EECS 388: Computer Systems and Assembly Language

Homework 5 Solution

1. (20) How many RTI interrupt events must occur to generate a 15 minute delay assuming the MCLK is operating at 2MHz and the RTR[2:0] bits are set for "110"? How do you set up the Real-Time Interrupt Control Register (RTICTL) (i.e., enable RTI and set RTI pre-scale) for this purpose?

According to the table on page 229 of your textbook, the period of a RTI interrupt is set by the MCLK frequency divided by a divisor stored in RTR[2:0]. Therefore:

RTR[2:0] = "110" implies a clock divider of 2^{18}.
MCLK = 2 MHz and a divider of 2^{18} implies that the frequency of RTI interrupts is 7.6294 Hz (2 MHz / 2^{18}).
The time delay, or period, between RTI events is thus 1/7.6294 Hz = 0.1311 seconds (because period = 1 / frequency).

Now, if we want a delay of 15 minutes:

15 minutes = 900 seconds.
Number of RTI events for 15 minutes =
- (900 seconds)/(0.1311 seconds per delay) = 6,867.
- Thus, it will require about 6,867 RTI events

One must write the appropriate values into the RTICTL in order to enable interrupts and to set the RTI frequency. The RTICTL register is a memory-mapped location located at address $0014. The RTIE bit is the MSB (bit 7), while the RTR[2:0] bits are the least-significant 3 bits (bits 2 through 0).

```
RTICTL      EQU $0014 ; Equate for the address of the RTICTL register

            LDAA %10000110       ; RTICTL mask (RTIE ='1', RTR[2:0] = "110")
            STAA RTICTL      ; Store the value into RTICTL
```

2. (15) Textbook, page 291. Advanced problem #4. **Change MCLK to 4 MHz**.

Assuming MCLK is at 4 MHz (as stated above), and the pre-scaler is set to 1, this means that the timing frequency is (4 MHz)/(2^1) = 2 MHz, or a period of 500 ns.

If the two counts (or timestamps) are $1993 and $07C8, then the period of the measured signal (assuming no counter rollovers) is $EE34, which is 60,980 counter ticks in decimal. This period, in real-time, is thus:

(($FFFF - $1993) + $07C8) + 1 → 60,981 ticks and 60,981 ticks * (500 ns / 1 tick) = 30.4905 ms.

3. (15) Textbook, page 291. Advanced problem #5.

If the period of the pulse being measured is greater than the rollover time for the counter, then one must make sure to detect counter rollovers in order to accurately measure the signal of interest. Think of this process as noting every New Years Eve from when you were born until the present time in order to figure out how old you are. This requires one to modify the program to log every counter rollover (pulse-accumulator overflow bit, or PAOVF).

Given the numbers from above (problem #2), we know that counter is adjusted by 1 every 500 ns, and that the counter will rollover when it reaches 2^{16}, or 65,536. This means that pulse length of interest can be found by counting counter rollovers:

Period= # rollovers + # of extra ticks = (500 ns / 1 tick) *[(# of rollovers)*(65,536 ticks/ 1 rollover)+ (current ticks)]

4. (25) Write a program to measure the period of a periodic signal connected to input channel 3 by measuring the count difference between two falling edges. Set PR2:PR0 = 011. Use polling method.

This program is very much like the example program found on pg. 273 of the textbook.

```
; ***********************************************
; Program Definitions
; ***********************************************
REG_BASE    EQU    $0000 ; Base address for calculating offsets to other registers
TMSK1               EQU    $8C     ; Offset for TMSK1 register
TMSK2               EQU    $8D     ; Offset for TMSK2 register
TCTL4       EQU    $8B     ; Offset for TCTL4 register
TIOS        EQU    $80     ; Offset for TIOS register
TC3H        EQU    $96     ; Offset for TC3H register
TSCR        EQU    $86     ; Offset for TSCR register
TFLG1               EQU    $8E     ; Offset for TFLG1 register
TCNT        EQU    $84     ; Offset for TCNT register
TMSK2_IN    EQU    $03     ; Set the pre-scale bits
TCTL4_IN    EQU    $80     ; Configure falling edges (10)
TIOS_IN     EQU    $00     ; Select channel 3 for input compare
TSCR_IN     EQU    $80     ; Enable timer
CLR_CH3     EQU    $08     ; Mask to clear channel 3 flag

        ; Data section
        ORG $6000

edge1       FDB    $0000 ; Reserve a word (16-bits) for edge measurement
period              FDB    $0000 ; Reserve a word (16-bits) for period
measurement

        ; Code section
        ORG $4000

        LDS #$8000   ; Initialize the stack pointer
        JSR     TIMERINIT      ; Initialize the timer
        JSR     MEASURE        ; Measure the period

        SWI ; end the program




; ***********************************************
; Function used to enable timer subsystem
; ***********************************************
TIMERINIT
         CLR TMSK1    ; disable interrupts
         LDX #REG_BASE      ; Load X with base address of registers

         LDAA #TMSK2_IN     ; Set pre-scale
         STAA TMSSK2, X

         LDAA #TCTL4_IN     ; Configure for falling edges
```

```
        STAA TCTL4, X

        LDAA #TIOS_IN       ; Select channel 3
        STAA TIOS_IN, X

        LDAA #TSCR_IN       ; Enable timer
        STAA TSCR_IN, X

        RTS    ; return

; ***************************************************
; Function used to measure signal period
; via polling method
; ***************************************************
MEASURE
       LDAA #CLR_CH3       ; Clear channel 3 flag to prepare measurements
       STAA TFLG1,X
       ; Grab measurement of first edge
WAIT1
       BRCLR TFLG1,X,$08,WAIT1   ; Wait for an edge

       LDD TCNT,X                ; Load in counter value
       STD edge1                 ; Save the measurement

       LDAA #CLR_CH3             ; Clear channel 3 flag again
       STAA TFLG1,X

       ; Grab measurement of second edge
WAIT2
       BRCLR TFLG1,X,$08,WAIT2   ; Wait for an edge
       LDD TCNT,X                ; Load in counter value
       SUBD edge1                ; Calculate the difference between edges
       STD period                ; Store the period result

       RTS     ; return
```

5. (25) Generate a 1500Hz square wave with a 40% duty cycle (ON/PERIOD) on output compare channel 2 (OC2). MCLK = 8MHz. Set the pre-scaler to divide by 4. Use interrupt.

This program is very much like the example program found on pg. 275 of the textbook except that it uses interrupts. If the MCLK runs at 8 MHz and the pre-scaler is set to 4, then the counter will adjust at a rate of (8 MHz)/4 = 2 Mhz, or with a period of 500 ns.

A 1500 Hz signal has a period of 666.67 microseconds, and a 40% duty cycle means that it will be high for 0.4*666.67 microseconds or 266.67 microseconds, and low for 400 microseconds. This translates to counter value of:

High counter:
= 266.67 microseconds * (1 tick / 0.5 microseconds)
= 534 ticks → $0216
Low counter:
= 400 microseconds * (1 tick / 0.5 microseconds)
= 800 ticks → $0320

```
; Program Equates for interrupts
INTCR    EQU  $001E
INTCR_IN      EQU  $60
; Program Equates for the timer circuitry
TMSK1  EQU  $008C
TMSK2  EQU  $008D
TCTL2   EQU  $0089
TIOS          EQU  $0080
TC2H          EQU  $0094
TSCR          EQU  $0086
TFLG1   EQU  $008E
TMSK1_IN      EQU  $04
TMSK2_IN      EQU  $02
TCTL2_IN      EQU  $10
TIOS_IN EQU  $04
TSCR_IN       EQU  $80
HIGH_TIME     EQU  $0216
LOW_TIME      EQU  $0320
   ORG $FFEA ; Register interrupt vector for timer channel 2
   FDB MY_IRQ

   ORG $4000 ; Initialize timer channel 2 and interrupts

   LDS #$8000;       Setup the stack

   MOVB #TMSK1_IN, TMSK1     ; Enable interrupts on channel 2
   MOVB #TMSK2_IN, TMSK2     ; Set prescale to 4
   MOVB #TCTL2_IN, TCTL2     ; OC2 toggle on compare
   MOVB #TIOS_IN, TIOS       ; Select channel 2 for OC
   MOVW #HIGH_TIME, TC2H     ; Setup initial high time
   MOVB #TSCR_IN, TSCR       ; Enable timer
   LDAA TFLG1
   ORAA #$04        ; Clear timer flag
   STAA TFLG1

   MOVB #INTCR_IN, INTCR     ; Setup interrupts
   CLI    ; Enable interrupts
```

```
LOOP   BRA LOOP    ; Infinitely loop

MY_IRQ
   LDAA TFLG1
   ORAA #$04        ; Clear timer flag
   STAA TFLG1
   ; Setup duty cycle for next pulse
   ; Compare to low, if low then switch to high and vice-versa
   LDD TC2H
   CPD #LOW_TIME
   BEQ SET_HI
   MOVW #LOW_TIME, TC2H      ; Setup next pulse width
   BRA DONE
SET_HI
   MOVW #HIGH_TIME, TC2H     ; Setup next pulse width

DONE
   RTI    ; Return from interrupt
```