ECE 4175 **REVISED**
Project Four **1-24-06**
Positioning System

Complete by:
Wednesday February 1st for an A+

References:

Chapter 10	ADC
Handout	Keypad/resistor schematic
Section 2.5	Reading from program mem.
KeyCodes.inc	File to help interpret keypad
Section 2.5	Indirect addressing
Figure 7-12	ASCII table

Overview

Your code from Project Three should continue to work with the addition of the entry of the number of steps to be taken using a keypad. (The rate at which the steps are taken continues to be the rate you entered for Project Three.)

New Variables

For this project you will need to define:

```
NUML        ;Low byte of entered number
NUMH        ;High byte of entered number
NUMSTPSL    ;Low byte of NUMSTPS, the number of steps to take
NUMSTPSH    ;High byte of NUMSTPS
KEYSTRG:4   ;A four-byte string to hold digit keycodes entered in succession
FLAGS       ;The bits of FLAGS are used as state bits
```

KeyCodes.inc File

This file, available from the piclab website, will be used to convert the ADC output read by the **ReadKeypad** subroutine to that key's ASCII coding. When you are ready to assemble your P3.asm file, put the KeyCodes.inc file into the same directory. At the end of your P3.asm file and just before the **end** assembler directive, add the line:

```
#include KeyCodes.inc
```

The assembler will use this to insert a 256-byte table into your code. When assembled, this file will reside at the last 256 addresses of program memory before the addresses where QwikBug resides. That is, it will range from 0x5f00 to 0x5fff. (QwikBug resides in addresses 0x6000 up to 0x7fff.)

ReadKeypad Subroutine

This subroutine sets the ADC to select the AN7/RE2 input from the keypad as well as a right-justified result. Then it waits for 15μs (see Figure 10-7) before initiating a conversion. When the **GO_DONE** bit in the **ADCON0** register goes to zero, signaling the completion of the conversion, just return.

ReadPot Subroutine Modification

Change this subroutine so as to first select the AN4/RA5 input from the pot as well as a left-justified result. Then wait for 15 μs before initiating a conversion. Upon the completion of the conversion, copy **ADRESH** to **POTVALUE** and return.

Keypad Subroutine

This subroutine, called each time around the mainline loop, is to check for a new key pressed. Call the **ReadKeypad**. If bit 0 of **ADRESH** is 1, then no key is pressed. Clear bits 0 and 1 of
✔ **FLAGS** and return.

If bit 0 of **ADRESH** is 0 (indicating that a key is pressed) and if bits 1 and 0 of **FLAGS** are

- 00 (indicating that no key was pressed ten milliseconds ago) then just set bit 0 of **FLAGS** and return. This will debounce the pressed keyswitch.
- 11 (indicating that the pressed key has already been acted upon) then return.
- 01 (indicating that a key was pressed for the first time ten milliseconds ago and that any keybounce has settled out) then set bit 1 of **FLAGS** and copy **ADRESL** to **TBLPTRL**. Load **TBLPTRH** with the value 0x5f. Use the **tblrd*** instruction to copy the ASCII value of the key pressed into **TABLAT**. What happens next depends upon whether the pressed key was a digit, an *, or a #.
  - If it was an *, then store the ASCII code for a + sign (0x2b) in **SIGN**, add **NUMH:NUML** to **NUMSTPSH:NUMSTPSL** and call a **SendStrg** subroutine to send to the PC display <CR>, <LF>, **SIGN**, the digits in **KEYSTRG**, <CR>, <LF>. In this string of characters, the <LF> is a "line feed" code and is coded as
✔ 0x0a. Then call a **ResetKeypad** subroutine to clear **NUMH:NUML**, clear the four bytes of **KEYSTRG**, and to reinitialize **FSR0** so that it points to **KEYSTRG.**
  - If it was a #, then store the ASCII code for a - sign (0x2d) in **SIGN**, subtract **NUMH:NUML** from NUMSTPSH:NUMSTPSL, call **SendStrg** and call
✔ **ResetKeypad**.
  - If it was anything else (i.e., a digit)
✔
    - If **NUMH** is not zero, then call **ResetKeypad** to start over and fall through to the following action:
    - If **NUMH** is zero, then copy **TABLAT** into **KEYSTRG** using the instruction **movff TABLAT, POSTINC0**. Then convert **TABLAT** to its digit value, multiply **NUML** by ten, copy the result into **NUMH:NUML** and add **TABLAT** to it.

ControlStepping Subroutine Modification

For this project, **NUMSTPSH:NUMSTPSL** is a twos-complement-coded number representing the number of steps still to be taken. Modify this **ControlStepping** subroutine so that no step is taken if **NUMSTPSH:NUMSTPSL** equals zero. If **NUMSTPSH:NUMSTPSL** is positive, then take a CW step and decrement **NUMSTPSH:NUMSTPSL**. If **NUMSTPSH:NUMSTPSL** is negative, then take a CCW step and increment **NUMSTPSH:NUMSTPSL**.