Experiment 5.1

Writing Programs in Tiny BASIC

Overview

During this experiment, you gain experience using the PEEK and POKE commands to read and write data to the individual bits of the system-defined ports.

Objectives

Upon completion of this experiment, you will be able to:

1. Read, using the PEEK command, data in the utility software module,
2. Write, using the POKE command, data to the LEDs,
3. Write a Tiny BASIC program to activate the LEDs sequentially,
4. Write a Tiny BASIC program to activate the motor relays,
5. Write a Tiny BASIC program to read the battery-sense bit, and
6. Write a Tiny BASIC program to read the charger-connection bit.

Prerequisites

1. Chapter 5, "A Command of Tiny BASIC."
2. A review of Appendix F, "Memory Map."

References

1. Chapter 5, "A Command of Tiny BASIC."
2. Appendix F, "Memory Map."
3. Appendix E, "Utility Software Module Routine Listings."
4. Section D, "Software," of the RB5X Reference Manual.

Technical Information

In addition to the four commands addressed in Chapter 5, "A Command of Tiny BASIC," you can use the PEEK and POKE commands to read and write values into memory in the following manner:

PEEK	B=@# **address**
POKE	@# **address** = 73

More detailed information on the use of these commands can be found in Section D, "Software," of the RB5X Reference Manual.

To use the PEEK and POKE commands, you must have an understanding of the layout of the electronics as far as the "address" of the various I/O ports is concerned. When a system is designed (particularly an open system, such as RB5X, which encourages expansion), there must be a plan so that (1) the existing circuitry can be easily accessed by software, and (2) additional circuitry can be added as peripherals are added. In RB5X, this plan is the memory map, as shown in Appendix F.

As you can see by this memory map, the lower portion of memory, addresses #0000 to #09FF, is occupied by the Tiny BASIC utility software module subroutines; addresses #3000 to #6FFF are for 16K of memory expansion; etc. The portion of the memory map that is most useful to you in programming RB5X are those pertaining to the I/O ports. RB5X has three ports of major interest: ports A, B, and C, which are located at addresses #7800, #7801, and #7802, respectively.

Port A (#7800) is the bumper contact register. When a bumper is pressed, a 0 appears in the corresponding location in this register. If no bumpers are pressed, then address #7800 contains eight 1s, or decimal 255. This is an input port and can only be read under program control. The programmer cannot insert 1s or 0s in this register through software.

Port B (#7801) is an output port register. Bit A, the least-significant bit (LSB), is the sonar-on bit; bit B is the infrared LED-on bit; bits C, D, E, and F are the four yellow LEDs under the dome located next to the four green LEDs; bit G is the fifth, or center, LED and the four clear LEDs around it; and bit H is the horn, located on the CPU board. A 1 in any of these locations switches the device on and a 0 switches the device off.

Port C (#7802) is half input and half output. Bits A, B, C, and D are outputs to the two motors through the four relays and the corresponding green LEDs on the LED board. A 1 switches on the motor and a 0 switches it off. Bits, E, F, G, and H are inputs. Bit E is battery charge, bit F looks for sonar transmit and senses contact with the charger, bit G looks for the reflected infrared LED signal, and bit H senses the sonar echo.

The figures below show these registers and their contents.

Port A, #7800

H	G	F	E	D	C	B	A

Bumper contact register (no contact=all 1s)

Port B, #7801

H	G	F	E	D	C	B	A

A = Sonar on
B = Infrared LED on
C, D, E, F = LEDs under dome
G = Fifth LED and pulsating LEDs
H = Horn

1 = on 0 = off

Port C, #7802

H	G	F	E	D	C	B	A

A, B, C, D = 2 motors through 4 relays (forward and reverse)
E = Battery charge
F = Sonar transmit and charger contact
G = Reflected Infrared LED signal
H = Senses sonar echo

Procedure

1. Boot the ASCII Pro diskette.

2. Set the computer baud rate to 300.

3. Switch on RB5X and allow it to initialize.

4. Press bumpers 2 and 4 to enter Tiny BASIC command mode. The computer returns the message STOP AT 30.

5. Type:

   NEW #1000<RET>

6. At the Tiny BASIC prompt >, type:

   NEW<RET>

   NOTE: The preceding six steps will be referred to hereafter as **"Set up the robot for Tiny BASIC."**

7. Enter the following Tiny BASIC commands:

   B=@#<RET>
   PRINT B<RET>

   Notice that these are Tiny BASIC commands, but that they are not given a line number. You enter these as commands in this format to cause direct (immediate) execution.

8. The computer now returns the decimal number 13. This is the memory contents in decimal of address #8008 on the utility software module. From the utility software module listing in Appendix E, you should be able to see that a hexadecimal OD is stored at address #8008 ($OD_{16} = 13_{10}$).

9. Write a series of Tiny BASIC commands to return the memory contents of addresses #8037, #8040, and #8044 on the utility software module. Record below the commands and the returned values.

10. The returned values should have been 62, 55, and 42. These are the decimal equivalent of the hexadecimal contents. What are the hexadecimal contents of these three locations?

11. The following sequence of commands writes (POKE) decimal 255 to address #8008. Enter these three commands:

```
@#8008=255
B=@#8008
PRINT B
```

Why did this return 13 instead of 255 from the utility software module machine-code subroutine?

12. The following command writes (POKE) data to the LEDs. Enter this command into the computer:

```
@#7801=124
```

Notice that the four yellow LEDs are on, while the amber and clear LEDs are flashing. The command @#7801=0 switches all the LEDs off.

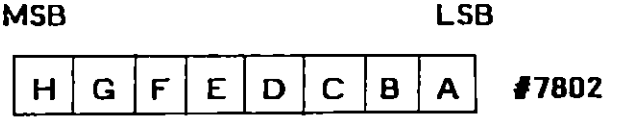
13. Now read (PEEK) the last data written into this register. Enter the following commands:

```
B=@#7801
PRINT B
```

14. Bits C, D, E, and F of this register control the LEDs. Write your own values into address #7801 and then read the contents back from this address.

15. Write a Tiny BASIC program to activate the LEDs at address #7801 in a clockwise fashion. (Be sure to include delays so the sequence can be viewed.)

16. The bits A, B, C, and D at address #7802 control the motor relays and the green LEDs on the LED board. The register at address #7802 is arranged in the following fashion:



17. Write a Tiny BASIC program to determine which bit of address #7802 corresponds to which motor relay. Complete the table below. (Example: Relay K0 = data bit A)

Relay No.	Data Bit
K0	_____
K1	_____
K2	_____
K3	_____

18. Figure 7 shows the motor control board and the relay designators.

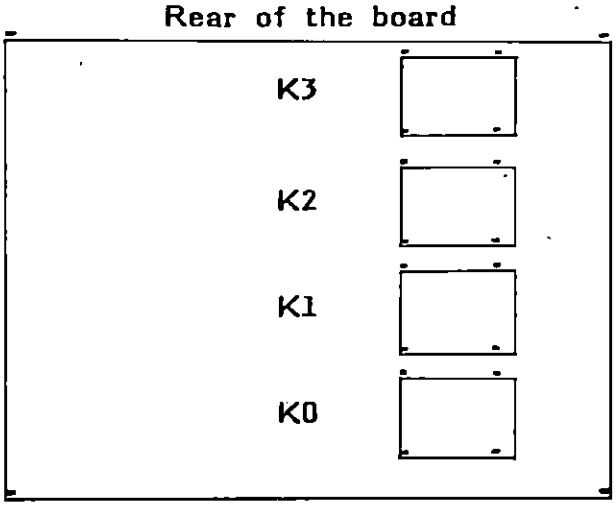


Figure 7. Motor Control Board and Relay Designator

Now read (PEEK) the value of address #7802 to check the battery sense bit. Bit E of #7802 senses the charge on the batteries. Bit F of #7802 senses the connection to the charger and the charging current. Enter the command below, which switches off all the motors.

```
@#7802=0
```

19. Now read (PEEK) the value of address #7802 to check to see which bits are 1s and which are 0s. Working with RobotLab, complete the table below, first with the charger plugged into the wall socket, next with the charger unplugged.

Bit	Contents- Charger Unplugged	Contents- Charger Plugged In
LSB A	_____	_____
B	_____	_____
C	_____	_____
D	_____	_____
E	_____	_____
F	_____	_____
G	_____	_____
MSB H	_____	_____

20. Write (POKE) decimal 255 into address #7802 and then read (PEEK) #7802 to observe the value stored there. Record the displayed value below:

_____

21. Write (POKE) decimal 15 into address #7802 and then read (PEEK) #7802 to observe the value stored there. Record the displayed value below:

_____

22. Why are the two returned values in Step 20 and 21 the same?

_____

_____

_____

23. Repeat Steps 20 and 21 with the charger unplugged. This should substantiate your response to Step 22. (The four upper bits of #7802 are input only and can only be read and not changed by software.)

Summary

During this experiment, you observed that you can write programs in Tiny BASIC to PEEK but not to POKE data at the utility software module subroutine addresses. You also observed that Tiny BASIC programs can be written to PEEK and .to POKE data to the LEDs and to the motor relays. You have also written Tiny BASIC programs to light the LEDs in a clockwise fashion. You should now understand that bit E of #7802 monitors the robot's battery charge (1=charged, 0=discharged), and you should also understand that bit F monitors contact and current flow with the battery charger (1=disconnected, 0=connected).

Questions

1. If the following program returned a decimal 195, what would be the hexadecimal value of the data stored at address #806C?

```
10 B=@#806C
20 PRINT B
```

2. What condition exists if bit E of address #7802 returned a binary 0?

3. Which bit of the system-defined port at address #7801 controls the flashing LEDs?

4. Write a Tiny BASIC program to activate the green LEDs in a counterclockwise pattern.

5. What condition exists when bit F of #7802 is a 1?

_____

_____

_____

Experiment 5.2

Writing More Programs in Tiny BASIC

Overview

In Chapter 5, "A Command of Tiny BASIC," you were instructed in the proper use of four commands unique to Tiny BASIC:

- LINK
- DELAY
- CLEAR
- DO/UNTIL

In this experiment, you write Tiny BASIC programs and modify existing Tiny BASIC programs using these four commands, along with PEEK and POKE.

Objectives

Upon completion of this experiment, you will be able to:

1. Write Tiny BASIC programs to change the data bits at the system-defined output ports.

2. Write Tiny BASIC programs to respond to the data bits at the system-defined input ports.

Prerequisites

1. Chapter 5, "A Command of Tiny BASIC."

References

1. Chapter 5, "A Command of Tiny BASIC."

2. Appendix F, "Memory Map."

3. Appendix G, "Functions."

4. Appendix H, "Motor Codes and Relay/Motion Assignment."

Technical Information

In Experiment 5.1, you practiced use of the PEEK and POKE commands. They are used for immediate execution; this is referred to as command mode. The use of line numbers with each statement places you in program execution mode. During this experiment, you operate in program execution mode.

In the execution of the DO/UNTIL statement, the exit condition is not tested until the end of the DO action. This means that each DO/UNTIL statement executes at least once in the program sequence. The CLEAR command must be used with caution because it clears all variables, including loop counters. When the LINK command is used, the machine-code subroutine called must end with a RETURN statement (#5C). Using a machine-code subroutine without a RETURN statement causes a program error or a crash of the program.

The DELAY command has a range of 1 through 1040, with 1 millisecond per count. If 0 is used following the DELAY statement, it defaults to a 1.04-second delay.

Procedure

1. Set up the robot for Tiny BASIC.

2. Type in the following program, remembering to end each line with a Return:

```
 10 CLEAR
 20 FOR I=1 TO 10:PRINT"":NEXT I:REM 10 BLANK LINES
 30 PRINT"PRESS AND HOLD BUMPERS 2 AND 4 TO
    CONTINUE"
 40 LINK#85A1:REM STARTS INITIALIZATION ROUTINE
 50 FOR I=1 TO 10:PRINT"":NEXT I:REM 10 BLANK LINES
 60 PRINT"PRESS BUMPER 1 TO CHANGE THE ROUTINE"
 70 DO:REM GREEN LEDS CCW
 80 @#7802=1:REM FIRST GREEN LED
 90 GOSUB 1000
100 @#7802=2:REM SECOND GREEN LED
110 GOSUB 1000
120 @#7802=4:REM THIRD GREEN LED
130 GOSUB 1000
140 @#7802=8:REM FOURTH GREEN LED
150 GOSUB 1000
160 B=@#7800:REM CHECKS BUMPERS
170 UNTIL (B=251):REM DO/UNTIL LOOP TILL BUMPER 1
    IS PRESSED
180 @#7802=0:REM ALL GREEN LEDS OFF
190 GOSUB 1000
840 FOR I=1 TO 25:PRINT"":NEXT I:REM CLEAR SCREEN
850 PRINT"PRESS AND HOLD BUMPER 3 TO REPEAT OR 5
    TO STOP"
860 B=@#7800:REM CHECKS BUMPERS
870 IF B=247 GOTO 20:REM IF BUMPER 3, REPEAT
880 IF B=239 GOTO 900:REM IF BUMPER 5, STOP
890 GOTO 860
900 STOP
1000 @#7801=128:REM START OF HORN AND DELAY
     ROUTINE
1010 DELAY 250
1020 @#7801=0:REM HORN OFF
1030 DELAY 1000
1040 RETURN:REM END OF HORN AND DELAY ROUTINE
```

3. Check your program for errors, save and run the program under the file name DISPLAY PROGRAM.

4. When RB5X finishes initialization, press bumpers 2 and 4 to move to Tiny BASIC command mode.

5. Explain what happens when you press bumpers 2 and 4.

_____

_____

_____

6. Press and hold bumper 1.

7. Explain what happens when you press bumper 1.

_____

_____

_____

8. When you press bumpers 2 and 4, the initialization routine ends and the green LEDs flash in a counterclockwise direction. The subroutine, beginning at line number 1000, beeps the horn for a 1/4-second at each step of the rotation.

   Explain what happens if you push any bumper except number 1, during the counterclockwise flashing of the green LEDs.

_____

_____

_____

9. When you pressed bumper 1, you moved out of the counterclockwise green LED flashing routine, into a holding loop waiting for you to decide either to stop or to repeat the sequence.

   Hold bumper 1 and note the system's response.

_____

10. Press bumper 5 to stop program execution.

    What does the computer's CRT display when you press bumper 5?

_____

11. Using Step 2 as an example, write the program steps to cause the

green LEDs to flash in a clockwise direction. Add it to your existing program starting at line 200. Use the following two lines to exit the program.

```
PRINT"PRESS AND HOLD BUMPER 6 TO CHANGE ROUTINE"
UNTIL (B=191)
```

12. Save and run this expanded program.

13. Note that the sequence used should be as follows:

    A. Initialization
    B. Bumper 2 and 4
    C. Green LEDs flash counterclockwise
    D. Bumper 1
    E. Green LEDs flash clockwise
    F. Bumper 6
    G. "PRESS AND HOLD BUMPER 3 TO REPEAT OR 5 TO STOP"

14. Use the skeleton program in Figure 9 to implement the flow chart functions shown in Figure 8. Use the subroutine in Figure 10, starting at line 2000.

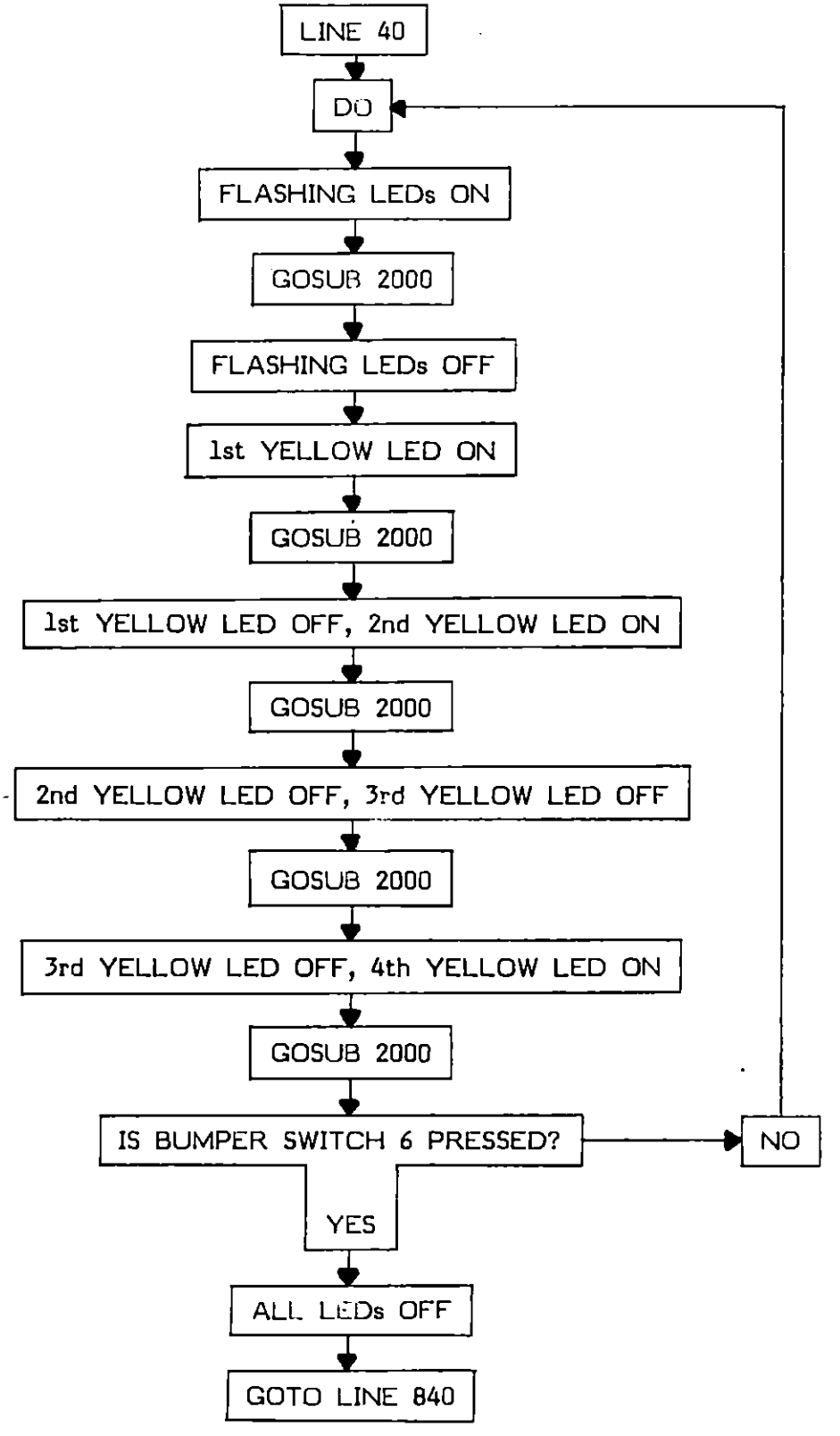


Figure 8. Flow Chart Function

```
400 DO
410 @#7801=____:REM 1ST YELLOW LED ON
420 GOSUB 2000
430 @#7801=____:REM 2ND YELLOW LED ON
440 GOSUB 2000
450 @#7801=____:REM 3RD YELLOW LED ON
460 GOSUB 2000
470 @#7801=____:REM 4TH YELLOW LED ON
480 GOSUB 2000
490 B=@#78___:REM CHECKS BUMPER SWITCHES
500 UNTIL (B=____):REM IF BUMPER 2,GOTO 510
510 @#7801=0:REM ALL LEDS OFF
```

Figure 9. Skeleton Program

NOTE: The bumper switch values can be found by holding the bumper and entering the following statement in command mode:

```
PRINT @#7800
```

NOTE: Wherever there is a blank in the skeleton program, you must figure out the number to activate the appropriate LED.

```
2000 B=@#7801:REM START OF 2ND SUBROUTINE
2010 @#7801=(128+B):REM HORN AND LED ON
2020 DELAY 250:REM 1/4 SECOND DELAY
2030 @#7801=D:REM HORN OFF, LED ON
2040 DELAY 1000:REM ONE SECOND DELAY
2050 RETURN:REM END OF 2ND SUBROUTINE
```

Figure 10. Subroutine

15. Save and execute this greatly expanded program.

16. Using the example of Step 13 and repeated execution of the program, explain the total program execution sequence below.

_____

_____

_____

Summary

In this experiment, you gained experience inputting data from and outputting data to the user-defined ports via Tiny BASIC programs. You should now have a better understanding of why the field of robotics is considered so technical. Not only must you be proficient in hardware, but also in software.

At this point in the course, if you haven't already done so, you are ready to take what you have learned about the robot from working with RobotLab and apply it to RB5X itself.

Questions

1. List the steps required to "Set up the robot for Tiny BASIC."

   A. _____

   B. _____

   C. _____

   D. _____

   E. _____

   F. _____

2. Complete the following table with the addresses for the user-defined ports.

   BUMPER SWITCHES = _____

   GREEN LEDs/MOTOR RELAYS = _____

   YELLOW LEDs/FLASHING LEDs = _____

3. What is the decimal value displayed on the CRT when you press bumper switch number 8?

   _____

4. What bumper or bumpers must be pressed to return a decimal 1?

   _____

5. Write a Tiny BASIC program that switches on all the green LEDs only when bumper 4 is pressed.

   _____

   _____

   _____

   _____

   _____

CHAPTER 6

GETTING THE BUS TO THE OUTSIDE WORLD

OVERVIEW

In Experiment 3.1, you learned how to trace the RB5X bus. This allows you to determine what signals are available at jacks J0 through J5 on the mother board.

In this chapter, you learn how to monitor and simulate signals at these jack connections, allowing you to expand the existing system by adding your own hardware -- an engineering task technicians are frequently called upon to perform. This expansion could entail either input or output circuitry and you need to consider the following:

- o Address decoding,
- o Data signal conditioning, and
- o Power access.

OBJECTIVES

Upon completion of this chapter, you will be able to identify the user-defined ports by:

1. Jack connector pins,

2. Programmable peripheral interface (PPI) pins, and

3. Memory locations.

PREREQUISITES

1. Chapter 5, "A Command of Tiny BASIC."

REFERENCES

1. CPU board schematic in Appendix D, "RB5X Schematics."

2. Mother board schematic in Appendix D, "RB5X Schematics."

3. Appendix 6, "Using NSC Tiny BASIC," in the RB5X Reference Manual.

4. Tables 3.2, 3.3, and 3.4 in Chapter 3, "The RB5X Bus System."

5. Appendix F, "Memory Map."

TECHNICAL INFORMATION

One of the first steps in getting the bus to the outside world is the identification of bus lines, their function, and their location. In Chapter 3, "The RB5X Bus System," you identified the system bus lines, their function, and their locations on jacks J0 through J5. The next logical step is to physically extend these jacks to an easily accessible breadboard. This allows greater flexibility in the layout of your circuitry.

RB5X can be expanded through user-defined ports, of which three are available, through jacks J1 and J2. They are identified as Ports A, B, and C, and are buffered through U20 (an INS8255 PPI) on the CPU board. Each of these ports is eight bits wide; the bits are labeled 0 through 7.

There are another three ports available through jacks J3 and J4. They are also labeled Ports A, B, and C, and are buffered through U23 (an INS8255 PPI) on the CPU board, are also eight bits wide, and are labeled 0 through 7.

These ports gain access to system data lines when their respective buffers (U20 and U23) are properly addressed, or enabled, by the microprocessor. These ports are bidirectional, the flow direction determined by the state of the WR and RD pins on the buffers. When the WR pin is low, the buffer writes (outputs) to the external circuitry. This process is similar to that used to "write" to memory. When the RD pin is low, the buffer reads (inputs) from the external circuitry, in the same way as you "read" from memory.

This allows your external circuitry to function as if it were physically located on the CPU board. With this electrical proximity, you are now able to access your external circuitry through software. This software access, which makes use of the Tiny BASIC PEEK and POKE commands, is the critical step in breathing life into your expansion circuitry.

Appendix F, "Memory Map," gives quick reference to the available user-defined ports. From this figure, you can see that J1 and J2 allow access to Ports A, B, and C. Let's trace these ports with more detail.

Port A of jack J1 and J2 is eight bits wide. The following table illustrates how the PPI pins are connected to J1 and J2.

Identifier	Bit	J1-J2 pins	U20 pins
2A0	0	M	4
2A1	1	11	3
2A2	2	N	2
2A3	3	12	1
2A4	4	P	40
2A5	5	13	39
2A6	6	R	38
2A7	7	14	37

Table 6.1 - PPI/J1 and J2 Connections

To send or to receive data from Port A, you treat it as if it were RAM with the address of #7808. When you place logic-level voltages at appropriate J1 or J2 pins, the microprocessor treats them as if they were an 8-bit word of 0s and 1s at memory address #7808. Likewise, Ports B and C look like 8-bit words at memory addresses #7809 and #780A, respectively. You can determine the actual pin connections from Table 3.2 in Chapter 3, "The RB5X System Bus."

The ports of PPI U23 can be accessed at the following memory locations:

Port	Memory Address
A	#7804
B	#7805
C	#7806

Table 6.2 - U23 Ports Memory Locations

The microprocessor can read and write to these ports just as if they were RAM. A quick survey of Table 3.3 in Chapter 3, "The RB5X System Bus," shows you the corresponding pin connections.

Let's set up a situation to read logic levels from our external breadboard circuit.

Given that jack J4 is wired to logic-level voltages in the following order:

Bit Identifier	J4 Pin	Logic Level
0	S	+5 Volts
1	15	+5 Volts
2	T	+5 Volts
3	16	+5 Volts
4	U	0 Volts
5	17	0 Volts
6	V	0 Volts
7	18	0 Volts

Table 6.3 - J4 Logic Level Voltages

To access these J4 pins, you need to access Port C of U23, by using the PEEK command to memory address #7806. The PRINT command displays a decimal 15 on the CRT. This should illustrate for you that the 0 bit is the least-significant bit (LSB) and the 7 bit is the most-significant bit (MSB). The following program can be used to accomplish this task:

```
NEW #1000
NEW
10 B=@#7806
20 PRINT B
30 END
```

The first two lines initialize the microprocessor by clearing the appropriate internal registers and erasing all previous programs.

Line 10 instructs the microprocessor to read (PEEK) the data at memory address #7806. The contents of address #7806 are placed in a buffer and given the label B.

Line 20 instructs the microprocessor to display on the CRT the decimal equivalent of the data read in line 10.

Line 30 stops the program.

The next program allows you to write a binary 15 to Port C of U23:

```
NEW #1000
NEW
10 @# 7806=15
20 END
```

Line 10 in this program does the reverse of line 10 in the previous program. It puts the number 15 into memory location #7806.

It should be apparent that U20 and U23 perform the address decoding for the user-defined ports. This greatly simplifies your expansion task. If U20 and U23 were not present on the CPU board, you would be required to build your own decoders. Another factor adding to the simplification of your expansion is the availability of +5 volts and ground at pins 2 and B and pins 1 and A, respectively, on each jack J1 through J5.

SUMMARY

You have discovered that RB5X was designed with expansion in mind. This is accomplished through six easily accessible, user-defined ports. Address decoding is accomplished through two INS8255 PPI chips on the CPU board. To further ease the expansion, power connections are furnished to each of the jacks. You should now understand a software technique for inputting and outputting data. You are now armed with the technical information to expand the hardware

capabilities of your RB5X.

QUESTIONS

Use the following information to answer questions 1 and 2.

The pins of jack J2 have the following voltages:

Pin 19, +5 volts
Pin 20, +5 volts
Pin 21, 0 volts
Pin 22, 0 volts
Pin W, +5 volts
Pin X, 0 volts
Pin Y, +5 volts
Pin Z, 0 volts

1. Write a Tiny BASIC program to display on the screen the decimal equivalent of the logic levels present at these pins.

2. What is the decimal number that displays on the CRT?

3. What is the chip identifier for the PPI chip that decodes address #7805?

4. What jack number and jack pin contains the MSB for memory address #780A?

Experiment 6.1

Building the Bus Extender

Overview

The bus extender is an easy means of extending the bus jacks through a flexible ribbon cable out to a breadboard. This allows you to lay out circuits for RB5X on a designer board. The bus extender allows you to test your circuits on the RobotLab and to quickly apply them to RB5X.

Objectives

Upon completion of this experiment, you will be able to:

1. Identify which pins of jacks J1 through J5 are connected to which lines of the flexible cable, and

2. Build the bus extender to fit jacks J1 through J5.

Prerequisites

1. An understanding of sound, soldering methods.

2. An understanding of the sequential nature of the standard resistor color code.

3. An understanding of the proper methods for stripping and tinning stranded wires.

4. Have the General Robotics Corporation jumper board and two feet of 44-conductor ribbon cable.

References

None.

Technical Information

The flexible ribbon cable used in this experiment has 44 lines, or conductors. They are color coded in the same sequence as the standard resistor color code. Notice that there are four complete counts of the color code and one count that stops at yellow. The sequence is counted from the side with the brown wire through the yellow wire. This cable is to be connected to the printed circuit (PC) board with a male connector that fits into jacks J1 through J5. This board, commonly referred to as the jumper board, is double-sided with 22 jack-pin contacts on each side. They are numbered 1 through 22 on one side and A through Z on the other. The letters G, I, O, and Q have been skipped in the alphabetic sequence, because it is often difficult to tell these letters from the numbers 6, 1, and 0 when they are used as identifiers.

The pads on the board are gold-plated to ensure a good electrical contact between the jack pins and the board and to retard contact corrosion.

NOTE: Do not allow solder to get on the plated contacts as this defeats the purpose of the plating. Note that some of the contacts (19 through 22 and U through Z) are not straight connections. This is done to allow the user to easily criss-cross certain lines when the jumper board is used for other purposes.

Procedure

1. Separate the lines of the ribbon cable by GENTLY pulling them apart. They should be separated for three inches back from both ends of the cable. Refer to Figure 11.

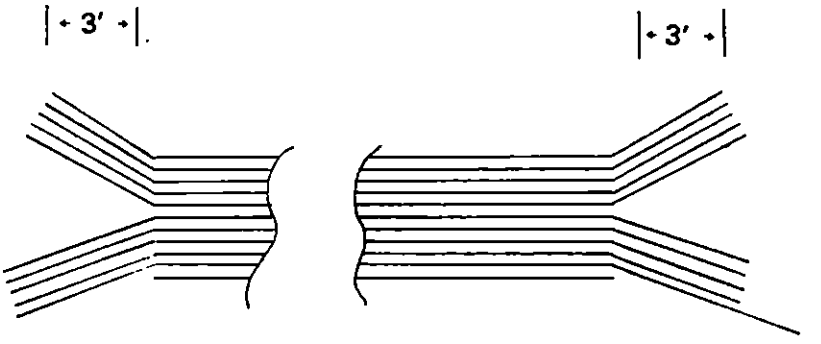


Figure 11

2. Strip each end of the wire 1/4-inch back. Exercise caution so as not to cut any of the strands of the wire. Refer to Figure 12 below.

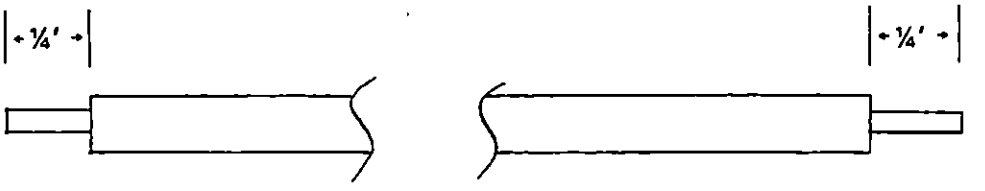


Figure 12

3. Lightly tin both ends of each of the stripped wires. Be careful not to apply too much solder as one end of this ribbon cable must later fit onto the foil leads of the jumper board.

4. Starting with the brown wire on the edge, sort every other wire so that they point up and perpendicular to the flat side of the cable (i.e., brown, orange, green, violet, white, etc., are up; red, yellow, blue, gray, etc., are down). Black equals the number 0, which is considered an even number. The even numbers go up and the odd numbers go down. Refer to Figure 13.

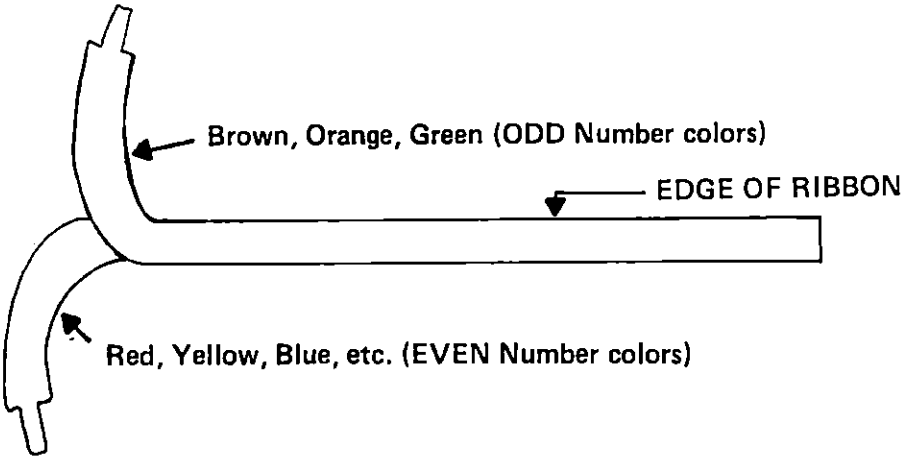


Figure 13

5. Place the odd numbers over the board so that they are on the side labeled 1 through 22.

6. Carefully solder each wire to the foil lead on each contact. Start with the end of the brown wire to the trace of contact 1, then the end of the orange wire to the trace of contact 2, then the end of the green wire to the trace of contact 3, etc. Be careful to keep contacts 19 through 22 in their proper sequence.

7. Carefully repeat the procedure to solder the rest of the wire ends to the foil lead on the other side (labeled A through Z). Start with the red wire to A, the yellow wire to B, the blue wire to C, etc. Be careful to keep contacts U through Z in their proper sequence.

8. Check to ensure that the odd-numbered wires of the ribbon cable go to the numbered side of the jumper board and that the even-numbered wires of the cable go to the lettered side of the jumper board.

Summary

During this experiment, you constructed a cable that allows you to extend jacks J1 through J5 out to a circuit work-surface. You will find this particularly useful when you start to develop your own circuits for use on RB5X.

Questions

1. Which letters on the connectors are not used as labels? Why?

2. Why is it necessary to tin the ends of the cable that are <u>not</u> connected to the jumper board?

3. What are the possible consequences of getting solder on the board contacts?

4. Why is the RB Robot Corporation jumper board wider than the jacks?

_____

_____

_____

_____

Experiment 6.2

Controlling the Extended Bus Lines

Overview

You have finished building a bus extender to allow you to easily lay out circuits for RB5X. You can now use this extender to input your own data at the user-defined ports. This helps you develop a better understanding of the interaction between hardware and software.

Objectives

Upon completion of this experiment, you will be able to:

1. Wire external switches to the bus extender;
2. Enter, save, and execute a Tiny BASIC program; and
3. Demonstrate the interaction between hardware and software.

Prerequisites

1. Chapter 5, "A Command of Tiny BASIC."
2. Have in your possession:

A. Bus extender board and cable

B. One eight-rocker DIP switch (SPST)

C. Eight 1K ohm 1/4W resistors.

References

1. CPU Board schematic in Appendix D, "RB5X Schematics."
2. Mother Board schematic in Appendix D, "RB5X Schematics."
3. Tables 3.2, 3.3, and 3.4 from Chapter 3, "The RB5X System Bus."
4. Appendix F, "Memory Map."

Technical Information

You have learned that the extender cable has given you easy access to RB5X's user-defined ports. As you remember, the ports are addressed as follows:

Port Designator	INS8255 Chip	Port Address
Port A	U20	#7808
Port B	U20	#7809
Port C	U20	#780A
Port A	U23	#7804
Port B	U23	#7805
Port C	U23	#7806

Table 6.4 - Addresses of User-Defined Ports

You are also familiar with the correlation between the bits at these ports and the wires on the bus extender. (Refer to Experiment 6.1, "Building the Bus Extender.")

Procedure

1. Switch off RB5X's power.

2. Plug the bus extender cable into J4 on the mother board.

3. Build the following circuit, which allows you to develop logic-level voltages to be used as data on the bus extender:

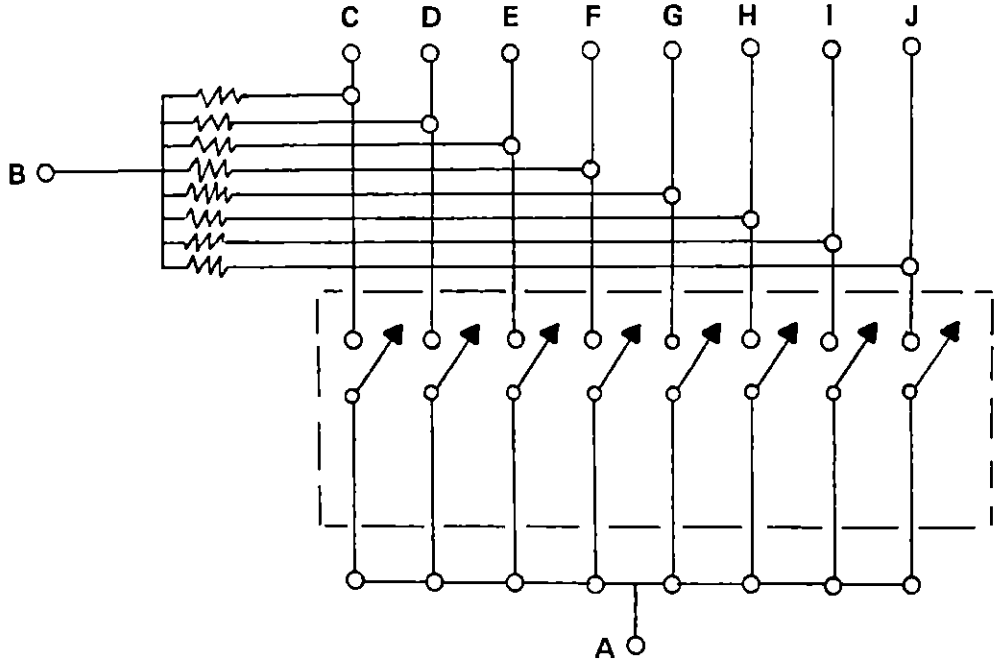


Figure 14 - Logic-Level Voltage Circuit

4. Connect the cable to the circuit using Table 6.4. (**NOTE:** Connect the constructed circuit to port A of U23.)

Schematic Connection	Cable Wire Color	J4 Pin Number	J4 Pin Signal
A	Brown (1st)	1	Ground
B	Orange (1st)	2	+5 Volts
C	Violet (3rd)	14	1A7 (MSB)
D	Gray (3rd)	R	1A6
E	Green (3rd)	13	1A5
F	Blue (3rd)	P	1A4
G	Orange (3rd)	12	1A3
H	Yellow (3rd)	N	1A2
I	Brown (3rd)	11	1A1
J	Red (3rd)	M	1A0 (LSB)

Table 6.4 - Cable/Circuit Connections

5. Switch RB5X on, making sure the RS-232 cable is connected to both the robot and your computer.

6. Boot the ASCII Pro diskette.

7. Press bumpers 2 and 4 simultaneously to enter Tiny BASIC command mode.

8. When the CRT displays the Tiny BASIC prompt >, enter the following BUS DEMO program, remembering to end each line with a Return:

```
NEW #1000
NEW
  5 REM BUS DEMO PROGRAM
 10 PRINT"TYPE THE DECIMAL VALUE YOU"
 20 PRINT"WISH TO ENTER ON THE DIP"
 30 PRINT"SWITCHES."
 40 PRINT"":PRINT""
 50 INPUT D
 60 FOR I=1 TO 5
 70 REM 5 TRIES AT MATCHING THE SETTINGS
 80 B=@#7804
 90 REM ASSIGNS THE DATA AT PORT A
100 REM OF U23 TO B
110 IF B=D GOTO 270
120 REM CHECK INPUT AGAINST PORT A DATA
```

```
130 PRINT"YOU SET THE DIP SWITCHES TO "
140 PRINT"DECIMAL ",B,"NOT TO ",D
150 PRINT"":PRINT""
160 PRINT"CHANGE THE DIP SWITCH"
170 PRINT"SETTINGS"
180 PRINT"":PRINT""
190 DELAY 1000:DELAY1000:DELAY1000
200 DELAY1000:REM 4 SECONDS
210 NEXT I
220 PRINT"DO YOU WISH TO TRY AGAIN?"
230 PRINT"(1=YES     0=NO)"
240 INPUT A
250 IN A=1 GOTO 10
260 GOTO 510
270 PRINT"":PRINT"":PRINT""
280 REM CORRECT DIP SETTING ROUTINE
290 PRINT"YOUR SWITCHES ARE SET TO "
300 PRINT"DECIMAL ",D
310 PRINT"":PRINT"":PRINT""
320 @#7802=1:@#7801=1
330 REM MOTOR RELAY AND LED ON
340 DELAY1000:REM 1 SECOND
350 @#7802=2:@#7801=2
360 REM MOTOR RELAY AND LED ON
370 DELAY1000:REM 1 SECOND
380 @#7802=4:@#7801=4
390 REM MOTOR RELAY AND LED ON
400 DELAY1000:REM 1 SECOND
410 @#7802=8:@#7801=8
420 REM MOTOR RELAY AND LED ON
430 DELAY1000:REM 1 SECOND
440 @#7802=0:@#7801=0
450 REM MOTOR RELAY AND LED OFF
460 PRINT"DO YOU WISH TO REPEAT"
470 PRINT"THE DEMO PROGRAM AGAIN?"
480 PRINT"(1=YES     0=NO)"
490 INPUT B
500 IF B=1 GOTO 10
510 FOR I=1 TO 25:PRINT"":NEXT I
520 PRINT"GOODBYE"
```

9. Check for errors and then save this program on diskette.

10. Type RUN<RET>.

11. When you see the Tiny BASIC prompt >, enter 7<RET>.

12. Fill in the blanks below for the correct, required settings of the DIP switches:

Switch	Open/Closed
C	_____
D	_____
E	_____
F	_____
G	_____
H	_____
I	_____
J	_____

13. When you see the question, "Do you wish to repeat the demo program again?" displayed on the CRT, enter Y for Yes.

14. At the Tiny BASIC prompt >, enter 246<RET>.

15. Complete the table below for the correct required settings of the DIP switches:

Switch	Open/Closed
C	_____
D	_____
E	_____
F	_____
G	_____
H	_____
I	_____
J	_____

16. Pick different values for Step 14 on your own. Repeat the procedure until you are comfortable with the operation.

    You are now able to input data into port A of U23 and to test that data.

17. Switch off RB5X's power and rewire the circuit from Step 3 so that it is connected to port B of U23.

18. Switch RB5X on and reload the BUS DEMO program.

19. Repeat Steps and 11.

    Did your DIP switch settings ever work properly? Can you figure out why not?

    You've probably realized that the circuit is now connected to port B and the software is addressed to port A. You must change the program address to access port B.

20. To change the address, change line 30 in the BUS DEMO program to read:

    30 B=@#7805

21. Save and run this new program.

22. Repeat the procedure for setting the DIP switches.

23. Do not disconnect this circuit from the bus extender, as you need it for the next experiment.

Summary

You now know how to enter data at the user-defined ports. You also know how to access this data through software. You should be aware that your expansion circuitry has a hardware and a software component. The technical and practical information you have gained from this experiment allows you to add digital expansion circuitry to your robot.

Questions

1. The DIP switches are set to the following sequences:

   C, D, E, G, I, and J - OPEN
   F and H - CLOSED

   What decimal value would be assigned to B in the software?

   _____

2. Complete the following table to connect the DIP switches to port C of U20 through jack J2.

Schematic Connection	Cable Wire Color	J2 Pin Number	J2 Pin Signal
A	_____	____	_____
B	_____	____	_____
C	_____	____	_____
D	_____	____	_____
E	_____	____	_____
F	_____	____	_____
G	_____	____	_____
H	_____	____	_____
I	_____	____	_____
J	_____	____	_____

3. If line 30 in the BUS DEMO program reads "30 B=@#7809", what port and chip would you be addressing?

   _____

4. If connections A and B were reversed in the circuit, and the DIP switches were set as follows:

   C, F, H, I, and J - OPEN
   D, E, and G - CLOSED

   what would be the decimal value read at the port?

   _____

Experiment 6.3

Know Your Addresses

Overview

In Experiment 6.2, you learned how to write programs to input data from an external port. This experiment, 6.3, allows you to INPUT data from internal and external ports, and to OUTPUT data to internal and external ports. As you may have noticed, this covers all possible combinations of inputs and outputs.

Objectives

Upon completion of this experiment, you will be able to:

1. Wire external LEDs to the user-defined ports,
2. Write Tiny BASIC programs to activate input and output ports, and
3. Modify existing Tiny BASIC programs to activate input and output ports.

Prerequisites

1. Chapter 5, "A Command of Tiny BASIC."

2. Have in your possesion:

   A. Bus extender and cable,

   B. One eight-rocker DIP switch (SPST),

   C. Eight 1K ohm 1/4W resistors,

   D. Eight 150 ohm 1/4W resistors (rocker switch pull-ups),

   E. One seven-segment LED display, and

   F. Eight LEDs.

References

1. CPU Board schematic in Appendix D, "RB5X Schematics."
2. Mother Board schematic in Appendix D, "RB5X Schematics."
3. Tables 3.2, 3.3, and 3.4 from Chapter 3, "The RB5X System Bus."
4. Appendix F, "Memory Map."
5. Appendix 5, "INS8255 Programmable Peripheral Interface," in the RB5X Reference Manual.

Technical Information

In Experiment 6.2, you learned that there are six ports at your disposal. These are called user-defined ports. There are also several at your disposal that are system-defined. These ports include the following:

1. Horn (located on the CPU board at address #7801);
2. LED board (located on the left side of RobotLab at address #7801);
3. Motor control board (located to the right of the LED board on the RobotLab at address #7802); and
4. Bumper switches (located in front of the motor control board on the RobotLab at address #7800).

You have seen these system-defined ports in action during initialization. The bumper switches are system-defined inputs only, while the horn, LEDs, and the motor controls are system-defined outputs only. As you can see from the list above, the horn and the LED board have the same address; they listen for different bits at that address. In the same way, the four motor controls at address #7802 listen to different bits. On the bumper switches at address #7800, each switch is tied to a specific bit at that address. This gives you 256 bit combinations at #7800, and thus 256 combinations of LEDs on and off.

The user-defined ports are usually set as input ports only upon initialization. To use the user-defined ports as outputs, you must work with the status byte, located at address #780B and #7807. The individual bits in the status byte allow you to decide which ports you wish to use as inputs and which as outputs. The status byte at address #780B controls ports A, B, and C of U20; the status byte at address #7807 controls ports A, B, and C of U23. Table 6.5 contains the input and output combinations possible for each INS8255 and the status byte value needed to effect this combination.

No.	Control Word Bits D7	D6	D5	D4	D3	D2	D1	D0	Port A	Port C (Upper)	Port B	Port C (Lower)
0	1	0	0	0	0	0	0	0	OUT	OUT	OUT	OUT
1	1	0	0	0	0	0	0	1	OUT	OUT	OUT	IN
2	1	0	0	0	0	0	1	0	OUT	OUT	IN	OUT
3	1	0	0	0	0	0	1	1	OUT	OUT	IN	IN
4	1	0	0	0	1	0	0	0	OUT	IN	OUT	OUT
5	1	0	0	0	1	0	0	1	OUT	IN	OUT	IN
6	1	0	0	0	1	0	1	0	OUT	IN	IN	OUT
7	1	0	0	0	1	0	1	1	OUT	IN	IN	IN
8	1	0	0	1	0	0	0	0	IN	OUT	OUT	OUT
9	1	0	0	1	0	0	0	1	IN	OUT	OUT	IN
10	1	0	0	1	0	0	1	0	IN	OUT	IN	OUT
11	1	0	0	1	0	0	1	1	IN	OUT	IN	IN
12	1	0	0	1	1	0	0	0	IN	IN	OUT	OUT
13	1	0	0	1	1	0	0	1	IN	IN	OUT	IN
14	1	0	0	1	1	0	1	0	IN	IN	IN	OUT
15	1	0	0	1	1	0	1	1	IN	IN	IN	IN

Table 6.5 - INS8255 Input/Output Combinations

After reviewing Table 6.5, you probably noticed that port C can operate both as input and output at the same time using only four bits for each function.

Let's look a little closer at how the status byte is programmed. As an example: You want to use port A as an input, port B as an output, port C-upper as an input, and port C-lower as an output (on U23). You use the status byte at line 12 of Table 6.5, or binary 10011000. Convert this binary number to decimal, or 272. To set the ports, use the command @#7807=272, which sets port A as and input, port B as an output, port C-upper as an input, and port C-lower as an output on U23.

Don't worry about the ports that you are <u>not</u> using when you set the status byte. As an example: You want port A as an input and port B as an output on U20 and don't care about port C. You could use any of the following commands:

```
@#780B=144 or
       =145 or
       =272 or
       =273
```

Procedure

1. Switch off power to the robot.

2. Plug the bus extender cable into J4 on the mother board.

3. Construct the following circuit on the designer-board end of the bus extender.

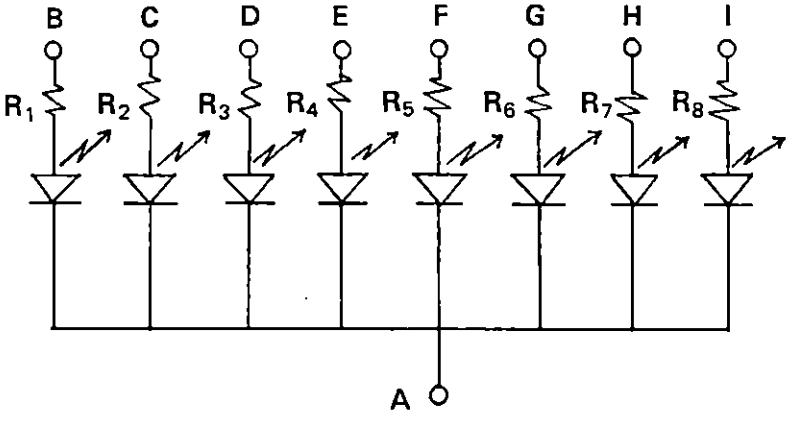


Figure 15 - Constructed Circuit

4. Connect the bus extender cable to the circuit using the following table. (**NOTE:** The constructed circuit connects to port B of U23 via jack J4 at address #7805.)

Schematic Connection	Cable Wire Color	J4 Pin Number	J4 Pin Signal
A	Brown (1st)	1	Ground
B	Yellow (5th)	Z	1B7 (MSB)
C	Orange (5th)	22	1B6
D	Brown (5th)	21	1B5
E	Red (5th)	Y	1B4
F	White (4th)	20	1B3
G	Black (4th)	X	1B2
H	Violet (4th)	19	1B1
I	Gray (4th)	W	1B0 (LSB)

Table 6.6 - Cable/Circuit Connections

5. Input the following LED DISPLAY program into the computer, remembering to end each line with a Return:

```
NEW #100
NEW
 10 REM LED AT USER PORT DISPLAY PROGRAM
 20 I=0:PRINT"":PRINT"":PRINT"":PRINT""
 30 PRINT"ENTER THE NUMBER TO DISPLAY:
 40 INPUT B
 50 I=I+1:REM LINE 60 SETS U23 AS INPUT
 60 @#7807=144
 70 REM LINE 90 ASSIGNS THE BINARY VALUE
 80 REM OF VARIABLE B TO PORT B OF U23
 90 @#7805=B
100 IF I=5 GOTO 120:REM 5 TRIES DONE
110 GOTO 30
120 PRINT"":PRINT"":PRINT""
130 PRINT"DO YOU WISH TO TRY AGAIN?"
140 PRINT"(1=YES     0=NO)"
150 INPUT C:IF C=1 GOTO 10
160 FOR I=1 TO 25:PRINT"":NEXT I
170 PRINT"GOODBYE"
```

6. Run this LED DISPLAY program.

7. Change line 60 of the LED DISPLAY program to a different status byte, one that still makes port B of U23 an output and port A an input. Write the change below:

_____

8. Rerun the LED DISPLAY program.

In the next part of this experiment, you use the switches on port A as an input and output port A's data to port B's LEDs. If you did not leave the switches hooked up from the last experiment, please refer to Experiment 6.2 for the proper wiring of the DIP switches.

9. Add the following lines to the LED DISPLAY program:

```
 10 REM DIP TO LED PROGRAM
 30 PRINT"SET THE DIP SWITCHES."
 33 DELAY 1000:REM 1 SECOND
 35 PRINT"ARE THE SWITCHES SET YET?"
 36 PRINT"(1=YES     0=NO)"
 37 INPUT A:IF A=0 GOTO 35
```

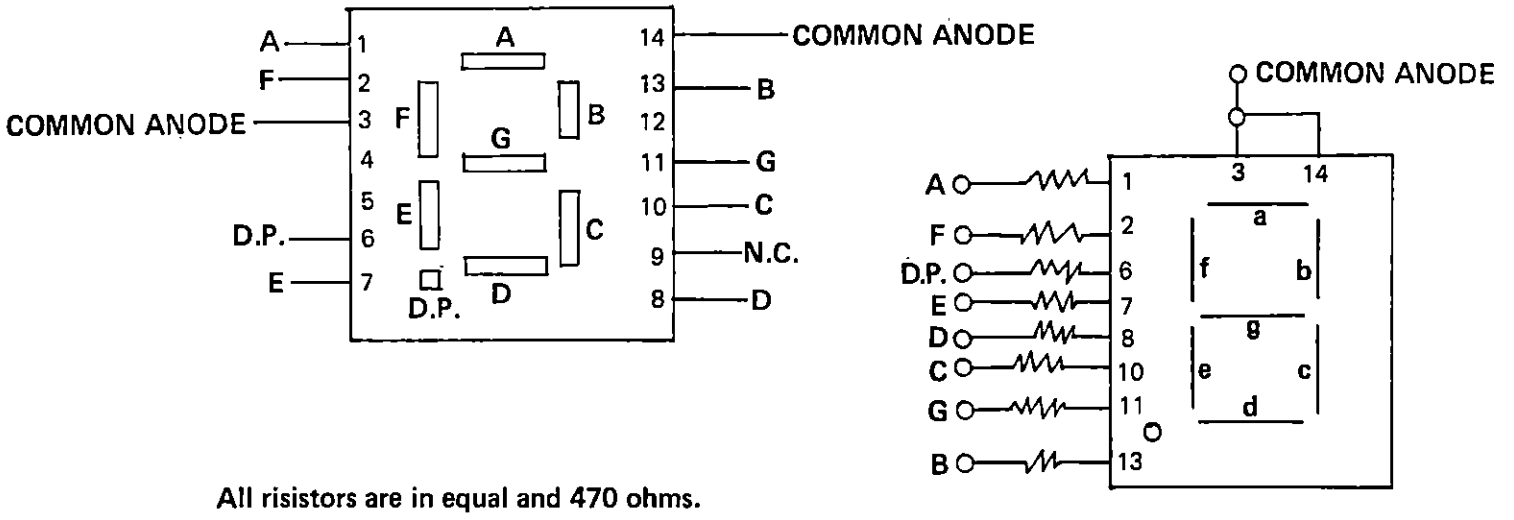
```
38 IF A<>1 GOTO 35
40 B=@#7804
45 REM LINE 40 ASSIGNS DATA AT PORT A
48 REM OF U23 TO B
93 PRINT"YOU HAVE SET THE SWITCHES "
95 PRINT"TO DECIMAL  ",B
96 PRINT"CHECK THE LEDS FOR DECIMAL  ",B
98 PRINT"":PRINT"":PRINT""
```

10. Run the DIP TO LED program you just entered.

11. Set the DIP switches to different values, and notice the LED and computer displays.

12. Disconnect both the LEDs and the DIP switches from your designer board.

13. Connect the following circuit using the seven-segment pinout shown below (using a TI 312 or equivalent):



All risistors are in equal and 470 ohms.

Figure 16 - Seven-Segment Pinout

14. Connect the bus extender cable to the following points (this is port A of U20, jack J2 at address #7808):

Port Identifier	Circuit Connection	Jack J2 Pin	Cable Color
2A0	A (pin 1)	M	Red (3rd)
2A1	B (pin 13)	11	Brown (3rd)
2A2	C (pin 10)	N	Yellow (3rd)
2A3	D (pin 8)	12	Orange (3rd)
2A4	E (pin 7)	P	Blue (3rd)
2A5	F (pin 2)	13	Green (3rd)
2A6	G (pin 11)	R	Gray (3rd)
2A7	DP (pin 6)	14	Violet (3rd)
+5 Volts	Common anode (pin 3, 14)	2	Orange (1st)

Table 6.7

15. Using Table 6.8, write a program that lights each individual LED segment one at a time. Note the negative logic employed because of the common anode connections. You may want to use the previous program to assist you in this task.

#7808 Port

MSB							LSB	
DP	G	F	E	D	C	B	A	
1	1	1	1	1	1	1	1	All LEDs off
1	1	1	1	1	1	1	0	A on
1	1	1	1	1	1	0	1	B on
1	1	1	1	1	0	1	1	C on
1	1	1	1	0	1	1	1	D on
1	1	1	0	1	1	1	1	E on
1	1	0	1	1	1	1	1	F on
1	0	1	1	1	1	1	1	G on
0	1	1	1	1	1	1	1	DP on

Table 6.8

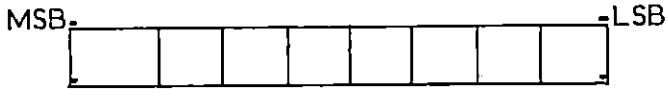
16. Write a program that causes the display to show a decimal 5.

17. Write a program that causes the display to show a decimal 3.

18. Modify the DIP TO LED program to allow you to input from the bumper switches instead of from the external DIP switches. (HINT: Lines 33, 35, 36, 37, and 38 can be deleted and an address change is needed at line 40.)

19. Verify which bumper switch corresponds to which bit in port B of U23 by filling in the blanks below:

#7805 Port



Summary

This experiment has given you experience in the following areas:

1. Internal input to external output,

2. External input to internal output, and

3. External input to external output.

You have had experience in writing, modifying, and executing Tiny BASIC programs. You are now ready to transfer data into and out of RB5X's memory.

Questions

1. Complete the following table to connect the external LEDs to port C of U20.

Port Identifier	Circuit Connection	Jack J2 Pin	Cable Color
Ground	A	_____	_____
2C0	I	_____	_____
2C1	H	_____	_____
2C2	G	_____	_____
2C3	F	_____	_____
2C4	E	_____	_____
2C5	D	_____	_____
2C6	C	_____	_____
2C7	B	_____	_____

2. To which jacks can the bus extender be connected to access port C of U20?

_____

3. Write the Tiny BASIC command to display decimal 196 on the LEDs wired to port B of U20.

_____

4. Write a Tiny BASIC program to input data from port A of U23 and to output that data to port A of U20.

_____

_____

_____

_____

_____

5. What decimal value would be useed at #7808 to have the seven-segment LED display a decimal 0 with no decimal point?

_____

NOTES:

Experiment 6.4

Activating RB5X

Overview

At this point in the course, you are familiar with the following:

1. Tiny BASIC commands;
2. ASCII Pro operation; and
3. The interaction of ASCII .Pro and Tiny BASIC to establish two-way communication between the robot and the computer, and between the user-defined ports and external circuitry.

Objectives

Upon completion of this text material, you will be able to:

1. Write Tiny BASIC programs to cause RB5X to move,
2. Write Tiny BASIC programs to cause RB5X to respond to bumper switches,
3. Write a Tiny BASIC program to cause RB5X's LEDs to light and flash,

4. Write Tiny BASCI programs to cause the RB5X's horn to sound, and
5. Activate the initialization routine on RB5X.

Prerequisites

1. Chapter 4, "Getting Started with ASCII Pro."
2. Chapter 5, "A Command of Tiny BASIC."

References

None.

Technical Information

RB5X's circuitry is identical to that of RobotLab, with the addition of the drive motors. This means that when the motor-control relays, and the corresponding green LEDs, activate, the robot moves.

To work with RB5X, place the robot in an open space on the floor, away from all hazards (such as stairs or other drop-offs), where it has plenty of room to move. If you have the option, RB5X moves best on tile; carpeting creates drag that more quickly drains the robot's batteries. The RS-232 cable connects to Port 1 on the robot's interface panel.

Keep in mind that any program you previously developed on RobotLab works on RB5X, realizing that when green LEDs are on, RB5X moves. The motor-drive relays are at address #7802, bits A, B, C, and D.

Remember that the addition of motors necessitates more frequent recharging of RB5X than was needed on RobotLab. If you want to RB5X without the RS-232 cable, you must have a NO ACTION pause at the beginning of the program, making the robot wait for a bumper press and giving you time to disconnect the cable before program execution begins.

Procedure

1. Remove the RS-232 cable from RobotLab and connect it to Port 1 on RB5X's interface panel.

2. Set up the robot for Tiny BASIC. (**NOTE:** Make sure that RB5X has the utility software module in the socket on the interface panel and that the 2K/4K switch is set to match the module's label.)

3. Enter the following program on your computer:

```
 10 CLEAR
 14 FOR I=1 TO 10:PRINT"":NEXT I:REM 10 BLANK LINES
 15 PRINT"PRESS AND HOLD BUMPERS 2 AND 4 TO
    CONTINUE"
 20 LINK#85A1:REM STARTS INITIALIZATION ROUTINE
 24 FOR I=1 TO 10:PRINT"":NEXT I:REM 10 BLANK LINES
 25 PRINT"PRESS AND HOLD BUMPER 1 TO CHANGE
    ROUTINE"
 30 DO:REM STARTS THE GREEN LEDS ROTATION CCW
    WHEN 2 & 4 ARE PRESSED
 40 @#7802=1:REM SWITCHES ON 1ST LED
 50 GOSUB 1000
 80 @#7802=2:REM SWITCHES ON 2ND LED
 90 GOSUB 1000
120 @#7802=4:REM SWITCHES ON 3RD LED
130 GOSUB 1000
160 @#7802=8:REM SWITCHES ON 4TH LED
170 GOSUB 1000
200 B=@#7800:REM CHECKS BUMPER SWITCHES
210 UNTIL (B=251):REM RUNS DO/UNTIL LOOP UNTIL
    BUMPER 1 IS PRESSED
220 @#7802=0:REM SWITCHES ALL GREEN LEDS OFF
225 GOSUB 1000
226 FOR I=1 TO 10:PRINT"":NEXT I:REM 10 BLANK LINES
227 PRINT"PRESS AND HOLD BUMPER 6 TO CHANGE
    ROUTINE"
230 DO:REM ROTATES GREEN LEDS CW
240 @#7802=8:REM SWITCHES ON 4TH LED
250 GOSUB 1000
260 @#7802=4:REM SWITCHES ON 3RD LED
270 GOSUB 1000
280 @#7802=2:REM SWITCHES ON 2ND LED
290 GOSUB 1000
300 @#7802=1:REM SWITCHES ON 1ST LED
```

```
310 GOSUB 1000
320 B=@#7800:REM CHECKS BUMPER SWITCHES
330 UNTIL (B=191):REM ENDS GREEN CW ROTATION
    WHEN BUMPER 6 IS PRESSED
340 @#7802=0:REM SWITCHES OFF ALL GREEN LEDS
344 FOR I=1 TO 10:PRINT"":NEXT I:REM 10 BLANK LINES
345 PRINT"PRESS AND HOLD BUMPER 2 TO CHANGE
    ROUTINE"
350 DO:REM START FLASHING AND AMBER LED
    ROUTINE
360 @#7801=64:REM FLASHERS ON
370 GOSUB 2000
380 @#7801=32:REM 4TH LED ON
390 GOSUB 2000
400 @#7801=16:REM 3RD LED ON
410 GOSUB 2000
420 @#7801=4:REM 2ND LED ON
430 GOSUB 2000
440 @#7801=8:REM 1ST LED ON
450 GOSUB 2000
460 B=@#7800:REM CHECKS BUMPER SWITCHES
470 UNTIL (B=254):REM ENDS FLASHER AND AMBER
    LED ROUTINE WHEN BUMPER 2 PRESSED
480 @#7801=0:REM CLEARS ALL LEDS
484 FOR I=1 TO 25:PRINT"":NEXT I:REM CLEAR CRT
485 PRINT"PRESS AND HOLD BUMPER 3 TO REPEAT OR
    1 TO STOP"
490 B=@#7800:REM CHECKS BUMPER SWITCHES
500 IF B=247 GOTO 14:REM IF BUMPER 3, REPEAT
510 IF B=251 GOTO 600:REM IF BUMPER 1, END
520 GOTO 490
600 STOP
1000 @#7801=128:REM START OF HORN AND DELAY
    ROUTINE
1010 DELAY 250
1020 @#7801=0:REM HORN OFF
1030 DELAY 1000
1040 RETURN:REM END OF HORN AND DELAY ROUTINE
2000 B=@#7801:REM START OF 2ND ROUTINE
2010 @#7801=(128+B):REM HORN AND LED ON
2020 DELAY 250
2030 @#7801=B:REM HORN OFF, LED ON
2040 DELAY 1000
2050 RETURN:REM END OF 2ND ROUTINE
```

4. Check this program for errors; when correct, save on diskette.

5. Press Return; after the ERROR 4 message, type RUN<RET>.

   Note RB5X's action and the CRT response. Does this look familiar as the initialization routine?

6. When RB5X completes the initialization routine, press bumpers 2 and 4, and continue responding to the CRT statements.

7. When the CRT displays the message PRESS AND HOLD BUMPER 3 TO REPEAT OR 1 TO END, press bumper 3 and repeat the routine.

8. When the CRT displays the message a second time, press bumper 1 to stop. The CRT then displays the message STOP AT 600.

9. Enter NEW<RET>.

10. Enter the following program segment:

```
 10 CLEAR
 20 @#7802=9:REM SWITCHES BOTH MOTORS ON
    FORWARD
 30 FOR I=1 TO 5:DELAY 1000:NEXT I:REM FORWARD 5
    SECONDS
 40 @#7802=6:REM SWITCHES BOTH MOTORS ON REVERSE
 50 FOR I=1 TO 5:DELAY 1000:NEXT I:REM REVERSE 5
    SECONDS
 60 @#7802=0:REM SWITCHES MOTORS OFF
 70 @#7802=10:REM SWITCHES MOTORS ON FOR CCW
    ROTATION
 80 FOR I=1 TO 13:DELAY 500:NEXT I:REM TURNS FOR
    6-1/2 SECONDS
 90 @#7802=0:REM SWITCHES MOTORS OFF
100 @#7802=5:REM SWITCHES MOTORS ON FOR CW
    ROTATION
110 FOR I=1 TO 13:DELAY 500:NEXT I:REM TURNS FOR
    6-1/2 SECONDS
120 @#7802=0:REM SWITCHES MOTORS OFF
130 B=@#7800:REM CHECKS BUMPERS
140 IF B<>251 GOTO 130:REM HOLDS ACTION UNTIL
    BUMPER 1 PRESSED
150 STOP
```

11. Check the program for errors; when correct, save on diskette as RB5X ACTION.

12. Type RUN<RET>. Notice RB5X's action.

13. Change the program so that RB5X moves forward and reverse for 10 seconds instead of five seconds.

    Record your changes below.

14. Change the program so that RB5X turns counterclockwise and clockwise for two turns instead of only once.

    Record your changes below.

15. Add the following lines to the RB5X ACTION program:

```
150 @#7802=9:REM BOTH MOTORS FORWARD
160 FOR I=1 TO 5:DELAY 1000:NEXT I:REM FORWARD FOR
    5 SECONDS
170 @#7802=0:REM SWITCHES MOTORS OFF
180 DO:REM START OF SQUARE ROUTINE
190 @#7802=5:REM CW TURN
200 FOR I=1 TO 3:DELAY 500:NEXT I:REM 90 DEGREE CW
    TURN
220 @#7802=9:REM SWICHES ON FORWARD MOTORS
230 FOR I=1 TO 5:DELAY 1000:NEXT I:REM FORWARD FOR
    5 SECONDS
250 B=@#7800:REM CHECKS BUMPER SWITCHES
260 UNTIL (B=239):REM DO/UNTIL LOOP UNTIL BUMPER 6
    PRESSED
270 @#7802=0:REM SWITCHES MOTORS OFF
```

16. Check for errors and save again on diskette as RB5X ACTION.

17. Once you type RUN<RET>, what bumper must you press to start RB5X on its square routine?

_____

18. Verify your answer by typing RUN<RET> and pressing that bumper.

19. What bumper must you press to terminate the square routine?

_____

Verify your answer by pressing that bumper.

20. Modify the DO/UNTIL loop to add a 1/4-second horn beep at each 90-degree turn.

21. Add another program segment beginning at line 300 to cause the amber LEDs to flash in a rotating pattern.

_____

_____

_____

Summary

In this experiment, you gained experience in exercising the various movement capabilties of RB5X. The programs in this experiment can be modified and used as the basis for your own programs, to put additional action into the robot.

Questions

1. What would happen if you changed program line 140 to the following:

```
140 IF B<>247 GOTO 130
```

_____

_____

2. What would happen if you entered the following direct execution command?

   @#7802=3

3. Write a program segment to have RB5X (1) move forward for 10 seconds, (2) beep its horn for one second, and (3) turn 180 degrees and then stop.

4. Change the DO/UNTIL portion of the RB5X ACTION program to allow the robot to perform a counterclockwise square routine. Record the changes below:

5. What would happen if line 30 of the RB5X ACTION program were changed to the following:

```
30 FOR I=1 TO 10:DELAY 500:NEXT I
```

_____

NOTES: