Quiz2 ECE2560 Spring 2022

Collaboration with other students is not allowed

This quiz is to make sure that you have i) CCS running on your computer ii) you have purchased a launchpad for this class iii) you can create an assembly language project in CCS iv) you can compile and run a program on your microcontroller v) you can use the debugger in CCS, and vi) you can navigate to the RAM and FRAM regions of the microcontroller in the memory browser.

Use Code Composer Studio and your Launchpad to write and execute the assembly language code given on the last two pages of this document. Sometimes copying and pasting from a pdf document to CCS can introduce errors, it is better to type code into CCS yourself. Make sure to watch the CSS Tutorial Screencasts first.

i) Take a screenshot of the "General" tab of the properties screen of your project. Note: In CCS, right click on your project folder in "project browser" and choose "properties" to get to the properties screen.

Enter debugger. Execute your program (green play button) and pause it. While the program is paused (suspended) take the following screenshots (use 16 bit hex, C style, to display the data):

ii) Screenshot of the memory browser showing the beginning of RAM

iii) Screenshot of the memory browser showing the beginning of FRAM

Take a photograph of your Launchpad

*

iv) Attach the picture to your solution

Use the **word template** and **instructions** contained on our web site to submit your screenshots to Carmen. Do not email directly to your TA or me. Files emailed to the TA or me will not be accepted.

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
        .def    RESET               ; Export program entry-point to
                                    ; make it known to linker.
;-------------------------------------------------------------------------------
        .data                       ; Assemble into program memory.
        .retain                     ; Override ELF conditional linking                  ; and retain current section.
        .retainrefs                 ; And retain any sections that have
                                    ; references to current section.
var1:           .word   0x1, 0x2, 0x3, 0x4    ; 4 16-bit variables initialized  to values 1, 2, 3 and 4
var2:   .space  6                                   ; set aside space for 6 bytes (3 words) initialized to zero
scon1:          .set    0x500                       ; Equate the symbol scon1 to constant 0x500
;-------------------------------------------------------------------------------
        .text                       ; Assemble into program memory.
        .retain                     ; Override ELF conditional linking                  ; and retain current section.
        .retainrefs                 ; And retain any sections that have
                                    ; references to current section.
con3:           .word   0x400                       ;
con4:     .word   0x2,  0x4
        .retain                     ; Override ELF conditional linking
                                    ; and retain current section
        .retainrefs                 ; Additionally retain any sections
                                    ; that have references to current
                                    ; section
;-------------------------------------------------------------------------------
RESET     mov.w   #__STACK_END,SP       ; Initialize stackpointer
StopWDT   mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer
;-------------------------------------------------------------------------------
; Main loop here
        mov.w   &var1,  R10                     ; move
        mov.w   #var1,  R10
        mov.w   #var1+2, R10
        mov.w   &var1+2, R10
```

```
        dec.w   &var1
        inc.w       &var2+4

        mov.w   &var1, &var2+4
                    mov.w  #scon1, &var1+2

                    dec.w   &con3
                    add.w   &con4, &var2

Loop:               jmp                 Loop

;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
        .global __STACK_END
        .sect   .stack

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
        .sect   ".reset"            ; MSP430 RESET Vector
        .short  RESET
```