OPTI 380B Intermediate Optics Laboratory

Lab 11

Microcontroller / Data Acquisition Project III

Objectives: This lab explores the software and hardware needed to run a stepper motor. The hardware is the "*Little Step-U*" controller board, sold by Parallax, and a standard, small-sized 4-phase unipolar stepper motor. The software will be supplied by you.

In particular, you will:
- learn how to control a stepper motor.
- learn how use a stepper motor controller board.
- learn about torque vs. speed tradeoffs when using a stepper motor.

Reading Assignment:

OPTI 380b WEB SITE

- Familiarize yourself with the set of high-level commands in "Stepper Motor Controller Documentation"

- Read about signed decimal values and the SDEC command in "Basic Stamp Syntax and Reference Manual".

PRELAB QUESTIONS

(PL1) The stepper motor that we will be using requires +12VDC to operate. This is not available on the Stamp board, and must be supplied from one of the variable supplies, part of the lab bench triple power supply. Where do you connect the 12V output and ground of this power supply to the "*Little Step-U*" controller board?

(PL2) What voltage is required to power the "*Little Step-U*" controller board?

(PL3) What is the maximum voltage that the "*Little Step-U*" controller board can supply to a stepper motor?

(PL4) What is the maximum current that the "*Little Step-U*" controller board can supply to a stepper motor?

(PL5) The "*Little Step-U*" controller board does NOT use the I2C bus. Instead, what pins on this board are used to talk to the Stamp microcontroller?

(PL6) What does the command {E17} do?

(PL7) What does the command {P1} do?

THE SOFTWARE

The following code is your starting point for this lab. The file is on our web site as "Basic Stepper Routine.txt"

This program asks for input of the:

- Speed [steps per second]	(speed VAR Byte)
- Ramp time [increments of 0.1 sec]	(ramp VAR Byte)
- Motor holding current [0="off", 1="on"]	(hc VAR Nib)
- Number of steps to take	(steps VAR Word)

and then uses the "E" command to move "steps" number of steps in the CW (clockwise) direction. For example, if you want to move 27 steps in the CW direction, the command sent to the "Little Step-U" controller board would be {E27}. Note that if you want to move in the CCW direction, the command would be {E-27}.

```
' Basic Stepper Routine.bpx
' {$STAMP BS2px}                        ‘ tells the Stamp board that we are using the BS2px CPU chip
' {$PBASIC 2.5}                         ‘ tells the Stamp board that we are using version 2.5 of PBasic

speed     VAR   Byte                    ‘ defines the variable “speed” as a Byte of length 8-bits
ramp      VAR   Byte                    ‘ defines the variable “ramp” as a Byte of length 8-bits
hc        VAR   Nib                     ‘ defines the variable “hc” as a Nibble of length 4-bits
steps     VAR   Word                    ‘ defines the variable “steps” as a Word of length 16-bits
baud      CON    1646                   ‘ defines the variable “baud” as a constant with value 1646

'baud is a number used to set the baud rate of the SERIN and SEROUT pins to 2400 baud for the BS2px chip


DEBUG CLS, "Little Step-U Demo", CR
PAUSE 100
DEBUG CRSRXY,0,1, "Enter the speed (steps/sec): "                              ‘ prints on line 1
DEBUGIN DEC speed
DEBUG CRSRXY,0,2, "Enter the ramp time (in 0.1 sec increments): "              ‘ prints on line 2
DEBUGIN DEC ramp
DEBUG CRSRXY,0,3,"Enter motor holding current 0=OFF, 1=ON: "                   ‘ prints on line 3
DEBUGIN DEC hc
DEBUG CRSRXY,0,4,"Enter the number of steps to take: "                         ‘ prints on line 4
DEBUGIN DEC steps

GOSUB CheckBusy                             ' Check Little Step-U is ready
SEROUT 10,baud,["{A",DEC speed,"}"]         ' Set speed
SEROUT 10,baud,["{B",DEC ramp,"}"]          ' Set ramp time
SEROUT 10,baud,["{P",DEC hc,"}"]            ' Stopped current on/off

' ===========================================================================
' Now do a simple move to a RELATIVE position using the parameters just set.
' Monitor the BUSY signal to determine when the move has been completed.
' ===========================================================================

DEBUG CRSRXY,0,6, "Moving ", DEC steps, " at ", DEC speed, " steps/sec."       ‘ prints on line 6
SEROUT 10,baud,["{E",DEC steps,"}"]
GOSUB CheckBusy                        ' Wait till it gets there
PAUSE 100
DEBUG CRSRXY,0,7, "FINISHED!!"                                                 ‘ prints on line 7


' ===========================================================================
' CheckBusy subroutine.
' Continually checks the state of the BUSY output from the Little Step-U and
' stays here until it becomes "not busy" (low).
' ===========================================================================

CheckBusy:
PAUSE 10
CB_loop:
IF IN12 = 1 THEN CB_loop      'Wait till not busy
PAUSE 1
RETURN

END
```

THE LABORATORY EXERCISE

Four points before we begin:

► Do NOT unplug the BS2px24 CPU chip from its socket, for any reason.

► Always turn the power to the board OFF ("0") when wiring up a circuit on the breadboard. Make any connections to the power supply ("V_{dd}" +5VDC, "V_{ss}" ground) or the I/O pins ("P0-P15") BEFORE turning on the power ("1").

► Do NOT connect an "output" pin directly to ground, or +5V. This will, most likely, destroy the CPU.

► <u>**Use the mini-grabber leads to make the following connections**</u>:
Connect ground of the bench power supply to pin Vom (-)
Connect +12VDC of the bench power supply to pin Vm (+)
<u>MAKE SURE THESE TWO LEADS ARE NOT SHORTED.</u>

"0". Manual control of the stepper motor ("as simple as it gets!!")

The shaft of a stepper motor is made to rotate by applying a positive voltage to each of the motor windings, <u>one-winding-at-a-time</u> and <u>in a particular sequence</u>. As Table 11.1 shows, if the voltage is first applied to Coil A (black wire), then to Coil B (orange wire), then to Coil C (brown wire), and finally to Coil D (yellow wire), the motor shaft will move in a clockwise (CW) direction. If the order is reversed, the motor will move in a counterclockwise (CCW) direction.

The "*Little Step-U*" controller board does this by taking "high-level" commands from the Stamp micro controller and, in the appropriate sequence, turning transistors on and off to supply +12VDC to each of the coils. As a result, the motor shaft rotates a certain number of steps, at a certain speed, in either the CW or CCW direction.

The same motion (albeit slowly) can be generated manually, using just a 9V battery. Refer to Table 11.1 to do the following:

1. Use a 9V battery to manually run the stepper motor. Connect the negative side of the battery to the motor's red (COMMON) wire, and alternatively touch (briefly) the positive side of the battery to the coils to make the motor move in a clockwise (CW) direction.

2. Repeat, but in the opposite sequence of coils, to make the motor move in a counterclockwise (CCW) direction.

3. Briefly (for just a few seconds) hold the positive lead of the battery to one of the coils and try to rotate the motor shaft by hand. The current flowing through the winding is generating a "holding torque" that is substantial. This current is called the "holding current." Rotate the motor shaft with no holding current, and feel the difference.

A. Wire up the "*Little Step-U*" Controller Board.

1. Make sure power to the Stamp microcontroller board is OFF.

2. Turn on the bench power supply. Adjust the right-most supply to give an output of +12VDC. Adjust the <u>current limit</u> control to its maximum setting.

3. TURN OFF THE BENCH POWER SUPPLY.

4. The "*Little Step-U*" controller board has been properly positioned on the white breadboard of the Stamp microcontroller at the start of the lab. Do not change its location. Note that the pins to connect to the external bench power supply, Vom(-) and Vm(+), are hanging over the edge of the white breadboard. This allows you to connect to them using the mini-grabber clip leads.

5. Make connections to all of the pins (except /IP1 and /IP2) on the "*Little Step-U*" controller board. Use the following table to properly connect the 5 wires to the stepper motor. Note that for this motor, the "Common" or ground wire is colored RED!!

Manufacturer	**?????????**
Degrees per Step	(you calculate this)
Steps per Revolution	(you measure this)
Phase 1 "Coil A"	Black
Phase 2 "Coil B"	Orange
Phase 3 "Coil C"	Brown
Phase 4 "Coil D"	Yellow
Common	Red

Table 11.1 Wiring Color Code for our Stepper Motor

6. <u>**Use the mini-grabber leads to make the following connections**</u>:
Connect ground of the bench power supply to pin Vom (-)
Connect +12VDC of the bench power supply to pin Vm (+)
<u>MAKE SURE THESE TWO LEADS ARE NOT SHORTED.</u>

7. Double-check your wiring. BEFORE turning on any power, have your TA DOUBLE-CHECK your wiring, as well. Be CERTAIN that you have connected +12VDC to pin Vm(+), and ground to pin Vom(-).

IF YOU USED A BANANA-to-BNC ADAPTER AT THE BENCH POWER SUPPLY, MAKE SURE THAT THE TAB IS CONNECTED TO THE BLACK GROUND JACK. DO NOT REVERSE THIS PLUG!!!!!!!!!!!!!!!!!!!!!!!!!!!

B. Initial checkout of the "*Little Step-U*" controller board and the stepper motor.

1. Turn on the bench power supply. Check that the output is still set to +12VDC.

2. Turn on power to the Stamp microcontroller board (IN THIS ORDER, to prevent transient spikes from the bench supply reaching the Stamp board).

3. Run the program "Basic Stepper Routine.bpx" for initial settings of:
- Speed = 10
- Ramp time = 10
- Motor holding current = 0
- Number of steps to take = 100

The motor should turn in a CW direction in a smooth, continuous fashion if everything is working correctly.

C. Measure the # of steps per revolution.

1. Run the program iteratively, changing just the number of steps, until you have a reasonably accurate value for the number of steps per revolution that the "*Little Step-U*" controller board and this particular stepper motor are running at.

HINTS:
- Make a mark on the gear to identify its location.
- Determine the number of steps required to rotate through an integer N number of revolutions (N>1) to increase your accuracy.

2. Calculate the value for the step size [degrees/revolution].

D. Reverse the direction of the stepper motor <u>in software</u>.

1. Suppose you want to run the stepper motor in a CCW direction. As the program is written, this is NOT as simple as entering a negative value for the "steps" variable.
- Why not?????
- How does the variable "steps" need to be defined if it can be set negative?

2. Modify the program to run the stepper motor in a CCW direction.
HINT: Learn about signed decimal values on pages 104-105 of the "Basic Stamp yntax and Reference Manual." In particular, learn about the SDEC modifier, and use it in place of DEC, wherever the "steps" variable is input or output.

3. Run the motor in both the CW and the CCW directions, to convince yourself that this syntax works!!

E. Reverse the direction of the stepper motor <u>in hardware</u>.

1. What other way can you reverse the direction of the stepper motor?

2. Try it!!!!!

F. Running and Holding currents.

1. Modify the program to PAUSE for 5 seconds, and then set the holding current to zero, just before the word "Finished!!" is printed.

2. Run the program and turn the holding current to OFF. After the program finishes, try to rotate the motor shaft by hand. Note how relatively easy it is to rotate.

3. Run the program and turn the holding current to ON. After the program finishes, try to rotate the motor shaft by hand. Note how much more difficult it has become to rotate, as long as the holding current is ON. What happens when the program finally switches the holding current OFF?

4. What is the running current [mA]?
(read the current meter in the bench power supply)

5. What is the holding current [mA]?
(read the current meter in the bench power supply)

G. Speed Limits

As discussed in class, a stepper motor operates as a constant power device. The product of the torque it can generate and the rate at which it turns is essentially a constant:

$$\text{Torque} \times \text{RPM} = \text{constant}$$

Even though our motor is not connected to anything, the torque it generates still has to overcome a certain amount of internal friction and rotational inertia. As a result, even with "no load" on the output shaft, there is still an upper limit to the speed at which this motor can be rotated. Beyond this point, an increase in the number of steps per second will not result in an increase in motor speed. And, depending on the motor, the rotation may become somewhat erratic and unpredictable.

NOTE: This is also a function of the "ramp" setting. No stepper motor can go from 0 to a finite RPM in zero seconds—the motor must be "ramped up" or accelerated to its speed setting, over some finite amount of time. This is determined by the "ramp" setting.

1. Start by setting the ramp variable to 50. This will ramp up the motor to full-speed over a time of 0.5 seconds (and decelerate it over 0.5 seconds, as well). Now begin to increase the speed, until the motor turns in an erratic fashion.

What is the maximum speed (for this acceleration time) that this motor can turn at?

2. Investigate the tradeoff between speed and ramp time. Can you find another ramp time that will allow the motor to run at a faster speed?