CpE 390 Microprocessor Systems

Lab 6: Timer Function

1. Introduction

The purpose of the timer module is to allow for time critical operations to be handled mostly by hardware, freeing up the software to perform other tasks. For example, generating or measuring waveforms can be done with minimal software interaction using the timer module.

The 68HC12 Standard Timer Module consists of a 16-bit counter timer that is clocked by a programmable pre-scaler. It provides eight input-capture/output-compare channels and a pulse accumulator. The eight inputs/outputs of the timer module share pins with Port T. The operation of the Timer Module is explained in detail in the textbook (Chapter 8) and also in your lecture notes.

In order to use the timer module, the programmer must set up and use a number of registers:

```
TIOS:   EQU  $0040    ; Timer Input / Output Select
TCNT:   EQU  $0044    ; Main Timer Counter Register
TSCR1:  EQU  $0046    ; Timer System Control Register 1
TCTL1:  EQU  $0048    ; Timer Control Register 1
TCTL2:  EQU  $0049    ; Timer Control Register 2
TCTL3:  EQU  $004A    ; Timer Control Register 3
TCTL4:  EQU  $004B    ; Timer Control Register 4
TIE:    EQU  $004C    ; Timer Interrupt Enable Register
TSCR2:  EQU  $004D    ; Time System Control Register 2
TFLG1:  EQU  $004E    ; Timer Interrupt Flag Register 1
TFLG2:  EQU  $004F    ; Timer Interrupt Flag Register 2
TC0:    EQU  $0050    ; Timer Capture/Compare Register 0
TC1:    EQU  $0052    ; Timer Capture/Compare Register 1
TC2:    EQU  $0054    ; Timer Capture/Compare Register 2
TC3:    EQU  $0056    ; Timer Capture/Compare Register 3
TC4:    EQU  $0058    ; Timer Capture/Compare Register 4
TC5:    EQU  $005A    ; Timer Capture/Compare Register 5
TC6:    EQU  $005C    ; Timer Capture/Compare Register 6
TC7:    EQU  $005E    ; Timer Capture/Compare Register 7
```

The TSCR1 register controls the basic behavior of the timer module. Bit 7 enables the timer. Bit 4 determines if interrupt flags are cleared automatically when data is read. The eight bits of the TIOS register determine whether each of the eight channels is to function as an input-capture (0) or output-compare (1) operation.

The TSCR2 register enables timer-counter overflow interrupts and selects the pre-scalar division ratio. The pre-scalar divides the bus clock by 2^n where $0 \leq n \leq 7$. This divided clock becomes the input clock to the main timer counter register. The TIE register enables interrupts on a per-channel basis when input-capture or output –compare events happen on any of the eight channels.

TCTL1 and TCTL2 select, on a per channel basis, the output action to be sent to an output pin when an event occurs on the corresponding output-compare channel. TCTL3 and TCTL4 select, on a per channel basis, the input transition that will trigger an event on the corresponding input-capture channel.

TFLG1 indicates, on a per-channel basis, when an input-capture/output-compare event has occurred on the corresponding channel. TFLG2 indicates when a timer-counter overflow event has occurred.

TC0~TC7 are the Input-Capture/Output-Compare data registers. In Input-Capture mode, they indicate the time at which an input event occurred. In Output-Compare mode, they indicate at what time an output event should be generated.

2. Input Capture Mode

(a) Use the HP signal generator on your bench to generate a 1.0 kHz, 0-5V square wave as an input signal to the EVI board. *(Make sure the signal generator waveform does not exceed 5V before connecting it to the EVB to avoid damage to the microcontroller).* We will be using input-capture channel 0 to measure the period of this waveform. The EVB board uses a 16 MHz crystal oscillator which is used to generate an 8 MHz E-clock. If the pre-scaler is set to divide by 8, what will be the units of time measured by the timer-counter clock?

(b) Connect the 1kHz square wave to pin PT0. Enter, assemble, load and execute the code on the following page. Examine the contents of location labeled *period*. Verify that the program correctly measures the period of waveform in units of the timer counter.

(c) Modify the program to measure the pulse-width (i.e., the time that the waveform is high). Do you get the value you expect?

```
TIOS:     EQU   $0040          ; Timer Input / Output Select
TSCR1:    EQU   $0046          ; Timer System Control Register 1
TCTL4:    EQU   $004B          ; Timer Control Register 4
TSCR2:    EQU   $004D          ; Time System Control Register 2
TFLG1:    EQU   $004E          ; Timer Interrupt Flag Register 1
TC0:      EQU   $0050          ; Timer Capture/Compare Register 0

          ORG   $5000
edge1:    DS.W  1              ; location to save first edge
period:   DS.W  1              ; location to save period (in pres-scaled cycles)

          ORG   $4000
          movb  #$80, TSCR1    ; enable timer counter
          bclr  TIOS, $01      ; enable input-capture 0
          movb  #$03, TSCR2    ; disable TCNT overflow and set pre-scale=8
          movb  #$01, TCTL4    ; capture rising edge of PT0 signal
          movb  #$01, TFLG1    ; clear the C0F flag
          brclr TFLG1, $01, *  ; wait for arrival of first edge
          ldd   TC0            ; get first edge and clear C0F flag
          std   edge1          ; save first edge
          movb  #$01, TFLG1    ; clear C0F flag
          brclr TFLG1, $01, *  ; wait for second edge
          ldd   TC0            ; get second edge
          subd  edge1          ; compute period (in pre-scaled cycles)
          std   period         ; save the result
          swi
```

3. Output Compare Mode

We will now use the timer module in output-compare mode to generate a 1 kHz square wave with a 30% duty cycle on pin PT5. Enter, assemble, load and execute the code on the following page. Check the output waveform on PT5 using the oscilloscope. Note that rather than polling the timer-counter flag, waiting for an event (as was done in part 1); this program uses interrupts to toggle the output. This frees the processor to perform other tasks. This will be useful in part 4.

```
TIOS:     EQU   $0040          ; Timer Input / Output Select
TCNT:     EQU   $0044          ; Main Timer Counter Register
TSCR1:    EQU   $0046          ; Timer System Control Register 1
TCTL1:    EQU   $0048          ; Timer Control Register 1
TIE:      EQU   $004C          ; Timer Interrupt Enable Register
TSCR2:    EQU   $004D          ; Time System Control Register 2
TFLG1:    EQU   $004E          ; Timer Interrupt Flag Register 1
TC5:      EQU   $005A          ; Timer Capture/Compare Register 5
HiCnt:    EQU   300            ; High period in timer-counter clock periods
LoCnt:    EQU   700            ; Low period in timer-counter clock periods

          ORG   $5000
HiLo:     DS.B  1              ; flag to indicate current output level (0 or 1)

          ORG   $4000
          movw  #OC5isr, $0FE4 ; set up OC5 interrupt vector
          movb  #$80, TSCR1    ; enable TCNT
          movb  #$03, TSCR2    ; disable TCNT overflow & set pre-scaler to 8
          bset  TIOS, $20      ; enable OC5
          movb  #$0C, TCTL1    ; configure OC5 action to “pull high”
          ldd   TCNT           ; start OC5 with delay =LoCnt
          addd  #LoCnt
          std   TC5
          movb  #$20, TFLG1    ; clear C5F flag
          clr   HiLo           ; set current output flag = 0
          bset  TIE, #$20      ; enable OC5 interrupt
          cli                  ; enable (global) interrupts
forever:  bra   forever        ; while OC5 generates output waveform

OC5isr:   movb  #$20, TFLG1    ; clear C5F flag
          tst   HiLo           ; what is old output level?
          bne   now_lo         ; if one, then we just changed to zero
now_hi:   movb  #$08, TCTL1    ; just changed to one – set action to pull low
          ldd   TCNT
          addd  #HiCnt         ; restart with delay=HiCnt
          std   TC5
          inc   HiLo           ; current output level=1
          rti
now_lo:   movb  #$0C, TCTL1    ; set action to pull high
          ldd   TCNT
          addd  #LoCnt         ; restart with delay=LoCnt
          std   TC5
          clr    HiLo          ; set current output level=0
          rti
```

4. Linking the Input Capture and Output Compare Codes

Now we are going to combine parts 2 and 3 to write a program that generates a 500Hz square wave which is high for a period of time equal to the period of the signal present on input-capture channel 0. Because the waveform generator (part 3) only uses interrupts to produce its output, it is possible to run these two programs simultaneously.

(a) Begin your program with the output compare code from part 3. This will set up the interrupt driven channel 5 waveform generator.
(b) At the end of the main program of part 3, add the input capture code from part 2 in place of the "bra forever" instruction. This added code will determine the period of the input waveform on channel 0. Note that parts (2) and (3) set up many of the same registers in the timer module. Make sure the codes from these two parts do not conflict.
(c) Instead of executing a **swi** at the end of the part 2 code, branch back and run the part 2 code in an infinite loop, continuously recalculating and updating the measured period of the signal on channel 0.
(d) In the interrupt service routine from part 3, use the measured "period" from the part 2 to set the high level delay (instead of using HiCnt). Set the low level delay so that the total period of the output waveform is 2 ms.
(e) When you execute this new code, you should initially see a 500 Hz square wave with a 50% duty cycle. Now change the frequency of the HP signal generator from its initial frequency of 1 kHz. You should see the high period of your output waveform track the period of the signal generator waveform. Show your oscilloscope display to your TA.