



Welcome to [E-XFL.COM](https://www.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	M68HC08
Core Size	8-Bit
Speed	8MHz
Connectivity	SCI, SPI
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 10x10b SAR
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/s908qb4e0cdte

List of Chapters

Chapter 1	General Description.	17
Chapter 2	Memory.	23
Chapter 3	Analog-to-Digital Converter (ADC10) Module.	37
Chapter 4	Auto Wakeup Module (AWU)	51
Chapter 5	Configuration Register (CONFIG)	57
Chapter 6	Computer Operating Properly (COP).	61
Chapter 7	Central Processor Unit (CPU).	65
Chapter 8	External Interrupt (IRQ).	77
Chapter 9	Keyboard Interrupt Module (KBI).	83
Chapter 10	Low-Voltage Inhibit (LVI).	89
Chapter 11	Oscillator Module (OSC)	93
Chapter 12	Input/Output Ports (PORTS)	103
Chapter 13	Enhanced Serial Communications Interface (ESCI) Module	109
Chapter 14	System Integration Module (SIM).	139

List of Chapters

Chapter 15	Serial Peripheral Interface (SPI) Module	155
Chapter 16	Timer Interface Module (TIM)	175
Chapter 17	Development Support	191
Chapter 18	Electrical Specifications	207
Chapter 19	Ordering Information and Mechanical Specifications	227

6.1	Introduction	61
6.2	Functional Description	61
6.3	I/O Signals	62
6.3.1	BUSCLKX4	62
6.3.2	STOP Instruction	62
6.3.3	COPCTL Write	62
6.3.4	Power-On Reset	62
6.3.5	Internal Reset	62
6.3.6	COPD (COP Disable)	62
6.3.7	COPRS (COP Rate Select)	63
6.4	Interrupts	63
6.5	Monitor Mode	63
6.6	Low-Power Modes	63
6.6.1	Wait Mode	63
6.6.2	Stop Mode	63
6.7	COP Module During Break Mode	63
6.8	Register	63

Chapter 7 Central Processor Unit (CPU)

7.1	Introduction	65
7.2	Features	65
7.3	CPU Registers	65
7.3.1	Accumulator	66
7.3.2	Index Register	66
7.3.3	Stack Pointer	67
7.3.4	Program Counter	67
7.3.5	Condition Code Register	68
7.4	Arithmetic/Logic Unit (ALU)	69
7.5	Low-Power Modes	69
7.5.1	Wait Mode	69
7.5.2	Stop Mode	69
7.6	CPU During Break Interrupts	69
7.7	Instruction Set Summary	70
7.8	Opcode Map	75

Chapter 8 External Interrupt (IRQ)

8.1	Introduction	77
8.2	Features	77
8.3	Functional Description	77
8.3.1	MODE = 1	79
8.3.2	MODE = 0	79
8.4	Interrupts	80

2.6.3 FLASH Mass Erase Operation

Use the following procedure to erase the entire FLASH memory to read as a 1:

1. Set both the ERASE bit and the MASS bit in the FLASH control register.
2. Read the FLASH block protect register.
3. Write any data to any FLASH address⁽¹⁾ within the FLASH memory address range.
4. Wait for a time, t_{NVS} .
5. Set the HVEN bit.
6. Wait for a time, t_{MErase} .
7. Clear the ERASE and MASS bits.

NOTE

Mass erase is disabled whenever any block is protected (FLBPR does not equal \$FF).

8. Wait for a time, t_{NVHL} .
9. Clear the HVEN bit.
10. After time, t_{RCV} , the memory can be accessed in read mode again.

NOTE

Programming and erasing of FLASH locations cannot be performed by code being executed from the FLASH memory. While these operations must be performed in the order as shown, other unrelated operations may occur between the steps.

CAUTION

A mass erase will erase the internal oscillator trim value at \$FFC0.

2.6.4 FLASH Program Operation

Programming of the FLASH memory is done on a row basis. A row consists of 32 consecutive bytes starting from addresses \$XX00, \$XX20, \$XX40, \$XX60, \$XX80, \$XXA0, \$XXC0, or \$XXE0. Use the following step-by-step procedure to program a row of FLASH memory

Figure 2-4 shows a flowchart of the programming algorithm.

NOTE

Do not program any byte in the FLASH more than once after a successful erase operation. Reprogramming bits to a byte which is already programmed is not allowed without first erasing the page in which the byte resides or mass erasing the entire FLASH memory. Programming without first erasing may disturb data stored in the FLASH.

1. Set the PGM bit. This configures the memory for program operation and enables the latching of address and data for programming.
2. Read the FLASH block protect register.
3. Write any data to any FLASH location within the address range desired.
4. Wait for a time, t_{NVS} .
5. Set the HVEN bit.

1. When in monitor mode, with security sequence failed (see 17.3.2 Security), write to the FLASH block protect register instead of any FLASH address.

8.8 Registers

The IRQ status and control register (INTSCR) controls and monitors operation of the IRQ module. The INTSCR:

- Shows the state of the IRQ flag
- Clears the IRQ latch
- Masks the IRQ interrupt request
- Controls triggering sensitivity of the $\overline{\text{IRQ}}$ interrupt pin

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	IRQF	0	IMASK	MODE
Write:						ACK		
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 8-3. IRQ Status and Control Register (INTSCR)

IRQF — IRQ Flag Bit

This read-only status bit is set when the IRQ interrupt is pending.

1 = $\overline{\text{IRQ}}$ interrupt pending

0 = $\overline{\text{IRQ}}$ interrupt not pending

ACK — IRQ Interrupt Request Acknowledge Bit

Writing a 1 to this write-only bit clears the IRQ latch. ACK always reads 0.

IMASK — IRQ Interrupt Mask Bit

Writing a 1 to this read/write bit disables the IRQ interrupt request.

1 = IRQ interrupt request disabled

0 = IRQ interrupt request enabled

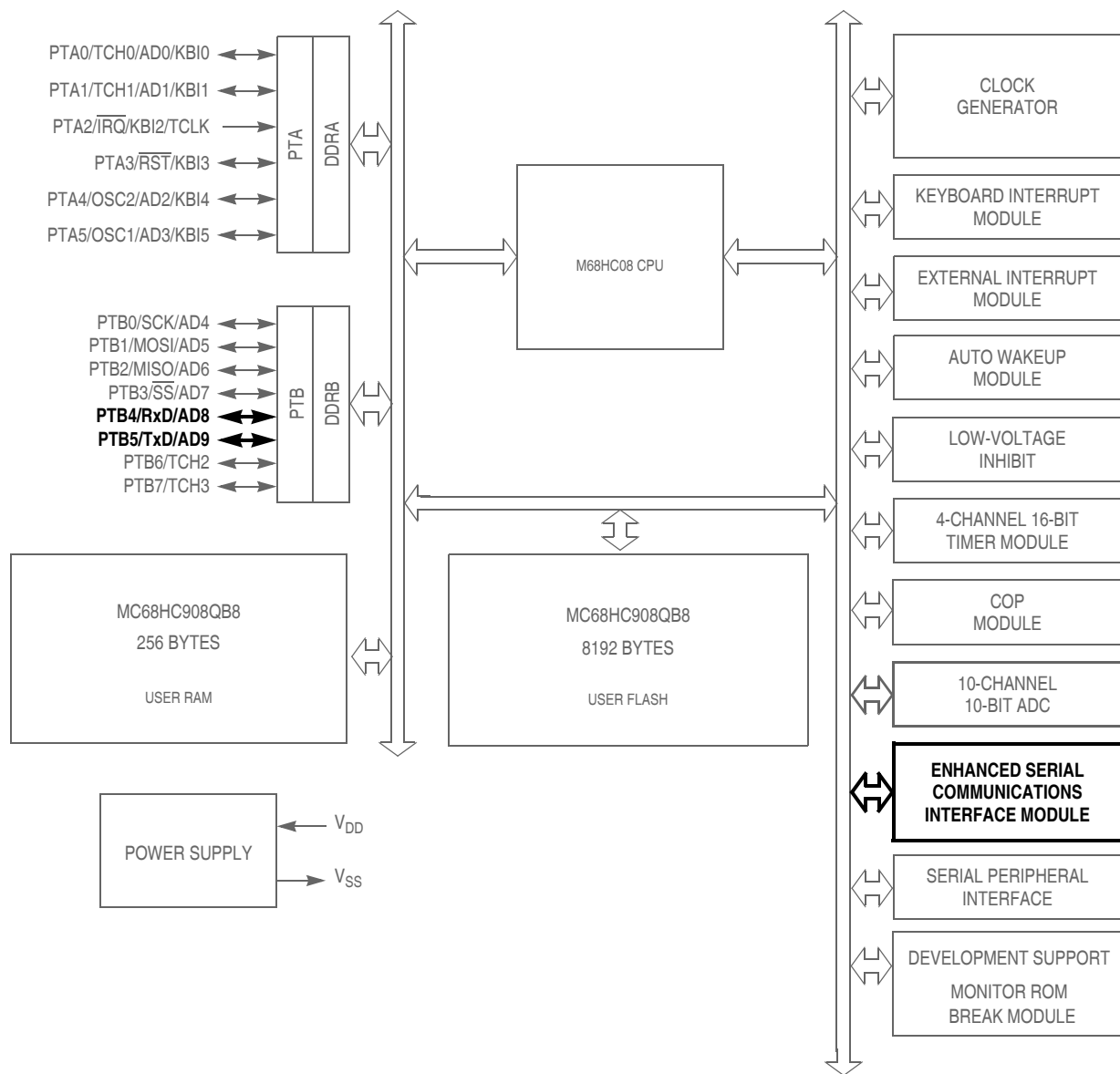
MODE — IRQ Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ}}$ pin.

1 = $\overline{\text{IRQ}}$ interrupt request on falling edges and low levels

0 = $\overline{\text{IRQ}}$ interrupt request on falling edges only

Enhanced Serial Communications Interface (ESCI) Module



$\overline{\text{RST}}$, $\overline{\text{IRQ}}$: Pins have internal pull up device
 All port pins have programmable pull up device
 PTA[0:5]: Higher current sink and source capability

Figure 13-1. Block Diagram Highlighting ESCI Block and Pins

13.8.1 ESCI Control Register 1

ESCI control register 1 (SCC1):

- Enables loop mode operation
- Enables the ESCI
- Controls output polarity
- Controls character length
- Controls ESCI wakeup method
- Controls idle character detection
- Enables parity function
- Controls parity type

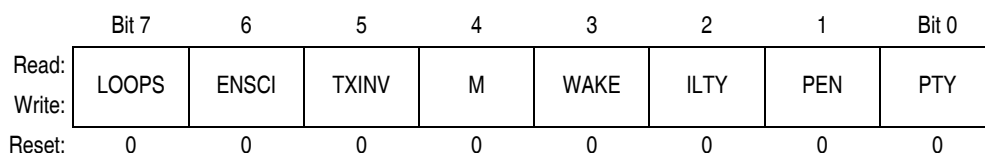


Figure 13-9. ESCI Control Register 1 (SCC1)

LOOPS — Loop Mode Select Bit

This read/write bit enables loop mode operation. In loop mode the RxD pin is disconnected from the ESCI, and the transmitter output goes into the receiver input. Both the transmitter and the receiver must be enabled to use loop mode.

- 1 = Loop mode enabled
- 0 = Normal operation enabled

ENSCI — Enable ESCI Bit

This read/write bit enables the ESCI and the ESCI baud rate generator. Clearing ENSCI sets the SCTE and TC bits in ESCI status register 1 and disables transmitter interrupts.

- 1 = ESCI enabled
- 0 = ESCI disabled

TXINV — Transmit Inversion Bit

This read/write bit reverses the polarity of transmitted data.

- 1 = Transmitter output inverted
- 0 = Transmitter output not inverted

NOTE

Setting the TXINV bit inverts all transmitted values including idle, break, start, and stop bits.

M — Mode (Character Length) Bit

This read/write bit determines whether ESCI characters are eight or nine bits long (see [Table 13-4](#)). The ninth bit can serve as a receiver wakeup signal or as a parity bit.

- 1 = 9-bit ESCI characters
- 0 = 8-bit ESCI characters

Enhanced Serial Communications Interface (ESCI) Module

- Enables the receiver
- Enables ESCI wakeup
- Transmits ESCI break characters

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 13-10. ESCI Control Register 2 (SCC2)

SCTIE — ESCI Transmit Interrupt Enable Bit

This read/write bit enables the SCTE bit to generate ESCI transmitter interrupt requests. Setting the SCTIE bit in SCC2 enables the SCTE bit to generate interrupt requests.

- 1 = SCTE enabled to generate interrupt
- 0 = SCTE not enabled to generate interrupt

TCIE — Transmission Complete Interrupt Enable Bit

This read/write bit enables the TC bit to generate ESCI transmitter interrupt requests.

- 1 = TC enabled to generate interrupt requests
- 0 = TC not enabled to generate interrupt requests

SCRIE — ESCI Receive Interrupt Enable Bit

This read/write bit enables the SCRF bit to generate ESCI receiver interrupt requests. Setting the SCRIE bit in SCC2 enables the SCRF bit to generate interrupt requests.

- 1 = SCRF enabled to generate interrupt
- 0 = SCRF not enabled to generate interrupt

ILIE — Idle Line Interrupt Enable Bit

This read/write bit enables the IDLE bit to generate ESCI receiver interrupt requests.

- 1 = IDLE enabled to generate interrupt requests
- 0 = IDLE not enabled to generate interrupt requests

TE — Transmitter Enable Bit

Setting this read/write bit begins the transmission by sending a preamble of 10 or 11 1s from the transmit shift register to the TxD pin. If software clears the TE bit, the transmitter completes any transmission in progress before the TxD returns to the idle condition (high). Clearing and then setting TE during a transmission queues an idle character to be sent after the character currently being transmitted.

- 1 = Transmitter enabled
- 0 = Transmitter disabled

NOTE

Writing to the TE bit is not allowed when the enable ESCI bit (ENSCI) is clear. ENSCI is in ESCI control register 1.

RE — Receiver Enable Bit

Setting this read/write bit enables the receiver. Clearing the RE bit disables the receiver but does not affect receiver interrupt flag bits.

- 1 = Receiver enabled
- 0 = Receiver disabled

System Integration Module (SIM)

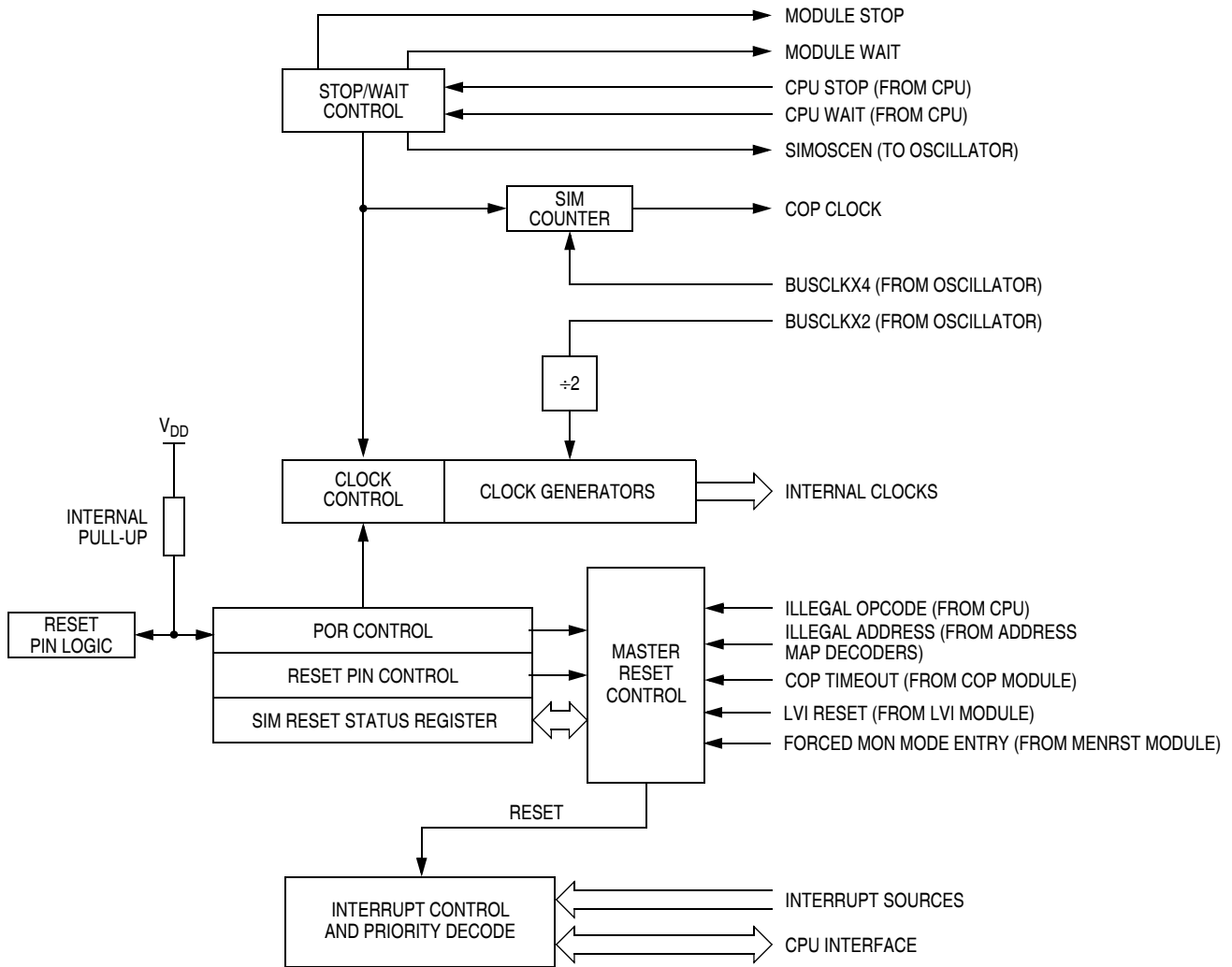


Figure 14-1. SIM Block Diagram

14.3 SIM Bus Clock Control and Generation

The bus clock generator provides system clock signals for the CPU and peripherals on the MCU. The system clocks are generated from an incoming clock, BUSCLKX2, as shown in [Figure 14-2](#).

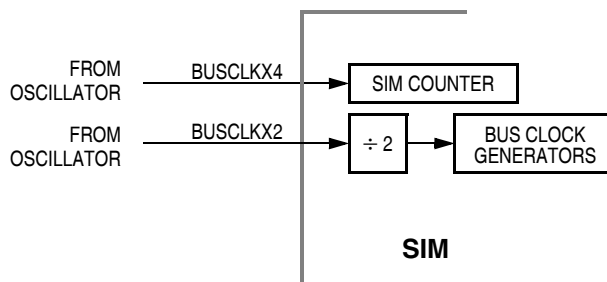


Figure 14-2. SIM Clock Signals

14.4.2 Active Resets from Internal Sources

The $\overline{\text{RST}}$ pin is initially setup as a general-purpose input after a POR. Setting the RSTEN bit in the CONFIG2 register enables the pin for the reset function. This section assumes the RSTEN bit is set when describing activity on the $\overline{\text{RST}}$ pin.

NOTE

For POR and LVI resets, the SIM cycles through 4096 BUSCLKX4 cycles. The internal reset signal then follows the sequence from the falling edge of $\overline{\text{RST}}$ shown in Figure 14-4.

The COP reset is asynchronous to the bus clock.

The active reset feature allows the part to issue a reset to peripherals and other chips within a system built around the MCU.

All internal reset sources actively pull the $\overline{\text{RST}}$ pin low for 32 BUSCLKX4 cycles to allow resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles (see Figure 14-4). An internal reset can be caused by an illegal address, illegal opcode, COP time out, LVI, or POR (see Figure 14-5).

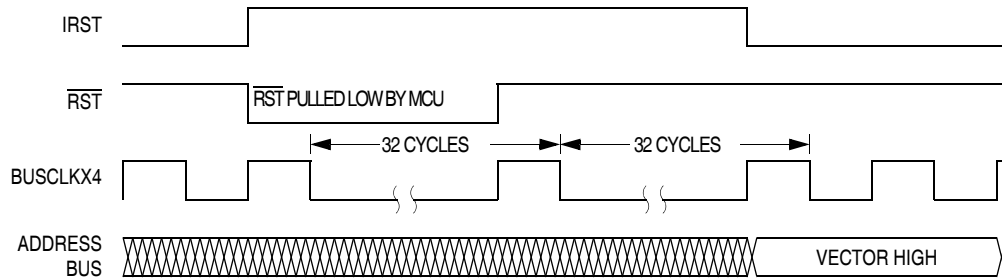


Figure 14-4. Internal Reset Timing

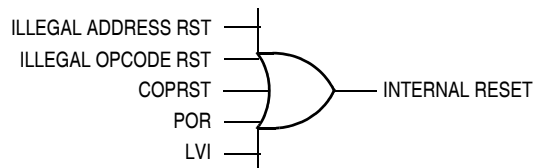


Figure 14-5. Sources of Internal Reset

Table 14-2. Reset Recovery Timing

Reset Recovery Type	Actual Number of Cycles
POR/LVI	4163 (4096 + 64 + 3)
All others	67 (64 + 3)

14.6 Exception Control

Normal sequential program execution can be changed in three different ways:

1. Interrupts
 - a. Maskable hardware CPU interrupts
 - b. Non-maskable software interrupt instruction (SWI)
2. Reset
3. Break interrupts

14.6.1 Interrupts

An interrupt temporarily changes the sequence of program execution to respond to a particular event. [Figure 14-7](#) flow charts the handling of system interrupts.

Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt can take precedence, regardless of priority, until the latched interrupt is serviced (or the I bit is cleared).

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. [Figure 14-8](#) shows interrupt entry timing. [Figure 14-9](#) shows interrupt recovery timing.

14.6.1.1 Hardware Interrupts

A hardware interrupt does not stop the current instruction. Processing of a hardware interrupt begins after completion of the current instruction. When the current instruction is complete, the SIM checks all pending hardware interrupts. If interrupts are not masked (I bit clear in the condition code register), and if the corresponding interrupt enable bit is set, the SIM proceeds with interrupt processing; otherwise, the next instruction is fetched and executed.

If more than one interrupt is pending at the end of an instruction execution, the highest priority interrupt is serviced first. [Figure 14-10](#) demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

The LDA opcode is prefetched by both the INT1 and INT2 return-from-interrupt (RTI) instructions. However, in the case of the INT1 RTI prefetch, this is a redundant operation.

NOTE

To maintain compatibility with the M6805 Family, the H register is not pushed on the stack during interrupt entry. If the interrupt service routine modifies the H register or uses the indexed addressing mode, software should save the H register and then restore it prior to exiting the routine.

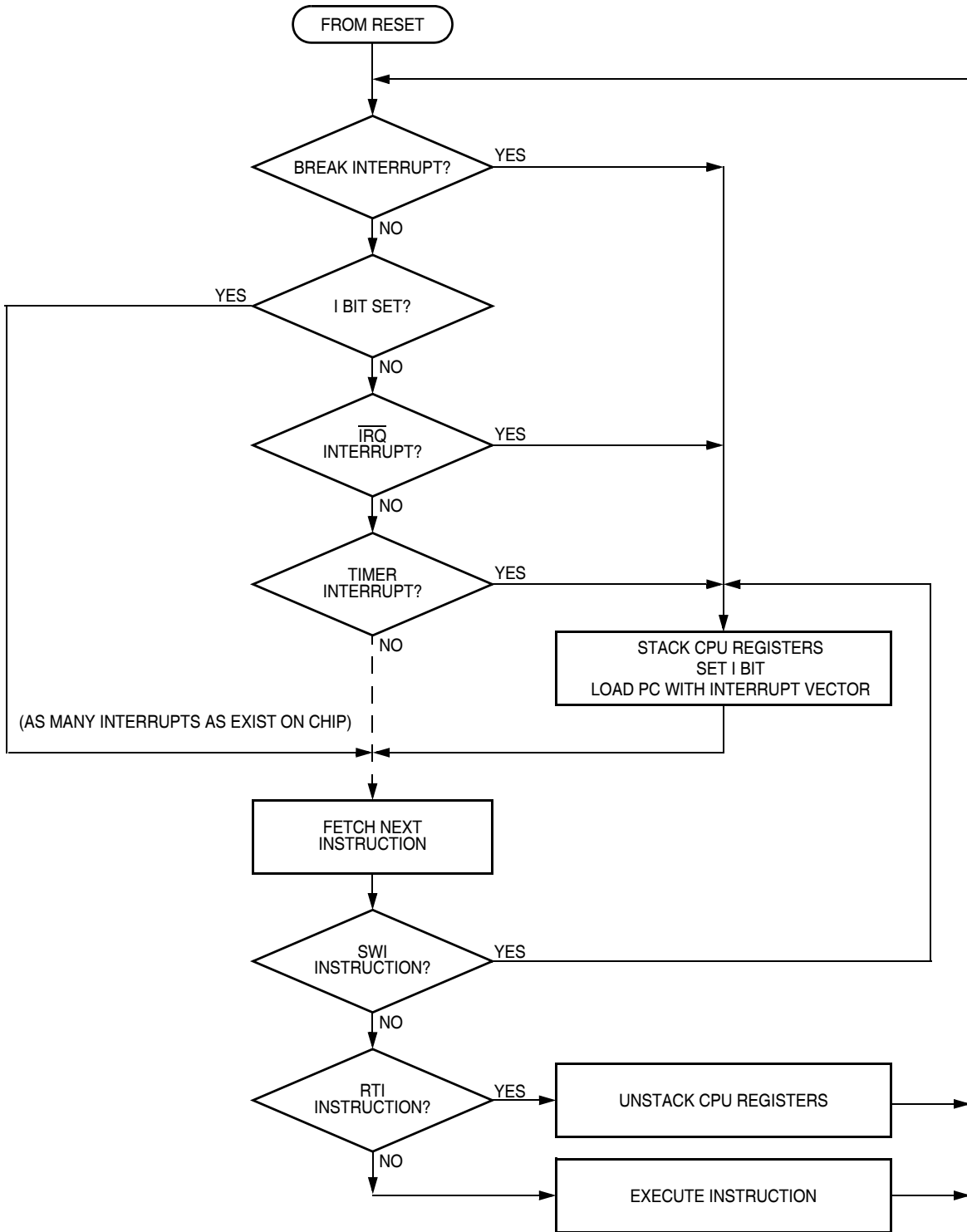


Figure 14-7. Interrupt Processing

14.6.3 Reset

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

14.6.4 Break Interrupts

The break module can stop normal program flow at a software programmable break point by asserting its break interrupt output. (See [Chapter 17 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.

14.6.5 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.

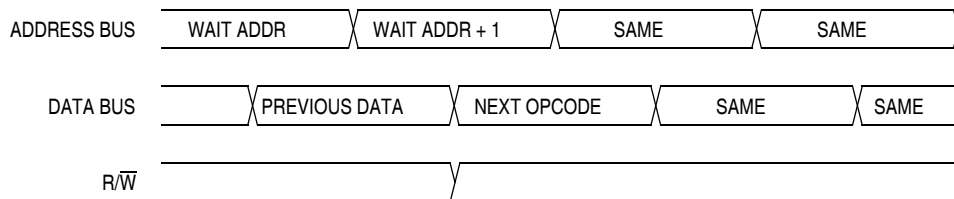
Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a two-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first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

14.7 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.

14.7.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. [Figure 14-14](#) shows the timing for wait mode entry.



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

Figure 14-14. 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 14-15 and Figure 14-16 show the timing for wait recovery.

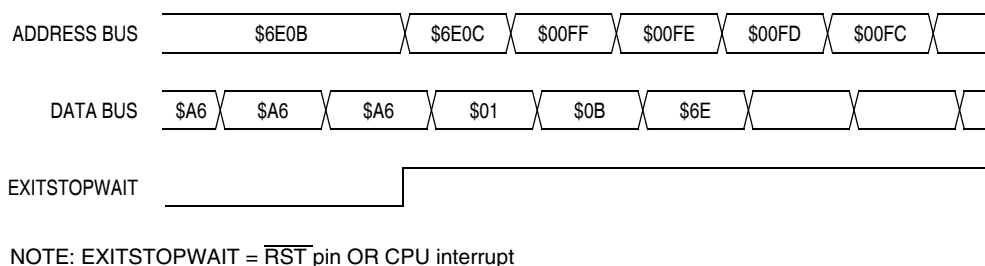


Figure 14-15. Wait Recovery from Interrupt

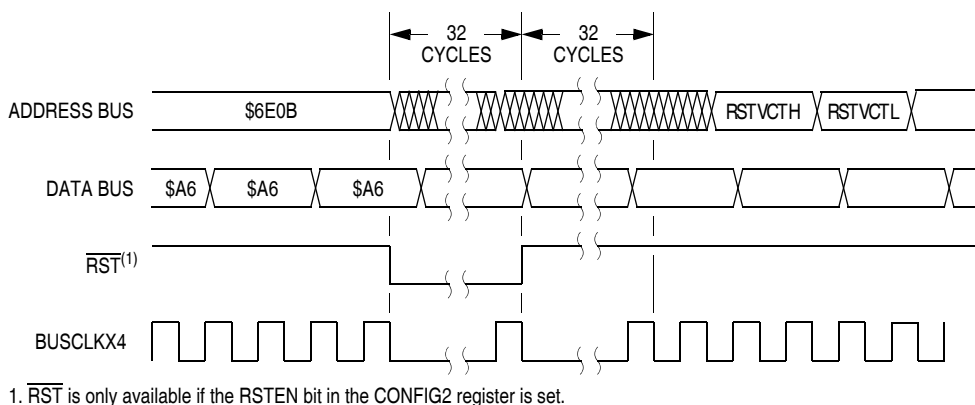


Figure 14-16. Wait Recovery from Internal Reset

14.7.2 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, BUSCLKX4 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.

SPTIE— SPI Transmit Interrupt Enable

This read/write bit enables interrupt requests generated by the SPTE bit. SPTE is set when a byte transfers from the transmit data register to the shift register.

- 1 = SPTE interrupt requests enabled
- 0 = SPTE interrupt requests disabled

15.8.2 SPI Status and Control Register

The SPI status and control register contains flags to signal these conditions:

- Receive data register full
- Failure to clear SPRF bit before next byte is received (overflow error)
- Inconsistent logic level on \overline{SS} pin (mode fault error)
- Transmit data register empty

The SPI status and control register also contains bits that perform these functions:

- Enable error interrupts
- Enable mode fault error detection
- Select master SPI baud rate

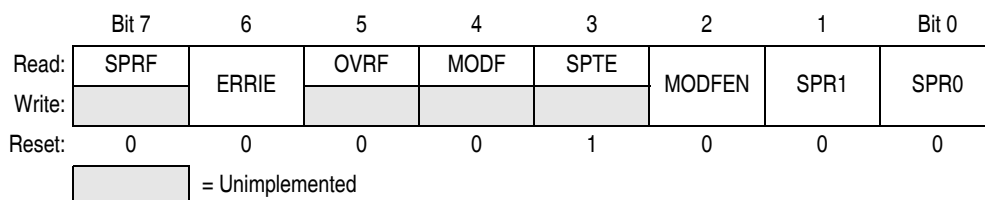


Figure 15-14. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full Bit

This clearable, read-only flag is set each time a byte transfers from the shift register to the receive data register. SPRF generates a interrupt request if the SPRIE bit in the SPI control register is set also.

During an SPRF interrupt, user software can clear SPRF by reading the SPI status and control register with SPRF set followed by a read of the SPI data register.

- 1 = Receive data register full
- 0 = Receive data register not full

ERRIE — Error Interrupt Enable Bit

This read/write bit enables the MODF and OVRF bits to generate interrupt requests.

- 1 = MODF and OVRF can generate interrupt requests
- 0 = MODF and OVRF cannot generate interrupt requests

OVRF — Overflow Bit

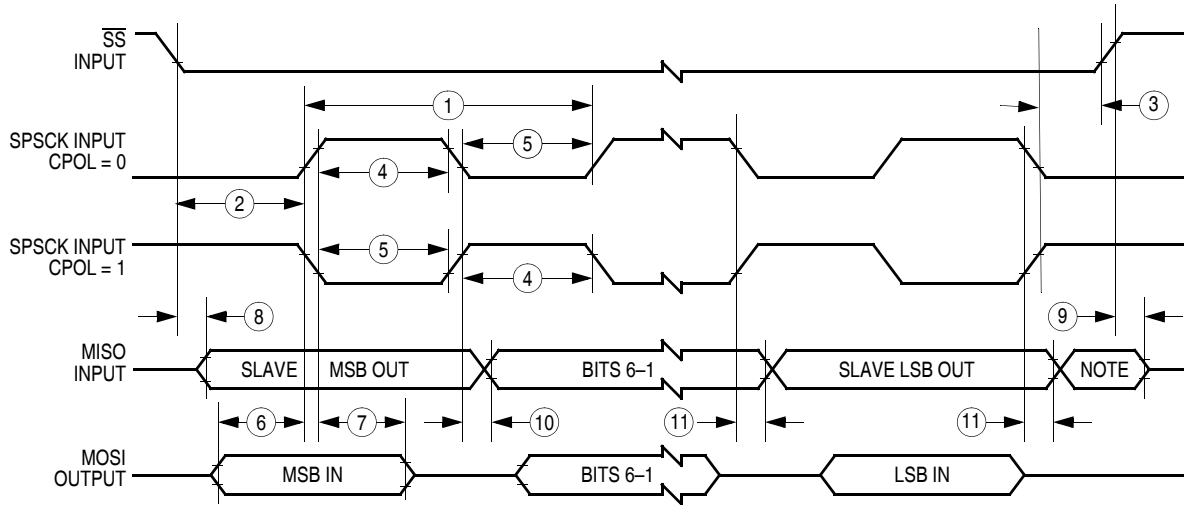
This clearable, read-only flag is set if software does not read the byte in the receive data register before the next full byte enters the shift register. In an overflow condition, the byte already in the receive data register is unaffected, and the byte that shifted in last is lost. Clear the OVRF bit by reading the SPI status and control register with OVRF set and then reading the receive data register.

- 1 = Overflow
- 0 = No overflow

18.15 3.0-Volt SPI Characteristics

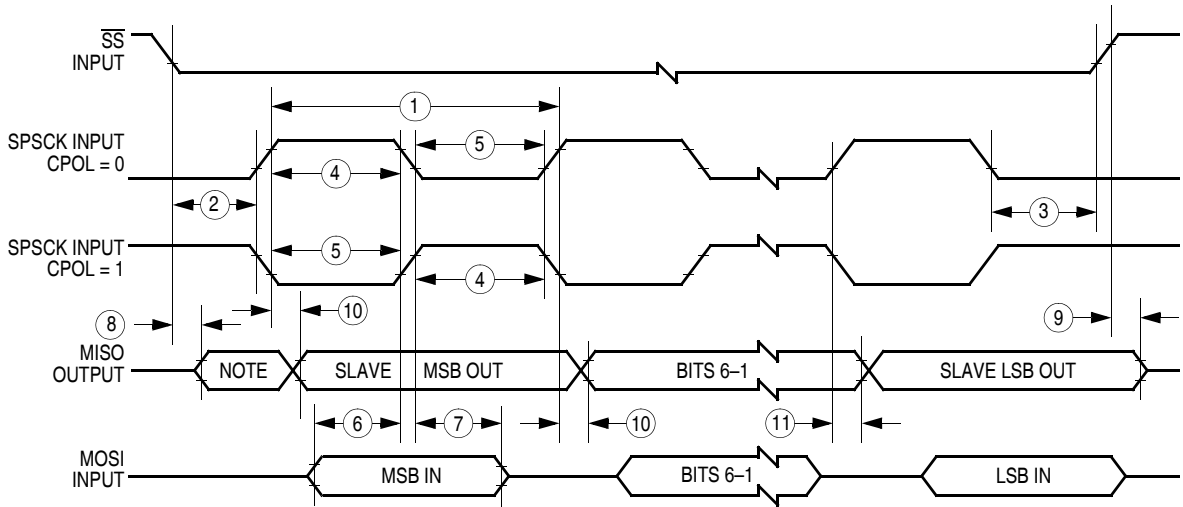
Diagram Number ⁽¹⁾	Characteristic ⁽²⁾	Symbol	Min	Max	Unit
	Operating frequency Master Slave	$f_{OP(M)}$ $f_{OP(S)}$	$f_{OP}/128$ DC	$f_{OP}/2$ f_{OP}	MHz MHz
1	Cycle time Master Slave	$t_{cyc(M)}$ $t_{cyc(S)}$	2 1	128 —	t_{cyc} t_{cyc}
2	Enable lead time	$t_{Lead(S)}$	1	—	t_{cyc}
3	Enable lag time	$t_{Lag(S)}$	1	—	t_{cyc}
4	Clock (SPSCK) high time Master Slave	$t_{SCKH(M)}$ $t_{SCKH(S)}$	$t_{cyc} - 35$ $1/2 t_{cyc} - 35$	$64 t_{cyc}$ —	ns ns
5	Clock (SPSCK) low time Master Slave	$t_{SCKL(M)}$ $t_{SCKL(S)}$	$t_{cyc} - 35$ $1/2 t_{cyc} - 35$	$64 t_{cyc}$ —	ns ns
6	Data setup time (inputs) Master Slave	$t_{SU(M)}$ $t_{SU(S)}$	40 40	— —	ns ns
7	Data hold time (inputs) Master Slave	$t_{H(M)}$ $t_{H(S)}$	40 40	— —	ns ns
8	Access time, slave ⁽³⁾ CPHA = 0 CPHA = 1	$t_{A(CP0)}$ $t_{A(CP1)}$	0 0	50 50	ns ns
9	Disable time, slave ⁽⁴⁾	$t_{DIS(S)}$	—	50	ns
10	Data valid time, after enable edge Master Slave ⁽⁵⁾	$t_{V(M)}$ $t_{V(S)}$	— —	60 60	ns ns
11	Data hold time, outputs, after enable edge Master Slave	$t_{HO(M)}$ $t_{HO(S)}$	0 0	— —	ns ns

1. Numbers refer to dimensions in [Figure 18-11](#) and [Figure 18-12](#).
2. All timing is shown with respect to 20% V_{DD} and 70% V_{DD} , unless noted; 100 pF load on all SPI pins.
3. Time to data active from high-impedance state
4. Hold time to high-impedance state
5. With 100 pF on all SPI pins



Note: Not defined but normally MSB of character just received

a) SPI Slave Timing (CPHA = 0)



Note: Not defined but normally LSB of character previously transmitted

b) SPI Slave Timing (CPHA = 1)

Figure 18-12. SPI Slave Timing

18.16 Timer Interface Module Characteristics

Characteristic	Symbol	Min	Max	Unit
Timer input capture pulse width ⁽¹⁾	t_{TH}, t_{TL}	2	—	t_{cyc}
Timer input capture period	t_{TLTL}	Note ⁽²⁾	—	t_{cyc}
Timer input clock pulse width ⁽¹⁾	t_{TCL}, t_{TCH}	$t_{cyc} + 5$	—	ns

1. Values are based on characterization results, not tested in production.
2. The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1 t_{cyc} .

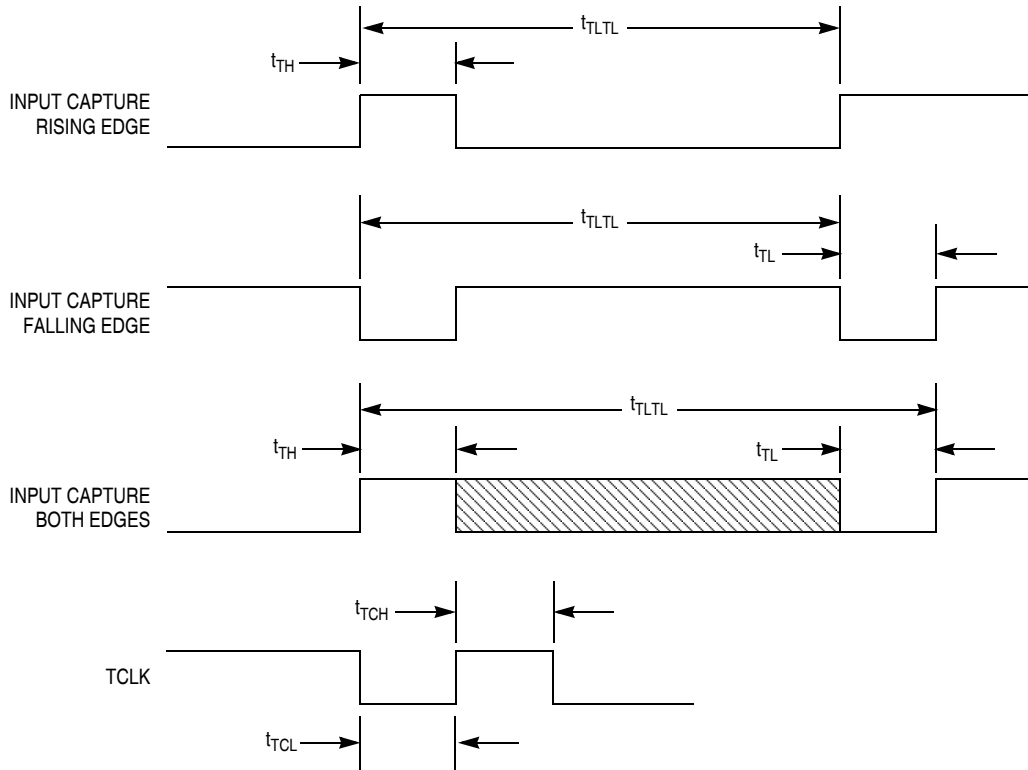


Figure 18-13. Timer Input Timing