ECE 4175
Project Two
Display of Scaled Pot Value

Complete by:
Wednesday Jan. 18th for an A+

Reference:
Chapter 6 - Structured Assembler
Chapter 10 - ADC (just scan, for now)

Overview

For this project, the code for Project One should continue to work. In addition, you will cause the PC's monitor to update a single digit, 0-8, every second. The digit value is to be taken from the potentiometer output, read by the analog-to-digital converter and scaled to this range.

You will also modify all of the code, old and new, to conform to the structured assembler requirements of Chapter 6. When you are ready to assemble your new file, use

```
sasm  P2
```

instead of

```
mpasmwin  P2
```

This invokes Jess Meremonte Johnson's structured assembler preprocessor to take the P2.asm file and to replace all of the structured assembler constructs with conditional and unconditional branch instructions in a file called P2.apr. Then this new file is subjected to the mpasmwin assembler to produce the P2.hex file that will be downloaded to the QwikFlash board and run.

Modification of Project One Code for Structured Assembler Constructs

Modify the code for Project One, replacing each of the following with one of the structured assembler constructs of Figure 6-1. Indent the code within each construct two spaces. For examples, refer to Figure 6-7. Here is some of the code needing modifying:

- "Loop ... bra Loop" in the Mainline program
- "bnz BAend" in the BlinkAlive subroutine
- "btfss PIR1,TMR2IF" in the LoopTime subroutine

Test your modified code before going on to the next part of this project. For all of your new code for the project, use the structured assembler constructs.

PotDisplay Subroutine

This subroutine, called each time around the mainline loop, decrements a one-byte variable, **HUNDRED**. If the result is zero, it reloads **HUNDRED** with a value of 100 and calls a **ReadPot** subroutine that converts the input from the potentiometer to the ADC's AN4 input as an eight-bit value and stores the result in a variable called **POTVALUE**. Then it multiplies **POTVALUE** by 9 and takes the upper byte of the two byte product in **PRODH:PRODL** and stores it in **SMALLPOT.** This value will range between 0 and 8. Finally, it calls the **TXbyte** subroutine described below twice. The first call sends the ASCII code for a "carriage return", 0x0d, to the PC. Then 0x30 is added to **SMALLPOT** and the result is put into **W** and this result (which is the ASCII code for the value in **SMALLPOT**) is sent to the PC. Finally, it returns (to

the mainline loop).

ReadPot Subroutine

The first two lines of the **Initial** subroutine have already been written to enable the analog-to-digital converter, to select the pot input on AN4 (what would otherwise be bit 5 of **PORTA**), and (when triggered to begin a conversion) to put the upper eight bits of the ten-bit converted value into **ADRESH**. Accordingly, whenever a conversion is desired, just call the following subroutine:

```
;;;;;;; ReadPot subroutine ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; This subroutine reads the potentiometer and puts the upper byte into ADRESH.

ReadPot
       bsf  ADCON0,GO_DONE              ;Initiate conversion
       REPEAT_
       UNTIL_  ADCON0,GO_DONE == 0      ;Wait for completion of conversion
       return                           ;Return with result in ADRESH
```

TXbyte Subroutine

```
;;;;;;; TXbyte subroutine ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; This subroutine first waits on the UART if a byte is in the process of being
; sent.  Then it sends the content of WREG to the PC.

TXbyte
       REPEAT_                          ;If an earlier transmission is still
       UNTIL_ PIR1,TXIF == 1            ;in progress, then wait
       movwf TXREG                      ;Send new byte from WREG
       return
```

Indenting of Your Stuctured Assembly Code

From the same DOS prompt

c:\Work>

used to assemble your code, use the "indent" feature of the structured assembler with the command line:

sasm -tabs -noasm P2.asm

as described on page 75. Then replace P2.asm with the generated P2.apr file:

copy P2.apr P2.asm

QwikPH Utility

Use Chris Twigg's QwikPH utility, described in Appendix Section A5.3, to update the program hierarchy in your file.

Comment Updates

Update the name of this code to P2 on the line 1 comment of your source file, P2.asm.
Update the description of what your code does in the first few lines of your source file.
Be sure to comment your new code with a comment on almost every line plus a header comment at the beginning of each new subroutine.