TIM08RM/AD
Rev. 1.0

HC08

TIM08
TIMER INTERFACE MODULE

REFERENCE MANUAL

MOTOROLA

TIM08

Timer Interface Module
Reference Manual

List of Sections

Revision History

This table summarizes differences between this revision and the previous revision of this reference manual.

Previous Revision	Original Release
Current Revision	1.0
Date	08/96
Changes	Format and organizational changes Incorporated changes reflected in Addendum (TIM08RMAD/AD)
Location	Throughout

Preface

All M68HC08 microcontrollers are modular, customer-specified designs. To meet customer requirements, Motorola is constantly designing new modules and creating new versions of exisitng modules.

The *TIM08 Reference Manual* introduces version B of the TIM08, the timer interface module of the Motorola HC08 Family. Future versions of the TIM08 will be attached as appendices in this reference manual.

Table of Contents

16-Bit Modulo Counter

List of Figures

List of Figures

List of Tables

Overview

Contents

Introduction

The timer interface module (TIM), a module in Motorola's HC08 Family of modular microcontrollers, is a simple yet flexible timer for use in systems where a moderate level of CPU control is required. The TIM can be implemented with two, four, six or eight channels. This manual will show the 4-channel version, as implemented in the MC68HC708XL36. The TIM can be broken into several submodules: the prescaler, the 16-bit modulo counter, and the capture/compare unit. **Figure 1** shows the major submodules of the TIM, the external pins associated with the TIM, and the internal bus signals used by the TIM.

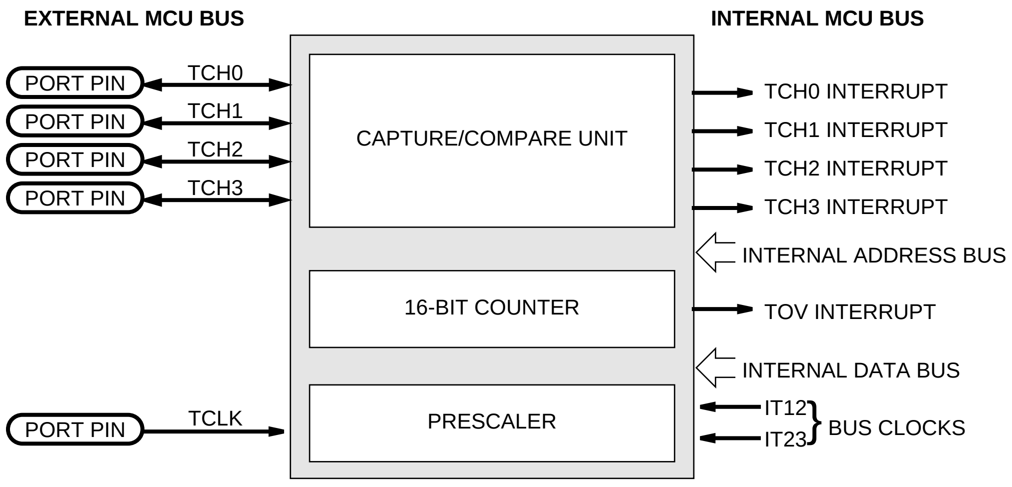


Figure 1. TIM Submodules

Figure 2 shows the structure of the TIM. The central component of the TIM is the 16-bit counter that can operate as a free-running counter or a modulo up-counter. The timer counter provides the timing reference for the input capture, output compare, and pulse-width modulation functions provided by the capture/compare unit. The timer counter modulo registers, TMODH:TMODL, control the modulo value of the timer counter. Software can read the timer counter value from the timer counter registers, TCNTH:TCNTL, at any time without affecting the counting sequence.

The capture/compare unit features two, four, six, or eight channels. These channels share the 16-bit counter (TCNTH:TCNTL) which receives its clock input from the six-stage prescaler or from the external clock input pin, TCLK. Each channel can be programmed as either an input capture channel, an unbuffered output compare channel, or an unbuffered pulse-width modulation channel. Two channels may be combined to provide one buffered output compare channel or one buffered pulse-width modulation channel.

If not needed for timing functions, any of the TIM pins can be used as general-purpose bidirectional input/output (I/O) port pins.

Figure 2 shows the registers in the TIM.

NOTE: *The TIM can be implemented with two, four, six or eight channels. This manual will show the 4-channel version, as implemented in the MC68HC708XL36.*

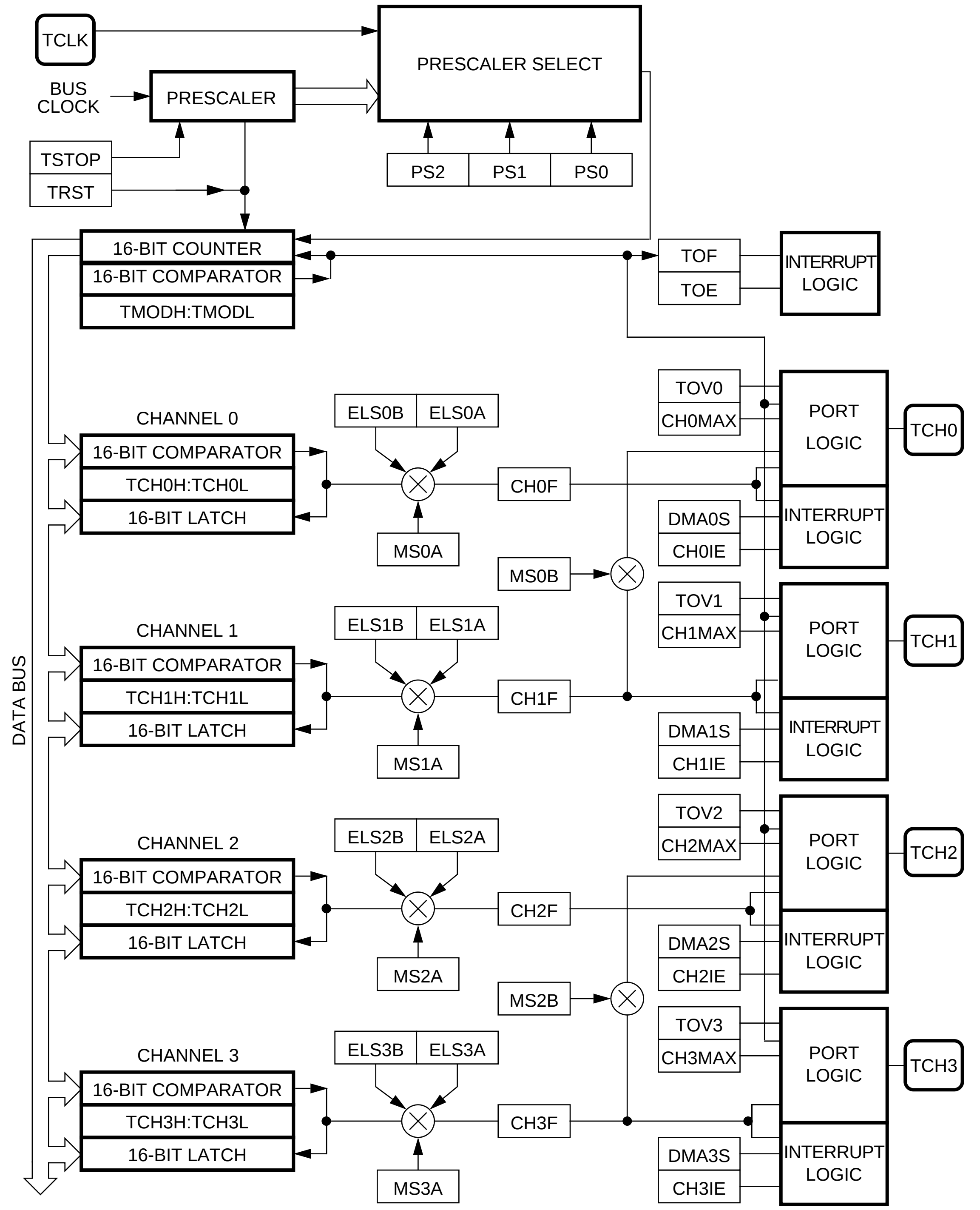


Figure 2. TIM Block Diagram

	7	6	5	4	3	2	1	0
TIMER STATUS AND CONTROL REGISTER (TSC)	TOF	TOE	TSTOP	TRST	0	PS2	PS1	PS0
TIMER DMA SELECT REGISTER (TDMA)	0	0	0	0	DMA3S	DMA2S	DMA1S	DMA0S
TIMER COUNTER REGISTER HIGH (TCNTH)	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
TIMER COUNTER REGISTER LOW (TCNTL)	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
TIMER MODULO REGISTER HIGH (TMODH)	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
TIMER MODULO REGISTER LOW (TMODL)	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CHANNEL 0 STATUS AND CONTROL REGISTER (TSC0)	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
CHANNEL 0 REGISTER HIGH (TCH0H	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
CHANNEL 0 REGISTER LOW (TCH0L)	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CHANNEL 1 STATUS AND CONTROL REGISTER (TSC1)	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
CHANNEL 1 REGISTER HIGH (TCH1H)	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
CHANNEL 1 REGISTER LOW (TCH1L)	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CHANNEL 2 STATUS AND CONTROL REGISTER (TSC2)	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
CHANNEL 2 REGISTER HIGH (TCH2H)	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
CHANNEL 2 REGISTER LOW (TCH2L)	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CHANNEL 3 STATUS AND CONTROL REGISTER (TSC3)	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
CHANNEL 3 REGISTER HIGH (TCH3H)	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
CHANNEL 3 REGISTER LOW (TCH3L)	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

Figure 2. TIM Block Diagram (Concluded)

Features

- Modular Architecture
- Up to Eight Input Capture Channels
  - Rising-Edge, Falling-Edge, or Any-Edge Input Capture Trigger
- Up to Eight Unbuffered Output Compare Channels or Four Buffered Output Compare Channels
  - Set, Clear, or Toggle Output Compare Action
  - Toggle Any Channel Pin on Counter Overflow
- Up to Eight Unbuffered Pulse Width Modulation (PWM) Channels or Four Buffered PWM Channels
  - 100% Duty Cycle Capability
  - Set, Clear, or Toggle Action on Pulse Width Match
  - Toggle Any Channel Pin on Counter Overflow
- Programmable TIM Clock Input
  - 7-Frequency Bus Clock Prescaler Selection
  - External TIM Clock Input
- Free-Running or Modulo Up-Count Operation
- Timer Counter Stop and Reset Bits
- CPU Interrupt Generation
- DMA Service Request Generation on Microcontrollers Containing a DMA Module

Input Capture (IC) Concepts

An input capture function has three basic parts: edge select logic, an input capture latch, and a 16-bit counter. The edge select logic determines the type of input transition to which the circuit responds. When an input transition occurs, an input capture function latches the contents of the counter into the input capture latch. This action sets a status flag indicating that an input capture has occurred. (See **Figure 3**.)

When the status flag is set, an interrupt is generated if enabled. The value of the count latched or "captured" is the time of the event. Because this value is stored in the input capture register when the actual event occurs, user software can respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

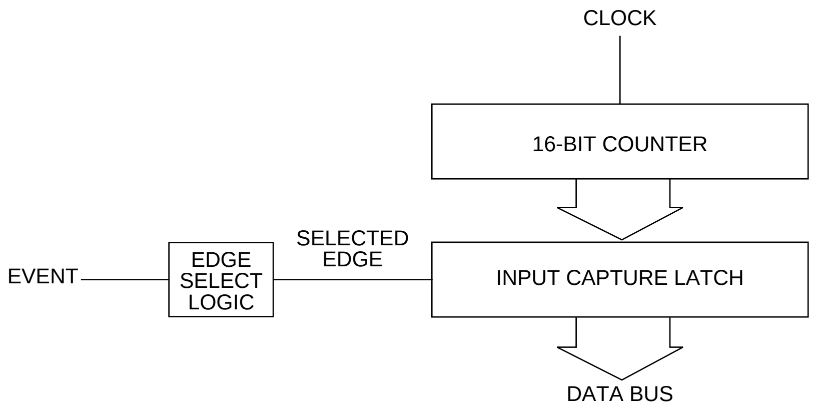


Figure 3. Input Capture Simplified Block Diagram

Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit. If interrupts are enabled, an interrupt can be directed to the CPU, or a service request can be directed to the DMA, when available. If the status flag is being polled by the software, an input capture subroutine can be executed when the status flag is set.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. For example, to measure the high time of a pulse, the input transition is captured at the rising edge and subtracted from the time captured for the subsequent falling edge. When the period or pulse width is less than the 16-bit modulo counter overflow period, the measurement is very straightforward. In practice, however, software usually must track the overflows of the 16-bit modulo counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal, a specific number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register. Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Output Compare (OC) Concepts

Output compare functions are used to program a specific time an event occurs. An output compare function has a 16-bit compare register and a 16-bit comparator. A 16-bit modulo counter provides the timing reference for output compares. When the contents of the compare register match the value of the counter, the comparator sets an output compare flag. (See **Figure 4**.)

Other events can occur when the flag is set. An interrupt can be generated if enabled. State changes can optionally occur on pins associated with the output compare function.

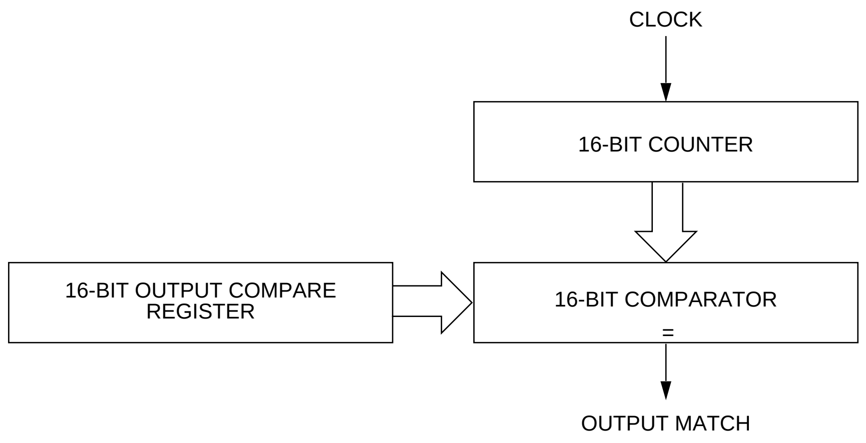


Figure 4. Output Compare Simplified Block Diagram

Software can determine that an output compare match has occurred by enabling output compare interrupts or by polling the status flag bit. If interrupts are enabled, an interrupt can be directed to the CPU, or a service request can be directed to the DMA, when available. If the status flag is being polled by the software, an output compare subroutine can be executed when the status flag is set.

The output compare function can generate an output of a specific duration and polarity. A 16-bit value corresponding to the time a pin state change will occur is written to the output compare register. The output compare function is configured to automatically generate a high or low output on the pin or toggle its state when the match occurs. The output compare register can be reprogrammed to a new value after the compare occurs. The new value corresponds to the time the next compare occurs. When the next match takes place, the pin automatically changes to the specified state. The output compare pin can also be configured to toggle its state when the 16-bit modulo counter overflows. Because pin state changes occur automatically at specific values of the counter, the pulse width can be controlled to the resolution of the counter independent of software latencies. A periodic pulse of a specific frequency and duty cycle can be generated by repeating the preceding steps.

Unbuffered Output Compares

Any TIM channel can generate unbuffered output compare pulses. The signal is unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the channel registers.

An unsynchronized write to the channel registers to change a pulse width could cause incorrect operation for up to two counter periods. For example, if a new output compare value is written to the compare registers before the previous output compare match occurred, but after the counter had reached the new value, no output compare match would occur during that counter period. Or if a new, small value is written during a timer overflow interrupt routine, but the output compare match is missed because the new value is not written until after the timer counter has passed that value, no output compare match would occur during that counter period.

The output compare interrupt occurs at the end of the current pulse, while the timer overflow interrupt occurs at the end of the current period.

In applications that cannot tolerate erroneous data during output compare pulse width changes, two methods are used to synchronize an unbuffered output compare pulse width. When changing to a longer pulse width, enable timer overflow interrupts and write the new pulse width during the timer overflow interrupt routine. When changing to a shorter pulse width, enable output compare interrupts and write the new pulse width value during the output compare interrupt routine.

Buffered Output Compares

A buffered output compare eliminates the synchronization problem inherent in the unbuffered output compare by providing two channel registers in which to store compare values. In this method, the current output compare value is contained in the first channel register, while a new output compare value is written into the second channel register. On counter overflow, the second channel register value is used to generate the output compare match. By writing new output compare match values only to the unused channel register, erroneous waveforms can be eliminated.

Two TIM channels can be linked to form one buffered output compare channel. When the contents of compare register A match the value of the counter, the comparator sets an output compare flag. Compare register A is used as the output compare value until a new value is written to output compare register B. At the next counter overflow, control switches from output compare register A to output compare register B. Control continues to switch between output compare registers A and B as a new value is written to each register. All control functions and output occur on channel A. The channel B control register is unused, and output pin B reverts to port control. (See **Figure 5**.)

NOTE: *In buffered output compare mode, do not write new compare values to the currently active channel registers. The software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered output compare signals.*

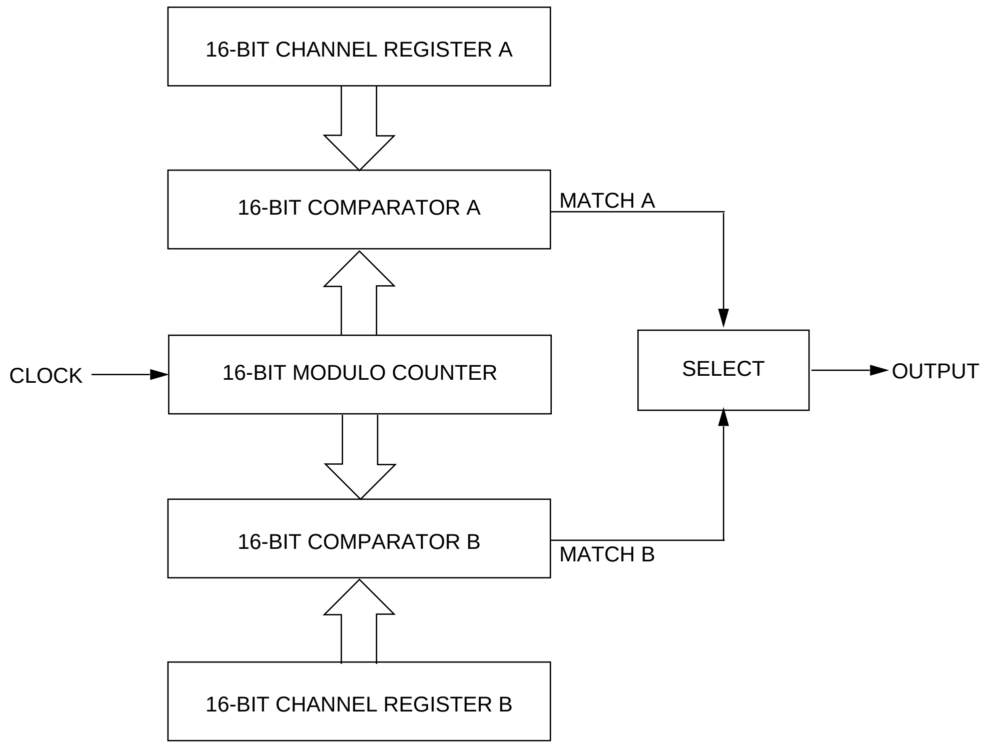


Figure 5. Buffered Output Compare Simplified Block Diagram

Pulse-Width Modulation (PWM) Concepts

A pulse-width modulated waveform is created when the high to low time ratio, or pulse-width, of a periodic signal can be varied. For example, if the waveform can be incrementally changed by 1/256 of its period, it has 8 bits of resolution. (See **Figure 6**.)

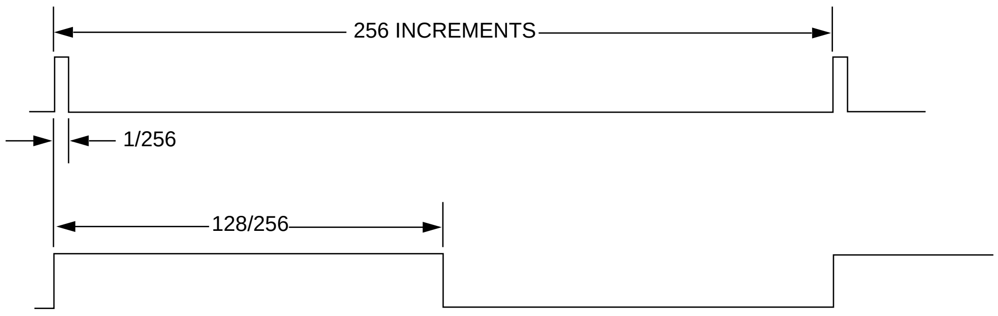


Figure 6. Pulse-Width Modulation Example

As shown in **Figure 7**, a PWM function has a 16-bit counter, two 16-bit comparators, and an output latch.

When the 16-bit counter reaches the modulo value in the 16-bit modulo registers, the 16-bit modulo comparator sets the output latch, indicating a modulo counter overflow. The modulo overflow is used as the reference to start the pulse, thereby setting the period of the waveform. As the counter is incremented, the counter value is compared with the contents of the 16-bit channel register. When a match occurs the latch is reset, ending the pulse. The duty cycle of the signal is varied by changing the value in the 16-bit channel register.

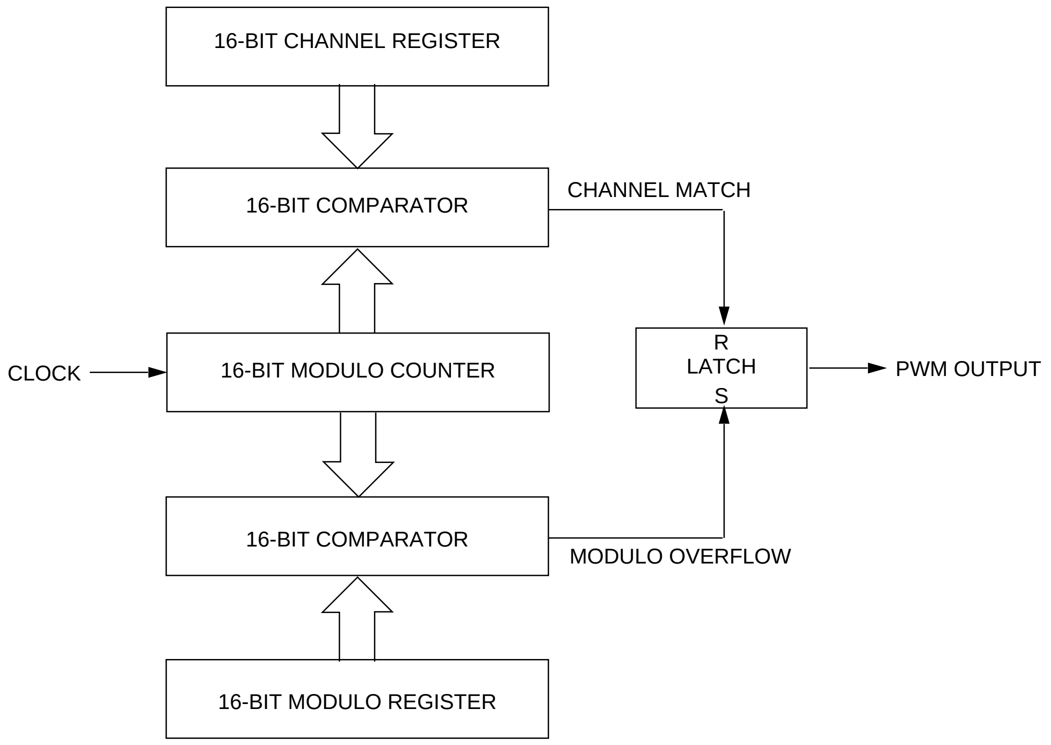


Figure 7. Pulse-Width Modulation Simplified Block Diagram

The input clock frequency to the 16-bit modulo counter, f_{TCNT}, determines the resolution of the PWM signal. The counter clock frequency is determined by the prescaler programming. By increasing the counter clock frequency, the resolution of the PWM signal becomes finer. For example, when using a counter clock of 2 MHz, the resolution of the counter is 500 ns. If f_{TCNT} is changed to 8 MHz, the resolution of the counter increases to 125 ns.

The value in the timer modulo registers determines the frequency of the PWM output. For example, the frequency of an 8-bit PWM signal is variable in 256 increments. Writing \$00FF (255) to the timer modulo registers produces a PWM frequency of $f_{TCNT} \div 256$.

The value in the timer channel registers determines the pulse width of the PWM output. The pulse width of an 8-bit PWM signal is variable in 256 increments. Writing \$0080 (128) to the timer channel registers produces a duty cycle of 128/256 or 50%.

The polarity of the pulse can be changed from a logic 1 state to a logic 0 state. Writing to a special control bit is required to obtain a 100% duty cycle (output high all of the time) or a 0% duty cycle (output low all of the time).

The PWM output can be used to electronically control the speed of a motor or the position of a servo. The PWM waveform drives an external switching amplifier which in turn controls the speed and direction of the motor. By adding a low-pass filter to a PWM output, the unit can be used as a D/A converter. The longer the high time of the output waveform, the higher the average value of output voltage produced. Other applications include data communication, where the pulse width indicates the data value.

Unbuffered PWM Signal Generation

Any TIM channel can generate unbuffered PWM signals. The signal is unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the channel registers.

An unsynchronized write to the channel registers to change a pulse width could cause incorrect operation for up to two counter periods. For example, if a new pulse width value is written to the channel registers before the previous pulse width match occurred, but after the counter had reached the new value, no pulse width match would occur during that counter period. Or if a new, small value is written during a timer overflow interrupt routine, but the pulse width match is missed because the new value is not written until after the timer counter is past that value, no pulse width match would occur during that counter period.

The PWM interrupt occurs at the end of the current pulse. The timer overflow interrupt, by contrast, occurs at the end of the current period.

In applications which cannot tolerate erroneous data during output compare pulse width changes, two methods are used to synchronize an unbuffered PWM pulse width. When changing to a longer pulse width, enable timer overflow interrupts and writes the new pulse width during the timer overflow interrupt routine. When changing to a shorter pulse width, enable PWM interrupts and writes the new pulse width value during the PWM interrupt routine.

Buffered PWM Signal Generation

A buffered PWM function eliminates the synchronization problem inherent in the unbuffered PWM by providing two channel registers in which to store pulse width values. In this method, the current pulse width value is contained in the first channel register, while a new pulse width value is written into the second channel register. On counter overflow, the second channel register value is used to generate the pulse width match. By writing new pulse width values only to the unused channel register, erroneous waveforms can be eliminated.

Two TIM channels can be linked to form one buffered PWM channel. When the contents of channel register A match the value of the counters, the comparator sets a PWM flag. Channel register A is used as the pulse width value until a new value is written to channel register B. At the next counter overflow, control switches from channel register A to channel register B. Control continues to switch between channel registers A and B as a new value is written to each register. All control functions and output occurs on channel A. The channel B control register is unused, and output pin B reverts to port control. (See **Figure 5**.)

NOTE: *In buffered PWM mode, do not write new compare values to the currently active channel registers. The software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered PWM signals.*

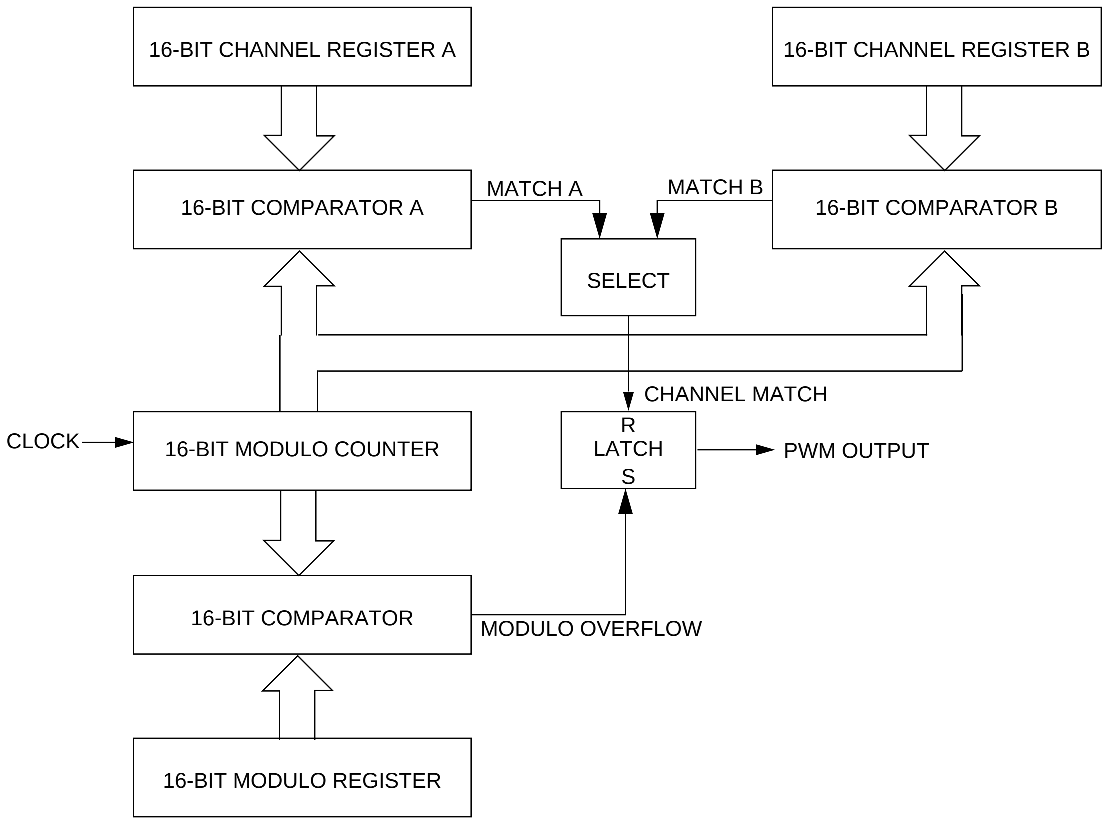


Figure 8. Buffered PWM Simplified Block Diagram

Signal Description

Contents

Introduction

The TIM has five signal pins that provide connections to the internal functions of the module. These pins are shared with port pins on the microcontroller. This section contains brief descriptions of the TIM input and output signals in their functional groups. See **Electrical Specifications** for timing information for these signals.

NOTE: *The TIM can be implemented with two, four, six or eight channels. This manual will show the 4-channel version, as implemented in the MC68HC708XL36.*

Signal Groups

The block diagram in **Figure 9** shows the signal pins. When the pins are not needed for their TIM function, they can be used for general-purpose input or output as part of a parallel data port. The port pins are part of a separate module. Refer to the applicable technical data book for information on the port module associated with the TIM. The block diagram also shows which TIM signals are bidirectional and which are either input or output only.

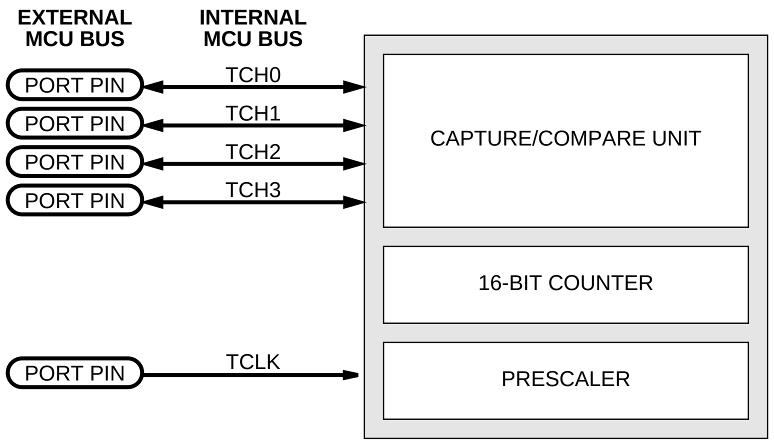


Figure 9. Function Signal Groups

Input Capture/Output Compare Pins (TCH0, TCH1, TCH2, TCH3)

Each of these pins is dedicated to one of the timer channels. These pins can be configured for an input capture, output compare, or PWM function. Each channel pin has one 16-bit register, which is used for holding either the input capture value or the output compare/PWM match value. When used as an input, the signal is conditioned so that any pulse longer than one bus clock period is guaranteed to pass. If this pin is not needed for either the input capture, output compare, or PWM function, it can be used for general-purpose I/O.

Input Capture Pins

With the input capture function, the TIM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TIM latches the contents of the timer counter into the channel registers, TCHxH:TCHxL. The polarity of the active edge is programmable. Input captures can generate CPU interrupts or DMA service requests on microcontrollers with a DMA module.

See **Input Capture (IC) Functions** for information on the operation of this function.

Output Compare Pins

With the output compare function, the TIM can generate an output signal at programmable intervals. When the counter reaches the value in the registers of an output compare channel, the TIM can set, clear, or toggle the channel pin. Output compares can generate CPU interrupts or DMA service requests on microcontrollers with a DMA module.

See **Unbuffered Output Compare (OC) Functions** and **Buffered Output Compare (OC) Functions** for information on the operation of these functions.

PWM Output Pins

With the PWM function, the TIM can generate a pulse-width modulated output signal. When the counter reaches the value in the registers of a PWM channel, the TIM can set, clear, or toggle the channel pin. A PWM function can generate CPU interrupts or DMA service requests on microcontrollers with a DMA module.

See **Unbuffered Pulse Width Modulation (PWM) Functions** and **Buffered Pulse Width Modulation (PWM) Functions** for information on the operation of these functions.

General-Purpose I/O

If not used as input capture, output compare, or PWM functions, these pins may be used as general-purpose I/O. This is accomplished by clearing the ELSxB and ELSxA bits in the TSCx register for each channel pin, as described in **Capture/Compare Unit**. Refer to the applicable technical data book for more information about using these pins as bidirectional I/O pins.

Auxiliary Timer Clock Input (TCLK)

TCLK is an external clock input that can be used as the clock source for the timer counter instead of the prescaled bus clock. Any TCLK pulse longer than one bus clock period is guaranteed to pass. See **Prescaler** for additional information on TCLK.

If this pin is not used as a clock input, it can be used as a general-purpose I/O pin. See **Table 1** for information on selecting the function of this pin. The minimum TCLK pulse width, $\text{TCLK}_{\text{LMIN}}$ or $\text{TCLK}_{\text{HMIN}}$, is:

$$\frac{1}{\text{bus frequency}} + t_{su}$$

The maximum TCLK frequency is:

$$\frac{\text{bus frequency}}{2}$$

PTE3/TCLK is available as a general-purpose I/O pin when not used as the TIM clock input. When the PTE3/TCLK pin is the TIM clock input, it is an input regardless of the state of the DDRE3 bit in data direction register E.

Prescaler

Contents

Introduction

The TIM has its own 16-bit counter as the main timing component. This counter clock is derived from the prescaler or the external input pin TCLK.

Prescaler

The prescaler generates six clock rates from the bus clock. The prescaler select bits, PS2:PS0, in the timer status and control register (TSC) select the bus clock, a prescaler divider tap, or the external clock, TCLK, as the input to the 16-bit counter. **Figure 10** shows the block diagram of the prescaler.

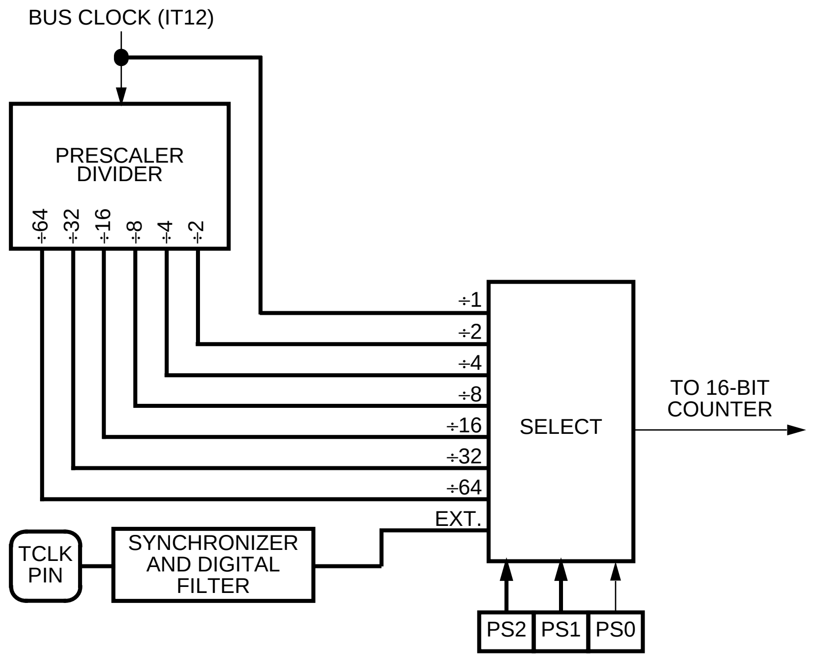


Figure 10. Prescaler Block Diagram

The bus clock is divided by a six-stage divider chain that provides outputs of the bus clock divided by 2, 4, 8, 16, 32, and 64. The outputs of the divider provide six inputs to a multiplexer (mux) which selects the clock input for the 16-bit counter. The remaining inputs to the mux are the bus clock and a synchronized external input from the TCLK pin. The mux provides one of the eight inputs to the 16-bit counter. The output of the mux is controlled by PS2:PS0 in the timer status and control register.

TCLK is an external clock input. TCLK can be used as the clock source for the 16-bit counter instead of the bus clock or its derivative. **Figure 11** shows the timing of TCLK and its synchronization to the bus clock.

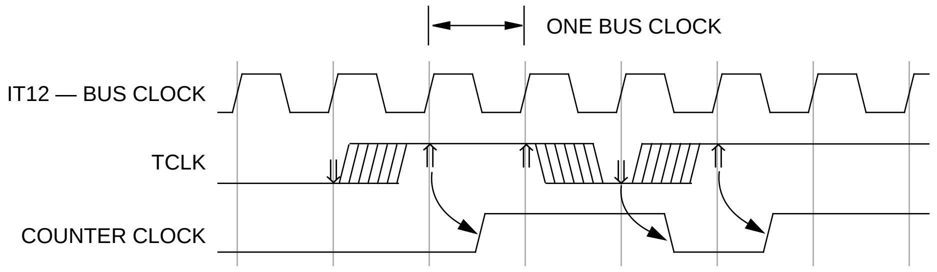


Figure 11. TCLK Timing

where

$$t_{cyc} = \frac{1}{bus\ frequency} + t_{su}$$

Timer Status and Control Register

TSC	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TOF	TOE	TSTOP	0	0	PS2	PS1	PS0
Write:	0			TRST				
Reset:	0	0	1	0	0	0	0	0

▨ Shading indicates this bit is not used in the prescaler section.

Figure 12. Timer Status and Control Register (TSC)

TOF — Unused in prescaler; see **Timer Status and Control Register** for description.

TOE — Unused in prescaler; see **Timer Status and Control Register** for description.

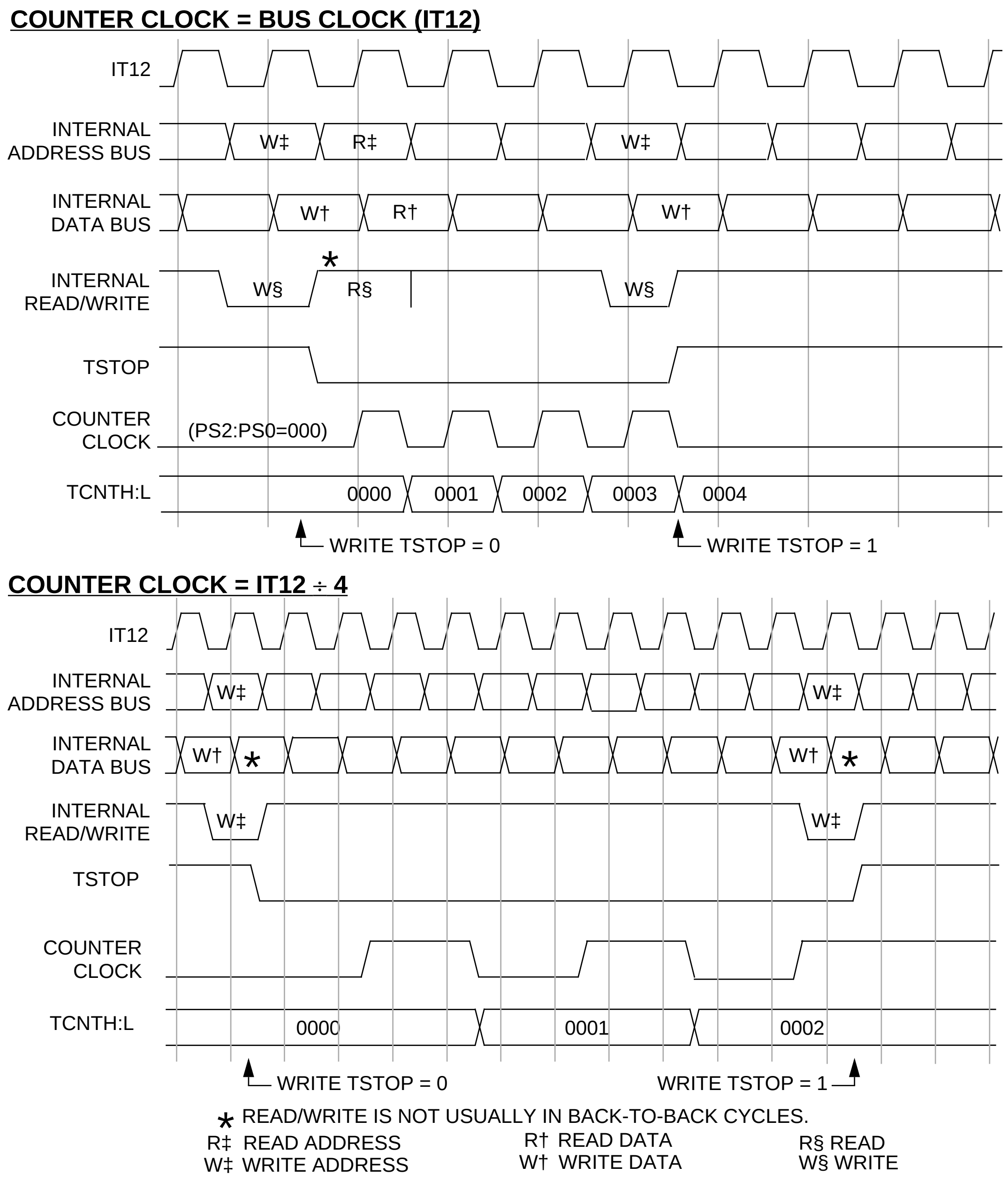


Figure 13. TSTOP Timing

TSTOP — Timer stop

This read/write bit stops the timer counter. Counting resumes when TSTOP is cleared. Reset sets the TSTOP bit, stopping the timer counter until the TIM is enabled.

1 = Timer counter stopped
0 = Timer counter active

To preserve the correct timing relationship, TSTOP stops the input clock to the prescaler. The relationship cannot be preserved when using the external TCLK as the counter clock. **Figure 13** for details on the timing of the TSTOP function.

TRST — Timer reset

Setting this write-only bit resets the timer counter and the timer prescaler. Setting TRST has no effect on any other registers. Counting resumes from $0000. TRST is cleared automatically after the timer counter is reset, and always reads 0. Reset clears the TRST bit.

1 = Prescaler and timer counter cleared
0 = No effect

See **Figure 14** for details on the timing of the TRST function.

NOTE: *Setting the TSTOP and TRST bits simultaneously stops the timer counter at a value of $0000.*

Bit 3 — Not used; always reads 0.

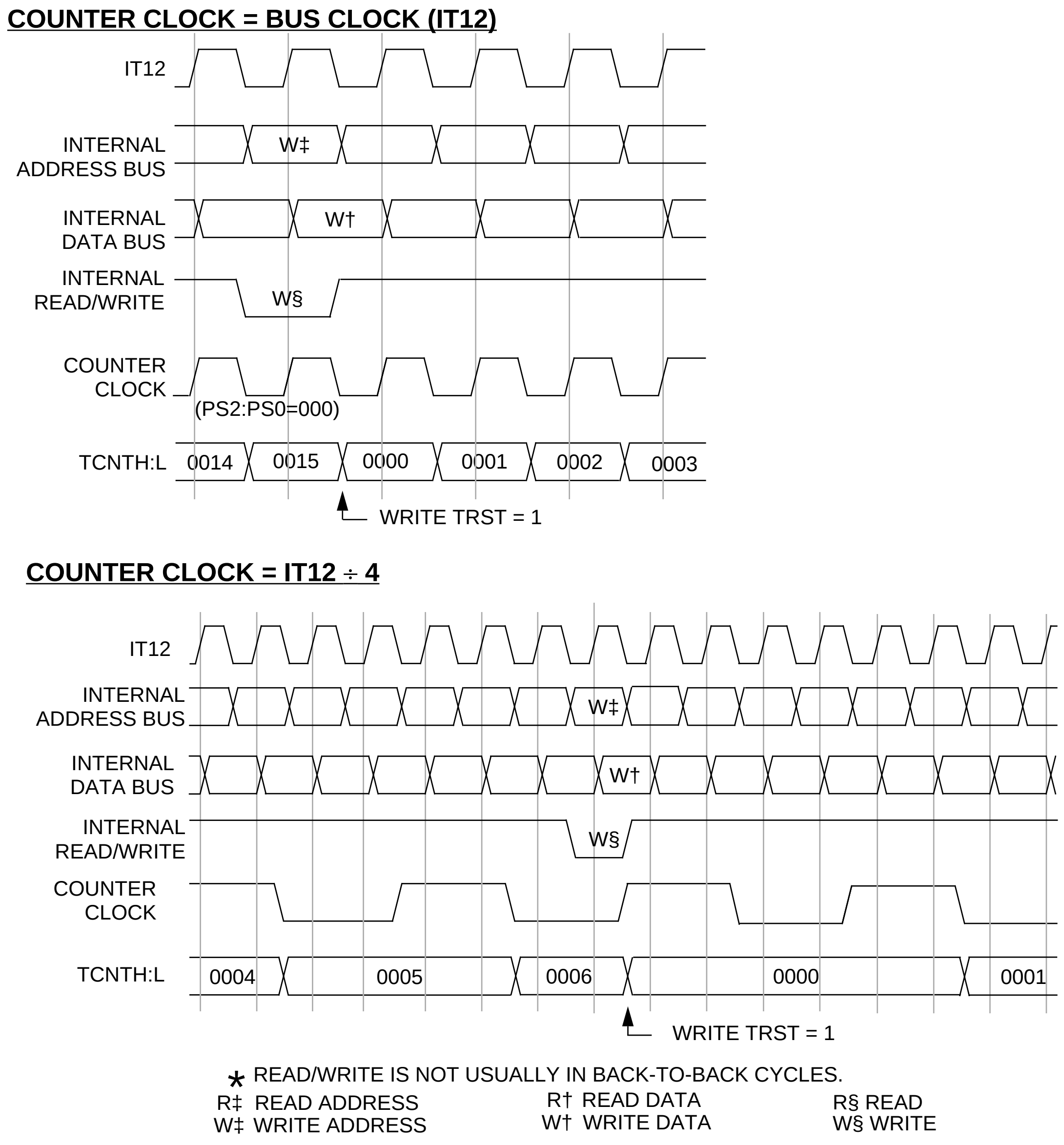


Figure 14. TRST Timing

PS2:PS0 — Prescaler bits

These read/write bits select the bus clock, one of the six prescaler outputs, or the TCLK pin as the input to the timer counter. **Table 1** shows the prescaler selection encoding, including the TIM clock source, and the function of the TCLK pin. Reset clears the PS2:PS0 bits.

Table 1. Prescaler Selection

PS2:1:0	TIM Clock Source	PORT/TCLK Function
000	Bus Clock	PORT
001	Bus Clock ÷ 2	PORT
010	Bus Clock ÷ 4	PORT
011	Bus Clock ÷ 8	PORT
100	Bus Clock ÷ 16	PORT
101	Bus Clock ÷ 32	PORT
110	Bus Clock ÷ 64	PORT
111	TCLK	TCLK

NOTE: *Stop the TIM before changing the prescaler output. Before writing to the prescaler select bits (PS2:PS0), set the timer stop bit (TSTOP).*

NOTE: *Changing the prescaler control bits while the TIM is running may cause an extra count if the input clock previously selected was a logic level 0 and the new input clock logic level is 1.*

Refer to **Stop Mode** for information on stopping the prescaler.

16-Bit Modulo Counter

Contents

Introduction

The timer counter provides the capture/compare units with a counter reference for the input capture (IC) functions, the output compare (OC) functions, and the pulse-width modulation (PWM) functions.

Timer Counter

The timer counter (TCNT) is the key timing component for the capture/compare unit. The timer counter is a 16-bit modulo counter with a programmable input clock and the capability to be stopped or reset by manipulating control bits in the timer status and control register. The 16-bit modulo counter consists of a 16-bit counter, a 16-bit comparator, a 16-bit modulo register, and interrupt generation logic. Refer to **Figure 15**.

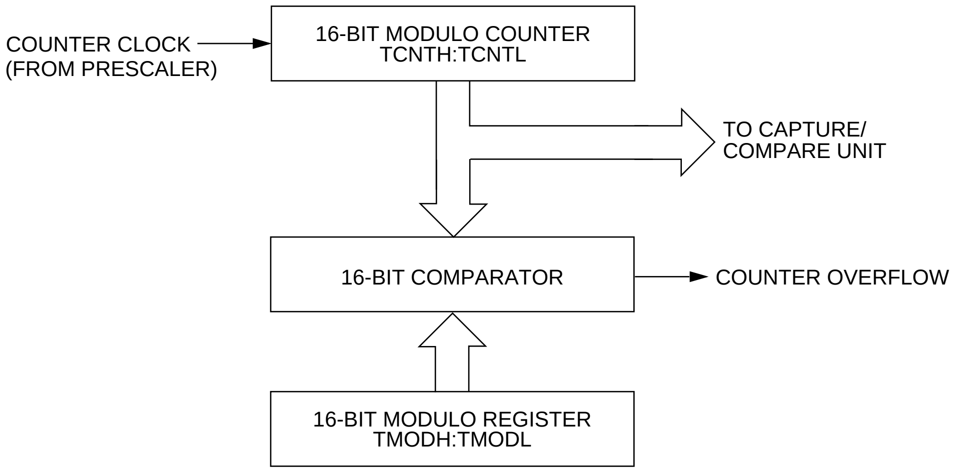


Figure 15. 16-Bit Modulo Counter Simplified Block Diagram

After reset, the TIM counter is stopped, and the bus clock is selected as the input to the counter. User software can configure the system to use the bus clock, one of six outputs from the prescaler, or an external clock through the TCLK input pin. See **Prescaler** for more details on prescaler operation and input clock selection.

After clearing the TSTOP bit in the timer status and control (TSC) register, the counter begins counting from $0000. When the contents of the modulo register (TMOD) match the value of the counter (TCNT), the comparator sets the timer overflow flag (TOF) in the TSC register. Other events can occur when the flag is set. An interrupt can be generated if enabled, and state changes can occur on pins associated with PWM functions in the capture/compare unit.

An interrupt is generated on a timer overflow if the interrupt enable bit, TOE, is set in the TSC register. See **CPU Interrupts** for information on interrupt operation and **DMA Service Requests** for information on service request operation.

A PWM output pin can be toggled on a counter overflow. See **Unbuffered Pulse Width Modulation (PWM) Functions** and **Buffered Pulse Width Modulation (PWM) Functions** for more information on using this function.

Timer Status and Control Register

TSC	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TOF	TOE	TSTOP	0	0	PS2	PS1	PS0
Write:	0			TRST	▨			
Reset:	0	0	1	0	0	0	0	0

▨	Shading indicates this bit is not used in the counter section.

Figure 16. Timer Status and Control Register (TSC)

TOF — Timer Overflow Flag

This clearable flag is set when the timer counter reaches the modulo value programmed in the timer modulo registers. Clear TOF by reading the timer status and control register when TOF is set and then writing a 0 to TOF. If another timer overflow occurs before the clearing sequence is complete, then writing 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Writing a 1 to this bit has no effect. Reset clears the TOF bit.

1 = Timer counter has reached modulo value.
0 = Timer counter has not reached modulo value.

TOE — Timer Overflow Enable

This read/write bit enables timer overflow interrupts when the TOF bit becomes set. Reset clears the TOE bit.

1 = Timer overflow interrupts enabled
0 = Timer overflow interrupts disabled

TSTOP — Timer STOP

This read/write bit stops the timer counter. Counting resumes when TSTOP is cleared. Reset sets the TSTOP bit, stopping the timer counter until the TIM is enabled.

1 = Timer counter stopped
0 = Timer counter active

To preserve the correct timing relationship, TSTOP stops the input clock to the prescaler. The relationship cannot be preserved when using the external TCLK as the counter clock. Refer to **Figure 17** for details on the timing of the TSTOP function.

NOTE: *Do not set the TSTOP bit before entering wait mode if the TIM is required to exit wait mode.*

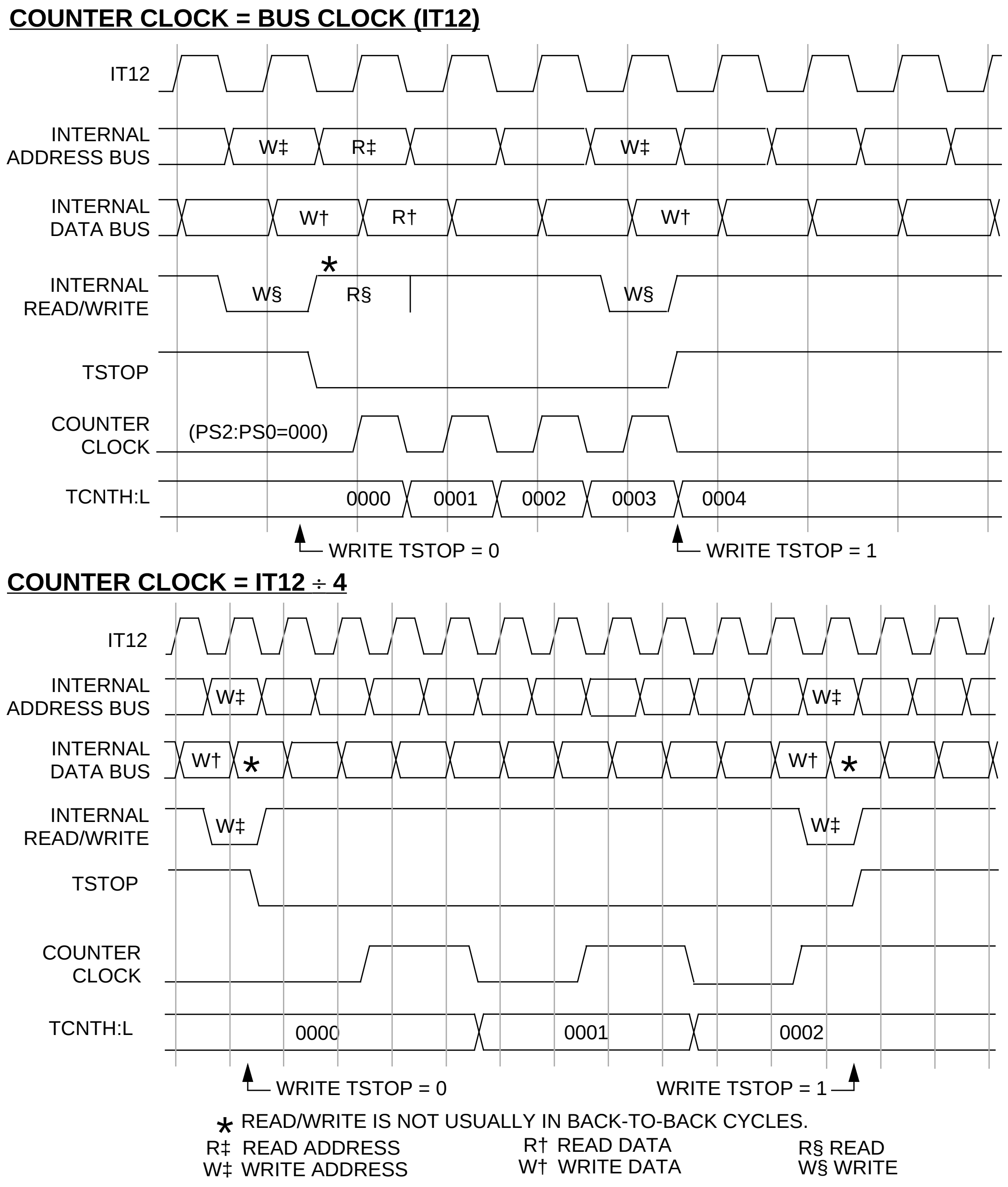


Figure 17. TSTOP Timing

TRST — Timer reset

Setting this write-only bit resets the timer counter and the timer prescaler. Setting TRST has no effect on any other registers. Counting resumes from $0000. TRST is cleared automatically after the timer counter is reset, and always reads 0. Reset clears the TRST bit.

1 = Prescaler and timer counter cleared
0 = No effect

See **Figure 18** for details on the timing of the TRST function.

NOTE: *Setting the TSTOP and TRST bits simultaneously stops the timer counter at a value of $0000.*

PS2:PS0 — Unused in counter; see **Timer Status and Control Register** for a description.

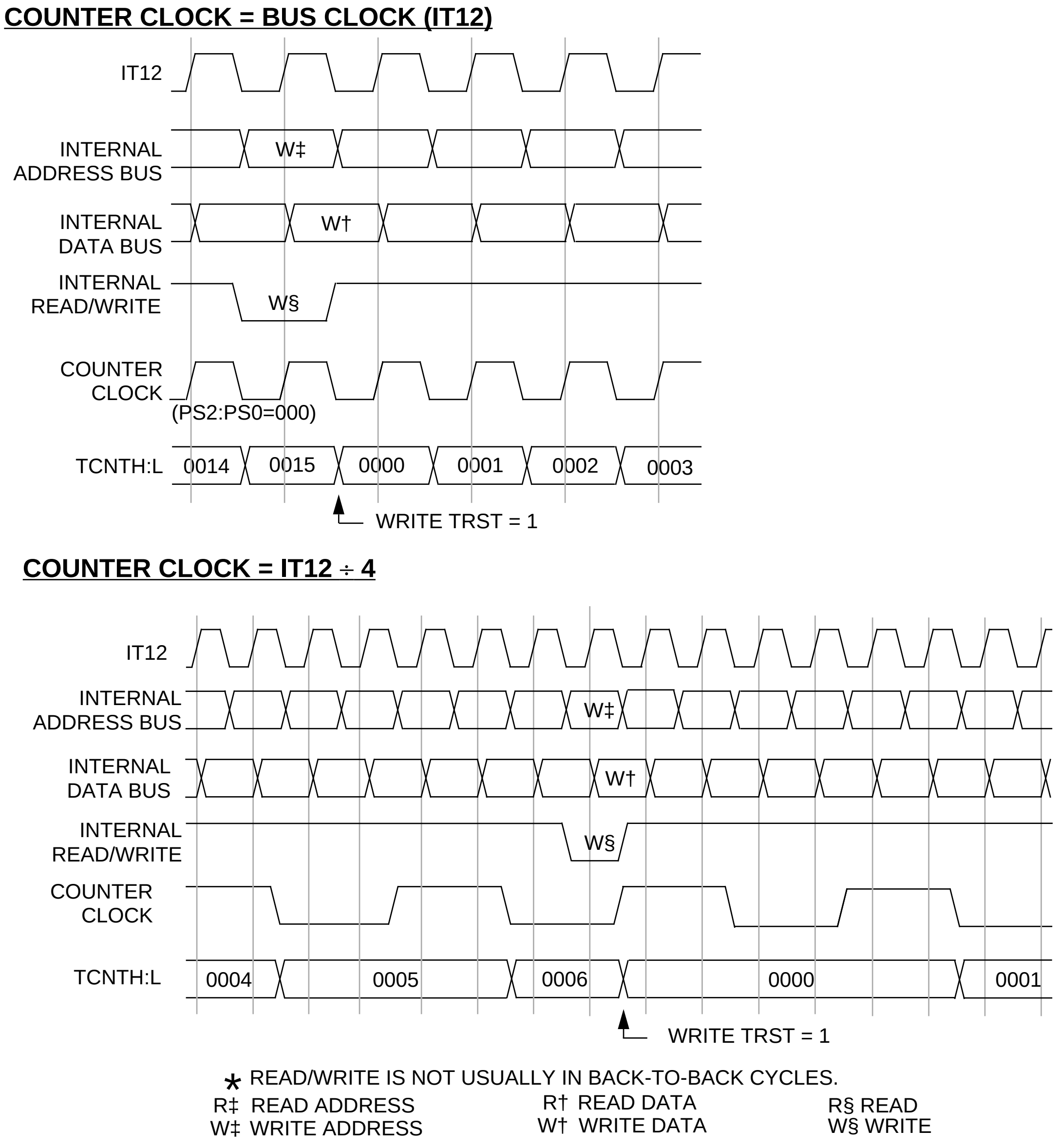


Figure 18. TRST Timing

Timer Counter Registers

These two read-only timer counter registers, TCNTH and TCNTL, contain the high and low bytes of the value in the timer counter. The counter value can be read at any time with user software without affecting its value. Reading the high byte (TCNTH) latches the contents of the low byte (TCNTL). Subsequent reads of TCNTH do not affect the latched TCNTL value until TCNTL is read. The LDHX instruction can be used to read a value from TCNT. The counter is set to $0000 on reset or when the timer reset bit (TRST) is set.

TCNTH	Bit 15	14	13	12	11	10	9	Bit 8
Read:	Bit 15	14	13	12	11	10	9	Bit 8
Write:								
Reset:	0	0	0	0	0	0	0	0

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

Figure 19. Timer Counter Registers (TCNTH:TCNTL)

Timer Counter Modulo Registers

These two read/write timer counter modulo registers, TMODH and TMODL, contain the high and low bytes of the modulo value for the timer counter. When the timer counter reaches the modulo value, the TOF flag is automatically set by hardware, and the timer counter resumes counting from $0000 at the next clock. The overflow flag (TOF) and overflow interrupts are inhibited after a write to the high byte (TMODH) until the low byte (TMODL) is written. The STHX instruction can be used to write values to TMOD, and the LDHX instruction can be used to read values from TMOD. Reset sets the timer counter modulo registers to $FFFF, enabling the modulo counter to act as a free-running counter.

TMODH	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	1	1	1	1	1	1	1	1

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

Figure 20. Timer Counter Modulo Registers (TMODH:TMODL)

NOTE: *If TMODH:TMODL is set to $0000, a TOF is generated on the first cycle in which the match occurs, but not subsequently.*

NOTE: *Stop and reset the timer counter before writing to the timer counter modulo registers.*

Capture/Compare Unit

Contents

Introduction

The capture/compare unit is one of the major submodules of the TIM. It contains the input capture (IC) functions, the output compare (OC) functions, and the pulse-width modulation (PWM) functions.

The function provided by each of the channels is determined by the configuration of the status and control register for each channel. The following sections describe how to set up the status and control registers to generate IC functions, unbuffered and buffered OC functions, and unbuffered and buffered PWM functions. See **Input Capture (IC) Concepts**, **Output Compare (OC) Concepts**, and **Pulse-Width Modulation (PWM) Concepts** for more information about these functions.

NOTE: *The TIM can be implemented with two, four, six or eight channels. This manual will show the 4-channel version, as implemented in the MC68HC708XL36.*

Input Capture (IC) Functions

Each TIM input capture pin TCH0–TCH3, when used as an input capture function, has a 16-bit register latch (TCHxH/L), input edge-detection/selection logic, and interrupt synchronization/generation logic. All of the input capture functions use the same 16-bit timer counter (TCNT). The latch captures the contents of the TCNT when the selected event occurs at the corresponding input capture pin.

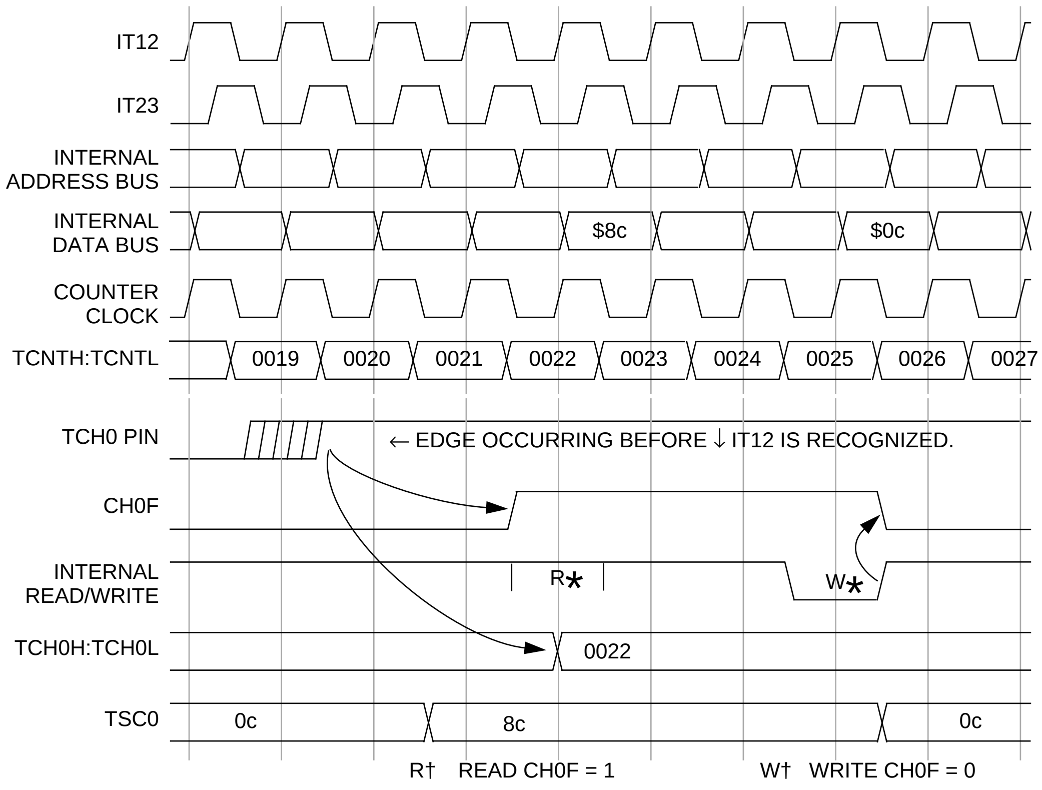
See **Input Capture (IC) Concepts** for additional information on the basic operation of an input capture function.

The edge detection logic contains control bits that allow user software to select the edge polarity to be recognized. These are the ELSxB and ELSxA bits in the timer status and control register (TSCx). Each of the input capture functions can be independently configured to detect rising edges only, falling edges only, any edge (rising or falling), or disable the input capture function. **Table 2** for the required bit patterns.

The input capture functions operate independently of each other and can capture the same TCNT value if the input edges are all detected within the same timer count cycle.

The interrupt generation logic includes a status flag, which indicates that an edge is detected, and a local interrupt enable bit, which determines if the corresponding input capture function will generate an interrupt or service request. The input capture sets the CHxF bit in the TSCx and can cause an interrupt or service request if the corresponding CHxIE bit is set in the TSCx. If the interrupt is disabled (CHxIE = 0), the input capture is operating in polled mode where software must read the status flag to recognize that an edge was detected. Refer to **CPU Interrupts** for additional details on interrupt operation and **DMA Service Requests** for additional details on service request operation.

Because input capture events are generally asynchronous to the timer counter, they are synchronized to the bus clock so that actual latching of the TCNT contents occurs just after the counter increments. The input is conditioned in such a way that any event longer than one bus clock is guaranteed to be captured. The relationship of the bus clock to the output of the synchronizer is shown in **Figure 21**. The value latched into the timer channel register by an input capture corresponds to the value of the counter one bus clock cycle after the input transition that triggered the edge detection logic. There can be up to one bus clock cycle of uncertainty in latching of the input transition. The maximum time is determined by the bus clock frequency, f_{OP}.



NAME	DESCRIPTION
IT12, IT23	Bus clocks used by CPU08, DMA, TIM and all modules on MCU
TCNTH:TCNTL	16-bit value in TCNT register
TCH0 pin	Timer channel 0 input pin
CH0F	CH0F bit in TSC0 register
TCH0H:TCH0L	16-bit value in TCH0 register
TSC0	Timer status and control register, channel 0

Figure 21. Input Capture Timing

Input captures are inhibited between reads from TCHxH and TCHxL. The LDHX instruction can be used to read the contents of the timer channel register TCHxH:TCHxL.

An input capture occurs every time a selected edge is detected, even if the input capture flag is already set. This means that the value read from the timer channel register corresponds to the most recent edge at the pin, which may not be the edge that caused the input capture flag to be set.

If any of the pins TCH0–TCH3 are not needed for an input capture function, they can be used as general-purpose input/output. For more information on general-purpose I/O ports, refer to the applicable technical data book.

Timer Channel Status and Control Registers

The timer channel status and control registers are 8-bit read/write registers. These registers are used to configure the timer channel to perform input capture functions. The state of these registers is reset to $00.

Each of the timer channel status and control registers does the following in input capture mode:

- Flags input captures
- Enables input capture interrupts
- Selects input capture mode of operation
- Selects rising, falling, or any edge as the active input capture trigger

TSC0	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC1	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC2	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC3	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

▨ Shading indicates this bit is not used for input capture functions.

Figure 22. Timer Channel Status and Control Registers (TSC0–TSC3)

CHxF — Channel x Flag

When channel x is an input capture channel, this clearable bit is set when an active edge occurs on the channel x pin.

When CPU interrupts are enabled (CHxE:DMAxS = 1:0), clear CHxF by reading the channel x status and control register with CHxF set and then writing a 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

When DMA service requests are available and enabled (CHxE:DMAxS = 1:1), clear CHxF by reading or writing to the low byte of the channel register (TCHxL).

Writing a 1 to this bit has no effect. Reset clears the CHxF bit.
1 = Input capture has occurred on channel x.
0 = No input capture has occurred on channel x.

CHxIE — Channel x Interrupt Enable

This read/write bit enables channel x interrupts. In microcontrollers with a DMA module, the DMAxS bit in the timer DMA select register selects channel x CPU interrupts or DMA service requests. Reset clears the CHxIE bit.
1 = Channel x interrupts enabled
0 = Channel x interrupts disabled

MSxB — Mode Select bit B

This bit should be cleared for input capture mode operation. MSxB exists only in the channel 0 and channel 2 status and control registers, TSC0 and TSC2.

This read/write bit selects buffered OC or buffered PWM operation. Setting MS0B disables the channel 1 status and control register, and reverts TCH1 to general-purpose I/O. Setting MS2B disables the channel 3 status and control register, and reverts TCH3 to general-purpose I/O. Reset clears the MSxB bit.
1 = Buffered OC/PWM mode enabled
0 = Buffered OC/PWM mode disabled

MSxA — Mode Select bit A

This bit should be cleared for input capture operation.

This read/write bit selects input capture mode or unbuffered OC/PWM mode. MS0A and MS1A are active only when MS0B = 0. MS2A and MS3A are active only when MS2B = 0. Reset clears the MSxA bit.

1 = Unbuffered OC/PWM operation
0 = Input capture operation

NOTE: *Stop and reset the TIM before changing a channel function. Before writing to the mode select bits (MSxB and MSxA), set the timer stop and timer reset bits (TSTOP and TRST) in the TSC register.*

ELSxB and ELSxA — Edge/Level Select bits

When channel x is an input capture channel, these read/write bits control the active edge sensing logic on channel x.

When ELSxB and ELSxA are both clear, channel x is not connected to the port, and pin TCHx is available as a general-purpose I/O pin. **Table 2** shows the configuration selected by ELSxB and ELSxA in input capture mode. Reset clears the ELSxB and ELSxA bits.

NOTE: *Before enabling the channel register for input capture, make sure that the PTE/TCHx pin is stable for a minimum of two bus clocks.*

Table 2. Input Capture Mode and Edge Selection

MSxB: MSxA	ELSxB: ELSxA	Mode	Configuration
00	00	Output preset	TCHx pin under port control; not used in IC mode
00	01	Input capture	Capture on rising edge only
00	10	Input capture	Capture on falling edge only
00	11	Input capture	Capture on rising or falling edge

TOVx — Toggle on Overflow

This bit is unused for input capture operation.

CHxMAX — PWM 100% duty cycle

This bit is unused for input capture operation.

Timer Channel Registers

These read/write registers are used as the 16-bit input capture register latches for input capture functions. These registers latch the value of TCNT when a specified transition is detected on the corresponding input capture pin. The state of these registers is indeterminate after reset.

In input capture mode (MSxB:MSxA = 0:0), reading the high byte of the channel register (TCHxH) inhibits input captures until the low byte (TCHxL) is read. This prevents another input capture from overwriting the TCHxH:TCHxL registers before the previous value has been read. An overwrite of TCHxH:TCHxL will occur if another IC is received before the TCHxH is read. The LDHX instruction may be used to read these registers.

NOTE: *Writing to the timer channel registers will overwrite any input capture data.*

If a timer channel register is not used for an input capture function, it can be used as a storage location.

TCH0H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							
TCH0L	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							
TCH1H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							
TCH1L	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1 0	Bit
Reset:	Indeterminate after Reset							
TCH2H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							
TCH2L	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1 0	Bit
Reset:	Indeterminate after Reset							
TCH3H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							
TCH3L	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							

Figure 23. Timer Channel Register (TCH0H/L–TCH3H/L)

Unbuffered Output Compare (OC) Functions

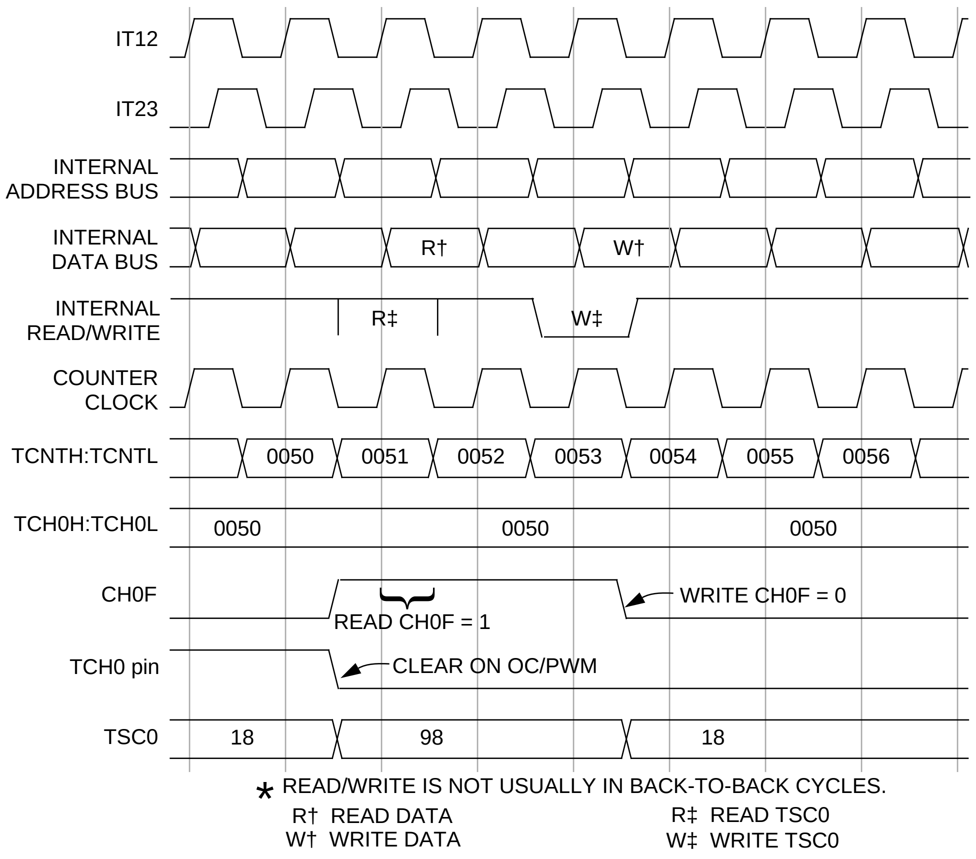
Each of the TIM output compare pins TCH0–TCH3, when used as an unbuffered output compare function, has a dedicated 16-bit compare register (TCHxH/L), a 16-bit comparator, and interrupt generation logic. The 16-bit modulo counter value, TCNT, is used as the timing reference for all unbuffered output compares. When the programmed contents of a timer channel register match TCNT, the 16-bit comparator generates an output compare match, and certain automatic actions are initiated. These automatic actions can be a hardware interrupt request and state changes at the related timer output pin.

See **Unbuffered Output Compares** for more information on the basic operation of this function.

When the output compare match occurs, a status flag CHxF is set in TSCx. **Figure 24** shows the timing of CHxF relative to the bus clocks, as well as the timing relationships for state changes described in the following paragraphs.

An output compare pin can be programmed to change states when an output compare match occurs. TSCx control bits ELSxA and ELSxB determine the output state of the pin on an output compare match. An output compare channel can be programmed to toggle, clear, or set the TCHx pin on an output compare match. See **Table 3** for the control bit configurations to select the state of an output compare pin.

If the interrupt enable bit (CHxIE) for this output compare function is set in TSCx, a CPU interrupt or DMA service request is generated on a successful output match. If the interrupt is disabled, CHxF can be polled by software to determine when an output compare match has occurred. See **CPU Interrupts** for details on interrupt operation and **DMA Service Requests** for details on service request operation.



NAME	DESCRIPTION
IT12, IT23	Bus clocks used by CPU08, DMA, TIM and all modules on MCU
TCNTH:TCNTL	16-bit value in TCNT register
TCH0H:TCH0L	Timer channel 0 data register for output compare value
CH0F	CH0F bit in TSC0 register
TCH0 pin	Timer channel 0 output pin
TSC0	Timer status and control register, channel 0

Figure 24. Unbuffered Output Compare Timing

Timer Channel Status and Control Registers

The timer channel status and control registers are 8-bit read/write registers. These registers are used to configure the timer channel to perform output compare functions. The state of these registers is reset to $00.

Each of the timer channel status and control registers does the following in unbuffered output compare mode:

- Flags output compares
- Enables output compare interrupts
- Selects initial level of TCHx output pin
- Selects unbuffered output compare mode operation
- Selects high, low, or toggling output on output compare match

TSC0	7	6	5	4	3	2	1	0
Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC1	7	6	5	4	3	2	1	0
Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC2	7	6	5	4	3	2	1	0
Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC3	7	6	5	4	3	2	1	0
Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

▨ Shading indicates this bit is not used for outout compares.

Figure 25. Timer Channel Status and Control Registers (TSC0–TSC3)

CHxF — Channel x Flag

When channel x is an output compare channel, CHxF is set when the value in the timer counter registers matches the value in the timer channel x registers.

When CPU interrupts are enabled (CHxE:DMAxS = 1:0), clear CHxF by reading the channel x status and control register with CHxF set and then writing a 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, writing 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

When DMA service requests are available and enabled (CHxE:DMAxS = 1:1), clear CHxF by reading or writing to the low byte of the channel register (TCHxL).

Writing a 1 to this bit has no effect. Reset clears the CHxF bit.

1 = Output compare on channel x
0 = No output compare on channel x

CHxIE — Channel x interrupt enable

This read/write bit enables channel x interrupts. In microcontrollers with a DMA module, the DMAxS bit in the timer DMA select register selects channel x CPU interrupts or DMA service requests. Reset clears the CHxIE bit.

1 = Channel x interrupts enabled
0 = Channel x interrupts disabled

MSxB — Mode select bit B

This bit should be cleared for unbuffered output compare operation. MSxB exists only in the channel 0 and channel 2 status and control registers, TSC0 and TSC2.

This read/write bit selects buffered OC or buffered PWM operation. Setting MS0B disables the channel 1 status and control register, and reverts TCH1 to general-purpose I/O.

Setting MS2B disables the channel 3 status and control register, and reverts TCH3 to general-purpose I/O. Reset clears the MSxB bit.

1 = Buffered OC/PWM mode enabled
0 = Buffered OC/PWM mode disabled

MSxA — Mode Select bit A

This bit has different functions, depending on the state of ELSxB and ELSxA.

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin. To select the level of your initial output, write the appropriate value to MSxA while ELSxB:A = 00. Then configure all bits in the TSCx as required for your application.

1 = Initial output level low
0 = Initial output level high

When ELSxB:A ≠ 00, this read/write bit selects input capture mode or unbuffered OC/PWM mode. **This bit should be set for unbuffered output compare operation.** MS0A and MS1A are active only when MS0B = 0. MS2A and MS3A are active only when MS2B = 0. Reset clears the MSxA bit.

1 = Unbuffered OC/PWM operation
0 = Input capture operation

NOTE: *Stop and reset the TIM before changing a channel function. Before writing to the mode select bits (MSxB and MSxA), set the timer stop and timer reset bits (TSTOP and TRST) in the TSC register.*

ELSxB and ELSxA — Edge/Level Select bits

When channel x is an output compare channel, ELSxB and ELSxA control the channel x output behavior when an output compare occurs.

When ELSxB and ELSxA are both clear, channel x is not connected to the port, and pin TCHx is available as a general-purpose I/O pin. **Table 3** shows the configuration selected by ELSxB and ELSxA in unbuffered output compare mode. Reset clears the ELSxB and ELSxA bits.

Table 3. Unbuffered Output Compare Mode and Level Selection

MSxB: MSxA	ELSxB: ELSxA	Mode	Configuration
X0	00	Output preset	Set initial output level high
X1	00	Output preset	Set initial output level low
01	00	Unbuffered OC/PWM	TCHx pin under port control; set initial output level low.
01	01	Unbuffered OC/PWM	Toggle output on OC match
01	10	Unbuffered OC/PWM	Clear output on OC match
01	11	Unbuffered OC/PWM	Set output on OC match

TOVx — Toggle on Overflow

This bit should be cleared for unbuffered output compare operation.

NOTE: *Although not typically used for output compare functions, this bit is active and will affect the operation of the TIM in output compare mode. Refer to **Timer Channel Status and Control Registers**.*

CHxMAX — PWM 100% duty cycle

This bit should be cleared for unbuffered output compare operation.

NOTE: *Although not typically used for output compare functions, this bit is active and will affect the operation of the TIM in output compare mode. Refer to **Timer Channel Status and Control Registers**.*

Timer Channel Registers

These read/write registers are used as the 16-bit compare register for output compare functions. These registers contain the output compare value for the output compare function. The state of the channel registers after reset is unknown.

In unbuffered output compare mode (MSxB:MSxA = 0:1), output compares are inhibited between writes to TCHxH and TCHxL. This prevents another output compare from occurring until the new output compare value is written. The STHX instruction can be used to write to TCHxH:TCHxL.

If a timer channel register is not used for an output compare function, it can be used as a storage location.

TCH0H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							
TCH0L	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							
TCH1H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							
TCH1L	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							
TCH2H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							
TCH2L	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							
TCH3H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							
TCH3L	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1 0	Bit
Reset:	Indeterminate after Reset							

Figure 26. Timer Channel Registers (TCH0H/L–TCH3H/L)

Buffered Output Compare (OC) Functions

TIM output compare pins TCH0 and TCH2 may be used as buffered output compare functions. This is accomplished by linking two unbuffered output compare channels together to form one buffered output compare channel. When used as buffered output compare pins, TCH0 and TCH2 have two dedicated 16-bit compare registers (TCHxH/L), two 16-bit comparators, and interrupt generation logic. The 16-bit modulo counter value, TCNT, is used as the timing reference for all unbuffered PWM functions. When the programmed contents of the active timer channel register matches TCNT, the active 16-bit comparator generates an output compare match, and certain automatic actions are initiated for that output compare function. These automatic actions can be a hardware interrupt request and state changes at the related timer output pin.

See **Buffered Output Compares** for information on the basic operation of this function.

When the output compare match occurs, a status flag CHxF is set in TSCx. **Figure 27** shows the timing of CHxF relative to the bus clocks, as well as the timing relationships for state changes described in the following paragraphs.

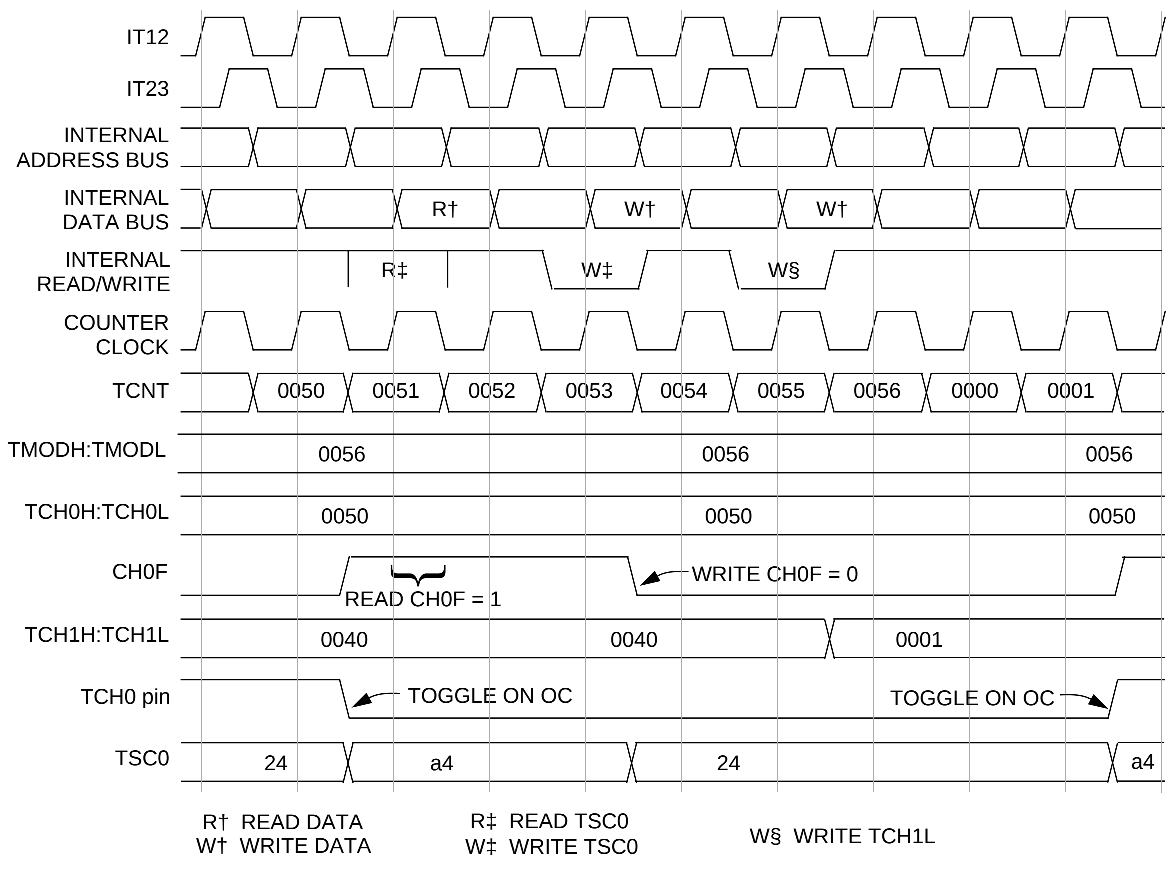
An output compare pin can be programmed to change states when an output compare match occurs. TSCx control bits ELSxA and ELSxB determine the output state of the pin on an output compare match. An output compare channel can be programmed to toggle, clear, or set the TCHx pin on an output compare match. See **Table 4** for the control bit configurations to select the state of an output compare pin.

If the interrupt enable bit (CHxIE) for this output compare function is set in TSCx, a CPU interrupt is generated on a successful output match. If the interrupt is disabled, CHxF can be polled by software to determine when an output compare match has occurred. See **CPU Interrupts** for details on interrupt operation.

NOTE: *DMA service requests are not available in buffered output compare mode.*

To use TCH0 as a buffered output compare pin, set the MS0B bit in TSC0. This will disable TCH1 as a TIM pin, and it will revert to port control. TSC0 controls and monitors the buffered output compare function, and TSC1 is unused. The channel 0 registers, TCH0H:TCH0L, initially control the output compare value on the TCH0 pin. Writing to the channel 1 registers enables the channel 1 registers to synchronously control the output compare value at the beginning of the next counter period. At each subsequent overflow, the channel registers (0 or 1) last written to control the output compare value. See **Figure 27**.

To use TCH2 as a buffered output compare pin, set the MS2B bit in TSC2. This will disable TCH3 as a TIM pin, and it will revert to port control. TSC2 controls and monitors the buffered output compare function, and TSC3 is unused. The channel 2 registers, TCH2H:TCH2L, initially control the output compare value on the TCH2 pin. Writing to the channel 3 registers enables the channel 3 registers to synchronously control the output compare value at the beginning of the next PWM period. At each subsequent overflow, the channel registers (2 or 3) last written to control the output compare value.



NAME	DESCRIPTION
IT12, IT23	Bus clocks used by CPU08, DMA, TIM and all modules on MCU
TCNTH:TCNTL	16-bit value in TCNT register
TCH0H:TCH0L	Timer channel 0 data register for output compare value
CH0F	CH0F bit in TSC0 register
TCH1H:TCH1L	Timer channel 1 data register for output compare value
TCH0 pin	Timer channel 0 output pin
TSC0	Timer status and control register, channel 0

Figure 27. Buffered Output Compare Timing

NOTE: *In buffered output compare mode, do not write new compare values to the currently active channel registers. The software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered output compare signals.*

Timer Channel Status and Control Registers

The timer channel status and control registers are 8-bit read/write registers. These registers are used to configure the timer channel to perform output compare functions. The state of these registers is reset to $00.

Each of the timer channel status and control registers does the following in buffered output compare mode:

- Flags output compares
- Enables output compare interrupts
- Selects initial level of TCHx output pin
- Selects buffered output compare mode of operation
- Selects high, low, or toggling output on output compare match

TSC0	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH0F	CH0IE	MS0B	MS0A ▨	ELS0B	ELS0A	TOV0 ▨	CH0MAX ▨
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC2	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH2F	CH2IE	MS2B	MS2A ▨	ELS2B	ELS2A	TOV2 ▨	CH2MAX ▨
Write:	0							
Reset:	0	0	0	0	0	0	0	0

▨ Shading indicates this bit is not used for output compares.

Figure 28. Timer Channel Status and Control Register (TSC0 and TSC2)

CHxF — Channel x flag

When channel x is a buffered output compare channel, CHxF is set when the value in the timer counter registers matches the value in the timer channel x registers.

When CPU interrupts are enabled (CHxE:DMAxS = 1:0), clear CHxF by reading the channel x status and control register with CHxF set and writing a 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

DMA service requests may not be used with buffered output compare functions.

Writing a 1 to this bit has no effect. Reset clears the CHxF bit.
1 = Output compare on channel x
0 = No output compare on channel x

CHxIE — Channel x interrupt enable

This read/write bit enables channel x interrupts. **In microcontrollers with a DMA module, the DMAxS bit in the timer DMA select register should be cleared to select channel x CPU interrupts.** DMA service requests cannot be used with buffered OC mode. Reset clears the CHxIE bit.
1 = Channel x interrupts enabled
0 = Channel x interrupts disabled

MSxB — Mode select bit B

This bit should be set for buffered OC operation. MSxB exists only in the channel 0 and channel 2 status and control registers, TSC0 and TSC2.

This read/write bit selects buffered OC or buffered PWM operation. Setting MS0B disables the channel 1 status and control register, and reverts TCH1 to general-purpose I/O. Setting MS2B disables the channel 3 status and control register, and reverts TCH3 to general-purpose I/O. Reset clears the MSxB bit.
1 = Buffered OC/PWM mode enabled
0 = Buffered OC/PWM mode disabled

MSxA — Mode select bit A

This bit has different functions, depending on the state of ELSxB and ELSxA.

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin. To select the level of your initial output, write the appropriate value to MSxA while ELSxB:A = 00. Then configure all bits in the TSCx as required for your application.

1 = Initial output level low
0 = Initial output level high

When ELSxB:A ≠ 00, **this bit is unused for buffered OC operation**.

NOTE: *Stop and reset the TIM before changing a channel function. Before writing to the mode select bits (MSxB and MSxA), set the timer stop and timer reset bits (TSTOP and TRST) in the TSC register.*

ELSxB and ELSxA — Edge/level select bits

When channel x is a buffered output compare channel, ELSxB and ELSxA control the channel x output behavior when an output compare occurs.

When ELSxB and ELSxA are both clear, channel x is not connected to the port, and pin TCHx is available as a general-purpose I/O pin. Table 4 shows the configuration selected by ELSxB and ELSxA in buffered output compare mode. Reset clears the ELSxB and ELSxA bits.

Table 4. Buffered Output Compare Mode and Level Selection

MSxB: MSxA	ELSxB: ELSxA	Mode	Configuration
X0	00	Output preset	Set initial output level high
X1	00	Output preset	Set initial output level low
1X	00	Buffered OC/PWM	TCHx pin under port control; set initial output level.
1X	01	Buffered OC/PWM	Toggle output on OC match
1X	10	Buffered OC/PWM	Clear output on OC match
1X	11	Buffered OC/PWM	Set output on OC match

TOVx — Toggle on overflow

This bit should be cleared for buffered output compare operation.

NOTE: *Although not typically used for output compare functions, this bit is active and will affect the operation of the TIM in output compare mode. See* ***Timer Channel Status and Control Registers****.*

CHxMAX — PWM 100% duty cycle

This bit should be cleared for buffered output compare operation.

NOTE: *Although not typically used for output compare functions, this bit is active and will affect the operation of the TIM in output compare mode. See* ***Timer Channel Status and Control Registers****.*

Timer Channel Registers

These read/write registers are used as the 16-bit compare register for output compare functions. These registers contain the output compare value for the output compare function. The state of the channel registers after reset is unknown.

In buffered output compare mode (MSxB = 1), output compares are inhibited between writes to TCHxH and TCHxL of the active channel. This prevents another output compare from occurring until the new output compare value is written. Output compares are allowed between writes to TCHxH and TCHxL of the inactive channel. The STHX instruction can be used to write to TCHxH:L.

If a timer channel register is not used for an output compare function, it can be used as a storage location.

TCH0H	Bit 15	14	13	12	11	10	9	Bit 8
Read:	Bit 15	14	13	12	11	10	9	Bit 8
Write:								
Reset:	Indeterminate after Reset							
TCH0L	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	6	5	4	3	2	1	Bit 0
Write:								
Reset:	Indeterminate after Reset							
TCH1H	Bit 15	14	13	12	11	10	9	Bit 8
Read:	Bit 15	14	13	12	11	10	9	Bit 8
Write:								
Reset:	Indeterminate after Reset							
TCH1L	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	6	5	4	3	2	1	Bit 0
Write:								
Reset:	Indeterminate after Reset							
TCH2H	Bit 15	14	13	12	11	10	9	Bit 8
Read:	Bit 15	14	13	12	11	10	9	Bit 8
Write:								
Reset:	Indeterminate after Reset							
TCH2L	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	6	5	4	3	2	1	Bit 0
Write:								
Reset:	Indeterminate after Reset							
TCH3H	Bit 15	14	13	12	11	10	9	Bit 8
Read:	Bit 15	14	13	12	11	10	9	Bit 8
Write:								
Reset:	Indeterminate after Reset							
TCH3L	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	6	5	4	3	2	1	Bit 0
Write:								
Reset:	Indeterminate after Reset							

Figure 29. Timer Channel Registers (TCH0H/L–TCH3H/L)

Unbuffered Pulse Width Modulation (PWM) Functions

Each of the TIM output compare pins TCH0–TCH3, when used as an unbuffered PWM function, has a dedicated 16-bit compare register (TCHxH/L), a 16-bit comparator, and interrupt generation logic. The 16-bit modulo counter value, TCNT, is used as the timing reference for all unbuffered PWM functions. When the programmed contents of a timer channel register match TCNT, the 16-bit comparator generates a PWM match, and certain automatic actions are initiated. These automatic actions can be a hardware interrupt request and state changes at the related timer output pin.

When generating an unbuffered PWM signal, the value in the timer channel registers determines the pulse width of the PWM signal, and the value of the timer modulo registers determines the period of the PWM signal. Refer to **Figure 30**. The toggle on overflow feature links the overflow of the 16-bit modulo counter to the unbuffered PWM channel. See **Pulse-Width Modulation (PWM) Concepts** and **Unbuffered PWM Signal Generation** for more information on the basic operation of this function.

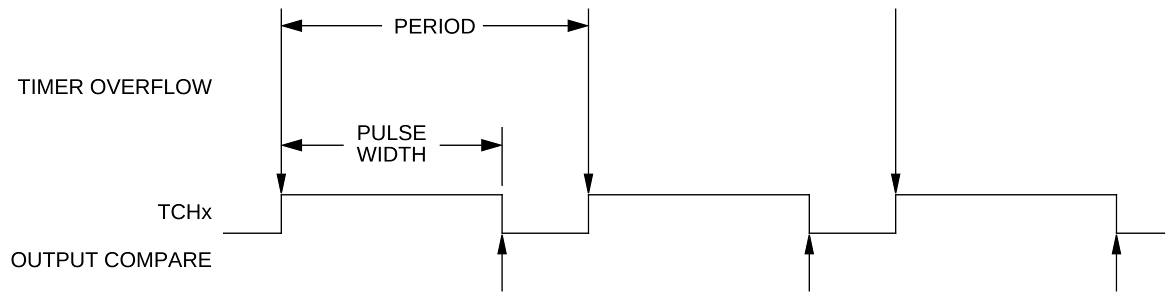


Figure 30. PWM Period and Pulse Width

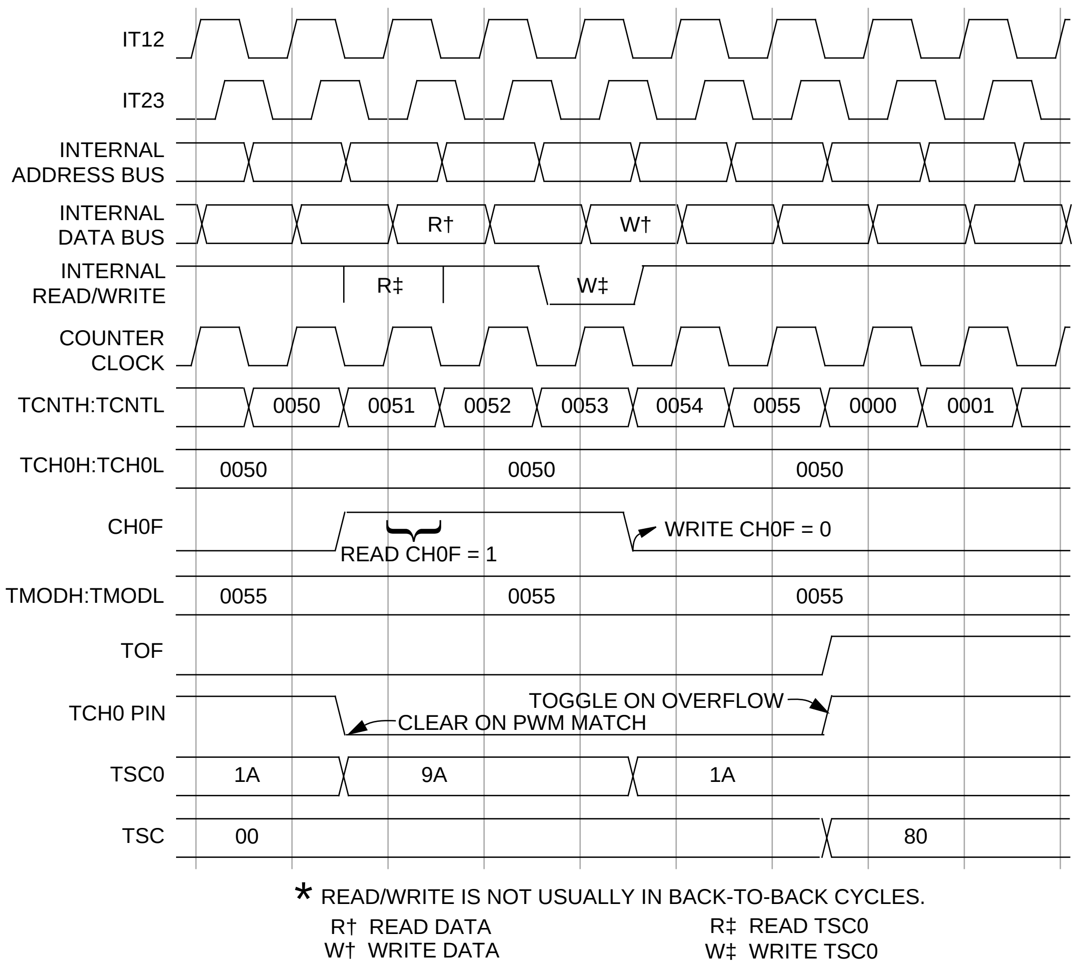
When the PWM match occurs, a status flag CHxF is set in TSCx. **Figure 31** shows the timing of CHxF relative to the bus clocks, as well as the timing relationships for state changes and interrupts described in the following paragraphs.

A PWM pin can be programmed to change states when a PWM match occurs, thereby determining the pulse width of the PWM signal. TSCx control bits ELSxA and ELSxB determine the output state of the pin on a PWM match. An unbuffered PWM channel can be programmed to toggle, clear, or set the TCHx pin on a PWM match. **Table 5** for the control bit configurations to select the state of a PWM signal pin.

If the interrupt enable bit (CHxIE) for this PWM function is set in TSCx, a CPU interrupt or DMA service request is generated on a PWM match. If the interrupt is disabled, CHxF can be polled by software to determine when a match has occurred. See **CPU Interrupts** for details on interrupt operation and See **DMA Service Requests** for details on service request operation.

To generate a signal with 100% duty cycle, use the CHxMAX bit in the TSCx. To generate a signal with 0% duty cycle, program the PWM channel to either set or clear (not toggle) on PWM match, then clear the TOVx bit to start a 0% duty cycle signal.

NOTE: *In PWM signal generation, do not program the PWM channel to toggle on PWM match. Toggling on PWM match prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on PWM match can also cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*



NAME	DESCRIPTION
IT12, IT23	Bus clocks used by CPU08, DMA, TIM and all modules on MCU
TCNTH:TCNTL	16-bit value in TCNT register
TCH0H:TCH0L	Timer channel 0 data register for output compare value
CH0F	CH0F bit in TSC0 register
TMODH:TMODL	Timer counter modulo register
TOF	Timer counter overflow flag (TOF) bit in TSC register
TCH0 pin	Timer channel 0 output pin
TSC0	Timer status and control register, channel 0
TSC	Timer status and control register

Figure 31. Unbuffered PWM Timing

Timer Channel Status and Control Registers

The timer channel status and control registers are 8-bit read/write registers. These registers are used to configure the timer channel to perform PWM functions. The state of these registers is reset to $00.

Each of the timer channel status and control registers does the following in unbuffered PWM mode:

- Flags pulse width matches
- Enables PWM interrupts
- Selects initial level of TCHx output pin
- Selects unbuffered PWM operation
- Selects high, low, or toggling output on PWM match
- Selects output toggling on timer overflow
- Selects 100% PWM duty cycle

TSC0	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC1	Bit 7	6	5	4	3	2	1	0
Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC2	Bit 7	6	5	4	3	2	1	0
Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC3	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Figure 32. Timer Channel Status and Control Registers (TSC0–TSC3)

CHxF — Channel x flag

When channel x is a PWM channel, CHxF is set when the value in the timer counter registers matches the value in the timer channel x registers.

When CPU interrupts are enabled (CHxE:DMAxS = 1:0), clear CHxF by reading the channel x status and control register with CHxF set and then writing a 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

When DMA service requests are available and enabled (CHxE:DMAxS = 1:1), clear CHxF by reading or writing to the low byte of the channel register (TCHxL).

Writing a 1 to this bit has no effect. Reset clears the CHxF bit.
1 = PWM match on channel x
0 = No PWM match on channel x

CHxIE — Channel x interrupt enable

This read/write bit enables channel x interrupts. In microcontrollers with a DMA module, the DMAxS bit in the timer DMA select register selects channel x CPU interrupts or DMA service requests. Reset clears the CHxIE bit.
1 = Channel x interrupts enabled
0 = Channel x interrupts disabled

MSxB — Mode select bit B

This bit should be cleared for unbuffered PWM operation. MSxB exists only in the channel 0 and channel 2 status and control registers, TSC0 and TSC2.

This read/write bit selects buffered OC or buffered PWM operation. Setting MS0B disables the channel 1 status and control register and reverts TCH1 to general-purpose I/O. Setting MS2B disables the channel 3 status and control register and reverts TCH3 to general-purpose I/O. Reset clears the MSxB bit.
1 = Buffered OC/PWM mode enabled
0 = Buffered OC/PWM mode disabled

MSxA — Mode select bit A

This bit has different functions, depending on the state of ELSxB and ELSxA.

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin. To select the level of your initial output, write the appropriate value to MSxA while ELSxB:A = 00. Then configure all bits in the TSCx as required for your application.
1 = Initial output level low
0 = Initial output level high

When ELSxB:A ≠ 00, this read/write bit selects input capture mode or unbuffered OC/PWM mode. **This bit should be set for unbuffered PWM operation.** MS0A and MS1A are active only when MS0B = 0. MS2A and MS3A are active only when MS2B = 0. Reset clears the MSxA bit.

1 = Unbuffered OC/PWM operation
0 = Input capture operation

NOTE: *Stop and reset the TIM before changing a channel function. Before writing to the mode select bits (MSxB and MSxA), set the timer stop and timer reset bits (TSTOP and TRST) in the TSC register.*

ELSxB and ELSxA — Edge/level select bits

When channel x is a PWM channel, ELSxB and ELSxA control the channel x output behavior when a pulse width match occurs.

When ELSxB and ELSxA are both clear, channel x is not connected to the port, and pin TCHx is available as a general-purpose I/O pin. **Table 5** shows the configuration selected by ELSxB and ELSxA in unbuffered PWM mode. Reset clears the ELSxB and ELSxA bits.

Table 5. Unbuffered PWM Mode and Level Selection

MSxB: MSxA	ELSxB: ELSxA	Mode	Configuration
X0	00	Output preset	Set initial output level high
X1	00	Output preset	Set initial output level low
01	00	Unbuffered OC/PWM	TCHx pin under port control; set initial output level low.
01	01	Unbuffered OC/PWM	Toggle output on PWM match
01	10	Unbuffered OC/PWM	Clear output on PWM match
01	11	Unbuffered OC/PWM	Set output on PWM match

TOVx — Toggle on overflow

When channel x is a buffered or unbuffered OC/PWM channel, this read/write bit controls the behavior of the channel x output when the timer counter overflows. Reset clears the TOVx bit

1 = Channel x pin toggles on timer counter overflow
0 = Channel x pin does not toggle on timer counter overflow

NOTE: *When TOVx is set, a timer counter overflow takes precedence over a channel x output compare if both occur at the same time.*

CHxMAX — PWM 100% duty cycle

This read/write bit forces the duty cycle of buffered and unbuffered OC/PWM signals to 100%. As **Figure 33** shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared. CHxMAX affects only the logic level of the channel x pin; pulse width matches can continue to occur and set the channel x flag. Reset clears the CHxMAX bit.

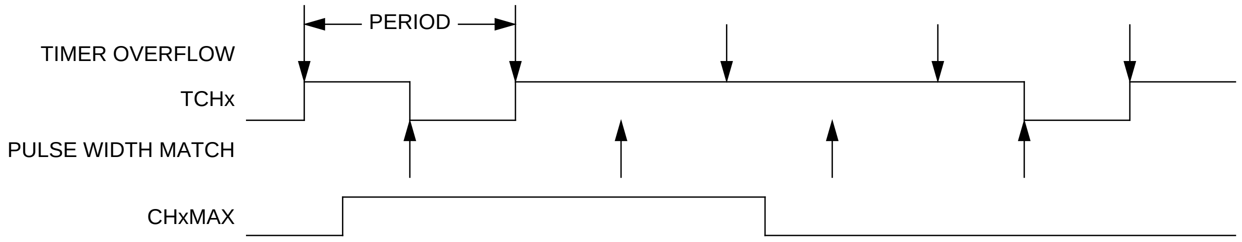


Figure 33. CHxMAX Latency

Timer Channel Registers

These read/write registers are used as the 16-bit compare register for PWM functions. These registers contain the pulse width match value for the PWM function. The state of the channel registers after reset is unknown.

In unbuffered PWM mode (MSxB:MSxA = 0:1), pulse width matches are inhibited between writes to TCHxH and TCHxL. This prevents another match from occurring until the new pulse width value is written. The STHX instruction can be used to write to TCHxH:L.

If a timer channel register is not used for a PWM function, it can be used as a storage location.

TCH0H	Bit 15	14	13	12	11	10	9	Bit 8
Read: / Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							

TCH0L	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							

TCH1H	Bit 15	14	13	12	11	10	9	Bit 8
Read: / Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							

TCH1L	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							

TCH2H	Bit 15	14	13	12	11	10	9	Bit 8
Read: / Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							

TCH2L	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							

TCH3H	Bit 15	14	13	12	11	10	9	Bit 8
Read: / Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							

TCH3L	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							

Figure 34. Timer Channel Registers (TCH0H/L–TCH3H/L)

Buffered Pulse Width Modulation (PWM) Functions

TIM pins TCH0 and TCH2 may be used as a buffered PWM function. This is accomplished by linking two unbuffered PWM channels together to form one buffered PWM channel. When used as buffered PWM pins, TCH0 and TCH2 each have two dedicated 16-bit compare registers (TCHxH/L), two 16-bit comparators, and interrupt generation logic. The 16-bit modulo counter value, TCNT, is used as the timing reference for all buffered PWM functions. When the programmed contents of a timer channel register match TCNT, the 16-bit comparator generates a PWM match, and certain automatic actions are initiated. These automatic actions can be a hardware interrupt request and state changes at the related timer output pin.

When generating a buffered PWM signal, the value in the timer channel registers determines the pulse width of the PWM signal, and the value of the timer modulo registers determines the period of the PWM signal. Refer to **Figure 30**. The toggle on overflow feature links the overflow of the 16-bit modulo counter to the buffered PWM channel. See **Pulse-Width Modulation (PWM) Concepts** and **Buffered PWM Signal Generation** for more information on the basic operation of this function.

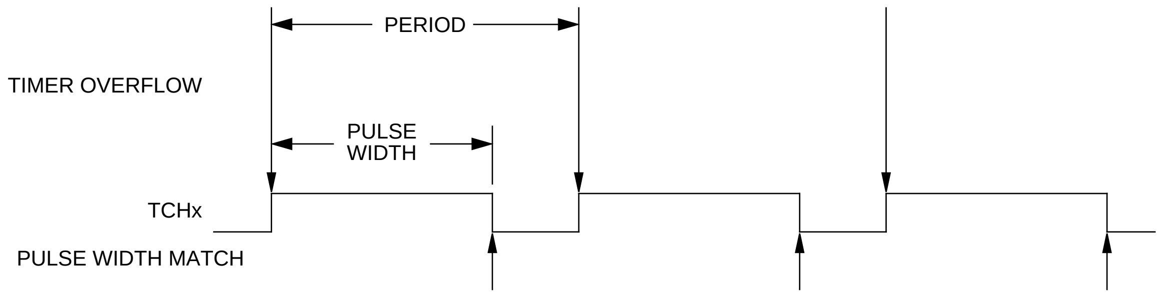


Figure 35. PWM Period and Pulse Width

When the PWM match occurs, a status flag CHxF is set in TSCx. **Figure 36**, in the previous section, shows the timing of CHxF relative to the bus clocks, as well as the timing relationships for state changes and interrupts described in the following paragraphs.

A PWM pin can be programmed to change states when a PWM match occurs, thereby determining the pulse width of the PWM signal. TSCx control bits ELSxA and ELSxB determine the output state of the pin on a PWM match. An unbuffered PWM channel can be programmed to toggle, clear, or set the TCHx pin on a PWM match. **Table 6** for the control bit configurations to select the state of a PWM signal pin.

If the interrupt enable bit (CHxIE) for this PWM function is set in TSCx, a CPU interrupt is generated on a PWM match. If the interrupt is disabled, CHxF can be polled by software to determine when a match has occurred. See **CPU Interrupts** for details on interrupt operation.

NOTE: *DMA service requests are not available in buffered PWM mode.*

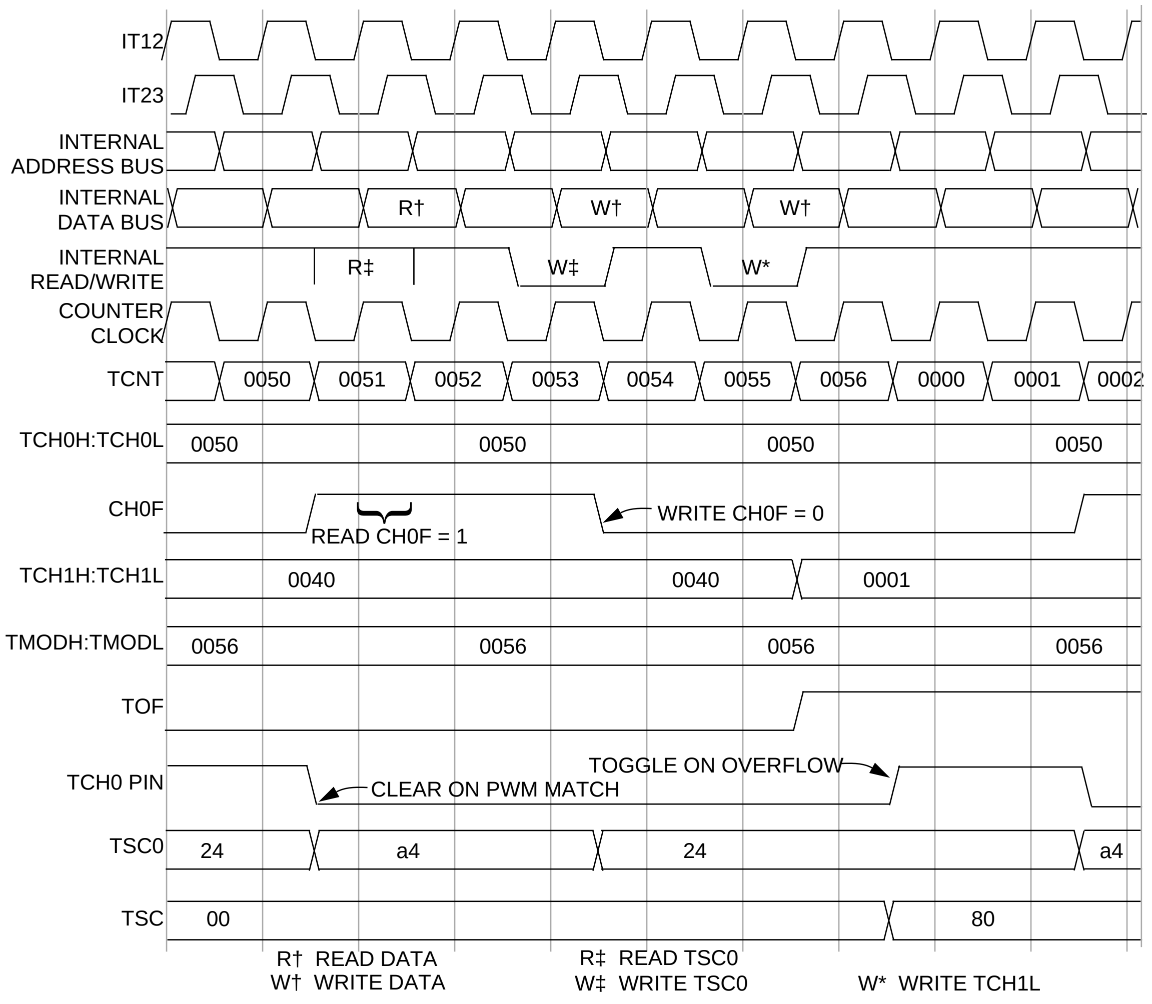
To generate a signal with 100% duty cycle, use the CHxMAX bit in the TSCx. To generate a signal with 0% duty cycle, program the PWM channel to either set or clear (not toggle) on PWM match, then clear the TOVx bit to start a 0% duty cycle signal.

NOTE: *In PWM signal generation, do not program the PWM channel to toggle on PWM match. Toggling on PWM match prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on PWM match can also cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

To use TCH0 as a buffered PWM pin, set the MS0B bit in TSC0. This will disable TCH1 as a TIM pin, and it will revert to port control. TSC0 controls and monitors the buffered PWM function, and TSC1 is unused. The channel 0 registers, TCH0H:TCH0L, initially control the pulse width on the TCH0 pin. Writing to the channel 1 registers enables the channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the channel registers (0 or 1) last written to control the pulse width. **Figure 36**.

To use TCH2 as a buffered PWM pin, set the MS2B bit in TSC2. This will disable TCH3 as a TIM pin, and it will revert to port control. TSC2 controls and monitors the buffered PWM function, and TSC3 is unused. The channel 2 registers, TCH2H:TCH2L, initially control the pulse width on the TCH2 pin. Writing to the channel 3 registers enables the channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the channel registers (2 or 3) last written to control the pulse width.

NOTE: *In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. The software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered PWM signals.*



NAME	DESCRIPTION
IT12, IT23	Bus clocks used by CPU08, DMA, TIM and all modules on MCU
TCNTH:TCNTL	16-bit value in TCNT register
TCH0H:TCHOL	Timer channel 0 data register for output compare value
CH0F	CH0F bit in TSC0 register
TMODH:TMODL	Timer counter modulo register
TOF	Timer counter overflow flag (TOF) bit in TSC register
TCH0 pin	Timer channel 0 output pin
TSC0	Timer status and control register, channel 0
TSC	Timer status and control register

Figure 36. Buffered PWM Timing

Timer Channel Status and Control Registers

The timer channel status and control registers are 8-bit read/write registers. These registers are used to configure the timer channel to perform PWM functions. The state of these registers is reset to $00.

Each of the timer channel status and control registers does the following in buffered PWM mode:

- Flags pulse width matches
- Enables PWM interrupts
- Selects initial level of TCHx output pin
- Selects buffered PWM operation
- Selects high, low, or toggling output on PWM match
- Selects output toggling on timer overflow
- Selects 100% PWM duty cycle

TSC0	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
Write	0							
Reset:	0	0	0	0	0	0	0	0

TSC2	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Figure 37. Timer Channel Status and Control Registers (TSC0, TSC2)

CHxF — Channel x flag

When channel x is a buffered PWM channel, CHxF is set when the value in the timer counter registers matches the value in the timer channel x registers.

When CPU interrupts are enabled (CHxE:DMAxS = 1:0), clear CHxF by reading the channel x status and control register with CHxF set and then writing a 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, writing 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

DMA service requests may not be used with buffered PWM functions.

Writing a one to this bit has no effect. Reset clears the CHxF bit.
1 = PWM match on channel x
0 = No PWM match on channel x

CHxIE — Channel x interrupt enable

This read/write bit enables channel x interrupts. **In microcontrollers with a DMA module, the DMAxS bit in the timer DMA select register should be cleared to select channel x CPU interrupts.** DMA service requests cannot be used with buffered PWM mode. Reset clears the CHxIE bit.
1 = Channel x interrupts enabled
0 = Channel x interrupts disabled

MSxB — Mode select bit B

This bit should be set for buffered PWM operation. MSxB exists only in the channel 0 and channel 2 status and control registers, TSC0 and TSC2.

This read/write bit selects buffered OC or buffered PWM operation. Setting MS0B disables the channel 1 status and control register, and reverts TCH1 to general-purpose I/O. Setting MS2B disables the channel 3 status and control register, and reverts TCH3 to general-purpose I/O. Reset clears the MSxB bit.
1 = Buffered OC/PWM mode enabled
0 = Buffered OC/PWM mode disabled

MSxA — Mode select bit A

This bit has different functions, depending on the state of ELSxB and ELSxA.

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin. To select the level of your initial output, write the appropriate value to MSxA while ELSxB:A = 00. Then configure all bits in the TSCx as required for your application.

1 = Initial output level low
0 = Initial output level high

When ELSxB:A ≠ 00, **this bit is unused for buffered PWM operation**.

NOTE: *Stop and reset the TIM before changing a channel function. Before writing to the mode select bits (MSxB and MSxA), set the timer stop and timer reset bits (TSTOP and TRST) in the TSC register.*

ELSxB and ELSxA — Edge/level select bits

When channel x is a buffered PWM channel, ELSxB and ELSxA control the channel x output behavior when a pulse width match occurs.

When ELSxB and ELSxA are both clear, channel x is not connected to the port, and pin TCHx is available as a general-purpose I/O pin. Table 6 shows the configuration selected by ELSxB and ELSxA in buffered PWM mode. Reset clears the ELSxB and ELSxA bits.

Table 6. Buffered PWM Mode and Level Selection

MSxB: MSxA	ELSxB: ELSxA	Mode	Configuration
X0	00	Output preset	Set initial output level high
X1	00	Output preset	Set initial output level low
1X	00	Buffered OC/PWM	TCHx pin under port control; set initial output level.
1X	01	Buffered OC/PWM	Toggle output on PWM match
1X	10	Buffered OC/PWM	Clear output on PWM match
1X	11	Buffered OC/PWM	Set output on PWM match

TOVx — Toggle on overflow

When channel x is a buffered or unbuffered OC/PWM channel, this read/write bit controls the behavior of the channel x output when the timer counter overflows. Reset clears the TOVx bit.

1 = Channel x pin toggles on timer counter overflow
0 = Channel x pin does not toggle on timer counter overflow

NOTE: *When TOVx is set, a timer counter overflow takes precedence over a channel x output compare if both occur at the same time.*

CHxMAX — PWM 100% duty cycle

This read/write bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As **Figure 38** shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared. CHxMAX affects only the logic level of the channel x pin; pulse width matches can continue to occur and set the channel x flag. Reset clears the CHxMAX bit.

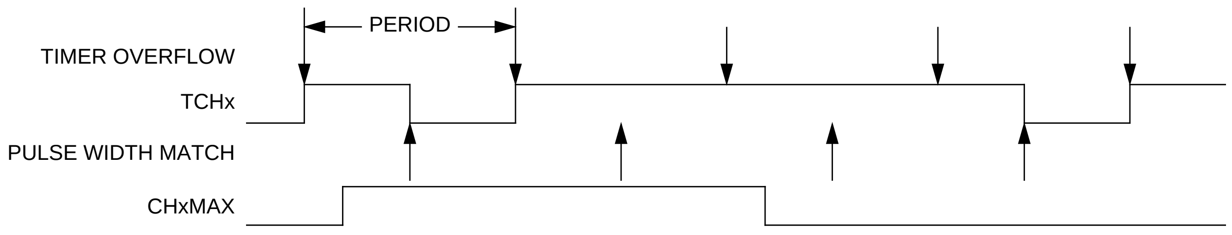


Figure 38. CHxMAX Latency

Timer Channel Registers

These read/write registers are used as the 16-bit compare register for PWM functions. These registers contain the pulse width match value for the PWM function. The state of the channel registers after reset is unknown.

In buffered PWM mode (MSxB = 1), pulse width matches are inhibited between writes to TCHxH and TCHxL of the active channel. This prevents another match from occurring until the new pulse width value is written. Output compares are allowed between writes to TCHxH and TCHxL of the inactive channel. The STHX instruction can be used to write to TCHxH:L.

If a timer channel register is not used for a PWM function, it can be used as a storage location.

TCH0H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							
TCH0L	**Bit 7**	**6**	**5**	**4**	**3**	**2**	**1**	**Bit 0**
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							
TCH1H	**Bit 15**	**14**	**13**	**12**	**11**	**10**	**9**	**Bit 8**
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							
TCH1L	**Bit 7**	**6**	**5**	**4**	**3**	**2**	**1**	**Bit 0**
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							
TCH2H	**Bit 15**	**14**	**13**	**12**	**11**	**10**	**9**	**Bit 8**
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							
TCH2L	**Bit 7**	**6**	**5**	**4**	**3**	**2**	**1**	**Bit 0**
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							
TCH3H	**Bit 15**	**14**	**13**	**12**	**11**	**10**	**9**	**Bit 8**
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							
TCH3L	**Bit 7**	**6**	**5**	**4**	**3**	**2**	**1**	**Bit 0**
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							

Figure 39. Timer Channel Registers (TCH0H/L–TCH3H/L)

Interrupts

Contents

Introduction

The TIM is capable of generating one interrupt for the timer counter overflow, and one for each of the channels. These interrupts can be masked by clearing the Interrupt Enable bit for each interrupt source. See **Timer Status and Control Register** for more information on setting up the timer counter overflow interrupt. See **Timer Channel Status and Control Registers**, **Timer Channel Status and Control Registers**, **Timer Channel Status and Control Registers**, **Timer Channel Status and Control Registers**, and **Timer Channel Status and Control Registers** for more information on setting up interrupts for each of the TIM channels.

NOTE: *The TIM can be implemented with two, four, six or eight channels. This manual will show the 4-channel version, as implemented in the MC68HC708XL36.*

Interrupts

Each interrupt can be directed to the central processor unit (CPU08), and the channel interrupts can generate service requests directed to the direct memory access (DMA) module for processing, if available. See **Timer DMA Select Register (TDMA)** for more information on selecting the interrupt destination for each timer channel.

NOTE: *DMA service requests cannot be enabled for buffered OC/PWM operation. This could cause incorrect flag clearing.*

Timer DMA Select Register (TDMA)

The timer DMA register selects either the CPU or the DMA module to service TIM interrupts. These bits are cleared on reset, selecting the CPU to process interrupts.

NOTE: *This register is available only on microcontrollers with a DMA module. If no DMA module is included, do not enable the bits described in* ***Figure 40****. If those bits are enabled, no CPU interrupts for that channel will be generated or serviced.*

NOTE: *The TIM can be implemented with two, four, six or eight channels. This manual will show the 4-channel version, as implemented in the MC68HC708XL36..*

TDMA	7	6	5	4	3	2	1	0
Read:	0	0	0	0	DMA3S	DMA2S	DMA1S	DMA0S
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 40. Timer DMA Select Register (TDMA)

DMA3S — DMA channel 3 select

This read/write bit enables the DMA to process TIM interrupts on timer channel 3. Reset clears the DMA3S bit.

1 = Timer channel 3 generates DMA service requests
0 = Timer channel 3 generates CPU interrupt requests

DMA2S — DMA channel 2 select

This read/write bit enables the DMA to process TIM interrupts on timer channel 2. Reset clears the DMA2S bit.

1 = Timer channel 2 generates DMA service requests
0 = Timer channel 2 generates CPU interrupt requests

DMA1S — DMA channel 1 select

This read/write bit enables the DMA to process TIM interrupts on timer channel 1. Reset clears the DMA1S bit.

1 = Timer channel 1 generates DMA service requests
0 = Timer channel 1 generates CPU interrupt requests

DMA0S — DMA channel 0 select

This read/write bit enables the DMA to process TIM interrupts on timer channel 0. Reset clears the DMA0S bit.

1 = Timer channel 0 generates DMA service requests
0 = Timer channel 0 generates CPU interrupt requests

CPU Interrupts

The TIM provides interrupt sources to the CPU08 through the system integration module (SIM). The SIM receives all interrupt requests for the CPU and decodes interrupt priorities when multiple requests are received. **The SIM is different on each microcontroller, and the interrupt priorities can change. A common TIM interrupt priority scheme, as implemented on the MC68HC708XL36, is described in this section.** See the applicable technical data book for the correct interrupt priority.

NOTE: *The TIM can be implemented with two, four, six or eight channels. This manual will show the 4-channel version, as implemented in the MC68HC708XL36.*

If more than one TIM interrupt is received by the SIM, the CH0 interrupt is processed first, followed by CH1, CH2, and CH3 interrupts, with the timer overflow interrupt processed last. The TIM channels will provide interrupts to the CPU if the associated DMA select bit is cleared, as discussed in **Timer DMA Select Register (TDMA)**.

Table 7. TIM Interrupt Priority

Source	Local Mask	Global Mask	Vector Address
CH0F	CH0IE	I	$FFF4–$FFF5
CH1F	CH1IE		$FFF2–$FFF3
CH2F	CH2IE		$FFF0–$FFF1
CH3F	CH3IE		$FFEE–$FFEF
TOF	TOE		$FFEC–$FFED

The interrupt flags (TOF, CH0F, CH1F, CH2F, and CH3F) are set and cleared regardless of the value of the interrupt enable bits (TOE, CH0IE, CH1IE, CH2IE and CH3IE). The interrupt flags are set when a timer overflow, input capture, output compare, or PWM match occurs. Each interrupt flag is cleared by reading the interrupt flag while it is set, and then writing a logic 0 to the flag. If another flag has been received between the read and write of the flag, the flag will not be cleared, indicating that another interrupt has occurred.

If the interrupt enable bit is set for a particular flag, a CPU interrupt will be generated.

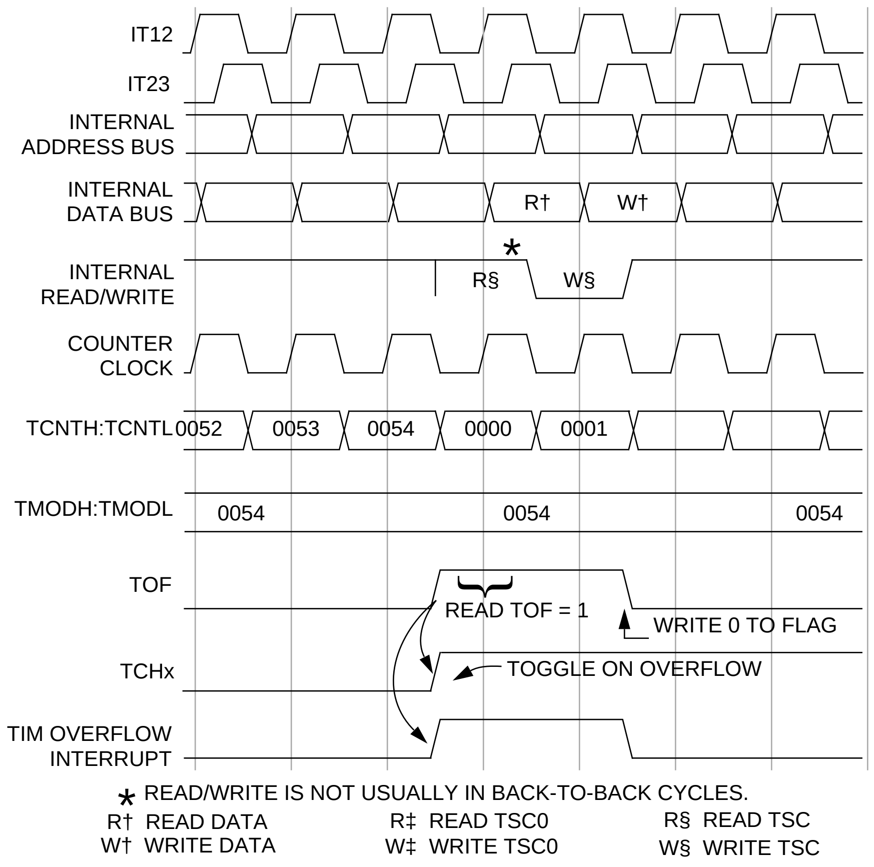
Timer Overflow Timing

Figure 41 shows the timing relationships between the MCU bus clocks, the 16-bit timer counter, and the timer overflow flag and interrupt. In this example, the modulo counter is programmed to overflow at \$0054. The counter clock is running at the fastest clock rate, which is the same frequency as IT12. The counter value changes on the falling edge of IT12. When the counter reaches the TMOD value, TOF is set at the next falling edge of IT12. When TOF is set, the TCHx pin is toggled if TOVx is set in the corresponding TSCx register. TOF also generates the timer overflow interrupt signal to the CPU08. As long as TOF = 1, the interrupt is pending. The interrupt is cleared by clearing TOF: read TOF = 1, then write TOF = 0.

Figure 41 shows another example of a timer overflow occurring. In this example, the counter clock is slower than the bus clock. Notice that on the counter overflow, the TOF is set on the falling edge of IT12 following the TMOD match. This immediately resets the timer counter to 0.

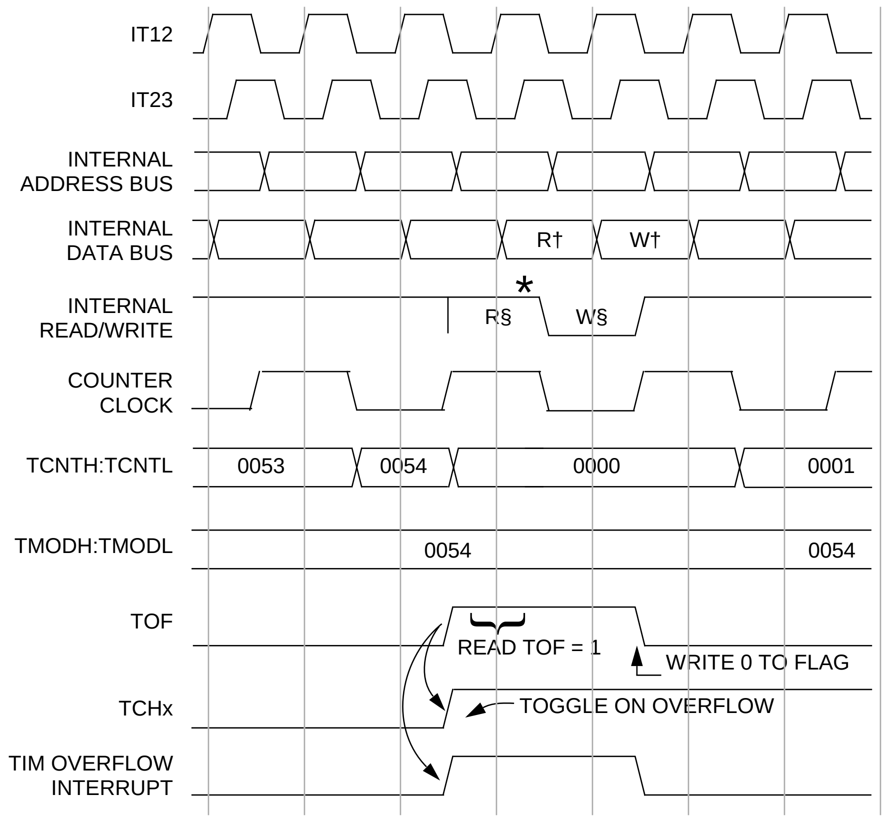
Input Capture Timing

Figure 43 shows the timing relationship between the bus clocks, the counter, and the input capture circuit for a CPU interrupts on timer channel 1. A change on the TCH1 input pin occurring before the falling edge of IT12 is recognized by the input capture circuit. On the next falling edge of IT12, CH1F is set and the channel 1 CPU interrupt signal, $t_{imintcx}$, is asserted. On the falling edge of IT23, the value of the counter is latched into TCH1. As long as CH1F = 1, the interrupt is pending. The interrupt is cleared by clearing CH1F: read CH1F = 1, then write CH1F = 0.



NAME	DESCRIPTION
IT12, IT23	Bus clocks used by CPU08, DMA, TIM and all modules on MCU
TCNTH:TCNTL	16-bit value in TCNT register
TMODH:TMODL	16-bit value in TMOD register
TOF	TOF bit in TSC register
TCHx	Timer channel pin with TOVx enabled in TSCx

Figure 41. CPU Counter Overflow Interrupt Timing Example A

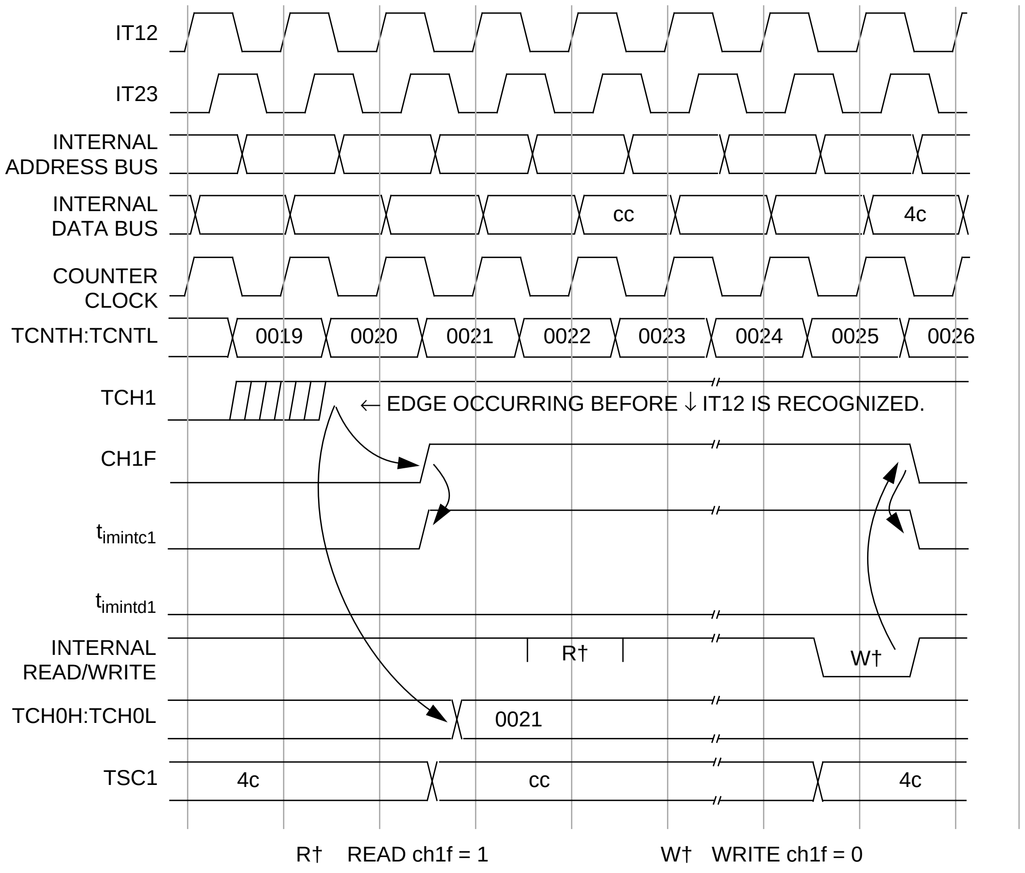


* READ/WRITE IS NOT USUALLY IN BACK-TO-BACK CYCLES.

R†	READ DATA	R§	READ TSC
W†	WRITE DATA	W§	WRITE TSC

NAME	DESCRIPTION
IT12, IT23	Bus clocks used by CPU08, DMA, TIM and all modules on MCU
TCNTH:TCNTL	16-bit value in TCNT register
TMODH:TMODL	16-bit value in TMOD register
TOF	TOF bit in TSC register
TCHx	Timer channel pin with TOVx enabled in TSCx

Figure 42. CPU Counter Overflow Interrupt Timing Example B

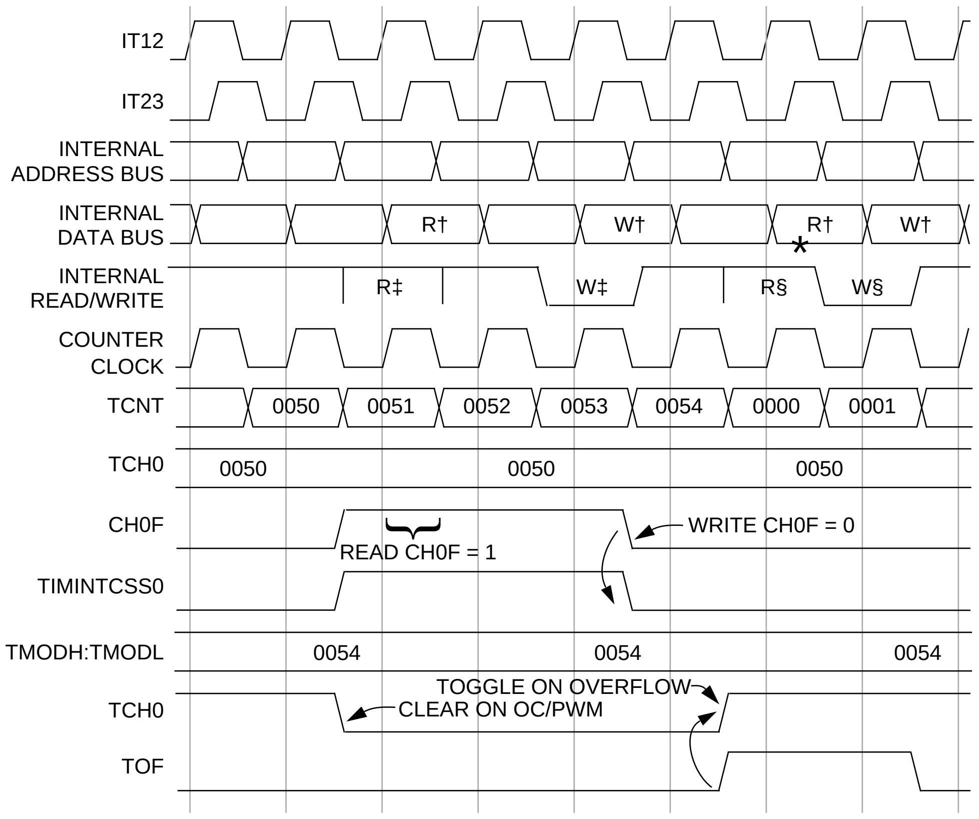


NAME	DESCRIPTION
IT12, IT23	Bus clocks used by CPU08, DMA, TIM and all modules on MCU
TCNTH:TCNTL	16-bit value in TCNT register
TCH1	Timer channel 1 input pin
CH1F	CH1F bit in TSC1 register
timintc1	TIM channel 1 CPU interrupt signal
timintd1	TIM channel 1 DMA service request signal
TCH0H:TCH0L	16-bit value in TCH0 register
TSC1	Timer status and control register, channel 1

Figure 43. CPU Input Capture Interrupt Timing Example

Output Compare/PWM Timing

Figure 44 shows the timing relationships between the MCU bus clocks, the 16-bit timer counter, and the timer output compare/PWM match flag and interrupt. In this example, the OC/PWM match is programmed to occur at $0050, and the modulo counter is programmed to overflow at $0054. The counter clock is running at the fastest clock rate, which is the same frequency as IT12. The counter value changes on the falling edge of IT12. When the counter reaches the TCH0 value ($0050), CH0F is set at the next falling edge of IT12. When CH0F is set, the TCH0 pin is cleared, as specified in the TSC0 register. CH0F also generates the TIM channel 0 CPU interrupt signal, $t_{imintc0}$, to the CPU08. While CH0F = 1, the interrupt is pending. The interrupt is cleared by clearing CH0F: read CH0F = 1, then write CH0F = 0. When the counter reaches the TMOD value, TOF is set, TCH0 toggles, and the timer overflow interrupt signal is asserted, as described in **Timer Overflow Timing**.



*READ/WRITE IS NOT USUALLY IN BACK-TO-BACK CYCLES.

R† READ DATA
W† WRITE DATA

R‡ READ TSC0
W‡ WRITE TSC0

R§ READ TSC
W§ WRITE TSC

NAME	DESCRIPTION
IT12, IT23	Bus clocks used by CPU08, DMA, TIM and all modules on MCU
TCNTH:TCNTL	16-bit value in TCNT register
TCHo	Timer status and control register, channel 0
CH0F	CH0F bit in TSC0 register
$t_{imintc0}$	TIM channel 0 CPU interrupt signal
TMODH:TMODL	16-bit value in TCH0 register
TCH0	Timer channel 0 output pin
TOF	Timer counter overflow flag in TSC

Figure 44. CPU Output Compare/PWM Interrupt Example

DMA Service Requests

NOTE: *DMA service requests are available only on microcontrollers with a DMA module. If no DMA module is included, do not enable the bits described in* ***Figure 40****. If those bits are enabled, no CPU interrupts for that channel will be generated or serviced.*

The TIM provides service requests to the DMA. The DMA service request priorities are determined by the DMA channel assignment. The TIM channels will provide service requests to the DMA if the associated DMA select bit is set, as discussed in **Timer DMA Select Register (TDMA)**.

NOTE: *The TIM can be implemented with two, four, six or eight channels. This manual will show the 4-channel version, as implemented in the MC68HC708XL36.*

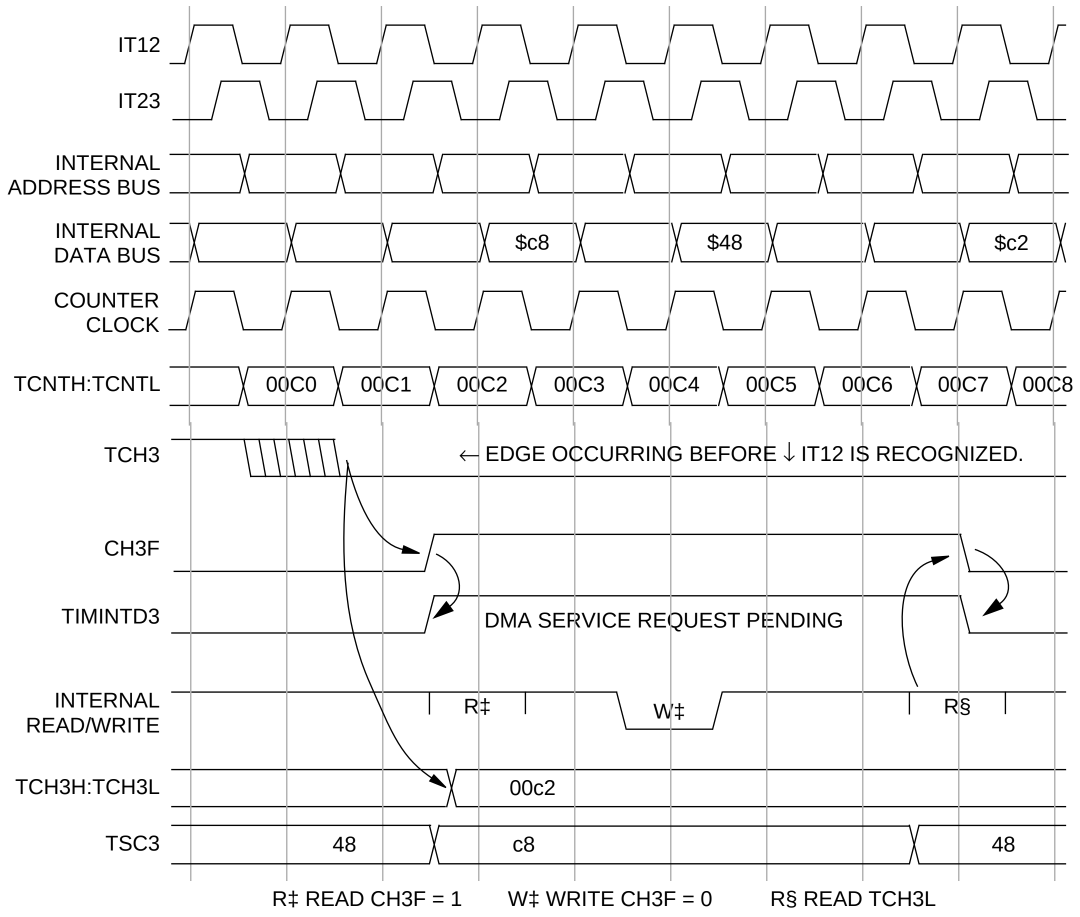
The interrupt flags (CH0F, CH1F, CH2F, and CH3F) are set and cleared regardless of the value of the interrupt enable bits (CH0IE, CH1IE, CH2IE, and CH3IE). The interrupt flags are set when an input capture or output compare occurs. Each interrupt flag is cleared by reading or writing to the low byte of the associated channel data register (TCHxL). Reading or writing the interrupt flag will have no affect on the condition of the flag.

If the interrupt enable bit is set for a particular flag, a DMA service request will be generated.

Input Capture Timing

Figure 45 shows the timing relationship between the bus clocks, the counter, and the input capture circuit for a DMA service request on timer channel 3. A change on the TCH3 input pin occurring before the falling edge of IT12 is recognized by the input capture circuit. On the next falling edge of IT12, CH3F is set and the channel 3 DMA service request signal, $t_{imintdx}$, is asserted. On the falling edge of IT23, the value of the counter is latched into TCH3. As long as CH3F = 1, the service request is pending. The service request is cleared when the DMA reads the low byte of the channel register, TCH3L. A write to TCH3L will also clear the

CH3F. Reads and writes of TSC3 do not clear the flag or pending service request.



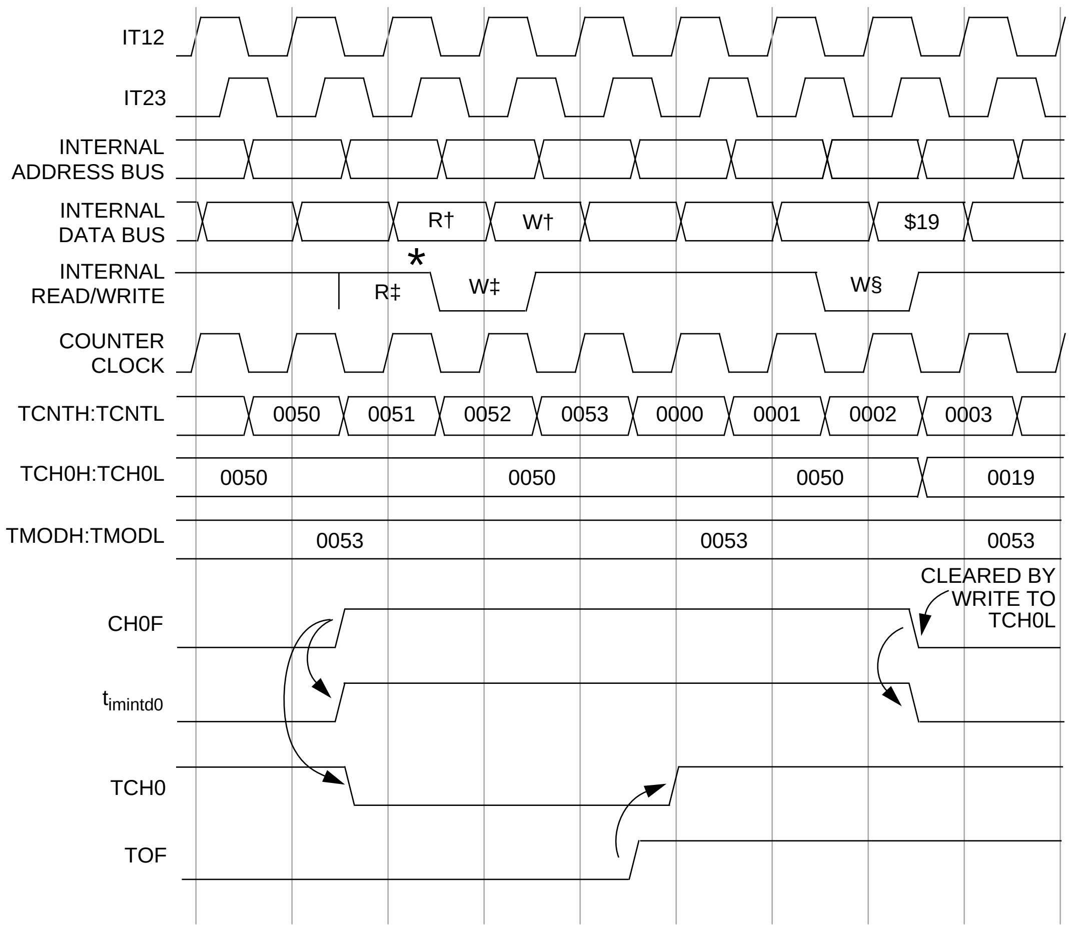
NAME	DESCRIPTION
IT12, IT23	Bus clocks used by CPU08, DMA, TIM and all modules on MCU
TCNTH:TCNTL	16-bit value in TCNT register
TCH3	Timer channel 3 input pin
CH3F	CH3F bit in TSC3 register
$t_{imintd3}$	TIM channel 3 DMA service request signal
TCH3H:TCH3L	16-bit value in TCH3 register
TSC3	Timer status and control register, channel 3

Figure 45. DMA Input Capture Service Request Timing Example

Output Compare/PWM Timing

Figure 46 shows the timing relationships between the MCU bus clocks, the 16-bit timer counter, and the timer output compare/PWM match flag and service request. In this example, the OC/PWM match is programmed to occur at $0050, and the modulo counter is programmed to overflow at $0054. The counter clock is running at the fastest clock rate, which is the same frequency as IT12. The counter value changes on the falling edge of IT12. When the counter reaches the TCH0 value ($0050), CH0F is set at the next falling edge of IT12. When CH0F is set, the TCH0 pin is cleared, as specified in the TSC0 register. CH0F also generates the TIM channel 0 DMA service request signal, timintd0, to the DMA module. While CH0F = 1, the service request is pending. The service request is cleared when the DMA writes to TCH0L. A read of TCH0L will also clear CH0F. Reads and writes of TSC3 do not clear the flag or pending service request. When the counter reaches the TMOD value, TOF is set, TCH0 toggles, and the timer overflow interrupt signal is asserted, as described in **Timer Overflow Timing**.

NOTE: *The DMA module will not service buffered OC/PWM interrupts. These bits should be cleared for both channels used in a buffered OC/PWM mode.*



* READ/WRITE IS NOT USUALLY IN BACK-TO-BACK CYCLES.

R†	READ TSC0 DATA	R‡	READ TSC0	W§	WRITE TCH0L
W†	WRITE TSC0 DATA	W‡	WRITE TSC0		

NAME	DESCRIPTION
IT12, IT23	Bus clocks used by CPU08, DMA, TIM and all modules on MCU
TCNTH:TCNTL	16-bit value in TCNT register
TCH0	Timer channel 0 output pin
CH0F	CH0F bit in TSC0 register
$t_{imintd0}$	TIM channel 0 DMA service request signal
TCH0H:TCH0L	16-bit value in TCH0 register
TSC0	Timer status and control register, channel 0

Figure 46. DMA Output Compare/PWM Service Request Example

Special Modes

Contents

Introduction

In addition to the user mode, the M68HC08 Family of microcontrollers supports several other operating modes. The function of the TIM is described for wait, stop, and monitor modes.

Wait Mode

Wait mode is entered by executing the WAIT instruction in the CPU08. Wait mode is a low-power mode in which the CPU clocks are stopped, and the bus clocks to the TIM continue to run. The TIM functions are active and all TIM counters and prescalers continue counting during wait mode. A timer interrupt may cause the CPU08 to exit wait mode if interrupts are not masked.

All TIM registers retain their states in wait mode.

If a DMA is available, it can service a TIM interrupt without causing the CPU to have to exit wait mode. This can be used to fill a buffer with input capture data or to transmit a buffer full of OC/PWM data. A DMA interrupt can be used to exit the CPU from wait and then to use or refill the buffer data as needed. This feature allows the MCU to conserve power, avoid long interrupt service latency, and potentially reduce code size due to fewer necessary interrupt service routines.

Stop Mode

Stop mode is entered by executing the STOP instruction in the CPU08. Stop mode is the lowest power-consumption mode, with the bus clocks stopped to both the CPU08 and the TIM. The MCU remains in STOP mode until the STOP bit is negated by the CPU or by a reset. All TIM counters and prescalers stop counting while the STOP bit is asserted.

All TIM registers retain their states in stop mode.

Monitor Mode

Monitor mode gives access to an on-chip monitor providing basic debugging facilities through a single wire serial interface to a host computer. The debug monitor provides basic commands to read and write the memory contents of the 68HC08 through an interface with a host computer. The monitor commands can be used to provide all the features normally found in a monitor, such as loading in user code, adding breakpoints, and modifying CPU, peripheral, and I/O register contents. If the MCU application is designed correctly, the monitor can be used to debug code in the embedded ROM part.

NOTE: *The monitor mode is different on each microcontroller, and the sequence used to enter this mode may change. monitor mode, as implemented on the MC68HC708XL36, is described in this section.*

On the MC68HC708XL36, monitor mode is entered by placing 2 x V_{DD} on pin $\overline{IRQ1}/V_{PP}$, grounding PTC1, and pulling PTC0 high. The execution of an SWI (software interrupt) instruction or the execution of a hardware reset while holding PTA0 high will start the monitor and send a break to the host computer. The communication interface with the host computer is a standard asynchronous communication interface with the transmitted and received data sent over the same wire. For more information on monitor mode, see the appropriate technical data book.

The TIM is active in monitor mode. Software loaded into RAM by the debug monitor can access TIM registers and interrupts can be enabled and generated.

Applications

Contents

Introduction

The TIM allows flexible configurations to service a variety of timing applications. Users familiar with the HC05 16-bit timer modules will discover that the TIM provides all the same features (input capture, output compare, interrupt on overflow) with additional functionality. Extra features include configurable channels, buffered/unbuffered PWM capability in conjunction with a modulo register, toggle on overflow, more control of the timer counter (stop and reset commands), and DMA servicing of the TIM on some devices.

The following applications are discussed in this section:

- General TIM Information
- PWM Controlled RC Digital to Analog Converter
- Servo Loop Motor Control
- Using the HC708XL36 DMA with the TIM

NOTE: *All software examples in this section were written for the MC68HC708XL36.*

General TIM Information

The following sections detail the basic functions of the TIM including input capture, output compare, unbuffered PWM, and buffered PWM by presenting the following applications:

- Input capture
- Output compare
- PWMs

Input Capture

The input capture function is useful for measuring pulse widths or periods of signals and for establishing functional time references (delays, state machines, etc.) to external events. The TIM allows processor interrupts or polling on rising and/or falling edges, time reference adjustment (via the modulo register), and time base adjustment (via the prescaler). The following example illustrates a general-purpose input capture routine that will measure the period of an incoming signal. **Table 8** details possible resolutions and frequency ranges for each timer prescaler. Range indicates the range of input frequencies that can be measured, as determined by the value in the modulo register. Resolution indicates the frequency at which the timer counter is incremented by one bit.

NOTE: *Higher input frequencies may be difficult to measure without using a DMA or external circuitry.*

Table 8. Range and Resolution for Period Input/Output*

Prescaler	Range (Hz)	Resolution
1	122 - 8.0 M	125 ns
2	61 - 4.0 M	250 ns
4	30.5 - 2.0 M	500 ns
8	15.3 - 1.0 M	1 µs
16	7.6 - 500 k	2 µs
32	3.8 - 250 k	4 µs
64	1.9 - 125 k	8 µs

* Assumes 8 MHz bus clock, 65K counts max

```
*
* Application example for TIM Reference Manual
*
* This software will feature the 68HC08XL36 Timer
* Interface Module in an input capture example.
* A specified channel will receive pulses or periods
* and interrupt the CPU on a specified edge.
*
* Register Equates
TSC     equ $20     ;Timer Status & Control Register $20
TDMA    equ $21     ;Timer DMA Select Register $21
TCNTH   equ $22     ;Timer Counter Register Hi $22
TCNTL   equ $23     ;Timer Counter Register Lo $23
TMODH   equ $24     ;Timer Counter Modulo Register Hi $24
TMODL   equ $25     ;Timer Counter Modulo Register Lo $25
TSC0    equ $26     ;Timer Channel 0 Status & Control Register $26
TCH0H   equ $27     ;Timer Channel 0 Register Hi $27
TCH0L   equ $28     ;Timer Channel 0 Register Lo $28
TSC1    equ $29     ;Timer Channel 1 Status & Control Register $29
TCH1H   equ $2a     ;Timer Channel 1 Register Hi $2a
TCH1L   equ $2b     ;Timer Channel 1 Register Lo $2b
TSC2    equ $2c     ;Timer Channel 2 Status & Control Register $2c
TCH2H   equ $2d     ;Timer Channel 2 Register Hi $2d
TCH2L   equ $2e     ;Timer Channel 2 Register Lo $2e
TSC3    equ $2f     ;Timer Channel 3 Status & Control Register $2f
TCH3H   equ $30     ;Timer Channel 3 Register Hi $30
TCH3L   equ $31     ;Timer Channel 3 Register Lo $31
PORTE   equ $08     ;Port E, TIM port
* Miscellaneous Equates
BIT0      equ $00
BIT1      equ $01
BIT2      equ $02
BIT3      equ $03
BIT4      equ $04
BIT5      equ $05
BIT6      equ $06
BIT7      equ $07
CHOFFSET equ $00       ;ch0 = 0 offset, ch1 = 3 offset, chx = 3x offset

* TIM status and control register bits
PRESCLR   equ $00       ;prescaler xxxxx000 (bus clock / 1)
TOE       equ $00       ;timer overflow interrupt enable x0xxxxxx

* Channel x status and control register bits
CHXIE     equ $40       ;channel interrupt enable x0xxxxxx
MODE      equ $00       ;mode select bits xx00xxxx (input capture)
EDGE      equ $04       ;edge select bits xxxx00xx (rising edge)

* Variables in RAM
        org $0050       ;start of RAM

* Application assembly code
        org $6e00       ;start of ROM
```

```
ICAP_INIT:

* set up channel x for input capture
        mov #TOE+$30+PRESCLR,TSC    ;stop and reset TIM, select prescaler, TOE
        ldx #TSC0+CHOFFSET          ;point to channel x SCR
        lda #CHXIE+MODE+EDGE
        sta ,x                      ;store channel ICAP control info

        cli                ;enable CPU interrupts (delete this line if polling)
        bclr BIT5,TSC      ;enable timer counter
        bra *              ;wait for interrupt (to poll monitor channel x flag)

* channel x interrupt service routine for ICAPx

CHISR   equ *
        pshh                     ;save h register
        lda TSC0+CHOFFSET        ;read channel x flag
        bclr BIT7, TSC0+CHOFFSET ;clear it
*    read time value from channel register
        pulh                     ;retrieve h register
        rti                      ;return from interrupt routine

* Application vectors

        org $ffee
        fdb CHISR
        fdb CHISR
        fdb CHISR
        fdb CHISR
        org $fffe
        fdb ICAP_INIT
```

Output Compare

Output compares are useful for generating one-shot or periodic output signals, or for internal software events (delays, periodic execution). The TIM allows control over the duration and polarity of an output signal, an option for buffered or unbuffered implementation, time reference adjustment (via the timer modulo register), and time base adjustment (via the prescaler). Note that buffered output compares can provide precise signal changes on the outputs, but each use two TIM channels. The following section of code uses an output compare port to generate a periodic output signal. See **Table 8** for a list of ranges and resolutions over all prescaler values. Range indicates the range of frequencies that can be output, as determined by the value in the modulo register. Resolution indicates the frequency at which the timer counter is incremented by one bit.

```
*
* Application example for TIM Reference Manual
*
* This software will feature the 68HC08XL36 Timer
* Interface Module in an output compare example.
* A signal with 50% duty cycle and specified period
* will be generated on a specified channel.
*
* Register Equates
TSC     equ $20     ;Timer Status & Control Register $20
TDMA    equ $21     ;Timer DMA Select Register $21
TCNTH   equ $22     ;Timer Counter Register Hi $22
TCNTL   equ $23     ;Timer Counter Register Lo $23
TMODH   equ $24     ;Timer Counter Modulo Register Hi $24
TMODL   equ $25     ;Timer Counter Modulo Register Lo $25
TSC0    equ $26     ;Timer Channel 0 Status & Control Register $26
TCH0H   equ $27     ;Timer Channel 0 Register Hi $27
TCH0L   equ $28     ;Timer Channel 0 Register Lo $28
TSC1    equ $29     ;Timer Channel 1 Status & Control Register $29
TCH1H   equ $2a     ;Timer Channel 1 Register Hi $2a
TCH1L   equ $2b     ;Timer Channel 1 Register Lo $2b
TSC2    equ $2c     ;Timer Channel 2 Status & Control Register $2c
TCH2H   equ $2d     ;Timer Channel 2 Register Hi $2d
TCH2L   equ $2e     ;Timer Channel 2 Register Lo $2e
TSC3    equ $2f     ;Timer Channel 3 Status & Control Register $2f
TCH3H   equ $30     ;Timer Channel 3 Register Hi $30
TCH3L   equ $31     ;Timer Channel 3 Register Lo $31

PORTE   equ $08     ;Port E, TIM port

* Miscellaneous Equates
BIT0      equ $00
BIT1      equ $01
BIT2      equ $02
BIT3      equ $03
BIT4      equ $04
BIT5      equ $05
BIT6      equ $06
BIT7      equ $07
CHOFFSET equ $00       ;ch0 = 0 offset, ch1 = 3 offset, chx = 3x offset
PERHI     equ $80       ;16.4ms period, 4.1ms offset from bus clock
PERLO     equ $00

* TIM status and control register bits
PRESCLR   equ $00       ;prescaler xxxxx000 (bus clock / 1)
TOE       equ $00       ;timer overflow interrupt enable x0xxxxxx

* Channel x status and control register bits
CHIE      equ $00       ;channel interrupt enable x0xxxxxx
MOD       equ $10       ;mode select bits xx00xxxx (output compare)
EDG       equ $04       ;edge select bits xxxx00xx (toggle output)
TOV       equ $00       ;toggle on overflow bit xxxxxx0x

* Variables in RAM
        org $0050       ;start of RAM

* Application assembly code
        org $6e00       ;start of ROM
```

```
OC_INIT:

* set up channel x for output compare
        mov #TOE+$30+PRESCLR,TSC    ;stop and reset TIM, select prescaler, TOE

        ldx #TSC0+CHOFFSET          ;point to channel x SCR
        lda #CHIE+MOD+EDG+TOV
        sta ,x                      ;store channel ICAP control info

        lda #PERHI                  ;set up signal period
        sta 1,x                     ;store in timer channel reg. (hi byte)
        lda #PERLO
        sta 2,x                     ;store low byte

        bclr BIT5,TSC      ;enable timer counter (generate signal)

        bra *

* interrupt service routines

* Application vectors

        org $fffe
        fdb OC_INIT
```

PWMs

PWMs are useful for generating signals to motor or gauge drivers, switching power supplies, and low-cost D/A converters. The TIM contains a special modulo register that controls the period of the PWM and a channel register that controls the pulse width (duty cycle). Other features include a buffered or unbuffered option, polarity control, toggle on overflow, and a 100% duty cycle option. The eight basic steps for setting up a PWM channel on the TIM are as follows:

1. Stop and reset the timer counter module using bits 4-5 of the timer status and control register.
2. Select a value for the timer counter modulo register and a timer prescaler with bits 0-2 of the timer status and control register to provide the required PWM period.
3. Load the appropriate timer channel register with the initial required pulse width.
4. Configure the timer channel for buffered/unbuffered output compare operation using bits 4 and 5 of the appropriate timer channel status and control register.
5. Select the timer counter “toggle on overflow” option using bit 1 of the appropriate timer channel status and control register.

6. Configure the timer channel to force the output to the “inactive” level on a successful compare (toggle on output compare should not be used) using bits 2 and 3 of the appropriate timer channel status and control register.
7. Enable output compare interrupts (if used) using bit 0 of the appropriate timer channel status and control register.
8. Enable the timer counter using bit 5 of the timer status and control register.

The following code details a general-purpose unbuffered PWM routine (all parameters are user-selectable).

```
*
* This software will feature the 68HC08XL36 Timer
* Interface Module in an unbuffered PWM example.
* A periodic waveform with specified duty cycle
* and period will be generated on a specified
* channel.
*
* Register Equates
TSC       equ $20    ;Timer Status & Control Register $20
TDMA      equ $21    ;Timer DMA Select Register $21
TCNTH     equ $22    ;Timer Counter Register Hi $22
TCNTL     equ $23    ;Timer Counter Register Lo $23
TMODH     equ $24    ;Timer Counter Modulo Register Hi $24
TMODL     equ $25    ;Timer Counter Modulo Register Lo $25
TSC0      equ $26    ;Timer Channel 0 Status & Control Register $26
TCH0H     equ $27    ;Timer Channel 0 Register Hi $27
TCH0L     equ $28    ;Timer Channel 0 Register Lo $28
TSC1      equ $29    ;Timer Channel 1 Status & Control Register $29
TCH1H     equ $2a    ;Timer Channel 1 Register Hi $2a
TCH1L     equ $2b    ;Timer Channel 1 Register Lo $2b
TSC2      equ $2c    ;Timer Channel 2 Status & Control Register $2c
TCH2H     equ $2d    ;Timer Channel 2 Register Hi $2d
TCH2L     equ $2e    ;Timer Channel 2 Register Lo $2e
TSC3      equ $2f    ;Timer Channel 3 Status & Control Register $2f
TCH3H     equ $30    ;Timer Channel 3 Register Hi $30
TCH3L     equ $31    ;Timer Channel 3 Register Lo $31

PORTE     equ $08    ;Port E, TIM port

* Miscellaneous Equates
BIT0      equ $00
BIT1      equ $01
BIT2      equ $02
BIT3      equ $03
BIT4      equ $04
BIT5      equ $05
BIT6      equ $06
BIT7      equ $07
CHOFFSET  equ $00    ;ch0 = 0 offset, ch1 = 3 offset, chx = 3x offset
DUTYHI    equ $80    ;50% duty cycle
DUTYLO    equ $00
PERHI     equ $FF    ;8.2ms period
PERLO     equ $FF
```

```
* TIM status and control register bits
PRESCLR  equ $00    ;prescaler xxxxx000 (bus clock / 1)
TOE      equ $00    ;timer overflow interrupt enable x0xxxxxx

* Channel x status and control register bits
CHIE     equ $40    ;channel interrupt enable x0xxxxxx
MOD      equ $10    ;mode select bits xx00xxxx (output compare)
EDG      equ $0c    ;edge select bits xxxx00xx (set output)
TOV      equ $02    ;toggle on overflow bit xxxxxx0x

* Variables in RAM
         org $0050  ;start of RAM

* Application assembly code
         org $6e00  ;start of ROM
UNBUF_INIT:

* set up channel x for unbuffered PWM
  mov #TOE+$30+PRESCLR, TSC  ;stop and reset TIM, select prescaler, TOE
  ldx #TSC0+CHOFFSET          ;point to channel x SCR (H reg = 00)
  lda #CHIE+MOD+EDG+TOV       ;enable i, OC, set output, toggle
  sta ,x                      ;store channel ICAP control info
  lda #DUTYHI                 ;specified duty cycle
  sta 1,x                     ;store in timer channel reg.
  lda #DUTYLO
  sta 2,x

  mov #PERHI,TMODH            ;store specified period
  mov #PERLO,TMODL

  cli                         ;enable CPU interrupts (delete this line if polling)
  bclr BIT5,TSC               ;enable timer counter

  bra *                       ;wait for interrupt (to poll monitor channel x flag)

* channel x interrupt service routine for OCx

CHISR    equ *
         pshh                        ;save h register
         lda TSC0+CHOFFSET           ;read channel x flag
         bclr BIT7,TSC0+CHOFFSET     ;clear it
* change duty cycle here (write to channel register)
         pulh                        ;retrieve h register
         rti                         ;return from interrupt routine

* Application vectors

         org $ffee
         fdb CHISR
         fdb CHISR
         fdb CHISR
         fdb CHISR

         org $fffe
         fdb UNBUF_INIT
```

The limitations of unbuffered PWMs are most apparent during the pulse width/duty cycle update process (for example, changing the timer channel register). The least reliable method is to randomly update the channel register, which can take up to two full periods to synchronize the output. A more reliable method is to update the channel register during a timer overflow interrupt routine, which can still cause one period of synchronization if the pulse width is decreased to 16 bus cycles or less (worst case interrupt latency before channel register update). The most reliable method is to update the channel register during an output compare interrupt routine. However, even this method will unsynchronize the output for one period when changing from a very high duty cycle to a very low duty cycle (for example, the interrupt routine latency exceeds the toggle and pulse width).

Since there are no infallible unbuffered PWM methodologies, the TIM has been designed to configure buffered PMW channels, at the trade-off of using two TIM channels. Buffered PWM channels are initialized using the same eight steps listed above and are implemented by alternately writing one of two timer channel registers. The software must track which channel register was last written. The following code details a general purpose buffered PWM routine using the TIM (all parameters are user selectable).

```
*
* This software will feature the 68HC08XL36 Timer
* Interface Module in an buffered PWM example.
* A periodic waveform with specified duty cycle
* and period will be generated on a specified
* channel.
*
* Register Equates
TSC      equ $20    ;Timer Status & Control Register $20
TDMA     equ $21    ;Timer DMA Select Register $21
TCNTH    equ $22    ;Timer Counter Register Hi $22
TCNTL    equ $23    ;Timer Counter Register Lo $23
TMODH    equ $24    ;Timer Counter Modulo Register Hi $24
TMODL    equ $25    ;Timer Counter Modulo Register Lo $25
TSC0     equ $26    ;Timer Channel 0 Status & Control Register $26
TCH0H    equ $27    ;Timer Channel 0 Register Hi $27
TCH0L    equ $28    ;Timer Channel 0 Register Lo $28
TSC1     equ $29    ;Timer Channel 1 Status & Control Register $29
TCH1H    equ $2a    ;Timer Channel 1 Register Hi $2a
TCH1L    equ $2b    ;Timer Channel 1 Register Lo $2b
TSC2     equ $2c    ;Timer Channel 2 Status & Control Register $2c
TCH2H    equ $2d    ;Timer Channel 2 Register Hi $2d
TCH2L    equ $2e    ;Timer Channel 2 Register Lo $2e
TSC3     equ $2f    ;Timer Channel 3 Status & Control Register $2f
TCH3H    equ $30    ;Timer Channel 3 Register Hi $30
TCH3L    equ $31    ;Timer Channel 3 Register Lo $31

PORTE    equ $08    ;Port E, TIM port

* Miscellaneous Equates
BIT0     equ $00
BIT1     equ $01
BIT2     equ $02
BIT3     equ $03
BIT4     equ $04
BIT5     equ $05
BIT6     equ $06
BIT7     equ $07
CHOFFSET equ $00    ;ch0 = 0 offset, ch2 = 6 offset
DUTYHI   equ $80    ;50% duty cycle
DUTYLO   equ $00
PERHI    equ $FF    ;8.2ms period
PERLO    equ $FF

* TIM status and control register bits
PRESCLR  equ $00    ;prescaler xxxxx000 (bus clock / 1)
TOE      equ $00    ;timer overflow interrupt enable x0xxxxxx

* Channel x status and control register bits
CHIE     equ $40    ;channel interrupt enable x0xxxxxx
MOD      equ $30    ;mode select bits xx00xxxx (buffered PWM)
EDG      equ $0c    ;edge select bits xxxx00xx (set output)
TOV      equ $02    ;toggle on overflow bit xxxxxx0x

* Variables in RAM
         org $0050  ;start of RAM
track    rmb 1
```

```
* Application assembly code
         org $6e00  ;start of ROM
BUFF_INIT:

* set up channel x for unbuffered PWM
         bset BIT0,track             ;set tracker flag

  mov #TOE+$30+PRESCLR,TSC    ;stop and reset TIM, select prescaler, TOE

  ldx #TSC0+CHOFFSET         ;point to channel x SCR (H reg = 00)
  lda #CHIE+MOD+EDG+TOV       ;enable i, buff, set output, toggle
  sta ,x                      ;store channel OC control info

  lda #DUTYHI                 ;specified duty cycle
  sta 1,x                     ;store in timer channel reg.
  lda #DUTYLO
  sta 2,x

  mov #PERHI,TMODH            ;store specified period in MOD reg.
  mov #PERLO,TMODL

  cli               ;enable CPU interrupts (delete this line if polling)
  bclr BIT5,TSC     ;enable timer counter (start PWM)

  bra *             ;wait for interrupt (to poll monitor channel x flag)

* channel x interrupt service routine for OCx

CHISR    equ *
         pshh      ;save h register
         lda TSC0+CHOFFSET        ;read channel x flag
         bclr BIT7,TSC0+CHOFFSET  ;clear it
         brset BIT0,track,JUMP    ;determine which reg. to update
         bset BIT0,track          ;set tracker flag
* change duty cycle (write to channel 1 or 3 register)
         bra EXIT
JUMP     bclr BIT0,track          ;clear tracker flag
* change duty cycle (write to channel 2 or 4 register)
EXIT     pulh                     ;retrieve h register
         rti                      ;return from interrupt routine

* Application vectors

         org $ffee
         fdb CHISR
         fdb CHISR
         fdb CHISR
         fdb CHISR
         org $fffe
         fdb BUFF_INIT
```

PWM Controlled RC Digital to Analog Converter

Pulse width modulation (PWM) is the process of using signal processing techniques to control the pulse width of a digital signal. The PWM signal is manipulated by software instructions and can be used in a variety of applications. One application is the design of a digital-to-analog converter (DAC). The PWM signal is sent out of the MCU and directly drives an RC circuit as shown in **Figure 47**. This circuit will produce a filtered average of the PWM signal on the output.

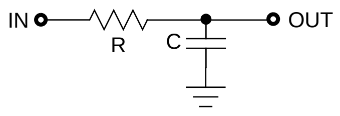


Figure 47. RC Circuit

For example, a 25% duty cycle PWM signal is shown in **Figure 48**. The “high” part of the signal is called the pulse width and in this example represents 25% of the PWM period. The “low” part of the signal represents 75% of the PWM signal. The entire cycle of the entire PWM signal is called the “PWM period.” The PWM frequency is 1/(PWM period).

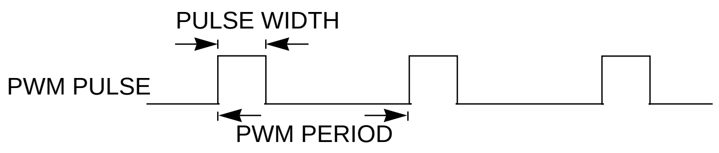


Figure 48. 25% Duty Cycle PWM Signal

When a PWM signal is fed into the RC circuit shown in **Figure 47**, a pseudo DC signal is created at the output of the RC circuit. This signal will have some low amplitude ripple dependent on the RC time constant of the circuit and the frequency of the PWM signal.

The average DC signal is related to the length of the pulse width of the PWM signal. The equation is as follows:

Average DC signal = Duty Cycle * (PWM high voltage — PWM low voltage)

For example, if there is a 5-volt, 50% duty cycle PWM signal driving the RC circuit in **Figure 47**, the average DC out voltage would be 2.5 volts. If the PWM duty cycle was changed to 85%, the average DC output voltage would be 4.25 volts. By varying the duty cycle, the user can design a simple digital-to-analog converter.

Analysis

There are four parameters that affect the DC output voltage: the capacitor, resistor, PWM frequency, and the PWM pulse width. As the preceding example has shown, the PWM pulse width (duty cycle) directly affects the filtered DC output level. The frequency and the RC time constant affect the ripple of the output. As the resistor and/or the capacitor are increased, there will be less ripple on the output due to the increased time constant of the circuit. Also, as the frequency increases, the ripple's frequency will increase, but its amplitude will decrease because the time for the voltage to rise or fall will be shortened.

When a rising voltage pulse is fed into the RC circuit, the output voltage responds according to the following equation:

$$V_{OUT} = V_{IN} - V_{IN}\, e^{-t/RC}$$

where V_{OUT} is the output voltage, V_{IN} is the input voltage, t is time after pulse, and RC is the time constant.

When a falling voltage pulse is fed into the RC circuit, the output voltage responds according to the following equation:

$$V_{OUT} = V_{IN}\, e^{-t/RC}$$

Figure 49 illustrates the transient voltage waveform of an RC network driven by a PWM signal. In this example, a 10-kHz, 50% duty cycle, 5-volt digital signal drives a resistor of 1 k and a capacitor of 1µF.

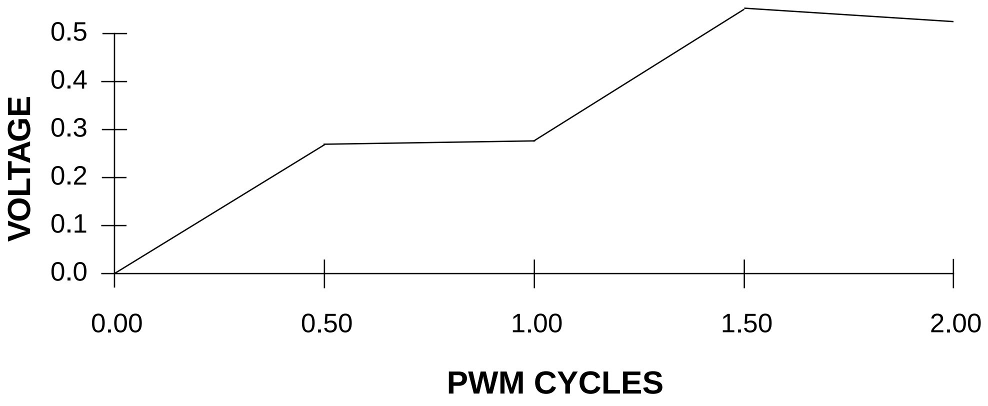


Figure 49. 10-kHz, 50% Duty Cycle RC Transient Response

Since the TIM uses a PWM signal, the equation to define the output voltage is not as obvious as the equations stated above. There are two different equations defining the output of the signal. A single equation cannot define the characteristics of the signal. In this case, the high part of the PWM signal has an equation and the low part of the PWM signal has an equation. The following equations define the voltage after the high part of the PWM signal (V_{OUT}HIGH the voltage at the low part of the PWM signal V_{OUT}LOW, and the average voltage of the high and low parts during the PWM period V_{OUT}AVG).

$$V_{OUT}\text{HIGH}^{(n)} = V_{IN} - V_{IN}\, e^{-[tc+(-RC \ln(\frac{V_{IN}-V_{OUT}LOW\,(n-1)}{V_{IN}}))]/RC}$$

$$V_{OUT}\text{LOW}^{(n)} = V_{OUT}\text{HIGH}^{(n)}\, e^{-t(1-d)/RC}$$

$$V_{OUT}\text{AVG}^{(n)} = \frac{V_{OUT}\text{HIGH}^{(n)} - V_{OUT}\text{HIGH}^{(n)}}{2} + V_{OUT}\text{LOW}^{(n)}$$

where *d* = duty cycle in decimal form
t = length of time of the signal's period
n = the number of PWM periods

Finding the output voltage at a particular time now becomes an iterative process. Write a small program with the preceding equations for your analysis. For example, use the circuit described earlier: a 5-volt 50% duty cycle, 10-kHz PWM signal driving a 1-k resistor and a 1-F capacitor network. Find the output voltage after the duty cycle of the second PWM period and the average voltage after one PWM period. After calculating with *t* = .0001 and *d* = 0.5, the following results are shown in **Table 9**.

Table 9. Output Voltages

Time Period	V_{OUT}HIGH	V_{OUT}LOW	V_{OUT}AVG
0	0	0	0
1	0.24385288	0.23196003	0.23790645
2	0.46450009	0.44184615	0.45317312

Therefore, the output voltage after the duty cycle of the second PWM period is 0.46450009 volts. The average voltage after one PWM period is 0.23790645 volts. A graph of the output voltage over 60 time periods is shown in **Figure 50**.

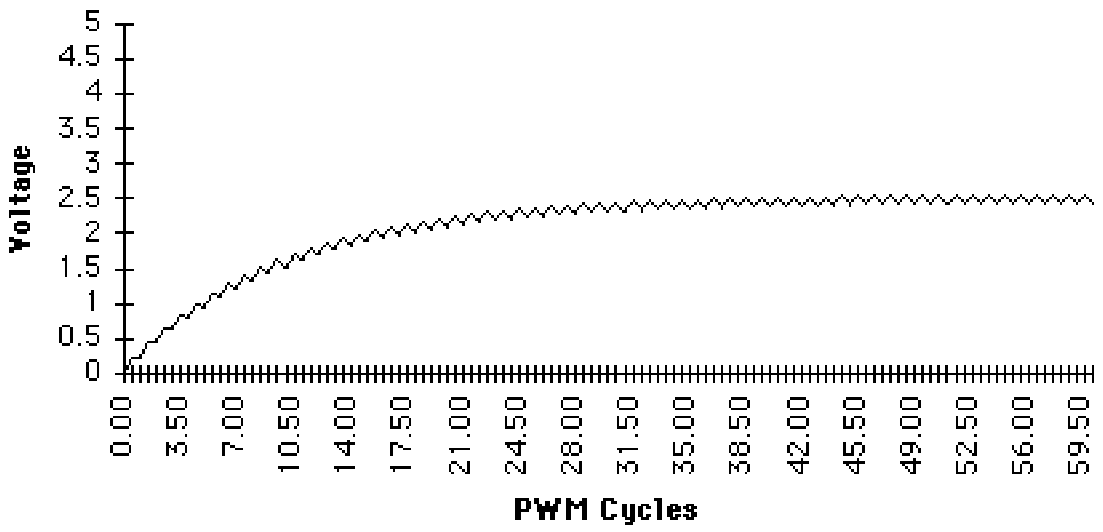


Figure 50. 10-kHz, 50% Duty Cycle RC Response

Hardware

Figure 47 shows that the DAC circuit is designed with an RC network. If the DAC were designed to drive a load, the extra load would change the time constant of the RC network. This in turn would change the transient and steady state responses. In order to buffer the RC network, feed the output into an op amp configured as a voltage follower. Since the inputs to the op amp are very high impedance, they should not affect the response of the RC network. The output of the op amp can now drive larger loads. This circuit is shown in **Figure 51**.

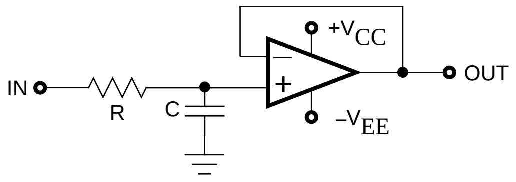


Figure 51. Buffered Output

Software

The following code initializes the timer for PWM and then enables the timer to start outputting the PWM signal.

```
****************************************************************************
****************************************************************************
*                                                                          *
*           Implementation of an unbuffered PWM with the Timer             *
*                                                                          *
****************************************************************************
*                                                                          *
* File Name: UB_PWM.RTN                    Copyright (c) Motorola 1994      *
*                                                                          *
* Curent Revision: 1.00                                                    *
* Current Release Level: ESS                                               *
* Current Revision Release Date: 03/23/94                                  *
*                                                                          *
* Current Release Written By: Mark Glenewinkel                             *
*                             Motorola CMCU Applications - Austin, Texas   *
*                                                                          *
* Assembled Under: IASM08 (P&E Microcomputer Systems, Inc.)    Ver.: 1.00   *
*                                                                          *
* Documentation File Name: UB_PWM.DOC             Revision: 1.00           *
*                                                                          *
* Brief Description of Routine Purpose:                                    *
*    This routine uses the Timer to implement a PWM signal.                *
*                                                                          *
* Part Family Software Routine Works With: HC08                            *
*                                                                          *
* Worst Case Execution (Cycles):   28                                      *
* Routine Size (Bytes):            20                                      *
* Stack Space Used (Bytes):         0                                      *
* RAM Used (Bytes):                 0                                      *
*                                                                          *
* Global Variables Used: None                                              *
* Subroutines Used: None                                                   *
*                                                                          *
* Full Functional Description of Routine Design:                           *
*   In order to setup a PWM waveform, the following must be configured:    *
*   1) Stop the timer, reset the timer counter (set TSTOP & TRST bits in TSC)*
*   2) Write to the 16 bit modulo counter to set the PWM period (TMODH:L)    *
*   3) Write to the 16 bit Output Compare register for the appropriate      *
*      timer channel. This sets the PWM pulse width (TCHxH:L)               *
*   4) Write to the timer channel status and control register (TSCx)       *
*      Disable the output compare interrupt (CHxIE = 0)                     *
*      Configure for unbuffered PWM (MSxB:A = 01)                           *
*      Configure to clear output line on PWM match (ELSxB:A = 10)           *
*      Configure overflow to toggle timer channel (TDVx = 1)                *
*      Disable max 100% duty cycle (CHxMAX = 0)                             *
*   5) Enable the timer to start counting (clear TSTOP in TSC)              *
*                                                                          *
****************************************************************************
*                                                                          *
* Update History:                                                          *
* Rev:      Author:     Date:        Description of Change:                *
* ----      -------     -----        ----------------------                *
* ESS 1.0   Glenewinkel 03/23/94     Original Release                      *
*                                                                          *
****************************************************************************
****************************************************************************
*                                                                         *
* Motorola reserves the right to make changes without further notice to any *
* product herein to improve reliability, function, or design. Motorola does *
* not assume any liability arising out of the application or use of any     *
* product, circuit, or software described herein; neither does it convey any*
* license under its patent rights nor the rights of others. Motorola        *
* products are not designed, intended, or authorized for use as components   *
* in systems intended for surgical implant into the body, or other           *
* applications intended to support life, or for any other application in     *
* which the failure of the Motorola product could create a situation where   *
* personal injury or death may occur. Should Buyer purchase or use Motorola *
* products for any such intended or unauthorized application, Buyer shall    *
* indemnify and hold Motorola and its officers, employees, subsidiaries,     *
* affiliatees, and distributors harmless against all claims, costs, damages, *
* and expenses, and reasonable attorney fees arising out of, directly or     *
* indirectly, any claim of personal injury or death associated with such     *
* unintended or unauthorized use, even if such claim alleges that Motorola   *
* was negligent regarding the design or manufacture of the part. Motorola    *
* and the Motorola Logo are registered trademarks of Motorola Inc.           *
*                                                                            *
****************************************************************************
****************************************************************************
*                                                                            *
*                   Part Specific Framework Includes Section                 *
*                                                                            *
$INCLUDE 'H708XL36.FRK'                 ;Device specific equates file
*
****************************************************************************
*                                                                            *
*                          Equates for Main Routine                          *
*                                                                            *
****************************************************************************
*                                                                            *
*       None
*                                                                            *
****************************************************************************
*                                                                            *
*                      RAM Definitions for Main Routine                      *
*                                                                            *
****************************************************************************
*                                                                            *
              org     RAM_Start
*       None
*                                                                            *
****************************************************************************
*                                                                            *
*                        Program Initialization                              *
*                                                                            *
* Code needed to initialize processor resources is placed here.              *
*                                                                            *
****************************************************************************

                org     EPROM_Start     ; start of HC708XL36 EPROM ($6E00)

Start           mov     #$30,TSC        ; stop the timer counter
                                        ; reset timer counter
                                        ; select prescaler (timclk = busclock)
*       Write the value for the required PWM frequency
*       With a 4 MHz bus, $01FF in the modulo register
*         creates a 7.8125kHz signal or a 128usec period

                mov     #$01,TMODH      ; modulo byte high
                mov     #$FF,TMODL      ; modulo byte low
```

```
*       Write the value for the required PWM pulse width
*       For a 25% pulse width, the pulse width will be equal
*         to 32usec
*       Write $0080 to timer channel 0 output compare

                mov     #$00,TCH0H      ; timer channel 0 high
                mov     #$80,TCH0L      ; timer channel 0 low

*       Write to timer channel 0 status and control reg
*       CH0E=0, disable output compare interrupt
*       MS0B:MS0A=01, unbuffered PWM, output compare
*       ELS0B:ELS0A=10, clear output line on compare
*       TOV0=1, enable timer counter toggle on overflow
*       CH0MAX=0, disable max 100% duty cycle

                mov     #%00011010,TSC0 ; timer CH0 status & ctrl

*       Enable the timer - start counter by clearing STOP bit

                bclr    5,TSC           ; timer status & ctrl

****************************************************************************
*                                                                          *
*                       Main Program Loop                                  *
*                                                                          *
****************************************************************************

LOOP            nop                     ; infinite loop while PWM
                bra     LOOP            ; is running


****************************************************************************
*                                                                          *
*               Interrupt Service Routines for Main Routine                *
*                                                                          *
****************************************************************************

                  org     RESET
                  fdb     Start

****************************************************************************
```

Servo Loop Motor Control

This application section details the implementation of a basic proportional derivative (PD) closed-loop speed control for a brush motor using four ICs (including an MC68HC08XL36), two opto discretes, and less than 200 bytes of code.

The TIM provides a flexible timing source for control systems which require extra features such as varying drive capability or non-volatile storage of parametric data. A typical MCU control system is mathematically modeled in the discrete time domain since data is sampled (not continuous). A straightforward control system using the MC68HC08XL36 and an MPM3004 TMOS H-bridge is characterized by the PD loop shown in **Figure 52**. The transfer function Gc(s) consists of the PD control, and Gp(s) represents a power amplifier, motor, and load, where s is a complex variable with both real and imaginary parts. In Gc(s), the proportional term, called Kp, is resolved (to a power of two) using simple shifts, and the derivative term, called KDs, of f(t) is approximately

$$\left.\frac{df(t)}{dt}\right|_{t=kT} \cong \frac{1}{\mathrm{T}}[f\ (kT) - f\ (k-1)T]$$

where f(kT) is the current sample of the controlled input and f(k-1)T is the previous sample. Finally, the term kDs is realized as the rate of change of the difference between the measured and desired period of motor shaft rotation.

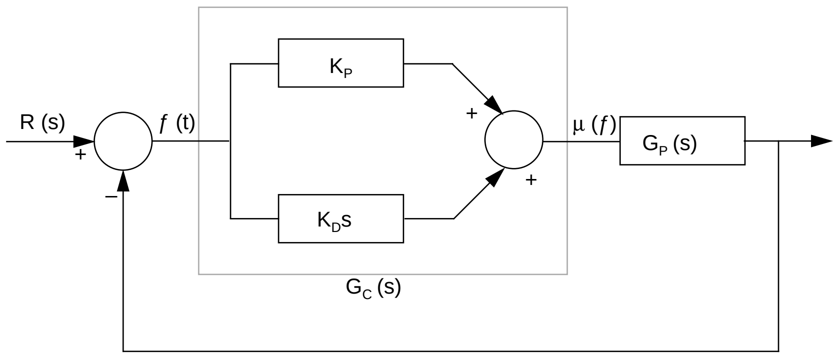


Figure 52. PD Loop Flow

The circuit in **Figure 53** represents a block diagram of servo loop motor control. The TIM is configured to input capture a feedback signal from an infrared detector on channel 1 and will output an 8-bit buffered PWM using channels 2 and 3. The H-bridge driver block is represented by an MPM3004, capable of controlling bidirectional currents of 10 A (continuous) at 60 V. It is driven indirectly by the microcontroller via the level shifter block, consisting of two MC34151 dual inverting gate drivers. Finally, the opto-sensor block is represented by an MRD750 Schmitt trigger detector, which is coupled to an MLED71 infrared emitter that monitors motor rotation through a slotted disc on the motor shaft.

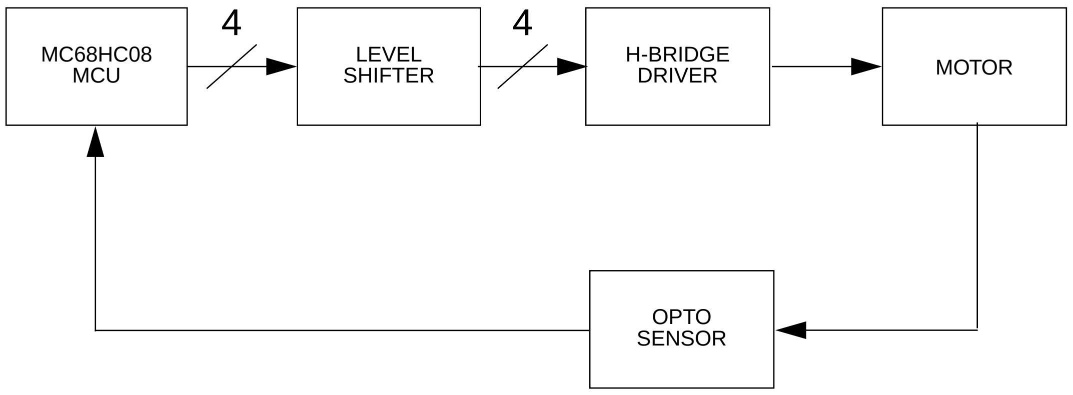


Figure 53. Servo Loop Motor Control Block Diagram

The complexity of the software algorithm executed by the microcontroller is dictated by the environment or desired performance of the motor. The justification for adding a derivative term can be understood by realizing that an underdamped proportional-only controller causes overshoot and ringing because of the system's attempt to reduce the error term to zero too rapidly. The derivative term works to compensate the rate of change of the error term. On the other hand, the derivative term also reduces the response time of the loop. Additionally, an integral term can be added (not included in this example) for low pass filtering or where the steady state error is potentially large.

The following code has been simplified for clarity to 8-bit math and will only drive the motor in one direction, so more stringent system specifications may require some additions to the code. Also, initialization, direction control, and diagnostic (for example, motor stall) routines have been omitted to keep this application generic. Finally, this application example is a conversion of Motorola Application Note *Basic Servo Loop Motor Control Using the MC68HC05B6 MCU*, order number AN1120/D. Please refer to it for more details.

*

```
* Application example for TIM Reference Manual
*
* This software features the 68HC08XL36 Timer Interface Module in an
* electronic motor speed control application. An input capture channel (IC1)
* will be used as a tachometer pulse input and two output compare channels (OC2
* & OC3) will be used in conjunction with a control line to a motor driver
* circuit (switching amplifier with 8-bit PWM input).
*
* Register Equates
TSC     equ $20     ;Timer Status & Control Register $20
TDMA    equ $21     ;Timer DMA Select Register $21
TCNTH   equ $22     ;Timer Counter Register Hi $22
TCNTL   equ $23     ;Timer Counter Register Lo $23
TMODH   equ $24     ;Timer Counter Modulo Register Hi $24
TMODL   equ $25     ;Timer Counter Modulo Register Lo $25
TSC0    equ $26     ;Timer Channel 0 Status & Control Register $26
TCH0H   equ $27     ;Timer Channel 0 Register Hi $27
TCH0L   equ $28     ;Timer Channel 0 Register Lo $28
TSC1    equ $29     ;Timer Channel 1 Status & Control Register $29
TCH1H   equ $2a     ;Timer Channel 1 Register Hi $2a
TCH1L   equ $2b     ;Timer Channel 1 Register Lo $2b
TSC2    equ $2c     ;Timer Channel 2 Status & Control Register $2c
TCH2H   equ $2d     ;Timer Channel 2 Register Hi $2d
TCH2L   equ $2e     ;Timer Channel 2 Register Lo $2e
TSC3    equ $2f     ;Timer Channel 3 Status & Control  Register $2f
TCH3H   equ $30     ;Timer Channel 3 Register Hi $30
TCH3L   equ $31     ;Timer Channel 3 Register Lo $31

PORTE   equ $08     ;Port E, where PWM happens!

* Miscellaneous Equates
BIT0      equ $00
BIT1      equ $01
BIT2      equ $02
BIT3      equ $03
BIT4      equ $04
BIT5      equ $05
BIT6      equ $06
BIT7      equ $07

* Variables in RAM
        org $0050     ;start of RAM
edge1h  rmb 1         ;high byte of first edge count
edge1l  rmb 1         ;low byte of first edge count
edge2h  rmb 1         ;high byte of second edge count
edge2l  rmb 1         ;low byte of second edge count
period  rmb 1         ;timer value of feedback signal
pwidth  rmb 1         ;pulsewidth for motor - must be initialized
desper  rmb 1         ;desired feedback period - must be initialized
deltnew rmb 1         ;current error value
deltold rmb 1         ;error value from last sample
deltadc rmb 1         ;rate of change compensated error
toggle  rmb 1         ;toggle for PWM buffers

* Application assembly code
        org $6e00     ;start of ROM
```

```
TIMER_INIT:

* set up channels 2 & 3 for PWM
        mov #$30,TSC       ;stop and reset the TIM, set prescaler to 1

        mov #$FF,TMODL     ;set 31.4 kHz frequency for PWM output
        clr TMODH          ;

        mov #$3A,TSC2      ;buff. PWM, no int., fall. edge, TOV2

        mov #pwidth,TCH2L  ;load initial pulsewidth into
        clr TCH2H          ;channel register
        bset BIT0,toggle   ;write channel 3 reg. on next update

*set up channel 1 for input capture
        mov #$48,TSC1      ;ICAP, fall. edge, enable int.

        cli                ;clear interrupt bit; enable CPU interrupts
        bclr BIT5,TSC      ;enable timer counter

        bra *
*
* Interrupt service routine(s)
*
* channel 1 interrupt service routine is entered
* when a falling edge is encountered on ICAP1

CH1ISR  equ *
        pshh                     ;save h register
        lda TSC1                 ;read channel 1 flag
        bclr BIT7,TSC1           ;clear it
gfly1   brclr BIT7,TSC1,gfly1    ;wait for next falling edge
        ldhx TCH1H               ;read timer counter value
        sthx edge1h              ;store first edge value
        lda #$FF
delay   dbnza delay              ;debounce time
        lda TSC1                 ;read channel 1 flag
        bclr BIT7,TSC1           ;clear it
gfly2   brclr BIT7,TSC1,gfly2    ;wait for next falling edge
        ldhx TCH1H               ;read timer counter value
        sthx edge2h              ;store second edge value
        lda edge2l               ;calculate period
        sub edge1l
        sta period               ;store period
        mov deltnew,deltold      ;read previous error & store it
        lda desper               ;load desired period
        sub period               ;subtract actual period
        blo incspd               ;if difference is < 0, inc. speed
        lsla                     ;multiply error by 2
        sta deltnew              ;store new error value
        lda deltold              ;read old error value
        sub deltnew              ;get rate of change of error
        sta deltadc              ;store rate of change of error
        lda deltnew              ;read new error value
        sub deltadc              ;apply de/dt correction
        sta deltadc              ;store it
        lda pwidth               ;read current motor pulse
        sub deltnew              ;apply correction
        bhi check                ;if positive, check magnitude
pwmdn   lda #$10
savepwm brset BIT0,toggle,set3   ;decide which channel reg. to write
```

```
        bset BIT0,toggle         ;write channel 3 next time
        sta TCH2L                ;store pulse length in channel 2 reg.
        clr TCH2H
        bra exit
set3    bclr BIT0,toggle         ;write channel 2 next time
        sta TCH3L                ;store pulse length in channel 3 reg.
        clr TCH3H
        bra exit
check   cmp #$10                 ;check for minimum
        blo pwmdn                ;set minimum
        bra savepwm              ;save decremented pulse length
incspd  lsla                     ;multiply error by 2
        sta deltnew              ;store new error value
        lda deltold              ;read previous error value
        sub deltnew              ;get rate of change of error
        sta deltadc              ;store rate of change compensation
        lda pwidth               ;read current pulse width
        sub deltadc              ;apply correction
        blo savepwm              ;check for saturation or correction = 0
exit    pulh                     ;retrieve h register
        rti

* Application vectors

        org $fff2
        fdb CH1ISR
        org $fffe
        fdb TIMER_INIT
```

Using the HC708XL36 DMA with the TIM

The MC68HC708XL36 timer module can be configured to let the DMA handle updating output compare registers when a timer interrupt occurs. When the contents of the timer counter registers matches the value stored in the timer output compare registers, a service request can be generated to signal the DMA to store a new value in the timer output compare registers, thus scheduling the next output compare. In the example that follows, the device is configured to have the DMA sequentially fetch a 16-bit value from a 32-byte table in RAM and store those values in the timer output compare registers as each output compare interrupt occurs. Using the DMA for this type of processing frees up the CPU for other tasks, since DMA processing can be set up to have minimal impact on CPU activity.

For example, the timer and DMA will be configured to continually repeat the waveform shown below.

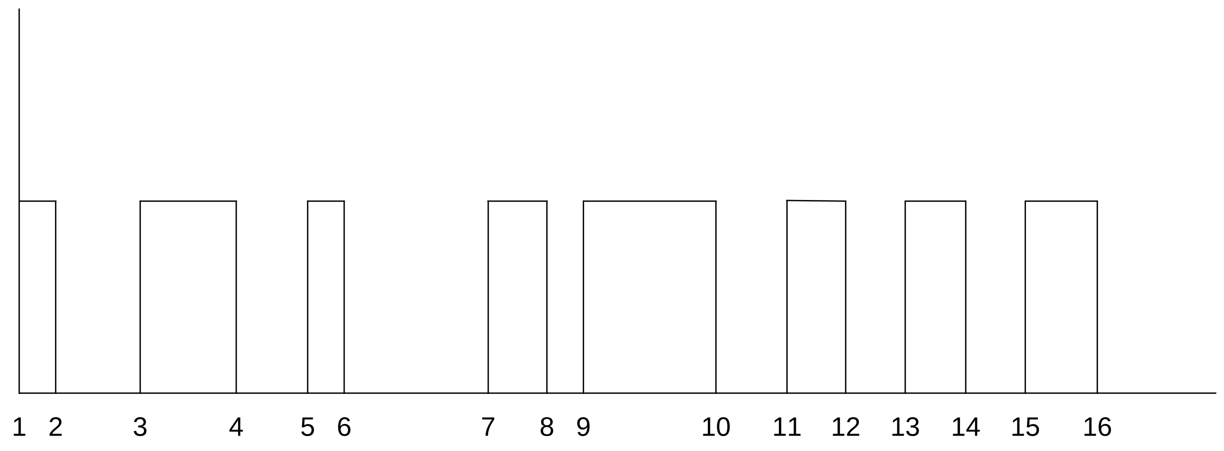


Figure 54. Waveform On Output Compare Pin (PTE5)

The output compare values that are used to generate the above waveform are stored in RAM beginning at location $60 as shown in the table below:

Table 10. Output Compare Values

Graph Point	Address in RAM	Data	
1	$60	$0000	(1st o/c)
2	$62	$03E8	(1st o/c + 125us)
3	$64	$0A28	(2nd o/c + 200us)
4	$66	$11F8	(3rd o/c + 250us)
5	$68	$1518	(4th o/c + 100us)
6	$6A	$1900	(5th o/c + 125us)
7	$6C	$2260	(6th o/c + 300us)
8	$6E	$2710	(7th o/c + 150us)
9	$70	$2AF8	(8th o/c + 125us)
10	$72	$3520	(9th o/c + 325us)
11	$74	$3A98	(10th o/c + 175us)
12	$76	$3F48	(11th o/c + 150us)
13	$78	$43F8	(12th o/c + 150us)
14	$7A	$48A8	(13th o/c + 150us)
15	$7C	$4D58	(14th o/c + 150us)
16	$7E	$5528	(15th o/c + 200us)

Functional Description of Program

The program uses DMA channel 0 for transfers and timer channel 1 for output compares. The bus rate is 8 MHz (125 ns per cycle). The modulo timer counter registers are set to generate a counter rollover at $6FFF.

The timer output compare registers (TCH1H:TCH1L) are initialized to $5555. The first output compare interrupt will occur when the value in the output compare registers matches the value in the timer counter registers. At that time the DMA will transfer the value of $0000 to the output compare registers to generate the first scheduled service request when the timer changes from $6FFF to $0000. The second scheduled interrupt will occur 125 µs later (1000 cycles * 125 ns per cycle = $03E8). The third output compare interrupt will occur 200 µs later (second output compare + third pulse = 125 µs + 200 µs = $0A28). Each output compare interrupt will generate a DMA service request. This process will therefore continue until all 16-bit table entries have been transferred. The waveform will be continually repeated from this point on each time the timer changes from $6FFF to $0000.

System Resource Configuration

DMA

- Write DMA Source Base Address (D0SH:D0SL = $0060)
  - Beginning address for the table of output compare values in RAM.
- Write DMA Destination Base Address (D0DH:D0DL = $002A)
  - Address of Timer Channel 1 Output Compare Registers (TCH1H:TCH1L).
- Configure DMA Channel 0 Control Register (D0C = $89)
  - Source/destination address set to increment/static.
  - Select word transfers.
  - DMA transfer source set to Timer Channel 1 service request.
- Configure DMA Status and Control Register (DSC = $10)
  - Enable loop mode for DMA Channel 0.

- Configure DMA Block Length Register (D0BL = $20)
  - Number of bytes in RAM table = 32 = $20.
- Configure DMA Control Register 1 (DC1 = 2)
  - Enable DMA Channel 0.

Timer

- Configure Timer Status and Control Register (TSC = $30):
  - Stop and reset Timer.
- Initialize Timer 1 Output Compare Registers (TCH1H:TCH1L):
  - First output compare at $5555.
- Configure Modulo Timer (TMODH:TMODL):
  - Counter rollover at $6FFF cycles.
- Configure Timer DMA Select (TDMA = 2):
  - DMA service for Timer Channel 1 service requests.
- Configure Timer Channel 1 Status and Control Register (TSC1 = $54):
  - Enable output compare interrupts.
  - Set up for output compare with toggle (see note below).
  - Disable toggle on Timer overflow.
- Configure Timer Status and Control Register (TSC = 0):
  - Start timer.
  - Prescaler set to system clock/1 (default condition).

NOTE: *When the output compare function is enabled, the associated output compare pin is set to a logic 1. The first toggle will be a negative transition and will occur when the timer counter reaches $5555.*

```
Main Routine:
      Branch to self forever (bra *)
end of main routine
.pagewidth !132
****************************************************************************
****************************************************************************
*                                                                            *
*                 Using the HC708XL36 DMA with the Timer                     *
*                                                                            *
****************************************************************************
*                                                                            *
* File Name: DMATEX.RTN                      Copyright (c) Motorola 1994     *
*                                                                            *
* Current Revision: 1.00                                                     *
* Current Release Level: PA                                                  *
* Current Revision Release Date: 03/13/94                                    *
*                                                                            *
* Current Release Written By: Mark Johnson                                   *
*                              Motorola CMCU Applications - Austin, Texas    *
*                                                                            *
* Assembled Under: IASM08 (P&E Microcomputer Systems, Inc.)    Ver.: 1.01    *
*                                                                            *
* Documentation File Name: DMATEX.TXT            Revision:1.00               *
*                                                                            *
* Brief Description of Routine Purpose: This routine uses the DMA to service *
*                                       Timer interrupts and schedule output *
*                                       compares.                            *
*                                                                            *
* Part Family Software Routine Works With: HC08                              *
*                                                                            *
* Worst Case Execution (Cycles): Infinite Loop                               *
* Routine Size (Bytes): 47                                                   *
* Stack Space Used (Bytes): 0                                                *
* RAM Used (Bytes): 32                                                       *
*                                                                            *
* Global Variables Used: None                                                *
* Subroutines Used: None                                                     *
*                                                                            *
* Full Functional Description Of Routine Design:                             *
*                                                                            *
*  The program uses DMA Channel 0 for transfers and Timer Channel 1 for      *
*  output compares. The bus rate is 8MHz (125 ns per cycle). The modulo      *
*  timer counter registers are set to generate a counter rollover at $6FFF.  *
*  The timer output compare registers (TCH1H:TCH1L) are initialized to       *
*  $5555. The first output compare interrupt will occur when the             *
*  value in the output compare registers matches the value in                *
*  the timer counter registers. At that time the DMA will transfer the value *
*  of $0000 to the output compare registers to generate the first scheduled  *
*  interrupt when the timer changes from $6FFF to $0000. The second          *
*  scheduled interrupt will occur 125us later (1000 cycles * 125ns per       *
*  cycle = $03E8). The third output compare interrupt will occur 200 us      *
*  later (125us + 200us = 1000 + 1600 = 2600 = $0A28). This process will     *
*  continue until all 16-bit table entries have been transferred. The        *
*  waveform will be continually repeated from this point on when the         *
*  timer changes from $6FFF to $0000.                                        *
*                                                                            *
* System Resource Configuration:                                            *
*                                                                           *
* DMA                                                                       *
*       1) Write DMA Source Base Address (D0SH:D0SL = $0060):               *
*               - beginning address for the table of output                 *
*                 compare values in RAM.                                    *
*       2) Write DMA Destination Base Address (D0DH:D0DL = $002A):          *
*               - address of Timer Channel 1 Output Compare Registers       *
*                 (TCH1H:TCH1L).                                            *
*       3) Configure DMA Channel 0 Control Register (D0C = $89):            *
*               - source/destination address set to increment/static.      *
*               - select word transfers.                                    *
*               - DMA transfer source set to Timer Channel 1                *
*                 service request.                                          *
*       4) Configure DMA Status and Control Register (DSC = $10):           *
*               - enable loop mode for DMA Channel 0.                       *
*       5) Configure DMA Block Length Register (D0DL = $20):                *
*               - number of bytes in RAM table = 32 = $20.                  *
*       6) Configure DMA Control Register 1 (DC1 = 2):                      *
*               - enable DMA Channel 0.                                     *
*                                                                           *
* Timer                                                                     *
*                                                                           *
*       1) Configure Timer Status and Control Register (TSC = $30):         *
*               - stop and reset Timer.                                     *
*       2) Initialize Timer 1 Output Compare Registers (TCH1H:TCH1L):       *
*               - first output compare at $5555.                            *
*       3) Configure Modulo Timer (TMODH:TMODL):                            *
*               - counter rollover at $6FFF cycles.                         *
*       4) Configure Timer DMA select (TDMA = 2):                           *
*               - DMA service for Timer Channel 1 interrupts.               *
*       5) Configure Timer Channel 1 Status and Control Reg. (TSC1 = $54):  *
*               - enable output compare interrupts.                         *
*               - set up for output compare with toggle                     *
*                 (* see note below).                                       *
*               - disable toggle on Timer overflow.                         *
*       6) Configure Timer Status and Control Register (TSC = 0):           *
*               - start Timer.                                              *
*               - prescaler set to system/1 (default condition).            *
*                                                                           *
* Note: When the output compare function is enabled, the associated output  *
*       compare pin is set to a logic 1. The first toggle will be a negative *
*       transition and will occur when the Timer counter reaches $5555.     *
*                                                                           *
* Main Routine:                                                             *
*                                                                           *
*       Branch to self forever (bra *)                                      *
*                                                                           *
*       end of main routine                                                 *
*                                                                           *
*                                                                           *
*****************************************************************************
*                                                                           *
* Update History:                                                           *
* Rev:      Author:     Date:        Description of Change:                 *
* ----      -------     -----        ----------------------                 *
* PA  1.0   Johnson     3/13/94      Original Release                       *
*                                                                           *
*****************************************************************************
```

```
****************************************************************************
*                                                                          *
* Motorola reserves the right to make changes without further notice to any  *
* product herein to improve reliability, function, or design. Motorola does  *
* not assume any liability arising out of the application or use of any      *
* product, circuit, or software described herein; neither does it convey any *
* license under its patent rights nor the rights of others. Motorola         *
* products are not designed, intended, or authorized for use as components   *
* in systems intended for surgical implant into the body, or other           *
* applications intended to support life, or for any other application in     *
* which the failure of the Motorola product could create a situation where   *
* personal injury or death may occur. Should Buyer purchase or use Motorola  *
* products for any such intended or unauthorized application, Buyer shall    *
* indemnify and hold Motorola and its officers, employees, subsidiaries,     *
* affiliates, and distributors harmless against all claims, costs, damages,  *
* and expenses, and reasonable attorney fees arising out of, directly or     *
* indirectly, any claim of personal injury or death associated with such     *
* unintended or unauthorized use, even if such claim alleges that Motorola   *
* was negligent regarding the design or manufacture of the part. Motorola    *
* and the Motorola Logo are registered trademarks of Motorola Inc.           *
*                                                                          *
****************************************************************************
****************************************************************************
$PAGE
*                                                                          *
****************************************************************************
*                                                                          *
*                   Part Specific Framework Includes Section              *
*                                                                          *
* Place the assembler statement ($INCLUDE) to include the part specific    *
* framework for the target part.                                           *
*                                                                          *
****************************************************************************
*
$NOLIST
$INCLUDE 'H708XL36.FRK'                ;Device specific equates file
*
$LIST
*
****************************************************************************
*                                                                          *
*                          Equates for Main Routine                       *
*                                                                          *
****************************************************************************
*
Table_start             equ     $60
*
****************************************************************************
*                                                                          *
*                      RAM Definitions for Main Routine                   *
*                                                                          *
****************************************************************************
                  org     Table_start
*
*     Set up table of output compare values in RAM starting at $0060
*
```

```
Table            fdb    0000         ;1st 16-bit entry
                 fdb    !1000        ;2nd entry
                 fdb    !2600        ;3rd entry
                 fdb    !4600        ;4th entry
                 fdb    !5400        ;5th entry
                 fdb    !6400        ;6th entry
                 fdb    !8800        ;7th entry
                 fdb    !10000       ;8th entry
                 fdb    !11000       ;9th entry
                 fdb    !13600       ;10th entry
                 fdb    !15000       ;11th entry
                 fdb    !16200       ;12th entry
                 fdb    !17400       ;13th entry
                 fdb    !18600       ;14th entry
                 fdb    !19800       ;15th entry
                 fdb    !21800       ;16th entry
*
$PAGE
****************************************************************************
*                                                                          *
*                        Program Initialization                            *
*                                                                          *
*      Code needed to initialize processor resources is placed here.       *
****************************************************************************
*
                 org    EPROM_Start   ;start of HC708XL36 EPROM ($6E00)
Start            equ    *
                 ldhx   #$450         ;load H:X with upper RAM boundary + 1.
                 txs                  ;move stack pointer to upper RAM
                                      ; boundary.
*
*     Initialization and configuration of DMA registers
*
                 ldhx    #Table       ;Write Source Base Address to D0SH:D0SL:
                 sthx    D0SH         ; -this is the RAM table starting address.
                 ldhx    #TCH1H       ;Write Destination Base Address to
                 sthx    D0DH         ; D0DH:D0DL. This is Timer Channel 1
                                      ; Output Compare Registers (TCH1H:TCH1L).
                 mov     #$89,D0C     ;Configure DMA Channel 0 Control Reg:
                                      ; -increment source/ static destination.
                                      ; -word transfer mode.
                                      ; -Timer Channel 1 service request.
                 mov     #$10,DSC     ;Configure DMA Status and Control Reg:
                                      ; -enable loop mode on DMA Channel 0.
                 mov     #$20,D0BL    ;Load Block Length Register 0 with
                                      ; output compare table size (!32 bytes).
                 mov     #2,DC1       ;Configure DMA Control Register 1:
                                      ; -disable CPU interrupt on loop restart.
                                      ; -enable DMA Channel 0.
*
```

```
*     Initialization and configuration of Timer registers
*
                  mov     #$30,TSC     ;Stop and reset Timer.
                  ldhx    #$5555       ;Initialize output compare regs to
                  sthx    TCH1H        ; generate first output compare.
                  ldhx    #$6FFF       ;Configure Modulo Timer:
                  sthx    TMODH        ; -counter rollover at $6FFF cycles.
                  mov     #2,TDMA      ;Configure Timer DMA select register:
                                       ; -DMA service for Timer Channel 1
                                       ;  interrupts.
                  mov     #$54,TSC1    ;Configure Timer Channel 1 Status and
                                       ; Control Register:
                                       ; -enable output compare interrupts.
                                       ; -set up for output compare w/ toggle.
                                       ; -disable toggle on timer overflow.
                  mov     #0,TSC       ;Start Timer.
****************************************************************************
*                                                                          *
*                          Main Program Loop                               *
*                                                                          *
****************************************************************************
*                                                                          *
Main              equ     *
                  bra     *            ;stay here forever
****************************************************************************
*                                                                          *
*               Reset Vectors for Main Routine                             *
*                                                                          *
****************************************************************************
                  org     RESET
                  fdb     Start
```

Electrical Specifications

Contents

Introduction

This section contains the electrical specifications and associated timing information for the TIM.

AC Characteristics

The following table provides information on the AC characteristics of the TIM.

Table 11. AC Characteristics

Characteristic	Symbol	Min	Max	Unit
Operating Frequency	f_{OP}	0	8[1]	MHz
TIM Counter Clock Frequency	f_{TCNT}	0	f_{OP}	MHz
TCLK Frequency	f_{TCLK}	0	$f_{OP}/2$	MHz
Cycle time	t_{cyc}	125[1]		ns
TIM Counter Period	t_{TCNT}	$1/f_{TCNT}$		ns
TCLK Pulse Width	t_{TCLK}	t_{cyc}		ns
Timer Resolution (IC, OC, & PWM)	t_{RESL}	t_{cyc}		ns

[1]The maximum operating speed (f_{OP}) of the TIM is the maximum bus speed of the MCU. That speed depends on the voltage and temperature range.

Timing Specifications

This section provides information on the timing relationships between the internal bus signals in the MCU. The CPU clock is referenced in the *M68HC08 Central Processor Unit Reference Manual*, order number CPU08RM/AD. The bus clocks, IT12 and IT23, are generated by the MCU clock generation module (CGM) and distributed to the MCU bus by the system integration module (SIM). The SIM also generates the internal address bus and internal data bus. All TIM timing is referenced to these internal bus signals from the SIM.

The rise/fall timing and hysteresis for the timer channel/port pins is dependent on the port circuit, and is specified in *MC68HC708XL36 Technical Summary*, order number MC68HC708XL36/D.

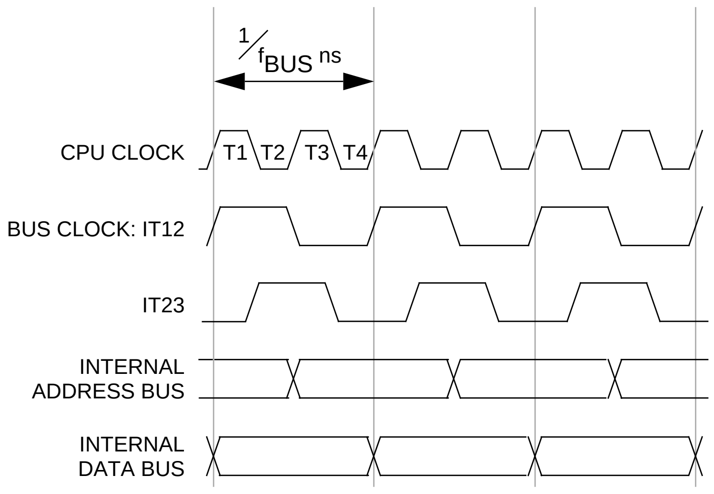


Figure 55. Internal Bus Signals

Memory Map and Registers

Contents

Introduction

This section contains an overview of the TIM registers. The following TIM registers are discussed in this section: timer status and control register (TSC), timer DMA select register (TDMA), timer counter registers (TCNTH:TCNTL), timer counter modulo registers (TMODH:TMODL), timer channel status and control registers (TSC0–TSC3), timer channel status and control registers (TSC0–TSC3), and timer channel registers (TCH0H/L–TCH3H/L).

NOTE: *The TIM can be implemented with two, four, six or eight channels. This manual will show the 4-channel version, as implemented in the MC68HC708XL36.*

Timer Status and Control Register

TSC	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TOF	TOE	TSTOP	0	0	PS2	PS1	PS0
Write:	0			TRST				
Reset:	0	0	1	0	0	0	0	0

Figure 56. Timer Status and Control Register (TSC)

TOF — Timer overflow flag

This clearable flag is set when the timer counter reaches the modulo value programmed in the timer modulo registers. Clear TOF by reading the timer status and control register when TOF is set and then writing a 0 to TOF. If another timer overflow occurs before the clearing sequence is complete, then writing 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Writing a 1 to this bit has no effect. Reset clears the TOF bit.

1 = Timer counter has reached modulo value.
0 = Timer counter has not reached modulo value.

TOE — Timer overflow enable

This read/write bit enables timer overflow interrupts when the TOF bit becomes set. Reset clears the TOE bit.

1 = Timer overflow interrupts enabled
0 = Timer overflow interrupts disabled

TSTOP — Timer stop

This read/write bit stops the timer counter. Counting resumes when TSTOP is cleared. Reset sets the TSTOP bit, stopping the timer counter until the TIM is enabled.

1 = Timer counter stopped
0 = Timer counter active

To preserve the correct timing relationship, TSTOP stops the input clock to the prescaler. The relationship cannot be preserved when using the external TCLK as the timer clock. See **Figure 13** for details on the timing of the TSTOP function.

NOTE: *Do not set the TSTOP bit before entering wait mode if the TIM is required to exit wait mode.*

TRST — Timer reset

Setting this write-only bit resets the timer counter and the timer prescaler. Setting TRST has no effect on any other registers. Counting resumes from $0000. TRST is cleared automatically after the timer counter is reset, and always reads zero. Reset clears the TRST bit.

1 = Prescaler and timer counter cleared
0 = No effect

See **Figure 14** for details on the timing of the TRST function.

NOTE: *Setting the TSTOP and TRST bits simultaneously stops the timer counter at a value of $0000.*

Bit 3 — Not used; always reads 0.

PS2–PS0 — Prescaler bits

These read/write bits select the bus clock, one of the six prescaler outputs, or the TCLK pin as the input to the timer counter. **Table 12** shows the prescaler selection encoding, including the TIM clock source, and the function of the TCLK pin. Reset clears the PS2–PS0 bits.

Table 12. Prescaler Selection

PS2: PS1: PS0	TIM Clock Source	PORT/TCLK Function
000	Bus Clock	PORT
001	Bus Clock ÷ 2	PORT
010	Bus Clock ÷ 4	PORT
011	Bus Clock ÷ 8	PORT
100	Bus Clock ÷ 16	PORT
101	Bus Clock ÷ 32	PORT
110	Bus Clock ÷ 64	PORT
111	TCLK	TCLK

NOTE: *Stop the TIM before changing the prescaler output. Before writing to the prescaler select bits (PS2–PS0), set the timer stop bit (TSTOP).*

NOTE: *Changing the prescaler control bits while the prescaler is running may cause an extra count if the input clock previously selected was a logic level 0 and the new input clock logic level is 1.*

See **Special Modes** for information on stopping the prescaler.

Timer DMA Select Register

NOTE: *This register is available only on microcontrollers with a DMA module. If no DMA module is included, do not enable the bits described in* ***Figure 57****. If those bits are enabled, no CPU interrupts for that channel will be generated or serviced.*

The timer DMA register selects either the CPU or the DMA module to service TIM interrupts. These bits are cleared on reset, selecting the CPU to process interrupts.

TDMA	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	DMA3S	DMA2S	DMA1S	DMA0S
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 57. Timer DMA Select Register (TDMA)

DMA3S — DMA channel 3 select

This read/write bit enables the DMA to process TIM interrupts on timer channel 3. Reset clears the DMA3S bit.

1 = Timer channel 3 generates DMA service requests.
0 = Timer channel 3 generates CPU interrupt requests.

DMA2S — DMA channel 2 select

This read/write bit enables the DMA to process TIM interrupts on timer channel 2. Reset clears the DMA2S bit.

1 = Timer channel 2 generates DMA service requests.
0 = Timer channel 2 generates CPU interrupt requests.

DMA1S — DMA channel 1 select

This read/write bit enables the DMA to process TIM interrupts on timer channel 1. Reset clears the DMA1S bit.

1 = Timer channel 1 generates DMA service requests.
0 = Timer channel 1 generates CPU interrupt requests.

DMA0S — DMA channel 0 select

This read/write bit enables the DMA to process TIM interrupts on timer channel 0. Reset clears the DMA0S bit.

1 = Timer channel 0 generates DMA service requests.
0 = Timer channel 0 generates CPU interrupt requests.

Timer Counter Registers

These two read-only timer counter registers (TCNT) contain the high and low bytes of the value in the timer counter. The counter value can be read at any time with user software without affecting its value. Reading the high byte (TCNTH) latches the contents of the low byte (TCNTL). Subsequent reads of TCNTH do not affect the latched TCNTL value until TCNTL is read. The LDHX instruction can be used to read a value from TCNT. The counter is set to $0000 on reset or when the timer reset bit (TRST) is set.

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

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

Figure 58. Timer Counter Registers (TCNTH:TCNTL)

Timer Counter Modulo Registers

These two read/write timer counter modulo registers (TMOD) contain the high and low bytes of the modulo value for the timer counter. When the timer counter reaches the modulo value, the TOF flag is automatically set by hardware, and the timer counter resumes counting from $0000 at the next clock. The overflow flag (TOF) and overflow interrupts are inhibited after a write to the high byte (TMODH) until the low byte (TMODL) is written. The STHX instruction can be used to write values to TMOD, and the LDHX instruction can be used to read values from TMOD. Reset sets the timer counter modulo registers to $FFFF, enabling the modulo counter to act as a free-running counter.

TMODH	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	1	1	1	1	1	1	1	1

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

Figure 59. Timer Counter Modulo Registers (TMODH:TMODL)

NOTE: *If TMODH:TMODL is set to $0000, a TOF is generated on the first cycle in which the match occurs, but not subsequently.*

NOTE: *Stop and reset the timer counter before writing to the timer counter modulo registers.*

Timer Channel Status and Control Registers

The timer channel status and control registers are 8-bit read/write registers. These registers are used to configure the timer channel to perform input capture, output compare, or PWM functions. The state of these registers is reset to $00.

Each of the timer channel status and control registers does the following:

- Flags input captures and output compares
- Enables input capture and output compare interrupts
- Selects input capture, unbuffered output compare, buffered output compare, unbuffered PWM, or buffered PWM operation
- Selects, high, low, or toggling output on output compare or PWM match
- Selects rising, falling, or any edge as the active input capture trigger
- Selects output toggling on timer overflow
- Selects 100% PWM duty cycle

TSC0	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC1	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC2	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

TSC3	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Figure 60. Timer Channel Status and Control Registers (TSC0–TSC3)

CHxF— Channel x flag

When channel x is an input capture channel, this clearable bit is set when an active edge occurs on the channel x pin. When channel x is an output compare or PWM channel, CHxF is set when the value in the timer counter registers matches the value in the timer channel x registers.

When CPU interrupts are enabled (CHxIE:DMAxS = 1:0), clear CHxF by reading the channel x status and control register with CHxF set and then writing a 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing zero to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

When DMA service requests are available and enabled (CHxIE:DMAxS = 1:1), clear CHxF by reading or writing to the low byte of the channel register (TCHxL). DMA service requests may not be used with buffered OC/PWM functions.

Writing a 1 to this bit has no effect. Reset clears the CHxF bit.
1 = Input capture or OC/PWM match on channel x
0 = No input capture or OC/PWM match on channel x

CHxIE — Channel x interrupt enable

This read/write bit enables channel x interrupts. In microcontrollers with a DMA module, the DMAxS bit in the timer DMA select register selects channel x CPU interrupts or DMA service requests for the input capture or unbuffered OC/PWM modes. DMA service requests cannot be used with buffered OC/PWM mode, therefore the DMAxS bit in the timer DMA select register should be cleared to select channel x CPU interrupts. Reset clears the CHxIE bit.
1 = Channel x interrupts enabled
0 = Channel x interrupts disabled

MSxB — Mode select bit B

MSxB exists only in the channel 0 and channel 2 status and control registers, TSC0 and TSC2.

This read/write bit selects buffered OC or buffered PWM operation. Setting MS0B disables the channel 1 status and control register, and reverts TCH1 to general-purpose I/O. Setting MS2B disables the channel 3 status and control register, and reverts TCH3 to general-purpose I/O. Reset clears the MSxB bit.
1 = Buffered OC/PWM mode enabled
0 = Buffered OC/PWM mode disabled

MSxA — Mode select bit A

This bit has different functions, depending on the state of ELSxB and ELSxA.

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin. To select the level of your initial output, write the appropriate value to MSxA while ELSxB:A = 00. Then configure all bits in the TSCx as required for your application.

1 = Initial output level low
0 = Initial output level high

When ELSxB:A ≠ 00, this read/write bit selects input capture mode or unbuffered OC/PWM mode, as shown in **Table 13**. MS0A and MS1A are active only when MS0B = 0. MS2A and MS3A are active only when MS2B = 0. Reset clears the MSxA bit.

1 = Unbuffered OC/PWM operation
0 = Input capture operation

NOTE: *Stop and reset the TIM before changing a channel function. Before writing to the mode select bits (MSxB and MSxA), set the timer stop and timer reset bits (TSTOP and TRST) in the TSC register.*

ELSxB and ELSxA — Edge/level select bits

When channel x is an input capture channel, these read/write bits control the active edge sensing logic on channel x. When channel x is an output compare or PWM channel, ELSxB and ELSxA control the channel x output behavior when an output compare or pulse width match occurs.

When ELSxB and ELSxA are both clear, channel x is not connected to the port, and pin TCHx is available as a general-purpose I/O pin. **Table 13** shows the configuration selected by ELSxB and ELSxA. Reset clears the ELSxB and ELSxA bits.

Table 13. Mode, Edge, and Level Selection

MSxB: MSxA	ELSxB: ELSxA	Mode	Configuration
X0	00	Output Preset	Set initial output level high
X1	00	Output Preset	Set initial output level low
00	00	Input Capture	TCHx Pin under port control; set initial output level high
00	01	Input Capture	Capture on Rising Edge Only
00	10	Input Capture	Capture on Falling Edge Only
00	11	Input Capture	Capture on Rising or Falling Edge
01	00	Unbuffered OC/PWM	TCHx Pin under port control; set initial output level low
01	01	Unbuffered OC/PWM	Toggle Output on OC/PWM Match
01	10	Unbuffered OC/PWM	Clear Output on OC/PWM Match
01	11	Unbuffered OC/PWM	Set Output on OC/PWM Match
1X	00	Buffered OC/PWM	TCHx Pin under port control; set initial output level
1X	01	Buffered OC/PWM	Toggle Output on OC/PWM Match
1X	10	Buffered OC/PWM	Clear Output on OC/PWM Match
1X	11	Buffered OC/PWM	Set Output on OC/PWM Match

NOTE: *Before enabling the channel register for input capture, make sure that the PTE/TCHx pin is stable for a minimum of two bus clocks.*

TOVx — Toggle on overflow

When channel x is a buffered or unbuffered OC/PWM channel, this read/write bit controls the behavior of the channel x output when the timer counter overflows. When channel x is an input capture channel, TOVx has no effect. Reset clears the TOVx bit.

1 = Channel x pin toggles on timer counter overflow.
0 = Channel x pin does not toggle on timer counter overflow.

NOTE: *When TOVx is set, a timer counter overflow takes precedence over a channel x output compare if both occur at the same time.*

CHxMAX — PWM 100% duty cycle

This read/write bit forces the duty cycle of buffered and unbuffered OC/PWM signals to 100%. As **Figure 61** shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared. CHxMAX affects only the logic level of the channel x pin; output compares or pulse width matches can continue to occur and set the channel x flag. Reset clears the CHxMAX bit.

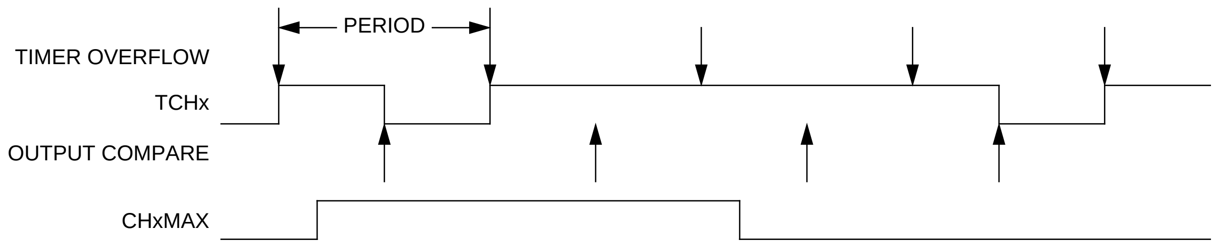


Figure 61. CHxMAX Latency

Timer Channel Registers

These read/write registers are used as the 16-bit input capture register latch for input capture functions, and as the 16-bit compare register for output compare and PWM functions. For input capture functions, these registers latch the value of TCNT when a specified transition is detected on the corresponding input capture pin. For output compare and PWM functions, these registers contain the output compare value for the output compare function or the pulse width match value for the PWM function. The state of the channel registers after reset is unknown.

In input capture mode (MSxB:MSxA = 0:0), reading the high byte of the channel register (TCHxH) inhibits input captures until the low byte (TCHxL) is read. This prevents another input capture from overwriting the TCHxH:TCHxL registers before the previous value has been read. An overwrite of TCHxH:TCHxL will occur if another IC is received before the TCHxH is read. The LDHX instruction may be used to read these registers.

Writing to the timer channel registers will overwrite any input capture data.

In unbuffered output compare/PWM modes, output compares/pulse width matches are inhibited between writes to TCHxH and TCHxL. This prevents another output compare/pulse width match from occurring until the new output compare/pulse width value is written. In buffered output compare/PWM modes, output compares/pulse width matches are inhibited between writes to TCHxH and TCHxL of the active channel. This prevents another output compare/pulse width match from occurring until the new pulse width value is written. Output compares are allowed between writes to TCHxH and TCHxL of the inactive channel. The sthx instruction can be used to write to TCHxH:L.

If a timer channel register is not used for an input capture, output compare, or PWM function, it can be used as a storage location.

TCH0H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							

TCH0L	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							

TCH1H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							

TCH1L	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							

TCH2H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							

TCH2L	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							

TCH3H	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Reset:	Indeterminate after Reset							

TCH3L	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	Indeterminate after Reset							

Figure 62. Timer Channel Registers (TCH0H/L–TCH3H/L)

Pin Summary

Contents

Introduction

This section summarizes the TIM pins, the pin configuration control bits, and the data on the external pins.

TIM Pin Functions

The following table summarizes the specific TIM pin functions.

Table 14. Pin Functions

Pin	Pin Function
GPI	General-purpose input
GPO	General-purpose output
ICx	TIM input capture
OCx	TIM output compare/PWM match
TCLK	Pin is used as external clock input.

TIM Pin Summary

The following tables summarize the individual TIM pins.

Table 15. TCH0 and TCH2 Pins

Function	MSxB: MSxA	ELSxB: ELSxA	DDRx	Pin Direction	Configuration
GPI	00	00	0	Input	PORT Pin
GPO	00	00	1	Output	PORT Pin
ICx	00	01	X	Input	ICx on Rising Edge Only
ICx	00	10	X	Input	ICx on Falling Edge Only
ICx	00	11	X	Input	ICx on Rising or Falling Edge
GPI	01	00	0	Input	PORT Pin
GPO	01	00	1	Output	PORT Pin
OCx	01	01	X	Output	Toggle Output on OCx
OCx	01	10	X	Output	Clear Output on OCx
OCx	01	11	X	Output	Set Output on OCx
GPI	1X	00	0	Input	PORT Pin
GPO	1X	00	1	Output	PORT Pin
OCx	1X	01	X	Output	Toggle Output on OCx
OCx	1X	10	X	Output	Clear Output on OCx
OCx	1X	11	X	Output	Set Output on OCx

Table 16. TCH1 and TC H3 Pins

Function	MS0B: MS2B	MS1A: MS3A	ELS1B: ELS1A ELS3B: ELS3A	DDREx	Pin Direction	Configuration
GPI	0	0	00	0	Input	PORT Pin
GPO	0	0	00	1	Output	PORT Pin
ICx	0	0	01	X	Input	Rising Edge Only
ICx	0	0	10	X	Input	Falling Edge Only
ICx	0	0	11	X	Input	Rising or Falling Edge
GPI	0	1	00	0	Input	PORT Pin
GPO	0	1	00	1	Output	PORT Pin
OCx	0	1	01	X	Output	Toggle Output
OCx	0	1	10	X	Output	Clear Output
OCx	0	1	11	X	Output	Set Output
GPI	1	X	XX	0	Input	PORT Pin
GPO	1	X	XX	1	Output	PORT Pin

Table 17. TCLK Pin

Function	PS2: PS1: PS0	DDRE3	Pin Direction	Configuration
GPI	0XX	0	Input	PORT Pin
GPI	X0X	0	Input	PORT Pin
GPI	XX0	0	Input	PORT Pin
GPO	0XX	1	Output	PORT Pin
GPO	X0X	1	Output	PORT Pin
GPO	XX0	1	Output	PORT Pin
TCLK	111	X	Input	TIM Clock Input

Glossary

$xxxx — The digits following the “$” are in hexadecimal format.

#xxxx — The digits following the “#” indicate an immediate operand.

A — Accumulator. See “accumulator.”

accumulator (A) — An 8-bit general-purpose register in the CPU08. The CPU08 uses the accumulator to hold operands and results of arithmetic and nonarithmetic operations.

address bus — The set of signals used to select a specific memory location so that the CPU can write information into the memory location or read its contents.

addressing mode — The way that the CPU obtains (addresses) the information needed to complete an instruction. The M68HC08 CPU has 16 addressing modes.

algorithm — A set of specific procedures by which a solution is obtained in a finite number of steps, often used in numerical calculations.

ALU — Arithmetic logic unit. See “arithmetic logic unit.”

arithmetic logic unit (ALU) — The portion of the CPU of a computer where mathematical and logical operations take place. Other circuitry decodes each instruction and configures the ALU to perform the necessary arithmetic or logical operations at each step of an instruction.

assembly language — A method used by programmers for representing machine instructions (binary data) in a more convenient form. Each machine instruction is given a simple, short name, called a mnemonic (or memory aid), which has a one-to-one correspondence with the machine instruction. The mnemonics are translated into an object code program which a microcontroller can use.

ASCII — American Standard Code for Information Interchange. A widely accepted correlation between alphabetic and numeric characters and specific 7-bit binary numbers.

asynchronous — Refers to circuitry and operations without common reference signals.

BCD — Binary-coded decimal. See “binary-coded decimal.”

binary — The binary number system using 2 as its base and using only the digits 0 and 1. Binary is the numbering system used by computers because any quantity can be represented by a series of ones and zeros. Electrically, these ones and zeros are represented by voltage levels of approximately Vdd (input) and Vss (ground), respectively.

binary-coded decimal (BCD) — A notation that uses binary values to represent decimal quantities. Each BCD digit uses 4 binary bits. Six of the possible 16 binary combinations are considered illegal.

bit — A single binary digit. A bit can hold a single value of zero or one.

Boolean — A mathematical system of representing logic through a series of algebraic equations that can only be true or false, using operators such as AND, OR, and NOT.

branch instructions — Computer instructions that cause the CPU to continue processing at a memory location other than the next sequential address. Most branch instructions are conditional. That is, the CPU will continue to the next sequential address (no branch) if a condition is false, or continue to some other address (branch) if the condition is true.

buffered — Utilizes a second storage register to prevent overwriting of valid data.

bus — A collection of logic signals used to transfer information.

bus clocks — There are two bus clocks, IT12 and IT23. These clocks are generated by the CGM and distributed throughout the MCU by the SIM. The frequency of the bus clocks, or operating frequency, is fop. While the frequency of these two clocks is the same, the phase is different. See **Figure 55. Internal Bus Signals.**

byte — A set of exactly eight binary bits.

C — Abbreviation for carry/borrow in the condition code register of the CPU08. The CPU08 sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some logical operations and data manipulation instructions also clear or set the C flag (as in bit test and branch instructions and shifts and rotates).

CCR — Abbreviation for condition code register in the CPU08. See “condition code register.”

central processor unit (CPU) — The primary functioning unit of any computer system. The CPU controls the execution of instructions.

CGM — Clock generation module. See “clock generation module.”

checksum — A value that results from adding a series of binary numbers. When exchanging information between computers, a checksum gives an indication about the integrity of the data transfer. If values were transferred incorrectly, it is unlikely that the checksum would match the value that was expected.

clear — To establish logic zero state on a bit or bits; the opposite of “set.”

clock — A square wave signal used to sequence events in a computer.

clock generation module (CGM) — A module in the M68HC08 Family. The CGM generates a base clock signal from which the system clocks are derived. The CGM may include a crystal oscillator circuit and or phase-locked loop (PLL) circuit.

comparator — A device that compares the magnitude of two inputs. A digital comparator defines the equality or relative differences between two binary numbers.

condition code register (CCR) — An 8-bit register in the CPU08 that contains the interrupt mask bit and five bits (flags) that indicate the results of the instruction just executed.

control bit — One bit of a register manipulated by software to control the operation of the module.

control unit — One of two major units of the CPU. The control unit contains logic functions that synchronize the machine and direct various operations. The control unit decodes instructions and generates the internal control signals that perform the requested operations. The outputs of the control unit drive the execution unit, which contains the arithmetic logic unit (ALU), CPU registers, and bus interface.

counter clock — The input clock to the TIM counter. This clock is an output of the prescaler sub-module. The frequency of the counter clock is ftcnt, and the period is ttcnt.

CPU — Central processor unit. See “central processor unit.”

CPU08 — The central processor unit of the M68HC08 Family.

CPU cycles — A CPU clock cycle is one period of the internal bus-rate clock, fop, normally derived by dividing a crystal oscillator source by two or more so the high and low times will be equal. The length of time required to execute an instruction is measured in CPU clock cycles.

CPU registers — Memory locations that are wired directly into the CPU logic instead of being part of the addressable memory map. The CPU always has direct access to the information in these registers. The CPU registers in an M68HC08 are:

- A (8-bit accumulator)
- H:X (16-bit accumulator)
- SP (16-bit stack pointer)
- PC (16-bit program counter)
- CCR (condition code register containing the V, H, I, N, Z, and C bits)

cycles — See “CPU cycles.”

cycle time — The period of the operating frequency: tcyc=1/fop.

data bus — A set of signals used to convey binary information from a CPU to a memory location or from a memory location to a CPU.

data direction register (DDR) — A register which determines the function (input or output) of each pin in a data port.

DDR — Data direction register. See “data direction register.”

decimal — Base ten numbering system that uses the digits zero through nine.

digital-to-analog converter (DAC) — A particular type of circuitry that changes digital information into analog information such as voltage or current.

direct memory access (DMA) — One of a number of modules that handle a variety of control functions in the modular M68HC08 Family. The DMA can perform interrupt-driven and software-initiated data transfers between any two CPU-addressable locations. Each DMA channel can independently transfer data between any addresses in the memory map. DMA transfers reduce CPU overhead required for data movement interrupts.

direct page — The first 256 bytes of memory ($0000–$00FF); also called page 0.

DMA — Direct memory access. See “direct memory access.”

duty cycle — A ratio of the amount of time the signal is on versus the time it is off. It is usually represented by a percentage.

effective address (EA) — The address where an instruction operand is located. The addressing mode of an instruction determines how the CPU calculates the effective address of the operand.

EPROM — Erasable, Programmable, Read-Only Memory. A non-volatile type of memory that can be erased by exposure to an ultraviolet light source.

execution unit (EU) — One of the two major units of the CPU containing the arithmetic logic unit (ALU), CPU registers, and bus interface. The outputs of the control unit drive the execution unit.

free-running counter — A device which counts from zero to a pre-determined number, then rolls over to zero and begins counting again.

H — Abbreviation for the upper byte of the 16-bit index register (H:X) in the CPU08.

H — Abbreviation for “half-carry” in the condition code register of the CPU08. This bit indicates a carry from the low-order four bits of the accumulator value to the high-order four bits. The half-carry bit is required for binary-coded decimal arithmetic operations. The decimal adjust accumulator (DAA) instruction uses the state of the H and C flags to determine the appropriate correction factor.

hexadecimal — Base 16 numbering system that uses the digits 0 through 9 and the letters A through F. One hexadecimal digit can exactly represent a 4-bit binary value.

high order — The leftmost digit(s) of a number.

H:X — Abbreviation for the 16-bit index register in the CPU08. See “index register.”

I — Abbreviation for “interrupt mask bit” in the condition code register of the CPU08. When I is set, all interrupts are disabled. When I is cleared, interrupts are enabled.

index register (H:X) — A 16-bit register in the CPU08. The upper byte of H:X is called H. The lower byte is called X. In the indexed addressing modes, the CPU uses the contents of H:X to determine the effective address of the operand. H:X can also serve as a temporary data storage location.

input/output (I/O) — Input/output interfaces between a computer system and the external world. A CPU reads an input to sense the level of an external signal and writes to an output to change the level on an external signal.

instructions — Operations that a CPU can perform. Instructions are expressed by programmers as assembly language mnemonics. A CPU interprets an opcode and its associated operand(s) and instruction.

instruction set — The instruction set of a CPU is the set of all operations that the CPU can perform. An instruction set is often represented with a set of shorthand mnemonics, such as LDA, meaning “load accumulator (A).” Another representation of an instruction set is with a set of opcodes that are recognized by the CPU.

interrupt — Provide a means to temporarily suspend normal program execution so that the CPU is freed to service sets of instructions in response to requests (interrupts) from peripheral devices. Normal program execution can be resumed later from its original point of departure. The CPU08 can process up to 128 separate interrupt sources, including a software interrupt (SWI).

interrupt service routine — An I/O software routine servicing a specific interrupt.

I/O — Input/output. See “input/output.”

$\overline{\textbf{IRQ}}$ — Interrupt request. The overline indicates an active-low signal.

least significant bit (LSB) — The rightmost digit of a binary value.

logic one — A voltage level approximately equal to the input power voltage (V_{DD}).

logic zero — A voltage level approximately equal to the ground voltage (V_{SS}).

low order — The rightmost digit(s) of a number.

LS — Least significant.

LSB — Least significant bit. See “least significant bit.”

M68HC08 — A Motorola family of 8-bit MCUs.

machine codes — The binary codes processed by the CPU as instructions. Machine code includes both opcodes and operand data.

MCU — Microcontroller unit. See “microcontroller.”

memory location — In the M68HC08 each memory location holds one byte of data and has a unique address. To store information into a memory location, the CPU places the address of the location on the address bus, the data information on the data bus, and asserts the write signal. To read information from a memory location, the CPU places the address of the location on the address bus and asserts the read signal. In response to the read signal, the selected memory location places its data onto the data bus.

memory map — A pictorial representation of all memory locations in a computer system.

microcontroller — Microcontroller unit (MCU). A complete computer system, including a CPU, memory, a clock oscillator, and input/output (I/O) on a single integrated circuit.

mnemonic — Three to five letters that represent a computer operation. For example, the mnemonic form of the “load accumulator” instruction is LDA.

modulo counter — A device capable of being programmed so that it counts to any number from zero to its maximum possible modulus.

most significant bit (MSB) — The leftmost digit of a binary value.

MS — Abbreviation for “most significant.”

MSB — Most significant bit. See “most significant bit.”

multiplexer (mux) — A digital device that can select one of a number of inputs and pass the logic level of that input on to the output.

N — Abbreviation for “negative,” a bit in the condition code register of the CPU08. The CPU sets the negative flag when an arithmetic operation, logical operation, or data manipulation produces a negative result.

nibble — Half a byte; 4 bits.

object code — The output from an assembler or compiler that is itself executable machine code, or is suitable for processing to produce executable machine code.

one — A logic high level, a voltage level approximately equal to the input power voltage (V_{DD}).

one’s complement — An infrequently used form of signed binary numbers. Negative numbers are simply the complement of their positive counterparts. One’s complement is the result of a bit by bit complement of a binary word: all ones are changed to zeros and all zeros changed to ones. One’s complement is two’s complement without the increment.

opcode — A binary code that instructs the CPU to do a specific operation in a specific way.

operand — The fundamental quantity on which a mathematical operation is performed. Usually a statement consists of an operator and an operand. The operator may indicate an add instruction; the operand therefore will indicate what is to be added.

oscillator — A circuit that produces a constant frequency square wave that is used by the computer as a timing and sequencing reference.

overflow — This condition occurs when the modulo counter reaches its modulo value and rolls over to zero.

page 0 — The first 256 bytes of memory ($0000–$00FF). Also called direct page.

PC — Program counter. See “program counter.”

pointer — Pointer register. An index register is sometimes called a pointer register because its contents are used in the calculation of the address of an operand, and therefore “points” to the operand.

port — A collection of individual I/O signals.

prescaler — A circuit that generates an output signal related to the input signal by a fractional scale factor such as 1/2, 1/8, 1/10 etc.

program — A set of computer instructions that cause a computer to perform a desired operation or operations.

programming model — The registers of a particular CPU.

program counter (PC) — A 16-bit register in the CPU08. The PC register holds the address of the next instruction or operand that the CPU will use.

pull — The act of reading a value from the stack. In the M68HC08, a value is pulled by the following sequence of operations. First, the stack pointer register is incremented so that it points to the last value saved on the stack. Next, the value at the address contained in the stack pointer register is read into the CPU.

pulse-width — The amount of time a signal is on as opposed to being in its off state.

pulse-width modulation (PWM) — Controlled variation (modulation) of a pulse width, with a constant frequency.

PWM period — The time required for one complete cycle of a PWM waveform.

push — The act of storing a value at the address contained in the stack pointer register and then decrementing the stack pointer so that it points to the next available stack location.

RAM — Random access memory. All RAM locations can be read or written by the CPU. The contents of a RAM memory location remain valid until the CPU writes a different value or until power is turned off.

RC circuit — A circuit consisting of capacitors and resistors having a defined time constant.

read — To transfer the contents of a memory location to the CPU.

registers — Memory locations wired directly into the CPU logic instead of being part of the addressable memory map. The CPU always has direct access to the information in these registers. The CPU registers in an M68HC08 are:
- A (8-bit accumulator)
- (H:X) (16-bit accumulator)
- SP (16-bit stack pointer)
- PC (16-bit program counter)
- CCR (condition code register containing the V, H, I, N, Z, and C bits)

Memory locations that hold status and control information for on-chip peripherals are called input/output (I/O) and control registers.

reset — Reset is used to force a computer system to a known starting point and to force on-chip peripherals to known starting conditions.

ROM — Read-only memory. A type of memory that can be read but cannot be changed (written). The contents of ROM must be specified before manufacturing the MCU.

servo loop — In a servo amplifier, the entire closed loop formed by feedback from output to input. In a position servo, the output position is compared to a command signal at the input.

set — To establish a logic one state on a bit or bits; opposite of “clear.”

signal groups — Groups of electronic signals related by function.

signed — A form of binary number representation accommodating both positive and negative numbers. The most significant bit is used to indicate whether the number is positive or negative, normally zero for positive and one for negative, and the other seven bits indicate the magnitude.

SIM — System integration module. See “system integration module.”

software interrupt (SWI) — An instruction that will cause an interrupt and its associated vector fetch.

SP — Stack pointer. See “stack pointer.”

stack — A mechanism for temporarily saving CPU register values during interrupts and subroutines. The CPU maintains this structure with the stack pointer (SP) register, which contains the address of the next available (empty) storage location on the stack. When a subroutine is called, the CPU pushes (stores) the low-order and high-order bytes of the return address on the stack before starting the subroutine instructions. When the subroutine is done, a return from subroutine (RTS) instruction causes the CPU to recover the return address from the stack and continue processing where it left off before the subroutine. Interrupts work in the same way except that all CPU registers are saved on the stack instead of just the program counter. The H register is not pushed on the stack during an interrupt.

stack pointer (SP) — A 16-bit register in the CPU08 containing the address of the next available (empty) storage on the stack.

status bit — One bit of a register used to store information about the status of the module.

subroutine — A sequence of instructions to be used more than once in the course of a program. The last instruction in a subroutine is a return from subroutine (RTS) instruction. At each place in the main program where the subroutine instructions are needed, a jump or branch to subroutine (JSR or BSR) instruction is used to call the subroutine. The CPU leaves the flow of the main program to execute the instructions in the subroutine. When the RTS instruction is executed, the CPU returns to the main program where it left off.

SWI — Software interrupt. See "software interrupt."

synchronous — Refers to two or more things made to happen simultaneously in a system by means of a common reference signal.

system integration module (SIM) — One of a number of modules that handle a variety of control functions in the modular M68HC08 Family. The SIM controls mode of operation, resets and interrupts, and system clock generation.

TIM — Timer Interface Module.

timer — A module used to relate events in a system to a point in time.

toggle — To change the state of an output from a logic zero to a logic one or from a logic one to a logic zero.

two’s complement — A means of performing binary subtraction using addition techniques. The most significant bit of a two’s complement number indicates the sign of the number (1 indicates negative). The two’s complement negative of a number is obtained by inverting each bit in the number and then adding 1 to the result.

unbuffered — Utilizes only one register for data; new data overwrites current data.

unsigned — Refers to a binary number representation in which all numbers are assumed positive. With signed binary, the most significant bit is used to indicate whether the number is positive or negative, normally zero for positive and one for negative, and the other seven bits indicating the magnitude.

variable — A value that changes during the course of executing a program.

waveform — A graphical representation in which the amplitude of a wave is plotted against time.

word — Two bytes or 16 bits, treated as a unit.

write — The transfer of a byte of data from the CPU to a memory location.

X — Abbreviation for the lower byte of the index register (H:X) in the CPU08.

Z — Abbreviation for zero, a bit in the condition code register of the CPU08. The CPU08 sets the zero flag when an arithmetic operation, logical operation, or data manipulation produces a result of $00.

zero — A logic low level, a voltage level approximately equal to the ground voltage (Vss).

Index

TIM08RM/AD