ECE 4175
Project Three
Variable Stepping Rate

Complete by:
Wednesday Jan. 25th for an A+

Reference:
Chapter 14 Math Subroutines

Overview

For this project, you will modify the code for Project Two, using the number, 0 to 8, derived for Project Two to produce nine discrete stepping rates of

0, 12, 24, 36, 48, 60, 72, 84, 96

steps/second. You will also display this rate on the PC monitor.

Rate Multiplier Algorithm

Each second when you form **SMALLPOT**, also multiply **SMALLPOT** (whose value ranges betwen 0 and 8) by 12, putting the result into **STEPRATE**.

Each time around the mainline loop, call a **ControlStepping** subroutine. This subroutine is to add **STEPRATE** to another one-byte RAM variable called **ACCUM**. If the result overflows and sets the carry bit, then it steps the stepper motor one step and also subtracts one hundred from **ACCUM**. Then it returns from the subroutine (whether or not the addition overflowed).

To see why this scheme works, note that if **STEPRATE** were equal to one, and if **ACCUM** had just overflowed and had 100 subtracted from the result, it would now have 256-100 = 156 in it. Then during the next 100 looptimes, **ACCUM** would be incremented up one count at a time to the point where the 100th time it would go from 255 back to 156 again and a single step would occur during these 100 looptimes (i.e., one step in exactly one second).

As another example, if **STEPRATE** were equal to two (and starting from **ACCUM** = 156), during the next 100 looptimes a net total of 2x100 = 200 counts would be added to **ACCUM**, resulting in the overflow of **ACCUM** twice and a final content of **ACCUM** of 156 again.

RateDisplay Subroutine

This subroutine, called only once a second after **SMALLPOT** and **STEPRATE** have been formed, is to convert **STEPRATE** to two ASCII-coded digits and send them out to the PC monitor, preceded by the <CR> carriage return code. Use the **FXD0808U** subroutine of Figure 14-4 to divide the value in **STEPRATE** by 10, to break out the tens and units digits. To do this, follow the directions of Section 14.4. Once you have executed the division subroutine, send the <CR> to the PC using the **TXbyte** subroutine. Then copy the quotient in **AARGB0** to **WREG** (using the **movf)** instruction and add 0x30 to it, to form its ASCII code equivalent, and send this to the PC monitor. Finally, convert the remainder in **REMB0** to its ASCII code equivalent and send it. Then return.

Scope Display of Useful Work and Stepping Rate

In your mainline code at the beginning of the mainline loop (i.e., just after the **LOOP_** construct), set RC2 (i.e., bit 2 of Port C). Then at end of the mainline loop, but just before the call of **LoopTime**, clear RC2. Note that this pin is brought out on the header at the top of the QwikFlash board and that we can monitor how long this pin is high to tell how long it takes to do useful stuff each time around the mainline loop.

Each time a step is taken, toggle **RB1**, which is also brought out on a pin located on the same header. We can use this to monitor the stepping rate of the stepper motor since its frequency will be exactly half of the stepping rate.