

# ECE 4175

## Project Four

### Faster Stepping

**Complete by:**  
Wednesday January 30<sup>th</sup> for an A+

**Reference:** Chapter 7 Interrupts

---

#### Overview

This project is a variation of Project Three. You will use high-priority interrupts to generate “ticks” every 1.6 milliseconds (i.e., about one-tenth of the loop time) and low-priority interrupts to awaken the CPU for its mainline tasks every 16 milliseconds (the same as the former loop time, produced with the watchdog timer in that case). You will do stepping from within the high-priority interrupt service routine. In this way, you will step at ten times the rate of the last project.

#### Initial Function

Initialize the Timer1 oscillator, Timer1, and Timer3 as described in Chapter 7 but for the loop time of 16 ms and the tick time of 1.6 ms. Also initialize any new variables.

#### Interrupt Service Routines

Modify the LPISR of Chapter 7 to produce 16 ms interrupts. Use the HPISR of Chapter 7 as a model. First thing, reload Timer3 to roll over again and generate the next interrupt in 1.6 milliseconds. Again use a signed char variable called **ACCUM** that initially equals 1. During each call, subtract **RATE** from **ACCUM**. If the result is negative, then add 62 to **ACCUM** and take a clockwise step. *Do not use function calls in HPISR*. As we will see in Chapter 12, a function call within an ISR causes the C compiler to set aside a massive variable list and to restore it at the end, taking up a lot of unnecessary time.

#### LCD

Change the former use of the LCD. Now show the stepping rate as a three-digit value centered on the display (i.e., from 10 S/S up to 600 S/S). Update the display once per second.

#### Measure Useful Work

Set **RB0** at the top of the main loop. Reset **RB0** just before executing the **Sleep** macro. Use this positive pulse to monitor how long the CPU is awake doing useful work. Note that each loop time will include about ten interrupt service routine executions. Set the scope’s trigger point to the left side of the scope’s screen. Use the scope’s AutoStore feature to measure the minimum and maximum value of this pulse. Use the scope’s time cursors to mark the minimum and the maximum pulse width. Let the waveform accumulate for over one second before halting and dumping the screen to the printer.

#### Measure Average Current Draw

As the stepper motor speeds up, note whether the average current draw and how it changes and by how much.

#### Qwik.lst File

When your compiled code works, print out two copies of the qwik.lst file generated by the compiler. Format the printout (i.e., left and right margins, font size, and possibly landscape mode) so that lines do not roll over. We will be discussing this file shortly. Turn one in with your project and bring the other to class.