EE 3221 DIGITAL SYSTEM II
3. STEPPER MOTOR, SERVO MOTOR AND IR RECEIVER INTERFACES

OBJECTIVE

To design a number of hardware and software interfaces based on the simple programming examples in Experiment 2.

REFERENCES

1. EE3221 Class Notes.
2. "Programmers Guide for the DECE Microprocessor and Peripherals" available on the course website, www.ece.unb.ca/EE3221/Courses/CPD
3. Lab documents for Experiment 1 and Experiment 2.

EQUIPMENT

1. Host PC with internet access and Hyperterminal configured for 9600 Bd, even parity, 1-stop bit and no flow control.
2. EEPROM programmer and WinPP2 application.
3. Xilinx 4020 board
4. IO interface board and 10-pin ribbon cables.
5. RS 232C serial link between the IO interface board and COM1 on the host PC.
6. 1-SPST switch and pullup interfaced to bit-7 of P23.
7. The on line DECE assembler accessible from the course website.
8. Stepper motor.
9. Servo motor.
10. IR receiver and Sony remote transmitter.

PREPARATION

1. Re-read Lab-1 to review the details and options available for completing the software development cycle and arrive at the lab with a copy of the document.
2. Arrive in the lab with electronic copies and paper copies of the source code developed for each of the Parts of the previous experiment.
3. Study each of the three Appendices in this document so you are well informed on the hardware and operation of the hardware before you arrive at the lab.
4. Create the source code for Part 3 of the Experiment and bring a paper copy and an electronic copy of the source code..

EXPERIMENT

Recall that bit-7 at connector P23, called Sel_Bus, allows you to select the information to be displayed as follows:

Sel_Bus = 1 ADDR Bus and DATA Bus
Sel_Bus = 0 Alu flags or F-Bus and Alu Result or R-Bus

While single stepping or stopped at a breakpoint, you may view this information at the two 16-bit hex displays on the Xilinx 4020 board.

1. Generating Stepper Motor Sequences

First debug operation of the stepper motor and verify integrity of the motor and cables. Refer to Figure 3.2. Connect the DIP switches to “Connector Y” on the stepper motor board and manually generate the step sequences defined in Table 3.1. Observe the change in the motor shaft position.

Now connect output port 0x0002 to “Connector Y” on the stepper motor board. Implement the last program from Part 3 of the previous Experiment and debug. Once you have debugged program operation, replace the breakpoints by the software delay shown below.

```
      LOADP   R2, 0       ; You can adjust the delay at input port 0.
SD0:  SUB     R2, R2, 1   ; The delay is (almost) proportional
      JMPNZ   SD0         ; to the initial value in R2.
```

What is the smallest value of R0 that permits stepper motor operation? What happens if you do not include a delay between steps? Why?

2. A Quadrature Position Encoder Interface

First debug operation of the position encoder and verify integrity of the cables. Refer to Figure 3.3. Connect $\ddot{o}_A$ and $\ddot{o}_B$ to a scope. Manually rotate the shaft of the motor and verify that $\ddot{o}_A$ and $\ddot{o}_B$ are being generated properly.

Attach the position encoder connector to the breadboard connector, P24. Note this will break the hyperterminal connection so you will not be able to debug using the monitor (unless you use one of the breakout connectors on the white breadboard and hardwire between bit-6 to bit-6 and bit-7 to bit-7. Implement the revised program from Part 4 of the previous Experiment and debug its operation. NOTE: you will need to read from input port 0x0018 instead of port 0x0000. Manually rotate the shaft by approximately 1 revolution in the positive direction. What is the observed numerical value of the Count?

What is the maximum positive value of the Count? Minimum negative value?
What is the approximate number of revolutions from the initial position to that when the Count has reached its maximum positive value? its minimum negative value?

Rather that displaying the incremental position at output port 0x0003, create a program that

displays the peak value, i.e., each iteration through your code should display the largest magnitude of the incremental position up to that instant. One way of incorporating this change follows.

```
W = Y;
If (Y < 0) W = 0 - Y;    // Take the absolute value of Y.
if (W > X) X = W;        // X holds the peak value.
```

Y is the incremental position read at port 0x0018. W is the absolute value of Y and X denotes the peak value. Debug and run your program. Now remove any breakpoints and turn off single stepping. Manually rotate the motor shaft. What is peak value of the incremental position that you can generate in the clockwise direction? In the counterclockwise direction?

3. Interfacing an IR Receiver

First debug operation of the IR Receiver and verify integrity of the cables. Refer to Figure 3.7. Connect the signal, B, to a scope. Point the remote transmitter at the IR receiver and press any key. Verify proper operation of the receiver.

Connect the output of the IR receiver, B, to bit-0 of input port 0x0001. Consider a program that waits for a 0 to 1 transition, i.e., the program should implement the operations,

i. Wait for a 0 at bit-0 of port 0x0001
ii. Wait for a 1 at bit-0 of port 0x0001

Recall that the source code from Part 1 of the previous experiment implemented the operation on line ii above. Create, implement and debug your program. First connect input port-1 to the DIP switches and manually generate a 0 to 1 transition at bit-0.

Then interface port-1 to the IR receiver and debug program operation.

Now modify the program so it continually counts the number of 0 to 1 transitions detected and displays the number at output port 0x0002. The program may be developed with the following sequence of operations.

```
N = 0;
do {
        Write N to output port 0x0003;
        Wait for a 0 to 1 transition
        N = N + 1;
        }
```

Debug the program. Reset the microprocessor. Press a key on the IR remote once. What is the value of N?

SUMMARY

1. From Part 1 of the Experiment, what is the smallest value of R0 that permits stepper motor operation? What happens if you set the step delay to 0x0001? Why?
2. In Part 2 of the Experiment what is the observed numerical value of the Count for approximately 1 revolution of the output shaft?

   What is the maximum positive value of the Count?
   What is the minimum negative value?
   What is the approximate number of revolutions from the initial position to that when the Count has reached its maximum positive value?

3. In Part 3 of the Experiment what is maximum peak value of the incremental position that you were able to manually generate?

4. In Part 4 of the Experiment, when a single key is pressed once, what is the observed value of N.

5. Submit paper copies of the .LST files for the final programs developed in Parts 1, 2 and 3 of the Experiment.

APPENDIX 1: Stepper Motor Operation

Stepper motors rotate or step from one fixed position to another at discrete instants in time. Many stepper motors have 2 or 4 coils. The stepper motor used in this experiment has 2 coils in which the direction of current flow is controlled by 6-bits at parallel output port 0x0002. By changing the direction of current flow in the coils in the *proper* sequence it is possible to make the motor move in either the clockwise or counterclockwise direction.

The stepper motor consists of a 4-pole (stationary) stator and a permanent magnet rotor as shown in Figure 3.1. The north pole on the permanent magnet is denoted by the arrow in Figure 3.1. Using basic electromagnetic theory you should be able to ascertain the four distinct steady state positions for the rotor as shown.

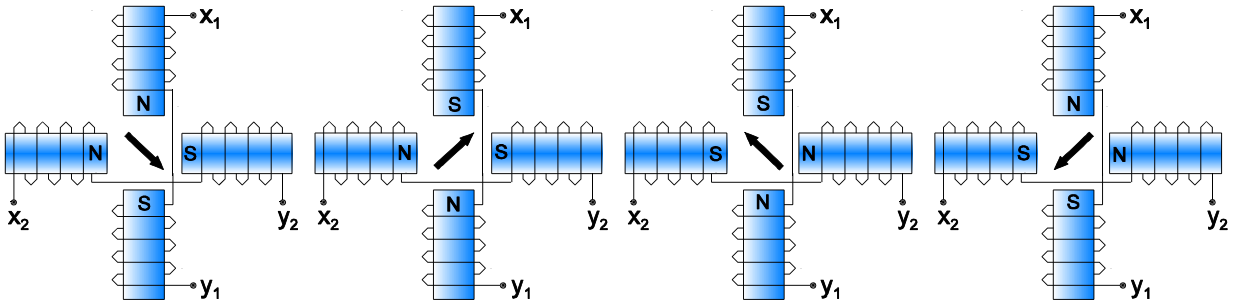

Figure 3.1. CCW Motion of the 2-Coil Stepper Motor

Each of the coils of the stepper motor is controlled by a driver circuit shown in Figure 3.2. The L6219DS driver turns the current on in each of the two coils in a direction that depends on the digital value supplied at $p_7\ p_6\ p_5\ p_4\ p_3\ p_2\ p_1\ p_0$. Each time the current changes in one of the coils the motor shaft will rotate one step. The Table below shows that the motor will step continuously in the clockwise direction if the sequence, 0x0000, 0x0010, 0x0011, 0x0001 is applied (over and over again) at $p_7\ p_6\ p_5\ p_4\ p_3\ p_2\ p_1\ p_0$.

	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0		Direction		Current	
Step	**x**	**I_{12}**	**I_{11}**	**$Phase_2$**	**x**	**I_{02}**	**I_{01}**	**$Phase_1$**	**Hex**	**CW**	**CCW**	**Coil 2**	**Coil 1**
1	0	0	0	0	0	0	0	0	00	↓	↑	I_O	I_O
2	0	0	0	1	0	0	0	0	10	↓	↑	$-I_O$	I_O
3	0	0	0	1	0	0	0	1	11	↓	↑	$-I_O$	$-I_O$
4	0	0	0	0	0	0	0	1	01	↓	↑	I_O	$-I_O$

Table 3.1. Step Sequence for Full Stepping

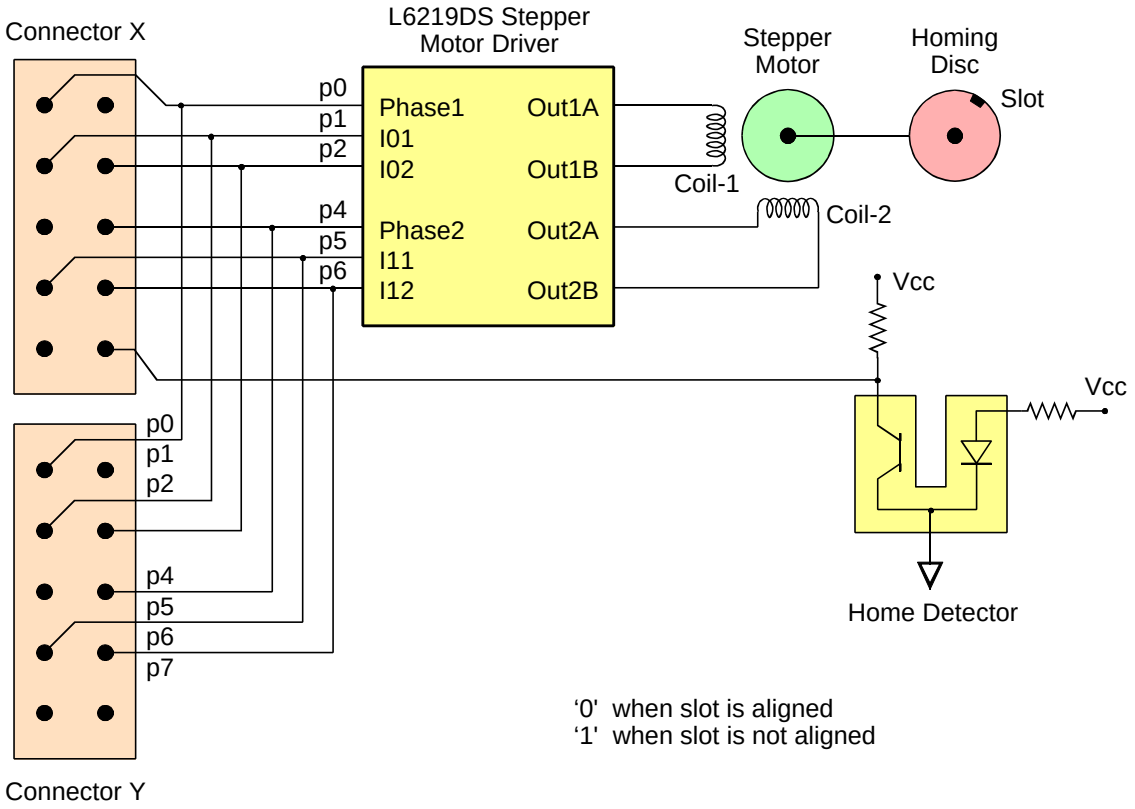

Figure 3.2. The Stepper Motor Board.

APPENDIX 2: Operation of the Quadrature Position Encoder Interface

A quadrature phase optical position encoder is a non-contact angular position sensor. It contains a rotating disk with N inscribed slots per revolution. ***Pulse*** outputs ,ö$_A$ and ö$_B$ are generated as each slot in a disk rotates past an optical detector as shown in Figure 3.3. The change in angular position is determined by counting the pulses. The Count Value has units of 2ð/N radians. The optical detector also includes decision electronics for encoding the direction of rotation as follows.

6. If ***ö$_A$ leads ö$_B$*** rotation is cw.
7. If ***ö$_B$ leads ö$_A$*** rotation is ccw.

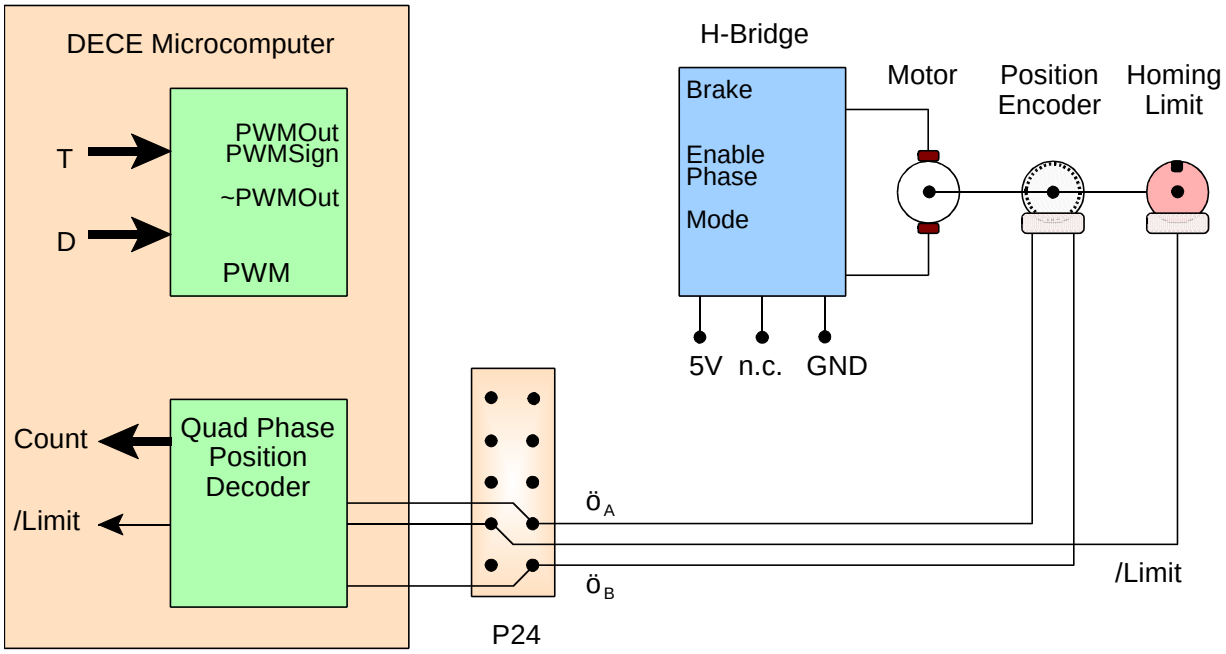

Figure 3.3. The Servo Motor Board with a Position Encoder.

The DECE microcomputer contains a built-in Quadrature Phase Position Decoder that accepts ö$_A$ and ö$_B$ and inputs in the form shown in Figure 3.4 and generates a 16-bit signed Count value. Whenever there is a pulse at ö$_A$ or ö$_B$ the decoder circuit automatically counts DOWN if ö$_A$ leads ö$_B$ or counts UP if ö$_B$ leads ö$_A$.

The incremental position is read as the 16-bit signed Count at input port 0x0018. The value of the Count register is automatically reset to 0x0000 after being read.

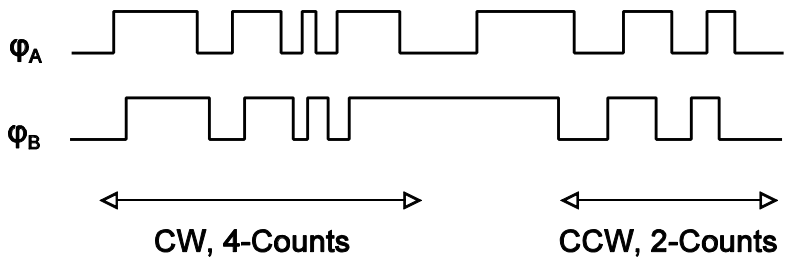

Figure 3.4. Position Encoder Outputs.

APPENDIX 3: Operation of the IR Receiver

An infrared remote transmitter generates digital data one bit at a time using the infrared emitter circuit shown in the Figure 3.5.

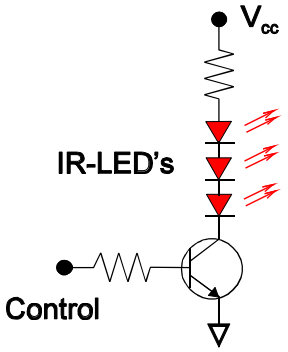

Figure 3.5. IR Transmitter.

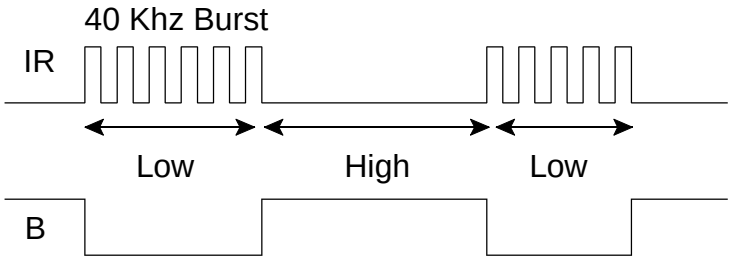

Figure 3.6. Transmitted Signal and Logic Representation.

By turning the control transistor on and off one can transmit logic '1's and '0's as the presence or absence of IR radiation. Most IR remote controls transmit information by modulating the serial data with a 40 KHz carrier to reduce the effect of ambient radiation (mostly caused by fluorescent lights) on the transmitted IR signal. A logic '0' is transmitted as a burst of IR radiation at 40 KHz and a logic '1' is transmitted as an absence of IR radiation as shown in the Figure 3.6.

The ***IR receiver module*** is interfaced to the DECE microcomputer as shown in Figure 3.7 so that software may be written to detect the 1-bit digital value at B.
The operation of an IR receiver involves detecting the presence of a 40KHz burst and representing the burst as a logic 'low' at B. In between bursts a logic high' is represented.

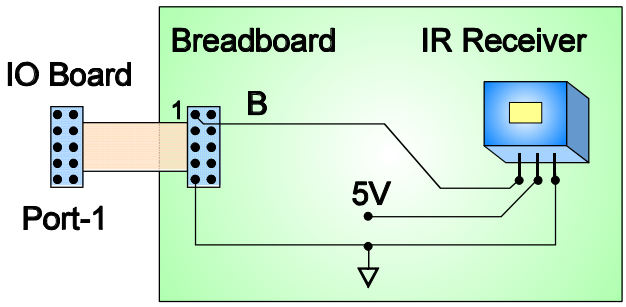

Figure 3.7. IR Receiver Interface.

One of the common standards for encoding information in IR transmissions uses pulse length modulation illustrated in Figure 3.8. Initially the line is in the idle state = '1'. A start bit, S, = '0' of duration TS = 2.4 ms signifies the beginning of a transmission. Each transmitted data bit is encoded by: a high level of duration, T = 0.52 ms, followed by a low level. If the ‘low’ level is of duration, T0 = 0.68 ms, then a '0' is represented. If the ‘low’ level is of duration, T1 = 1.25 ms, then a '1' is represented. Notice that a logic '1' takes more time to transmit than a logic '0'.

Information is encoded in 13-bits beginning with the start bit, S, as shown in Figure 3.8. The start bit is used to synchronize the reading of the remaining 12-data bits, $A_0\,A_1\,A_2\,A_3\,A_4\,A_5\,A_6\,A_7\,A_8\,A_9\,A_{10}\,A_{11}$.

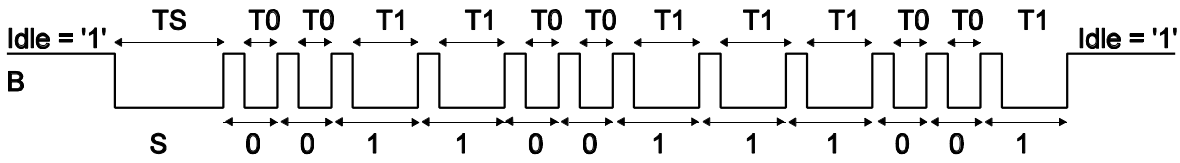

Figure 3.8. Pulse Length Encoding.

S	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11
0	0	0	1	1	0	0	1	1	1	0	0	1
S	LSB					12-Bit Data						MSB

Figure 3.8. Sony IR Remote Bit Format.