ECE 3731: Microproc & Embedded Sys Lab

Lab 6: Timer Subsystem - Flag Polling and Interrupts

August 2nd, 2018

Summer 2018

Honor Code: I have neither given nor received unauthorized assistance on this graded report.

x Brian Vera-Burgos   x Srinivas Simhan

Table of Contents

Objective **3**

Equipment Used **3**

Lab Assignment **4**

- Flowchart 4
- CPU Modification 4
- Change NUMTEXT 5

Post Lab Assignment **6**

Conclusion **10**

- Assignment 10
  - Flowchart 10
  - CPU Modification 10
  - Change NUMTEXT 10
- Post-Lab 10

Objective

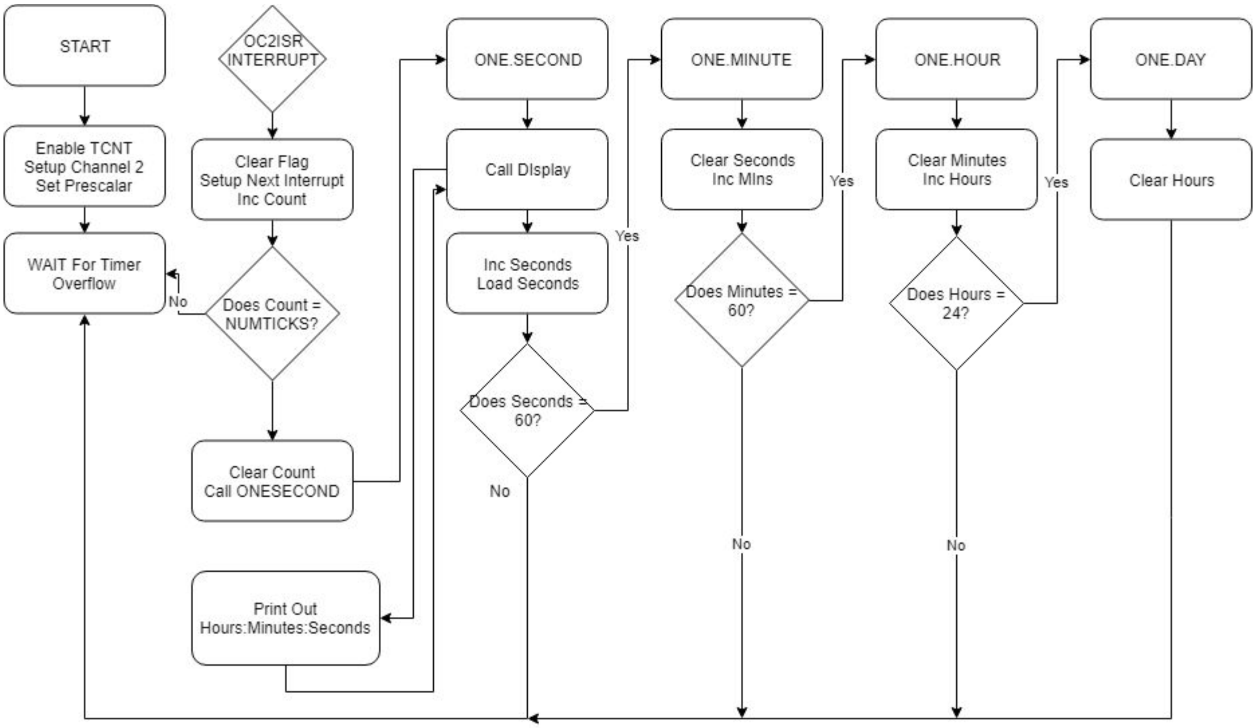
- To become familiar with CodeWarrior
- To become familiar with HCS Dragon12-Light
- To become familiar with setting up timers using the TCNT register
- To become familiar with using TOC and TOF when using the TCNT register
- To become familiar with “polling” interrupt flags
- To understand and create flowcharts to be used in explanations in terms of documentation for our code

Equipment Used

- CodeWarrior
- HCS Dragon12-Light

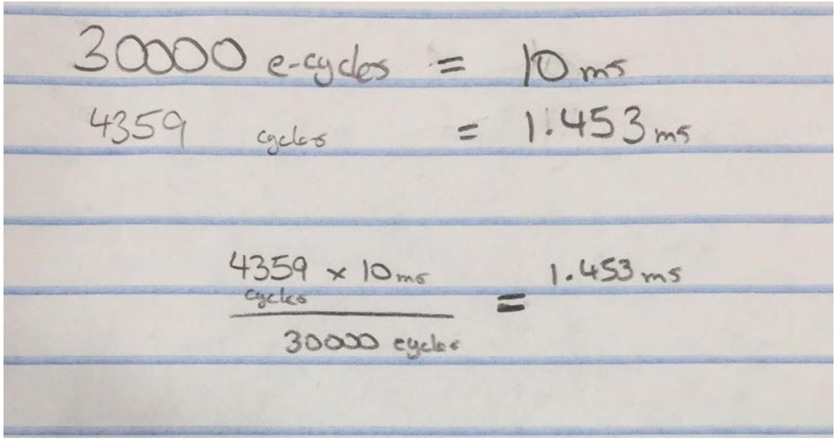
Lab Assignment

A. Flowchart



B. CPU Modification

a. In Part B, we were asked to replace adding 30,000 in CPU register D, and instead add the last 4 digits of our UMID (4359). The behavior that we saw was the the clock's speed increased very quickly. At 30,000 cycles, there was a delay incurred of about 10ms. After doing calculations, at 4359 cycles, there was a delay of about 1.453ms. Based on the visible behavior of the clock, and our calculations, we believe this to be accurate.



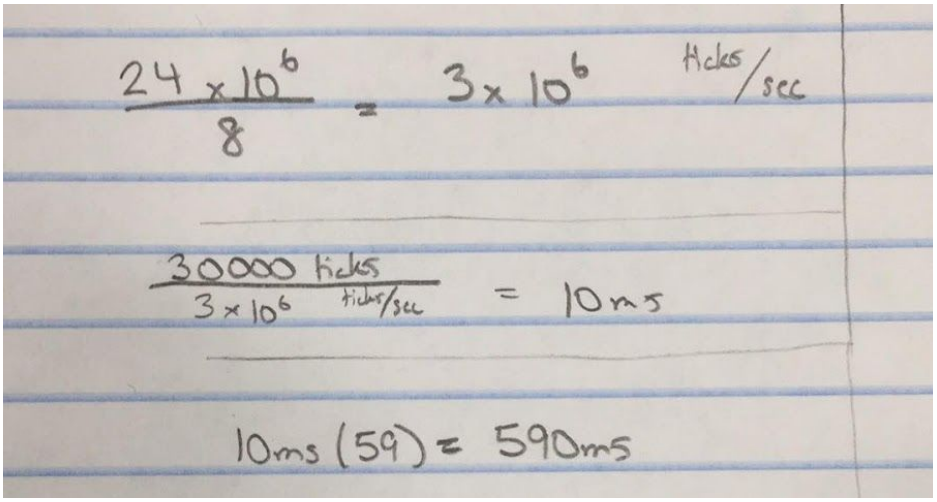
30000 e-cycles $= 10$ ms

4359 cycles $= 1.453$ ms

$$\frac{4359 \times 10\text{ms}\ \text{cycles}}{30000 \text{ cycles}} = 1.453\text{ms}$$

C. Change NUMTEXT

a. In this section, we changed the value of NUMTICKS to 59. This changed the timing for 1 “second” to 590ms. Calculations that led to this are seen below:



$$\frac{24 \times 10^6}{8} = 3 \times 10^6 \text{ ticks/sec}$$

$$\frac{30000 \text{ ticks}}{3 \times 10^6 \text{ ticks/sec}} = 10\text{ms}$$

$$10\text{ms}\,(59) = 590\text{ms}$$

Post Lab Assignment

- The Post Lab question asks us to create a timer using a polling method instead of a standard timer interrupt. The pace of this timer can be modified using NUMTICKS.
- Full Code:

```
;*****************************************************************
;* ClockASM.ASM
;
;Code Entry, Assembly, and Execution
;(Put your name and date here)
;---------------------------------------
;* -this is the sample code for Lab1
;* -for Full Chip Simulation or Board -- select your target
;* DO NOT DELETE ANY LINES IN THIS TEMPLATE
;* --ONLY FILL IN SECTIONS
;*****************************************************************
; export symbols
          XDEF Entry, _Startup          ; export 'Entry' symbol
          ABSENTRY Entry        ; for absolute assembly: mark this as application entry point

; Include derivative-specific definitions
                           INCLUDE 'derivative.inc'

;--------------------------------------------------
; Equates Section
;----------------------------------------------------
ROMStart    EQU  $2000  ; absolute address to place my code
TEN     EQU   $80
OC2VEC  EQU   $3E6A  ; vector under D-bug12 (board)
OC2VECSIM EQU $FFEA  ; simulation uses actual vector
C2F     EQU   $04
C2I     EQU   $04
IOS2    EQU   $04
;----------------------------------------------------
; Variable/Data Section
;----------------------------------------------------
          ORG RAMStart   ; loc $1000  (RAMEnd = $3FFF)
; Insert here your data definitions here

COUNT       DS   1
NUMTICKS    DS   1
SECONDS     DS   1   ;keeps track of seconds
MINUTES     DS   1   ;keeps track of minutes
HOURS       DS   1   ;keeps track of hours


      INCLUDE 'utilities.inc'
      INCLUDE 'LCD.inc'

;----------------------------------------------------
; Code Section
;----------------------------------------------------
          ORG   ROMStart  ; loc $2000
```

```
Entry:
_Startup:
            ; remap the RAM &amp; EEPROM here. See EB386.pdf
 ifdef _HCS12_SERIALMON
            ; set registers at $0000
            CLR   $11                  ; INITRG= $0
            ; set ram to end at $3FFF
            LDAB  #$39
            STAB  $10                  ; INITRM= $39

            ; set eeprom to end at $0FFF
            LDAA  #$9
            STAA  $12                  ; INITEE= $9
            JSR   PLL_init      ; initialize PLL
  endif

;-----------------------------------------------------
; Insert your code here
;-----------------------------------------------------

MAIN
*SET UP THE (interrupt) SERVICE & INITIALIZE
        SEI             ; turn off interrupts while initializing intr.
        JSR   TermInit  ; Initialize Serial Port when not
                     ; ...using built-in DBUG12 utilities
                CLR   COUNT
        CLR   SECONDS
        CLR   MINUTES
        CLR   HOURS
        MOVB  #100,NUMTICKS  ; number of ticks (interrupts) for 1 second
                                        ; this was changed to MOVB #59, NUMTICKS

*SET UP THE SERVICE (ISR) & INITIALIZE -continued

       ;bset    DDRT,%00100000   ; PT5 (spkr) is output
       movb    #$80,TSCR1     ; enable TCNT
       bset    TIOS,IOS2      ; choose OC2 for timer ch. 2
       movb    #$20,TSCR2     ; set prescaler to 32
       movb    #C2F,TFLG1    ; clear  C2F flag initially
       bset    TIE,C2I     ; arm OC2
       SEI                  ; disallow interupts
POLL
         BRCLR TFLG2, %10000000, CLEAR   ;Check if overflow is high
         JSR OC2ISR                     ;Call counting function
CLEAR

       BRA     POLL
*====END OF MAIN ROUTINE

*============= SERVICE PROCESS
OC2ISR
        MOVB   #$80,TFLG2   ; clear flag
                                    ; remove cycles since we’re doing polling for the second half
        INC    COUNT   ; one more interrupt interval counted
        LDAB   COUNT
        CMPB   NUMTICKS
        BNE    DONE  ; not one second yet so return
```

```
    CLR    COUNT
    JSR    ONE.SECOND  ; one second has elapsed
DONE    RTS
*============= END OF SERVICE ROUTINE
ONE.SECOND    ; what to do every second
    JSR   DISPLAY
    INC   SECONDS
    LDAA  SECONDS
    CMPA  #60
    BEQ   ONE.MINUTE
    RTS
ONE.MINUTE
    CLR   SECONDS
    INC   MINUTES
    LDAA  MINUTES
    CMPA  #60
    BEQ   ONE.HOUR
    RTS
ONE.HOUR
    CLR   MINUTES
    INC   HOURS
    LDAA  HOURS
    CMPA  #24
    BEQ   ONE.DAY
    RTS
ONE.DAY
    CLR   HOURS
    RTS

DISPLAY  ; DISPLAY THE TIME AS HH:MM:SS
    PSHB
Simulation  EQU  1
   ifndef  Simulation
; Simulation--cannot interpret backspace character
    LDAB  #8  ; backpace to beginning of display line
    JSR   putchar
    JSR   putchar
    JSR   putchar
    JSR   putchar
    JSR   putchar
    JSR   putchar
    LDAB   #$0D
    JSR   putchar
   endif

    LDAB   HOURS
    JSR    OUTDEC
    LDAB   #':'
    JSR    putchar
    LDAB   MINUTES
    JSR    OUTDEC
    LDAB   #':'
    JSR    putchar

    LDAB   SECONDS
    JSR    OUTDEC
    LDAB   #$0D
```

```
        JSR   putchar
        LDAB   #$0A
        JSR   putchar
        PULB
        RTS

HEX2BCD  ; assumes value to be converted is in ACC A and result in A
        TFR    A,B   ; make copy in B
UP      CMPB   #10
        BLO    DONE2
        SUBB   #10
        ADDA   #6
        BRA    UP
DONE2
       RTS

OUTDEC
        TFR    B,A   ; HEX2BCD takes input from A
        JSR    HEX2BCD
        TFR    A,B     ; putchar needs value in B
        LDX   #0  ;
        JSR   out2hex   ; output B as 2 hex digits
        RTS


;**************************************************************
;*                 Interrupt Vectors                          *
;**************************************************************
            ORG   Vreset
            DC.W  Entry         ; Reset Vector

                  ORG   Vtimch2       ; setup  OC2 Vector
            DC.W  OC2ISR
```

Conclusion

A. Assignment

a. Flowchart

i. In Part A we learned about setting up a timer to count and also how to us that timer to set off an interrupt. This interrupt was used to be a more accurate timer than the one we made in a previous lab.

b. CPU Modification

i. We learned that by changing the number of cycles, the behavior of the clock also changes. In our case, we reduced the number of cycles from 30000 to 4359 cycles, and as a result we saw an increased speed from approximately 10ms to 1.453ms.

c. Change NUMTEXT

i. In part C we further learned to manipulate this timer setup by changing the value of NUMTICKS. This value directly corresponds to the seconds value and allows us to decide what a “second” is.

B. Post-Lab

a. In the postlab we learned how to utilize the exact same program but by polling for the overflow bit rather than using an interrupt. This method works exactly the same as before but the only way to modify the timing is by using NUMTICKS and with a prescaler. The comparator was disabled in order to accomplish this program.