

ECE 4175

Project Five

Stepper Motor Control via Interrupts

Complete by:
Thursday February 5th for an A+

References: Chapter 7 Interrupt Use
Chapter 8 Stepper Motor Control

Objective

For this project, you will switch from using the watchdog timer to awaken the MCU periodically to using Timer1, clocked via its 32768 crystal oscillator, for this purpose. This provides several beneficial results:

- The loop times are held to almost exactly 10 ms rather than to approximately 16 ms.
- The crystal oscillator will also be used to clock Timer3, allowing it to be used as a scale-of-N scaler. We will use this to control the stepping rate of a stepper motor, with the interval between steps set to $N/32768$ seconds $\approx 30.5 \times N$ microseconds.
- These timers can count crystal oscillator cycles regardless of whether the chip is asleep or awake.

You will also learn how to control the turning rate of a stepper motor.

To Do

Begin this project with your code from P4B.c. Add to this from the T3.c template program:

- the high- and low-priority interrupt stuff
- the initialization of Timer1 and Timer3 in Initial
- an initial display, for one second, of "STEPRATE"

Note that the low-priority interrupt is designed to wake up the chip every 10 milliseconds, as controlled by Timer1's 32768 Hz oscillator and Timer1. This sets the loop time to 10 ms. You will no longer enable the watchdog timer to awaken the chip.

You will modify the ReadPot function of the last project so that every tenth loop time (i.e., every tenth of a second) it will load ADRES with the *int* value from the ten-bit conversion of the pot.

Add a new function call, ControlStepRate, in the main loop that will also be called every tenth loop time and that will take ADRES, add one to it (so that we never deal with a value of zero), display this value on the LCD, and convert it to the number of counts of the 32768 Hz oscillator needed to produce a step rate equal to the value of ADRES + 1. Subtract this number from 65536 (the number of counts in a sixteen-bit counter). Temporarily disable interrupts with

```
INTCONbits.GIEH = 0;
```

copy this subtracted number into the *int* variable, STEPCNT, and reenable interrupts. (This will prevent the high-priority interrupt service routine from breaking into the middle of this update, possibly producing a bogus number.)

With no further modification, the high-priority interrupt service routine will produce steps at the desired step rate.