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 7.63 Hz (2 MHz / 2^{18}).
- The time delay, or period, between RTI events is thus 1/7.63 Hz = 0.13 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)/(0.13 seconds per delay) = 6,293.08.
  - Thus, it will require about 6,294 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 $0C78, then the period of the measured signal (assuming no counter rollovers) is $D1B, which is 3,355 counter ticks in decimal. This period, in real-time, is thus:

3,355 ticks * (500 ns / 1 tick) = 1677500 ns → 1.6775 ms.

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

If the period of the pulse being measured is greater than the rollover time fo 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    $96      ; Offset for TCNT register
TMSK2_IN     EQU    $03      ; Set the pre-scale bits
TCTL4_IN     EQU    $C0      ; Configure falling edges
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

DONE    BRA DONE ; Infinitely loop to halt 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,WTFLG         ; 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,WTFLG         ; 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. 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

The program on p. 275 can be modified to perform the 40% duty cycle switch by changing the following:

- TMSK2_IN needs to be changed to $04
  - This sets the correct pre-scale for our problem.
  - Pre-scale of 4 ($2^2 = 4$).
- The first instance of #$03E8 (in TIMERINIT) needs to be changed to the counter value for our high counter, or #$0216.
- The instance of #$03E8 (in SQWAVE) needs to be changed to alternate between the high and low-counter values using a conditional branch to implement an IF statement.
  - This can be easily done by reserving a word (double-byte) in memory to contain the count value. This value will "flip-flop" between the high and low counter value during every iteration of the loop (see the following modifications.

```
;***********************
; New data section
;***********************
ORG $6000
COUNT_INC        FDB $0000     ; Location to hold counter increment

.
.
.
<Within TIMERINIT>

        ; Initialize COUNT_INC to be low counter value
        LDD $0320
        STD COUNT_INC

<Replace SQWAVE with the following>
SQWAVE
        BRCLR       TFLG1,$04,SQWAVE        ; Poll for counter flag
        LDD         TC2H                    ; Load in counter value

        LDX   COUNT_INC                     ; Load in current COUNT_INC
        CPX   #$0216                                ; Compare to high-count
        BEQ   ADD_LOW                       ; If high → add low

        ADDD #$0216                         ; Otherwise, add high
        LDX   #$0216                                ; Update COUNT_INC
        STX   COUNT_INC

        BRA   ENDIF
ADD_LOW
        ADDD #$0320                         ; Add low
        LDX   #$0320                                ; Update COUNT_INC
        STX   COUNT_INC
ENDIF
        STD TC2H     ; Setup next transition time
        JSR CLEARFLAG ; generate repetitive signal
        RTS     ; return
```