Engineering Bulletin

EB389/D
Rev. 1, 3/2002

TOF Considerations when Measuring a Long Input Capture Event

By Donnie Garcia and Gianni Filippi
Microcontroller Applications Engineering
Austin, Texas

Introduction and Background

Most M68HC08 Family products contain timer modules that can be used to perform a multitude of tasks. One application the timer is used for is measuring time between input-capture events. This document clarifies unclear text from earlier M68HC08 Family data books, reaffirming the information contained in the *TIM08 Timer Interface Module Reference Manual,* Motorola document order number TIM08RM/AD. In this engineering bulletin, early timer overflow flag (TOF) is defined, a real-world situation is discussed, and two separate software solutions are outlined.

NOTE: *References to Timer 1, Timer 2, Timer A, or Timer B will be made in the following text by omitting the timer number/letter. For example, TMOD generically refers to TAMOD, TBMOD, T1MOD, or T2MOD. Refer to the appropriate data book.*

Definition of Early TOF

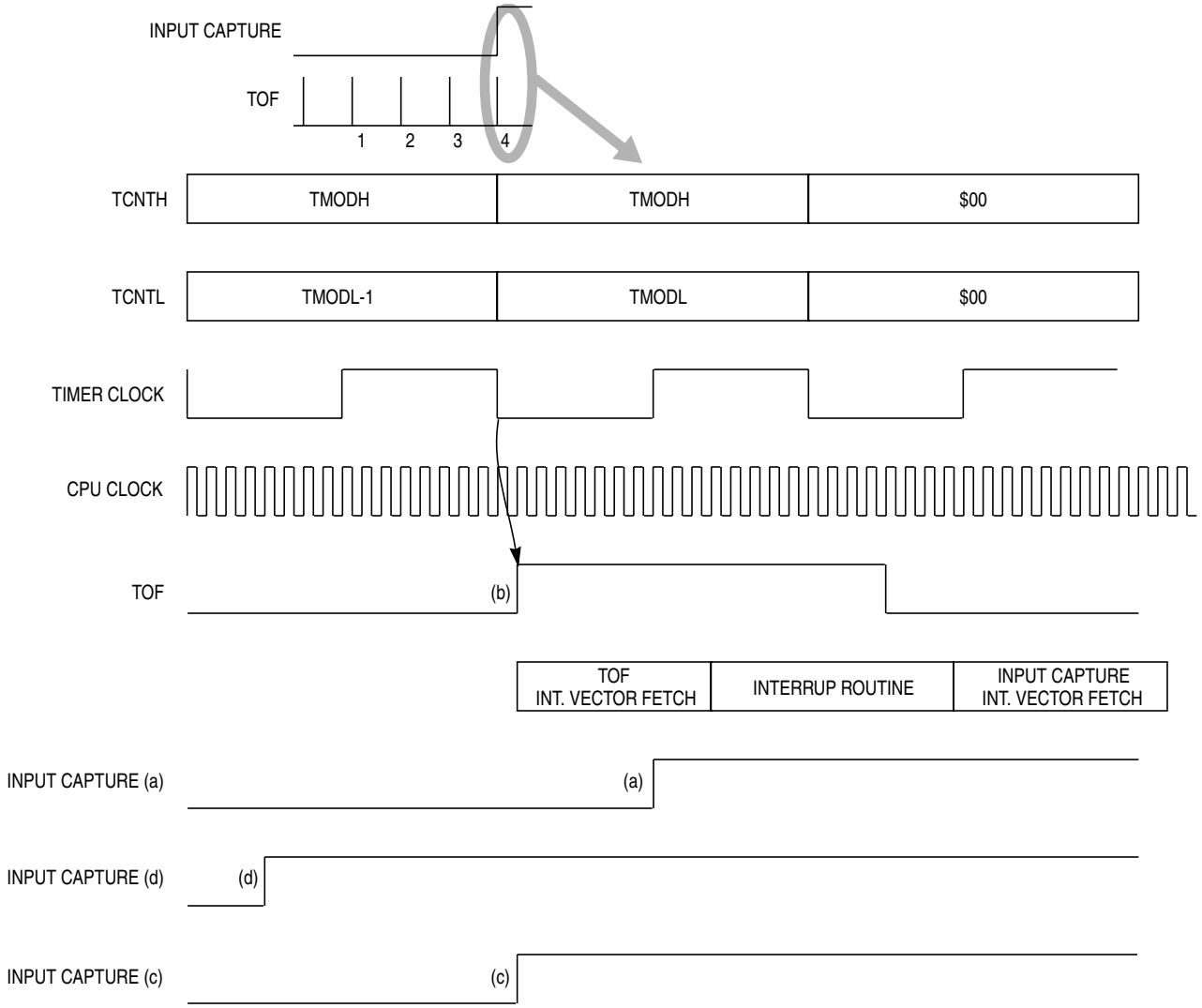
When trying to calculate the time of a pulse width that spans many timer overflows, an application program must count the number of overflows that occur. The final time interval between input captures equals:

Equation 1: (Ovf * (TMOD + 1)) + TCHX

Where: Ovf = The number of overflows
TMOD = The value in the timer counter modulo register
TCHX = The contents of the timer channel register

To accomplish this, a counter must be placed in the timer overflow subroutine. Every time the overflow interrupt occurs, the counter is incremented to keep track of the number of overflows. The early TOF arises because the overflow interrupt occurs soon after the timer reaches the value in the timer counter

modulo register, as stated in the Interrupts section of the *TIM08 Timer Interface Module Reference Manual*. When this occurs, the overflow counter can be incremented before the timer counter reaches its next state ($0000). This leads to an inaccurate count of the number of overflows that have occurred if an input capture occurs during the modulo transition time. See **Figure 1(a)**.



(a) An input-capture event occurs when the TCNT equals TMOD.

(b) TOF is set one bus cycle after the TCNT goes from TMOD-1 to TMOD.

(c) An input-capture event occurs when the TCNT equals TMOD before TOF is set.

(d) An input-capture event occurs when the TCNT does not equal TMOD.

Figure 1. Early TOF Cases

A Discussion of Real-World Early TOF

To better understand early TOF, the following discussion was generated during resolution of a customer issue.

Symptom:

A company developed C code that used the timer interface module to measure time lengths of input-captured pulses longer than the timer's overflow range. During the debugging process, they sometimes experienced the abnormal phenomenon of seeing the overflow interrupt having a higher priority than that of the channel interrupt, contrary to the data book specification. This was causing erroneous time measurement when an input-captured signal occurred while the timer counter was at the modulo value.

Explanation:

It turned out that the priorities are respected, but the TOF is set when the TIM 16-bit counter reaches the modulo value, rather than when it resets to $0000 after reaching the modulo value programmed in the timer counter modulo register. For example if the timer counter modulo registers (TMODH and TMODL) contain $FFFF, the TOF is set when the timer counter goes from $FFFE to $FFFF. See **Figure 1(b)**. The timer overflow interrupt service routine will then be entered and no other interrupts will be granted until that one is completed. Therefore, if an input-capture signal occurs, it will not trigger an immediate interrupt service routine, even if the channel interrupt has a higher priority than the overflow one. However, if the channel receives an input signal while the timer counter is counting $FFFF, a time measurement error will occur. See **Figure 1(a)**. As the timer clock source can be divided by the internal bus clock (by 1, 2, 4, 8, 16, 32, 64 or be driven by the TCLK pin), this error will appear when the timer clock source is set to a value significantly slower than the internal bus clock. In fact, only in those circumstances will there be a chance of experiencing an early TOF.

Software Solution 1: Overflow ISR Software Technique:

To ensure correct timing calculations, modify the overflow interrupt service routine (ISR) to detect whether an input capture signal occurred while the timer counter was at the timer modulo value. If this is the case, the overflow interrupt service routine will be aborted and the program will return to the main. See **Figure 2**. Then it will enter the input capture interrupt service routine, because it is a higher priority than the overflow. After that, the program will return to the previously aborted overflow interrupt service routine. Therefore, a miscalculation of time will not occur.

The flow chart and the software for this solution are shown here.

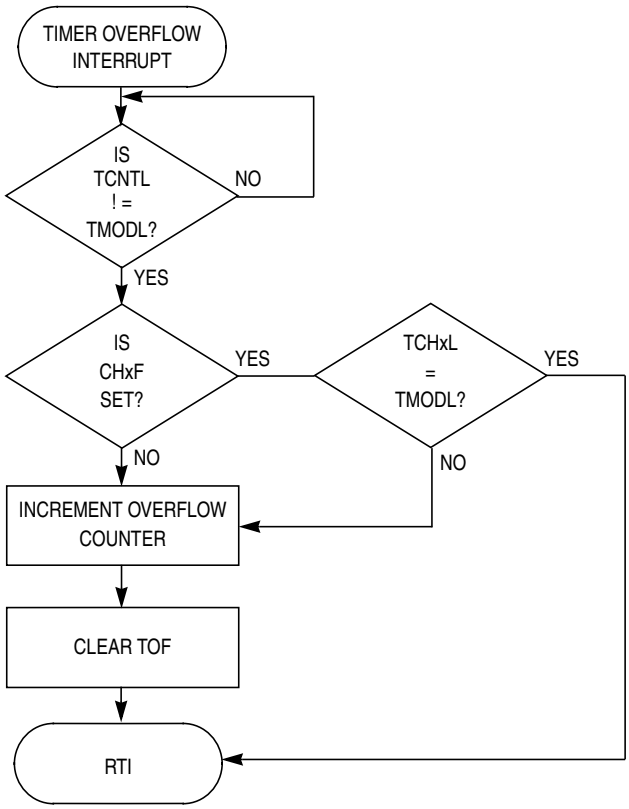


Figure 2. Overflow ISR Technique Flowchart

Flow Chart Explanation

1. Wait until the lower byte of the free-running counter overflows the value stored in the TMODL register. Read the lower byte of the timer register and wait until a number different from TMODL is read. (This is required to ensure that the overflow counts are performed after there has been a “true” overflow.)
2. To determine if a channel interrupt occurred, check whether CHXF is set.

   If a channel interrupt has occurred, the next task is to find out exactly when this interrupt has occurred before exiting. (Go to 3)

   If CHXF=0, then no channel interrupt has occurred. The routine carries on as normal and increments the “count of overflows,” clears the TOF flag, and exits the routine: RTI.
3. Read the lower byte of the timer channel registers (TCHXL) and compare with TMODL.

   If equal, the channel interrupt occurred when the free-running counter was counting the value stored in the TMODL. This means that the “count of overflows” must not be incremented until after the timer channel interrupt service routine is serviced first. Therefore the routine is aborted, RTI, without any further execution.

   If TCHXL is not equal to TMODL, then the timer channel interrupt happened after the free running counter did a “true” overflow and therefore the “count of overflows” is incremented, the TOF flag is cleared, and the routine is executed: RTI.
4. After returning from the timer overflow interrupt routine to the main program, the timer channel interrupt service routine is entered immediately.

   Note that if the channel interrupt occurred at TMOD, then the CPU will have two interrupts pending as the TOF flag was not cleared. The CPU now fetches the timer channel interrupt as it has a higher priority.
5. After returning from the timer channel interrupt service routine, the channel flag (CHXF) will be cleared. The CPU will now fetch the vectors, for a second time, for the timer overflow event. When it enters the timer overflow interrupt service routine, the CHXF will now be cleared, and thus the “count of overflows” will be incremented as normal, the TOF will then be cleared, and the routine will be exited: RTI.

TIM Overflow Interrupt Service Routine – Assembler Code

```
TIM_OVL_isr
read_again        LDA     TCNTL           ;Check the low part of the TIM counter
                                          ;and wait for a 'true' overflow condition.
                  CMP     TMODL           ;Is TCNTL=TMODL
                  BEQ     read_again      ;This loop will not exit until the TIM
                                          ;counter overflows to $0000.

                  BRSET   7,TSCX,test     ;Has there been a channel interrupt?

normal_overflow   INC     OVL_CTR         ;Increment the overflow counting meter
                  LDA     TSC             ;Read the TIM Status and Control Register
                  BCLR    7,TSC           ;and write 0 to bit 7 to complete the
                                          ;clearing sequence (clear TOF)
                  JMP     exit            ;Jump to RTI
test              LDA     TCHXL           ;Read the lower byte of the timer channel register.
                  CMP     TMODL           ;TCHXL=TMODL
                  BNE     normal_overflow ;If the value is not equal to the modulo value
                                          ;carry on with the normal routine, otherwise RTI

exit              RTI                     ;Return to main
```

TIM Overflow Interrupt Service Routine – C Code

```
#define CHXF 0x80                                   /* Bit mask for bit 7 in TSCX */
@interrupt void TIM_OVL (void)
{
   while (TCNTL = = TMODL);                         /* Wait here for a rollover
                                                       from modulo to $00. Note that
                                                       there  is no need to check TCNTH,
                                                       in fact it will be equal to
                                                       the timer modulo value. */

   if (!((TSCX & CHXF) && (TCHXL = = TMODL)))       /* If a Channel-Interrupt did not
                                                       occur or (if it occurred but)
                                                       the timer modulo value was not
                                                       the latched value into the TIM
                                                       channel register, proceed normally,
                                                       therefore carry on with the
                                                       Overflow-Interrupt service routine.
                                                       Note that there is no need to
                                                       check TCHXH, in fact it will contain
                                                       the timer modulo high byte. */
{
        /* The Normal Timer-Overflow Interrupt service routine must be inserted here */
   }
}       /* Otherwise (if the Input Capture Interrupt occurred and also the value latched
           into the TIM channel register was the timer modulo value) RTI, therefore abort
           the Overflow-Interrupt service routine and go back to the main, which will
           jump into the Input Capture Interrupt service routine as it will be pending and
           it has an higher priority that the Overflow-Interrupt request. */
```

Testing for Output Compare ISR Software Solution

Testing was completed using an MMDS0508 emulator and an M68EML08GP32 emulator module. The timer modulo value was set to the default, $FFFF. Three test conditions were identified and tested using Metrowerks CodeWarrior IDE.

- **Test condition 1** — An input capture occurs when the timer channel register does not equal $FFFF (**Figure 1(d)**).
- **Test condition 2** — An input capture occurs when the timer channel register equals $FFFF after a TOF interrupt has occurred (**Figure 1(a)**).
- **Test condition 3** — An input capture occurs when the timer register equals $FFFF before a TOF interrupt has occurred (**Figure 1(c)**).

For test condition 1, timer channel 1 was set up for output compare. PortD4 was connected to PortD5 (timer channel 1) so that when timer channel 1 output compare occurred at $7FFF, an input capture was triggered on timer channel 0 (**Figure 1(c)**). For test condition 2, PortD4 was connected to PortC4. Once a TOF interrupt occurred, PortC was toggled so that an input capture would occur while the timer channel register was equal to $FFFF (**Figure 1(a)**). Only divide-by 64 and divide-by 32 were tested for this condition because the smaller divide-by values did not allow proper timing. For test condition 3, timer channel 1 was set up for output compare. Port D4 was connected to PortD5 (timer channel 1) so that when timer channel 1 output compare occurred at $FFFF, an input capture was triggered on timer channel 0 (**Figure 1(c)**).

Software Solution 2: Input Capture ISR Software Technique:

The input capture ISR software solution does a series of checks once an input-capture interrupt occurs to resolve the early TOF behavior. From examination of the timer, it was determined that the early TOF occurs soon after the timer reaches the value in the timer counter modulo register. Thus the first check that the input capture ISR does is to see if the contents of the timer channel register contain the value in the timer counter modulo register (the input capture occurred at the value in the timer modulo register). See **Figure 1(a)**. If the input capture did not occur at the value in the timer counter modulo register, then the normal calculation of the input capture time should be done. See **Equation 1**. If the input capture did occur at the value in the timer modulo register, then the software must handle two special cases. Case one is when the TOF interrupt occurs, and then an input capture occurs when the timer is still at the value in the timer counter modulo register. See **Figure 1(a)**. This is the early TOF condition. The calculation of the input capture time is done with **Equation 2**:

Equation 2: $((Ovf - 1) * (TMOD + 1)) + TCHX$

Where Ovf = The number of overflows
TMOD = The value in the timer counter modulo register
TCHX = The contents of the timer channel register

This equation accounts for the extra overflow in the overflow counter. Case two occurs when the timer reaches the value in the timer modulo register and an input capture occurs before the TOF interrupt. See **Figure 1(c)**. This case must use the normal calculation of the input capture time, **Equation 1**. These two special cases are handled using the status of the TOF flag in the timer status and control register. Refer to the flowchart in **Figure 3**. The C code is shown here.

```
//TMOD = Timer Modulus Register
//TCHX = Timer channel register
//TSC  = Timer Status and control register
//TSCX = Timer Channel Status and control register
//numb = Total number of timer counts since last input capture
//n    = Overflow counter
//i    = constant value equal to Timer Modulo +1

void interrupt TIC_ISR(void)
{                                   //Input Capture Interrupt Service Routine. The software
                                    //solution is implemented here

if (TCHX == TMOD)                   //Check the Timer channel register for the value
                                    //in the Timer counter modulo register
{
        if ((TSC & 0x80) == 0)      //If Timer channel register is the value in the
                                    //Timer Modulo Register then check to see
                                    //if there is a pending TOF interrupt
        {numb = ((n - 1) * i) + TCHX;}  //If there is no pending interrupt then n must be
                                        //modified to account for early TOF
        else
               {numb = (n * i) + TCHX;} //If there is a pending TOF interrupt
                                        //then use the normal calculation

}
        else   {numb =(n * i) + TCHX ;} //If Timer channel register did not equal
                                        //the value in the Timer Modulo Register then use
                                        //the normal equation

TSCX = TSCX & 0X7f;                 //Clear the Interrupt flag in the status and
                                    //control register for channel X
}
```

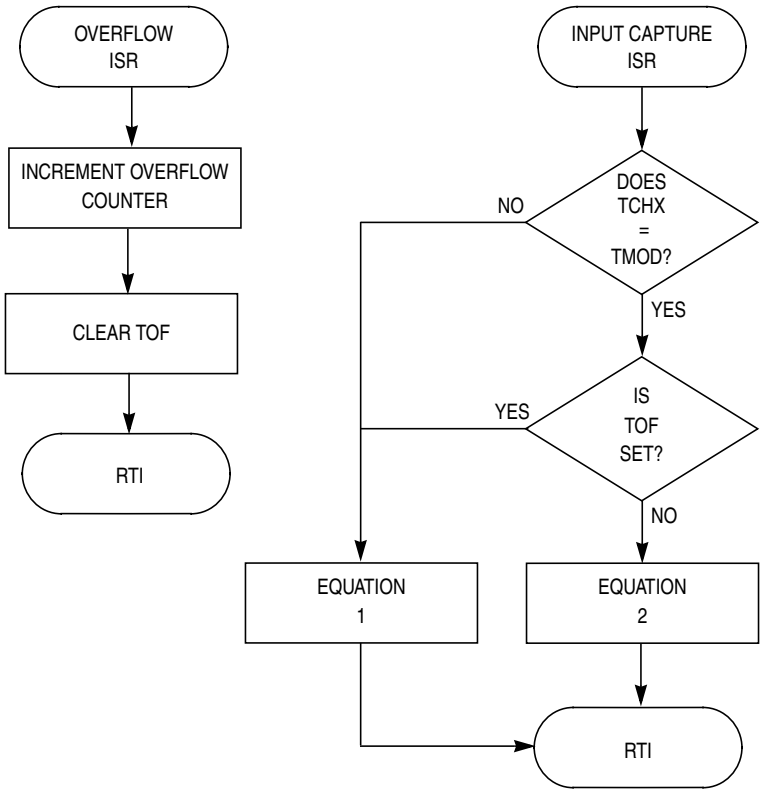


Figure 3. Input Capture ISR Technique Flowchart

Flow Chart Explanation

1. Once an input capture has occurred, software checks to see whether the value in the timer channel register is equal to the value in the timer modulo register.
2. If the value in the timer channel register is not equal to the value in the timer modulo register, then Equation 1 must be used.
3. If the value in the timer channel register is equal to the value in the timer modulo register, then software checks to see whether there is a pending TOF interrupt.
4. If the value in the timer channel register is equal to the value in the timer modulo register and there is a pending TOF interrupt, then Equation 1 must be used.
5. If the value in the timer channel register is equal to the value in the timer modulo register and there is no pending TOF interrupt, then Equation 2 must be used.
6. Overflow ISR simply increments the overflow counter.

Equation 1: (Ovf * (TMOD + 1)) + TCHX

Where: Ovf = The number of overflows
TMOD = The value in the timer counter modulo register
TCHX = The contents of the timer channel register

Equation 2: ((Ovf – 1) * (TMOD + 1)) + TCHX

Where Ovf = The number of overflows
TMOD = The value in the timer counter modulo register
TCHX = The contents of the timer channel register

Testing for Input Capture ISR Software Solution

Testing was completed using an MMDS0508 and an M68EML08AB32 emulator module. The timer modulo value was set to the default, $FFFF. Three test conditions were identified and tested using Metrowerks CodeWarrior IDE.

- **Test condition 1** — An input capture occurs when the timer channel register does not equal $FFFF (**Figure 1(d)**).
- **Test condition 2** — An input capture occurs when the timer channel register equals $FFFF after a TOF interrupt has occurred (**Figure 1(a)**).
- **Test condition 3** — An input capture occurs when the timer register equals $FFFF before a TOF interrupt has occurred (**Figure 1(c)**).

For test condition 1, a pullup resistor was used to toggle PortE2 (timer channel 0) so that an input capture would occur at some random time where the timer channel register did not equal $FFFF (**Figure 1(d)**). For test condition 2, PortE2 was connected to PortC4. Once a TOF interrupt occurred, PortC was toggled so that an input capture would occur while the timer channel register was equal to $FFFF (**Figure 1(a)**). Only divide-by 64 and divide-by 32 were tested for this condition because the smaller divide-by values did not allow proper timing. For test condition 3, timer channel 1 was set up for output compare. PortE2 was connected to PortE3 (timer channel 1) so that when timer channel 1 output compare occurred at $FFFF, an input capture was triggered on timer channel 0 (**Figure 1(c)**).

Conclusion

Both software techniques provided in this document outline reasonable solutions to the early TOF issue. Use either of these techniques to obtain the full functionality of the input-capture feature.

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.; SPS, Technical Information Center,
3-20-1, Minami-Azabu Minato-ku, Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.;
Silicon Harbour Centre, 2 Dai King Street,
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

http://www.motorola.com/semiconductors

MOTOROLA