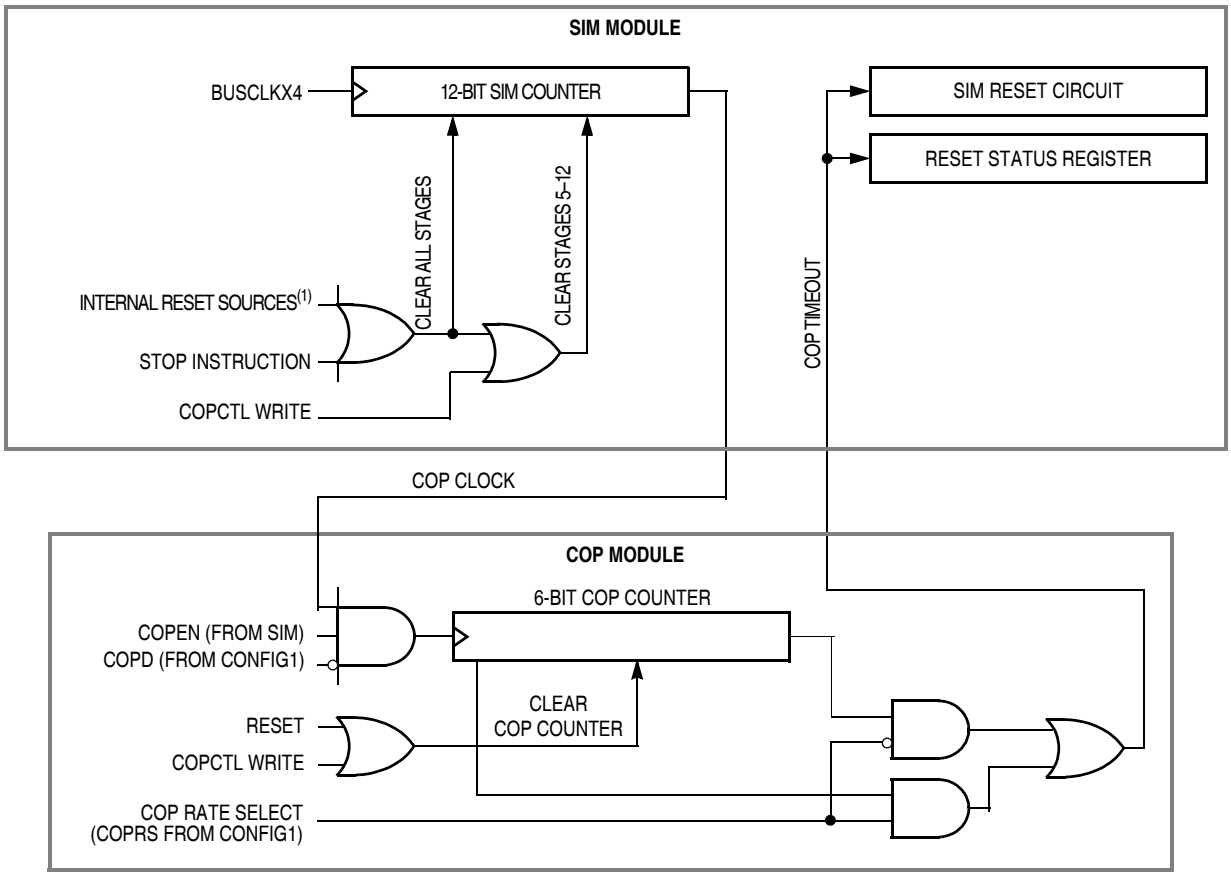
1 = COP module disabled
0 = COP module enabled

Chapter 6
Computer Operating Properly (COP)

6.1 Introduction

The computer operating properly (COP) module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by clearing the COP counter periodically. The COP module can be disabled through the COPD bit in the configuration 1 (CONFIG1) register.

6.2 Functional Description



1. See Chapter 13 System Integration Module (SIM) for more details.

Figure 6-1. COP Block Diagram

Table 7-1. Instruction Set Summary (Sheet 2 of 6)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BHS *rel*	Branch if Higher or Same (Same as BCC)	PC ← (PC) + 2 + *rel* ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH *rel*	Branch if $\overline{\text{IRQ}}$ Pin High	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL *rel*	Branch if $\overline{\text{IRQ}}$ Pin Low	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT #*opr* BIT *opr* BIT *opr* BIT *opr*,X BIT *opr*,X BIT ,X BIT *opr*,SP BIT *opr*,SP	Bit Test	(A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A5 B5 C5 D5 E5 F5 9EE5 9ED5	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
BLE *opr*	Branch if Less Than or Equal To (Signed Operands)	PC ← (PC) + 2 + *rel* ? (Z) ∣ (N ⊕ V) = 1	–	–	–	–	–	–	REL	93	rr	3
BLO *rel*	Branch if Lower (Same as BCS)	PC ← (PC) + 2 + *rel* ? (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BLS *rel*	Branch if Lower or Same	PC ← (PC) + 2 + *rel* ? (C) ∣ (Z) = 1	–	–	–	–	–	–	REL	23	rr	3
BLT *opr*	Branch if Less Than (Signed Operands)	PC ← (PC) + 2 + *rel* ? (N ⊕ V) =1	–	–	–	–	–	–	REL	91	rr	3
BMC *rel*	Branch if Interrupt Mask Clear	PC ← (PC) + 2 + *rel* ? (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI *rel*	Branch if Minus	PC ← (PC) + 2 + *rel* ? (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS *rel*	Branch if Interrupt Mask Set	PC ← (PC) + 2 + *rel* ? (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE *rel*	Branch if Not Equal	PC ← (PC) + 2 + *rel* ? (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL *rel*	Branch if Plus	PC ← (PC) + 2 + *rel* ? (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA *rel*	Branch Always	PC ← (PC) + 2 + *rel*	–	–	–	–	–	–	REL	20	rr	3
BRCLR *n*,*opr*,*rel*	Branch if Bit *n* in M Clear	PC ← (PC) + 3 + *rel* ? (Mn) = 0	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN *rel*	Branch Never	PC ← (PC) + 2	–	–	–	–	–	–	REL	21	rr	3
BRSET *n*,*opr*,*rel*	Branch if Bit *n* in M Set	PC ← (PC) + 3 + *rel* ? (Mn) = 1	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET *n*,*opr*	Set Bit *n* in M	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BSR *rel*	Branch to Subroutine	PC ← (PC) + 2; push (PCL) SP ← (SP) – 1; push (PCH) SP ← (SP) – 1 PC ← (PC) + *rel*	–	–	–	–	–	–	REL	AD	rr	4
CBEQ *opr*,*rel* CBEQA #*opr*,*rel* CBEQX #*opr*,*rel* CBEQ *opr*,X+,*rel* CBEQ X+,*rel* CBEQ *opr*,SP,*rel*	Compare and Branch if Equal	PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (X) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 2 + rel ? (A) – (M) = $00 PC ← (PC) + 4 + rel ? (A) – (M) = $00	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 4 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask	I ← 0	–	–	0	–	–	–	INH	9A		2

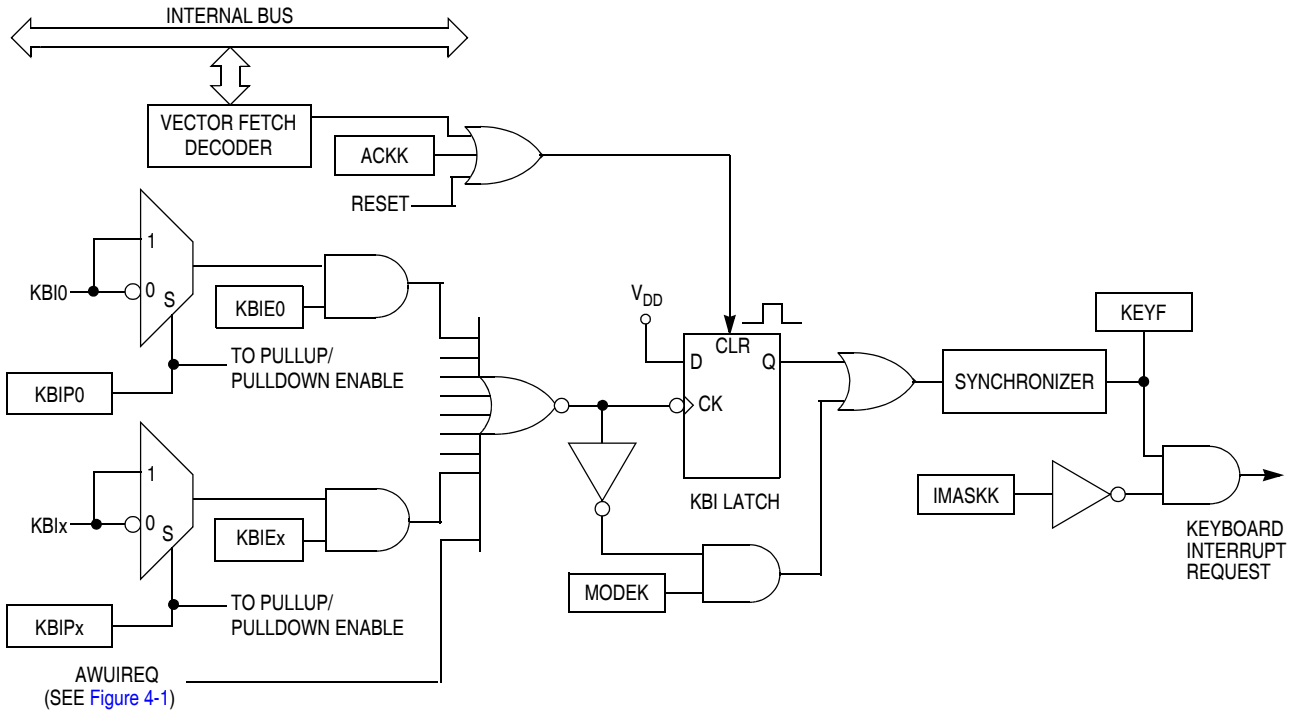


Figure 9-2. Keyboard Interrupt Block Diagram

The KBI vector fetch or software clear and the return of all enabled keyboard interrupt pins to a deasserted level may occur in any order.

Reset clears the keyboard interrupt request and the MODEK bit, clearing the interrupt request even if a keyboard interrupt input stays asserted.

9.3.1.2 MODEK = 0

If the MODEK bit is clear, the keyboard interrupt inputs are edge sensitive. The KBIPx bit will determine whether an edge sensitive pin detects rising or falling edges. A KBI vector fetch or software clear immediately clears the KBI latch.

The keyboard flag bit (KEYF) in KBSCR can be read to check for pending interrupts. The KEYF bit is not affected by IMASKK, which makes it useful in applications where polling is preferred.

NOTE

Setting a keyboard interrupt enable bit (KBIEx) forces the corresponding keyboard interrupt pin to be an input, overriding the data direction register. However, the data direction register bit must be a 0 for software to read the pin.

The LVI module contains a bandgap reference circuit and comparator. When the LVITRIP bit is cleared, the default state at power-on reset, V_{TRIPF} is configured for the lower V_{DD} operating range. The actual trip points are specified in 16.5 5-V DC Electrical Characteristics and 16.8 3-V DC Electrical Characteristics.

Because the default LVI trip point after power-on reset is configured for low voltage operation, a system requiring high voltage LVI operation must set the LVITRIP bit during system initialization. V_{DD} must be above the LVI trip rising voltage, V_{TRIPR}, for the high voltage operating range or the MCU will immediately go into LVI reset.

After an LVI reset occurs, the MCU remains in reset until V_{DD} rises above V_{TRIPR}. See Chapter 13 System Integration Module (SIM) for the reset recovery sequence.

The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR) and can be used for polling LVI operation when the LVI reset is disabled.

The LVI is enabled out of reset. The following bits located in the configuration register can alter the default conditions.

- Setting the LVI power disable bit, LVIPWRD, disables the LVI.
- Setting the LVI reset disable bit, LVIRSTD, prevents the LVI module from generating a reset.
- Setting the LVI enable in stop mode bit, LVISTOP, enables the LVI to operate in stop mode.
- Setting the LVI trip point bit, LVITRIP, configures the trip point voltage (V_{TRIPF}) for the higher V_{DD} operating range.

10.3.1 Polled LVI Operation

In applications that can operate at V_{DD} levels below the V_{TRIPF} level, software can monitor V_{DD} by polling the LVIOUT bit. In the configuration register, LVIPWRD must be cleared to enable the LVI module, and LVIRSTD must be set to disable LVI resets.

10.3.2 Forced Reset Operation

In applications that require V_{DD} to remain above the V_{TRIPF} level, enabling LVI resets allows the LVI module to reset the MCU when V_{DD} falls below the V_{TRIPF} level. In the configuration register, LVIPWRD and LVIRSTD must be cleared to enable the LVI module and to enable LVI resets.

10.3.3 LVI Hysteresis

The LVI has hysteresis to maintain a stable operating condition. After the LVI has triggered (by having V_{DD} fall below V_{TRIPF}), the MCU will remain in reset until V_{DD} rises above the rising trip point voltage, V_{TRIPR}. This prevents a condition in which the MCU is continually entering and exiting reset if V_{DD} is approximately equal to V_{TRIPF}. V_{TRIPR} is greater than V_{TRIPF} by the typical hysteresis voltage, V_{HYS}.

10.3.4 LVI Trip Selection

LVITRIP in the configuration register selects the LVI protection range. The default setting out of reset is for the low voltage range. Because LVITRIP is in a write-once configuration register, the protection range cannot be changed after initialization.

NOTE

The MCU is guaranteed to operate at a minimum supply voltage. The trip point (V_{TRIPF}) may be lower than this. See the Electrical Characteristics section for the actual trip point voltages.

11.3.1.2 XTAL Oscillator Clock (XTALCLK)

XTALCLK is the XTAL oscillator output signal. It runs at the full speed of the crystal (f_{XCLK}) and comes directly from the crystal oscillator circuit. Figure 11-2 shows only the logical relation of XTALCLK to OSC1 and OSC2 and may not represent the actual circuitry. The duty cycle of XTALCLK is unknown and may depend on the crystal and other external factors. The frequency of XTALCLK can be unstable at start up.

11.3.1.3 RC Oscillator Clock (RCCLK)

RCCLK is the RC oscillator output signal. Its frequency is directly proportional to the time constant of the external R (R_{EXT}) and internal C. Figure 11-3 shows only the logical relation of RCCLK to OSC1 and may not represent the actual circuitry.

11.3.1.4 Internal Oscillator Clock (INTCLK)

INTCLK is the internal oscillator output signal. INTCLK is software selectable to be nominally 12.8 MHz, 8.0 MHz, or 4.0 MHz. INTCLK can be digitally adjusted using the oscillator trimming feature of the OSCTRIM register (see 11.3.2.1 Internal Oscillator Trimming).

11.3.1.5 Bus Clock Times 4 (BUSCLKX4)

BUSCLKX4 is the same frequency as the input clock (XTALCLK, RCCLK, or INTCLK). This signal is driven to the SIM module and is used during recovery from reset and stop and is the clock source for the COP module.

11.3.1.6 Bus Clock Times 2 (BUSCLKX2)

The frequency of this signal is equal to half of the BUSCLKX4. This signal is driven to the SIM for generation of the bus clocks used by the CPU and other modules on the MCU. BUSCLKX2 will be divided by two in the SIM. The internal bus frequency is one fourth of the XTALCLK, RCCLK, or INTCLK frequency.

11.3.2 Internal Oscillator

The internal oscillator circuit is designed for use with no external components to provide a clock source with a tolerance of less than ±25% untrimmed. An 8-bit register (OSCTRIM) allows the digital adjustment to a tolerance of ACC_{INT}. See the oscillator characteristics in the Electrical section of this data sheet.

The internal oscillator is capable of generating clocks of 12.8 MHz, 8.0 MHz, or 4.0 MHz (INTCLK) resulting in a bus frequency (INTCLK divided by 4) of 3.2 MHz, 2.0 MHz, or 1.0 MHz respectively. The bus clock is software selectable and defaults to the 3.2-MHz bus out of reset. Users can increase the bus frequency based on the voltage range of their application.

Figure 11-3 shows how BUSCLKX4 is derived from INTCLK and OSC2 can output BUSCLKX4 by setting OSC2EN.

11.3.2.1 Internal Oscillator Trimming

OSCTRIM allows a clock period adjustment of +127 and –128 steps. Increasing the OSCTRIM value increases the clock period, which decreases the clock frequency. Trimming allows the internal clock frequency to be fine tuned to the target frequency.

All devices are factory programmed with trim values that are stored in FLASH memory at locations $FFC0 and $FFC1. The trim value is **not** automatically loaded into the OSCTRIM register. User software must

11.8 Registers

The oscillator module contains two registers:

- Oscillator status and control register (OSCSC)
- Oscillator trim register (OSCTRIM)

11.8.1 Oscillator Status and Control Register

The oscillator status and control register (OSCSC) contains the bits for switching between internal and external clock sources. If the application uses an external crystal, bits in this register are used to select the crystal oscillator amplifier necessary for the desired crystal. While running off the internal clock source, the user can use bits in this register to select the internal clock source frequency.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	OSCOPT1	OSCOPT0	ICFS1	ICFS0	ECFS1	ECFS0	ECGON	ECGST
Write:								
Reset:	0	0	1	0	0	0	0	0

= Unimplemented

Figure 11-4. Oscillator Status and Control Register (OSCSC)

OSCOPT1:OSCOPT0 — OSC Option Bits

These read/write bits allow the user to change the clock source for the MCU. The default reset condition has the bus clock being derived from the internal oscillator. See 11.3.2.2 Internal to External Clock Switching for information on changing clock sources.

OSCOPT1	OSCOPT0	Oscillator Modes
0	0	Internal oscillator (frequency selected using ICFSx bits)
0	1	External oscillator clock
1	0	External RC
1	1	External crystal (range selected using ECFSx bits)

ICFS1:ICFS0 — Internal Clock Frequency Select Bits

These read/write bits enable the frequency to be increased for applications requiring a faster bus clock when running off the internal oscillator. The WAIT instruction has no effect on the oscillator logic. BUSCLKX2 and BUSCLKX4 continue to drive to the SIM module.

ICFS1	ICFS0	Internal Clock Frequency
0	0	4.0 MHz
0	1	8.0 MHz
1	0	12.8 MHz — default reset condition
1	1	Reserved

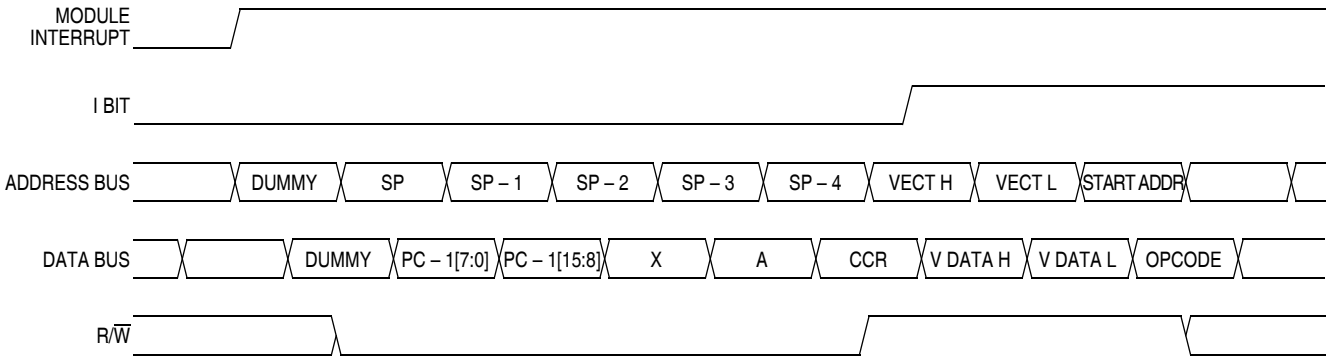


Figure 13-8. Interrupt Entry

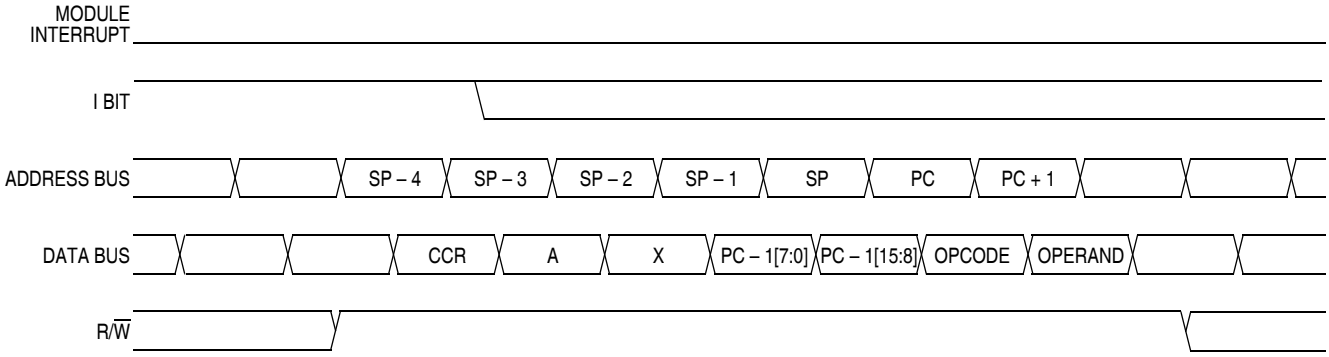


Figure 13-9. Interrupt Recovery

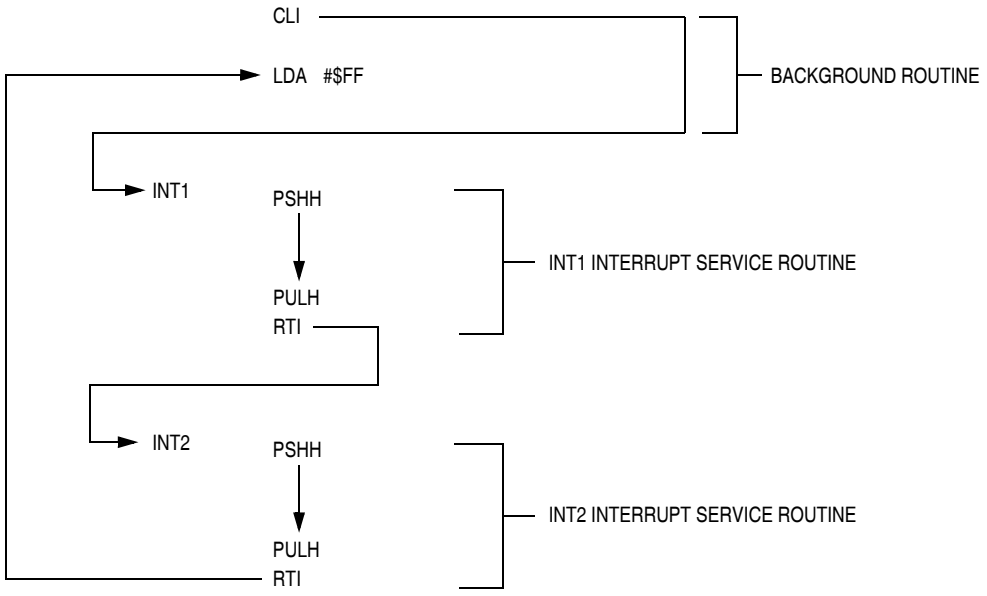


Figure 13-10. Interrupt Recognition Example

13.6.3 Reset

All reset sources always have equal and highest priority and cannot be arbitrated.

13.6.4 Break Interrupts

The break module can stop normal program flow at a software programmable break point by asserting its break interrupt output. (See Chapter 15 Development Support.) The SIM puts the CPU into the break state by forcing it to the SWI vector location. Refer to the break interrupt subsection of each module to see how each module is affected by the break state.

13.6.5 Status Flag Protection in Break Mode

The SIM controls whether status flags contained in other modules can be cleared during break mode. The user can select whether flags are protected from being cleared by properly initializing the break clear flag enable bit (BCFE) in the break flag control register (BFCR).

Protecting flags in break mode ensures that set flags will not be cleared while in break mode. This protection allows registers to be freely read and written during break mode without losing status flag information.

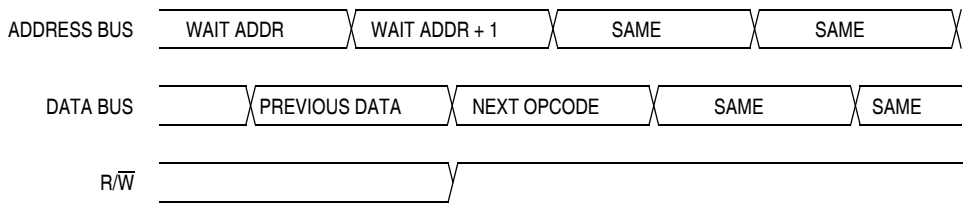
Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a two-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first step is accomplished prior to entering break mode. Upon leaving break mode, execution of the second step will clear the flag as normal.

13.7 Low-Power Modes

Executing the WAIT or STOP instruction puts the MCU in a low power- consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur.

13.7.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. Figure 13-14 shows the timing for wait mode entry.



NOTE: Previous data can be operand data or the WAIT opcode, depending on the last instruction.

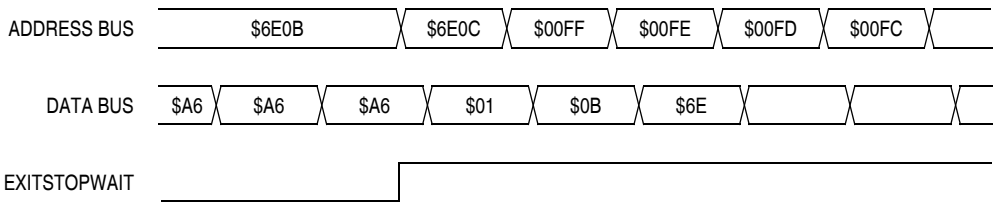
Figure 13-14. Wait Mode Entry Timing

A module that is active during wait mode can wake up the CPU with an interrupt if the interrupt is enabled. Stacking for the interrupt begins one cycle after the WAIT instruction during which the interrupt occurred.

In wait mode, the CPU clocks are inactive. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

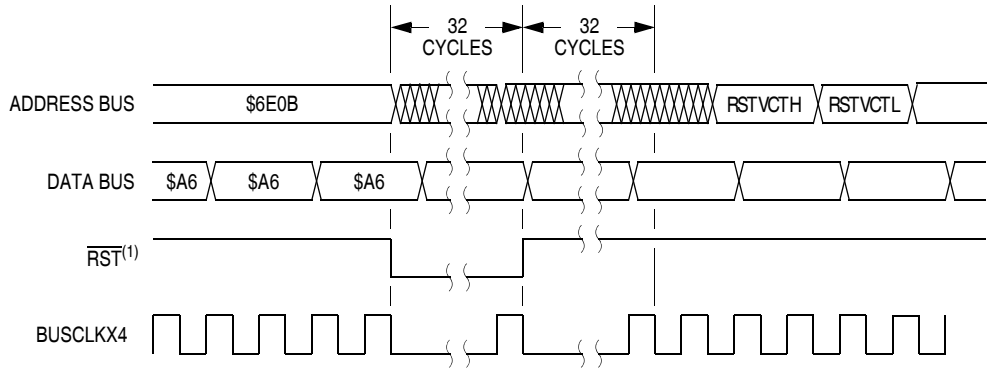
Wait mode can also be exited by a reset (or break in emulation mode). A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the break status register (BSR). If the COP disable bit, COPD, in the configuration register is 0, then the computer operating properly module (COP) is enabled and remains active in wait mode.

Figure 13-15 and Figure 13-16 show the timing for wait recovery.



NOTE: EXITSTOPWAIT = $\overline{\text{RST}}$ pin OR CPU interrupt

Figure 13-15. Wait Recovery from Interrupt



1. $\overline{\text{RST}}$ is only available if the RSTEN bit in the CONFIG1 register is set.

Figure 13-16. Wait Recovery from Internal Reset

13.7.2 Stop Mode

In stop mode, the SIM counter is reset and the system clocks are disabled. An interrupt request from a module can cause an exit from stop mode. Stacking for interrupts begins after the selected stop recovery time has elapsed. Reset or break also causes an exit from stop mode.

The SIM disables the oscillator signals (BUSCLKX2 and BUSCLKX4) in stop mode, stopping the CPU and peripherals. If OSCENINSTOP is set, BUSCLKX2 will remain running in STOP and can be used to run the AWU. Stop recovery time is selectable using the SSREC bit in the configuration register 1 (CONFIG1). If SSREC is set, stop recovery is reduced from the normal delay of 4096 BUSCLKX4 cycles down to 32. This is ideal for the internal oscillator, RC oscillator, and external oscillator options which do not require long start-up times from stop mode.

NOTE

External crystal applications should use the full stop recovery time by clearing the SSREC bit.

14.8.2 TIM Counter Registers

The two read-only TIM counter registers contain the high and low bytes of the value in the counter. Reading the high byte (TCNTH) latches the contents of the low byte (TCNTL) into a buffer. Subsequent reads of TCNTH do not affect the latched TCNTL value until TCNTL is read. Reset clears the TIM counter registers. Setting the TIM reset bit (TRST) also clears the TIM counter registers.

NOTE

If you read TCNTH during a break interrupt, be sure to unlatch TCNTL by reading TCNTL before exiting the break interrupt. Otherwise, TCNTL retains the value latched during the break.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-5. TIM Counter High Register (TCNTH)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-6. TIM Counter Low Register (TCNTL)

14.8.3 TIM Counter Modulo Registers

The read/write TIM modulo registers contain the modulo value for the counter. When the counter reaches the modulo value, the overflow flag (TOF) becomes set, and the counter resumes counting from $0000 at the next timer clock. Writing to the high byte (TMODH) inhibits the TOF bit and overflow interrupts until the low byte (TMODL) is written. Reset sets the TIM counter modulo registers.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Reset:	1	1	1	1	1	1	1	1

Figure 14-7. TIM Counter Modulo High Register (TMODH)

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reset:	1	1	1	1	1	1	1	1

Figure 14-8. TIM Counter Modulo Low Register (TMODL)

NOTE

Reset the counter before writing to the TIM counter modulo registers.

Case 751G page 1 of 2