LABORATORY 26

WORK CELL INTERFACING

INTRODUCTION

You have studied many aspects of robotics up to this point and now you are probably ready to take on a more complex task. Having robots work with other pieces of equipment and to connect to the "outside" world is as involved as any problem you will encounter in the area of industrial robotics. The development of a complete workcell that not only moves the parts around but also checks the quality of the parts and performs other necessary jobs is the real power that robots and computer bring to the manufacturing environment.

In this lab, you will develop a complex robot workcell that uses all the fearures of the Rhino system that you have studied. Some new information will be presented in this lab and it will be explained in detail; however, this lab will not give you the step-by-step instructions in such detail as used earlier. In this lab the major tasks will be laid out and you will be invited to develop your own solutions.

The instructions in this lab are geared toward the Mark II controller. If you have a Mark III controller you can use the non-designated I/O lines to accomplish the same tasks. That would make your workcell more like the ones in industry. You can operate the Mark III non-designated I/O lines in several different ways, by writing a program in RoboTalk (TM), by writing a progrm in BASIC or other appropriate language, by using the special kernel commands available in the operating system of the Mark III controller, or by using the branching and I/O capabilities of the Mark III Rhino teach pendant.

OBJECTIVES

When you have finished this lab you will be able to:

1. Install and operate an interface card in the game port of the Apple computer.

2. Use the CONVERT program to generate data files for robot moves.

3. Use the RENUMBER program to renumber and merge programs.

4. Design a workcell that makes different moves based on the conditions of different sensor switches.

5. Integrate BASIC programs to implement the work cell described above.

LIST OF EQUIPMENT

1. Rhino XR Series robot and controller.

2. Apple IIe computer system.

3. Rhino belt conveyor (FG0864) or chain belt conveyor (no longer available).

4. Rhino rotary carousel (FG0034) or tilting, rotary carousel (FG1004).

5. Interface circuit card with sensor switches.

6. Rhino Lab Disk for Apple IIe.

7. Facilities to fabricate switch mounting brackets.

8. Rhino Com Language Card. (FG0702)

RELATED INFORMATION

This is the last lab in the manual. When you finish this lab, you should have a good idea how to design and implement automated production applications using the Rhino system. This knowledge will make you aware of the technical details and concerns that have to be considered in the design of any flexible manufacturing installation.

In the previous labs you have operated magnetic and vacuum hands and you have developed work cells that used the rotary carousel, the conveyor and linear slide base. You may have even done some work with the X-Y table. The lab that introduced you to the language card illustrated the use of the microswitches on the robot for sensing the position of the robot. With all of this variety in programming, you have not had the robot react to circumstances external to the robot system. This lab will have you set up a circuit that allow the robot system to execute preprogrammed sequences based on the closure of up to fifteen different switches.

Connecting the switches to control the robot is done through the computer. In the case of the Apple computer,

the switches are connected to a special circuit that is plugged into the game port on the circuit board of the computer. The special circuit that connects between the switches and the computer is a device called a data selector. This chip works by having a four bit binary code placed on the data select lines. By using four binary lines, the binary codes for the numbers 0-15 can be implemented in binary. As the count varies, it selects one of 16 different lines. Each of these lines can be tested to see if it is high or low. Of course the high or low is interpreted as a 1 or 0.

As the data on the different lines varies, a program can be developed that will have the robot do one sequence of moves if one switch is closed and a another move sequence if a different switch is closed. This lab will take you through all the steps necessary to complete this process for three switches. If you understand the programs and procedures presented to utilize the first three switches, you will be able to expand the system to use the full fifteen switches. By expanding the use of this interface you could have the robot respond to all types of sense inputs; including sound, light, sonar, touch and even some voice commands. This circuit and program will allow you to make your robot a lot smarter!

Three important steps are involved in the accomplishment of all this.

1. Understand the hardware. To make full use of the interface capabilities of this circuit, you will need to fully understand the circuits that are used and how they are implemented in the Apple computer.

2. Examine a program that exercises the hardware. The program is what makes it possible to operate the hardware. The program is written in BASIC so you will have no trouble understanding how it works. You will need to know every detail of the program since you will need to modify the program to have itread 16 switches instead of 3.

3. Integrate the software that operates the robot with the software that you use to teach the robot the sequences that will be controlled by the switches. The CONVERT program is used to change teach pendant developed robot moves into data statements that can be used by the INTERFACE program. Several modifications will have to be made to a variety of programs to get them all to work together.

The operation of the interface process and the other

steps can be better understood by first studying the program that makes the sytem operate.

THE INTERFACE PROGRAM

The following listing is the program that implements the interface card and directs the moves of the Rhino.

```
10  HOME:PRINT  "SWITCH CONTROLLED"
12  PRINT"        MOVE"
```

```
 14 PRINT"      PROGRAM"
 15 PRINT: PRINT: PRINT: PRINT
 16 INPUT "ENTER'1 TO RUN AND '2'TO STOP;X
 18 IF X>1 THEN END
 30 REM***************
 32 REM CYCLE SWITCHES
 34 REM***************
 40 FOR X= 0 TO 9
 50 GOSUB 140
 60 GOSUB 270
 80 NEXT X
 90 GOTO 40
100 REM ***************
110 REM  READ SWITCHES
120 REM ***************
140 LET M=X
150 IF M>= 8 THEN M=M-8: P=PEEK(49241): GOTO 170
160 P=PEEK(49240)
170 IF M>= 4 THEN M=M-4: P=PEEK(49243): GOTO 190
180 P=PEEK(49242)
190 IF M>= 2 THEN M=M-2: P=PEEK(49245): GOTO 210
200 P=PEEK(49244)
210 IF M=1 THEN P=PEEK(49247):RETURN
220 P=PEEK(49246):RETURN
230 REM ***************
235 REM  SWITCH IS SET
240 REM ***************F
270 P=PEEK(49249)
275 HOME
278 PRINT:PRINT
280 IF P>127 THEN Y=X
282 IF P<=127 THEN Y=0
285 SC=Y
295 GOSUB 320
300 RETURN
305 REM ***********
310 REM SELECT MOVE
315 REM ***********
320 IF SC=1 THEN N1=1:GOTO 400
325 IF SC=2 THEN N=1:N1=2: GOTO 355
330 IF SC=3 THEN N=2:N1=3: GOTO 355
332 PRINT "SCANNING SWITCHES"
335 RETURN
340 REM ***********
345 REM  SKIP DATA
350 REM ***********
355 FOR Q=1 TO N
360 READ Y
365 FOR R=1 TO Y
370 READ X,X,X,X,X,X,X,X
375 NEXT R
380 NEXT Q
385 REM ************
390 REM MAKING MOVES
395 REM ************
```

```
400 D$= CHR$(4):PRINT D$+"PR#2"
405 PRINT " "
410 HOME
412 PRINT "MAKING MOVE #"N1
415 READ X
420 FOR J=1 TO X
425 READ H,G,F,E,D,C,B,A
430 IF A=9999 THEN 470: REM HARDWARE HOME
435 IF A=-9999 THEN 485: REM DELAY
440 & STEP H,G,F,E,D,C,B,A
445 NEXT J
450 RESTORE
455 GOTO 16
460 END
464 REM ***************
465 REM  HARDWARE HOME
466 REM ***************
470 & HOME
475 GOTO 445
479 REM *********
480 REM   DELAY
481 REM *********
485 T=10
490 & WAIT T
495 GOTO 445
497 REM ******************
498 REM  MOVE RIGHT - M1
499 REM ******************
500 DATA 7
510 DATA 0,0,0,0,0,0,0,0
520 DATA 0,0,-650,1020,0,-850,0,-70
530 DATA -9999, -9999, -9999, -9999, -9999,
-9999, -9999, -9999
540 DATA 0,0,0,0,0,0,0,0
550 DATA 0,0,0,0,0,0,0,0
560 DATA 0,0, 650, -1020, 0, 850, 0, 70
570 DATA 0,0,0,0,0,0,0,0
680 REM  ****************
685 REM   MOVE LEFT - M2
690 REM  ****************
700 DATA 7
710 DATA 0,0,0,0,0,0,0,0
720 DATA 0,0,640,1050,0,-810,0,-40
730 DATA -9999, -9999, -9999, -9999, -9999,
-9999, -9999, -9999
740 DATA 0,0,0,0,0,0,0,0
750 DATA 0,0,0,0,0,0,0,0
760 DATA 0,0,-640, -1050, 0,810,0,40
770 DATA 0,0,0,0,0,0,0,0
900 REM ******************
910 REM MOVE FORWARD - M3
920 REM ******************
930 DATA 7
940 DATA 0,0,0,0,0,0,0,0
950 DATA 0,0,0,1080, 0, -820,-50,-50
```

```
960 DATA -9999, -9999, -9999, -9999, -9999, -9999, -
9999, -9999
970 DATA 0,0,0,0,0,0,0,0
980 DATA 0,0,0,0,0,0,0,0
990 DATA 0,0,0, -1080, 0, 820,50,50
1000 DATA 0,0,0,0,0,0,0,0
```

This program has four major portions. The first part of the program scans switches to see if they are open or closed. When the scanning process determines that a switch is closed, it remembers which switch was closed and goes to another portion of the program. The next section selects what is to happen based on which switch was closed. If switch 1 was closed, the program goes directly to the move part of the program If switches 2 or 3 are closed, the program reads portions of the data to effectively skip over the data statements that are not wanted. After the data is skipped or not, it takes several instructions to read the data and make the required moves. The final postion of the program is the data. The data statements are generated with one of the teach pendant programs and then added to the interface program. This process is easier than calculating the moves for each axis to complete a task.

The following comments explain each part of the program in detail to help you understand its operation.

Lines 10-18 are the start up sequence for the program.

Lines 40-90 are the basic scanning routine that keeps the computer continually checking the switches until one is closed. Subroutine 140 sets the values of memory locations to either 1 or 0. Subroutine 270 checks a flag location to see if it contains a 1 or 0. If the flag is 1, then the number (in this case 0 -9) that caused the flag location to contain a 1, is set as the number of the switch that was closed. This switch number is labeled "SC" and is used in a later portion of the program. Notice that line 90 has the computer return to do the loop continuously.

Lines 140-220 set the contents of memory locations to either 1 or 0, based on the number, M, that is being addressed are 49240-49247. These are 8 locations that the Apple computer uses to implement the game port. These numbers are in decimal and the equivalent hex numbers are C058-C05F.

These 8 locations are set up to be set by "peeking". This means that if a peek instruction is done to location 49240, it will set the contents of the

location to a 0. If 49241 is "peeked", it will be set to contain a 1. Within these 8 locations, the even locations are set to equal 0 and the odd locations are set to equal 1. The program evaluates a number according to its equivalent binary value and then peeks the necessary locations to place the binary number on the game port. This process sets four lines that go to a data selector chip. If the number 7 is put on the lines, in binary (0111), this causes the chip to check the voltage on pin 7 of its 16 pins that can have data on them. If pin 7 is a 0 (0v), then the number 256 is stored in memory location 49249 (C061). This effectively sets a 1 in the left bit of the memory location that will be checked by subroutine 270. If pin 7 has a 1 (5v) on it, nothing is done.

Line 140 sets M to equal the value of the loop generated by the line 40 program. If M equals 7, for example, then line 150 will be skipped and line 160 will set location 49240 to a 0 value. Line 170 would check to see if the number was larger than 4. Since 7 is larger than 4, the line would subtract 4 from 7 and then redefine M as the remainder (3). The line would then set the contents of memory location 49243 to 1. This process converts the number from decimal to binary and sets the locations accordingly. After setting 49243, the program skips to step 190. This process continues until all the digits are evaluated and then the program returns to line 60. Line 60 sends the compute to line 270.

Lines 270-300 check the flag (memory location 49249) and then go to subroutine 320. It is important to realize that the flag is checked for all values of X and SC is set to equal X only when the flag is set. SC is 0 at all other times. 127 in base 10 is 1111111 in base 2. The 1s take up the 7 lower significant bits of memory location 49249. If bit 8 is a 1, it means that the number is greater than 127. This eighth bit is the one that is used as the flag to indicate a grounded pin on data selector chip which signals a closed switch. Step 280 sets Y to equal X if P is greater than 127. Line 282 resets Y to 0 if P is not greater than 127. The program will not work properly without this step. It will repeat move 1 if the program is not totally restarted. If P is greater than 127 then SC is set to equal X, which can only be 1,2 or 3. The program then goes to line 320.

Lines 320-335 check the value of SC. If it is not 1, 2 or 3, the program prints "SCANNING SWITCHES" and returns to the scanning routine. Lines 320-330

direct the program to either location 400 or 355. If SC is 1 a count indicator, N1, is set to 1 and then the program goes to the move routine that starts at line 400. If SC is a 2, counter, N, is set to 1 and N1 is set to 2. The program then goes to line 355. SC being a 3 causes the same variables to be set and the program to go to line 355 also. The variable N1 is used in the move routine to print out which move is being made. N is used in the routine at 355.

Lines 355-380 have the computer read data and then do nothing with it. This in effect skips over the data. This process is made possible because the first data bit in each move file is a number that tells how many data lines are to be moved. In this case, that number will indicate how many lines to skip. N was set to 1 if the second switch was closed (SC=2). This causes line 355 to do the next loop 1 time. 360 reads the first data bit, that indicates the number of data fields, and uses the number to set a counter that reads the required number of data lines. If switch 3 was closed, N is a 2 and two sets of data are skipped. Once the necessary data is skipped, the program goes to line 400 to make the moves.

Lines 400-495 read the data and move the robot. Lines 400-412 set up the communication card and print the number of the move that is being made. 415 reads the data that tells how many data lines to read. 420 is the start of the read and move loop. 425 actually reads the data. 430 and 435 check for the data that indicates hardware home or delay. 9999 indicates a hardware home and that -9999 is a delay. If either of these are detected for data A, then the program goes to 470 or 485 to execute the appropriate command. Line 440 moves all 8 axes of the robot. 445 closes the J loop and 450 restores the data pointer to the beginning of the data file.

Lines 500-1000 are the data files. Notice that they are organized so that a single data entry indicates the number of 8 field data lines that are in the move. Using this structure, the program will execute as many moves as you want. If any data is added or subtracted from the move sequence, it is critical that this counter is changed to reflect the correct number of moves.

SUMMARY OF THE PROGRAM OPERATION

The program does each of the following steps, in the order listed, to operate the interface circuit board and

program.

1. The operator is given a prompt that presents the choice of running the program or returning to BASIC.

2. Once the program is running, the switches are scanned by a program that places the number 0-9 on the game port pins. These pins are connected to a data selector chip that checks the selected line to see if it is grounded or at 5v. If the switch is closed, the program sets a flag. The number of the switch that set the flag is transferred to another program.

3. Selecting the move to be made is done by going to the proper set of data. This is accomplished by going directly to the data is switch 1 was closed and by skipping over appropriate data is switches 2 or 3 were closed.

4. Moving the robot is accomplished with the special language commands that are available on the Rhino Com Language card. The program checks for hardware home data as well as data that indicates a delay.

5. Once the progam has made one move, the program returns to the opening prompt so the operator can select to make another move or quit.

PROCEDURE

As mentioned earlier, the procedure in this lab, will be different than in your previous labs. In this lab you will be instructed how to implement and use the interface circuit and program. The actual set up and use of the programs to develop complex applications with the Rhino system will be left up to you. This is a reasonable expectation since you have completed several labs with the Rhino system before doing this lab.

OPERATING THE INTERFACE SYSTEM

The first steps in this sequence have you set up and operate a demonstration program. Once you have completed the demonstration, you will go through the process of teaching the robot several sequences of steps and then converting the resultant programs so they can be used with the interface system.

_____1. Install the interface circuit in the Apple by removing the cover and carefully inserting the connector into the game socket. The game socket is located in the top right hand corner of the circuit board, very near the video connector. The socket is the WHITE Ceramic socket labeled GAME I/O. You may want to leave the cover off the Apple when this circuit is connected.

_____2. Install the communication card and set up a robot and controller ready to operate. Make sure the robot has room to extend fully to the right, left and front. Use one of the programs to locate the robot in the hardware home position with the hand vertical and the fingers closed.

_____3. Load and run the SWITCHES program. This is the program used to implement the interface system. As you close each switch, the robot should move to the right when switch 1 is closed, to the left with switch 2 and forward with switch 3. Operate the program several times to make sure you know exactly how it works. Compare the moves the robot is making with the data statements listed in this lab. If you list the program that is in the computer, you may have different line numbers for the statements than the numbers listed in this lab, but the program will be the same.

These simple steps illustrate how easy it is to operate the interface system once it is set up to run. The possibilities of this system should stir your

imagination. All you have to do to have the robot execute any preprogrammed sequence is close the appropriate switch and the robot will do what you have "trained" it to do.

The really exciting part of this is that you can design other senses so they indicate a closed switch to the robot when the senses are activated. It would be very simple (but beyond the purpose of this manual) to build a circuit that would have the robot turn on a light if a loud noise was detected.

In addition to sensing single switch closures, the program could be modified to respond to any combination of the 16 switches. This capability gives the Rhino system tremenedous flexibility.

DEVELOPING NEW MOVES

To add your own moves to the switches program requires the following steps:

1. Develop and save the programs you want the robot to do, using one of the teach pendant programs. Make sure all of the programs start and stop in EXACTLY the same place. This is critical since any error will affect the other programs. Make sure all 8 axes are reset to the home position. Name each of the programs you develop in a way that indicates their operation.

2. Convert each of the programs to a BASIC program using the CONVERT program. You may want to review the lab on the communication card to refresh your memory of the proper operation of the CONVERT program.

3. Use the RENUMBER program to add the data statements to the switches program. You will load the switches program and then merge it with the new data statements. As you add data, you will need to delete the move part of the BASIC program that is obtained as the part of the conversion process.

4. After you have all the data statements added to the program, you will need to alter the switches program to reflect the number of programs and switches your programs use.

The following steps will lead you through the process of integrating several moves of the robot with the switches program. These steps require that you know how to use the programs mentioned before you do this lab.

____4. Record three separate moves of the robot,

being sure to have each set of moves return to your 8 axis home position.

____5. Convert each of the programs and save the resultant programs.

____6. Load RENUMBER into the computer.

____7. Load each of the move programs and delete the moves potion of the program so all that is left are the data statments. Be sure to not remove the first data statement that indicates how many moves will be made. Renumber them so the first set of data starts at 1000, the second at 2000 and so forth. Be sure to resave each program.

____8. Load SWITCH-D into the computer. This is a version of the switches program that has had the data statements deleted. List this program to make sure it is in the computer.

____9. Merge each of the moves programs with the switch program. You should now have a complete program. If you used 3 switches, you are ready to operate the program. If you used more or less than 3 switches, do the following steps. If not, operate your program to check out its performance.

____10. Change each of the following lines to fit the number of switches in your configuration.

A. Line 40 - Make the maximum number match the number of switches you have. You don't need to change this if you have less than 9 switches.

B. Lines 320-330 - Add or subtract lines. Make sure the values of N and N1 are appropriate.

Now that you can have your robot respond to control switches, you are ready to have it do a variety of tasks.

Set up at least one of the following ideas to illustrate the power of this technology.

INTERFACING IDEAS

The following ideas will each require considerable imagination and work. You may have other ideas that are more appropriate for your lab facilities. These ideas are only offered to indicate the range of things that might be done.

1. Have the robot place objects on a balance beam, scale, or any other device that will weigh the objects and has a moving indicator to determine the weight. Set a switch on the scale so objects that are too heavy will flip the switch. Then program the robot to pick items from the conveyor, weigh them, and place the good items in one place and objects that are too heavy in another location.

2. Sort objects according to size. You may want to use the microswitches that are on the robot hand, or add more, for this problem.

3. Stack objects from the conveyor onto a pallet until the stack reaches a set height. Then have the robot move the pallet to a new location. You may want to keep this application simple and use switches to indicate where to stack each new item on the pallet, with a final switch indicating the movement of the pallet.

These are only three simple ideas. You will probably think of a doxen that are more interesting and challenging than these. If you can, have two or three people help you develop some of these complex applications. It is best to do relatively easy applications until you explore all the difficulties that are involved.

NAME _____

DATE _____

CLASS _____

INSTRUCTOR _____

QUESTIONS

1. Which program is used develop the data values for the robot moves?

a. CONVERT
b. TEACH
c. RENUMBER
d. ALL MOVES

2. Which memory locations are used to store the binary version of the number 2 in the scanning portion of the program?

a. 49240
b. 49243
c. 49245
d. 49247

3. What does the SKIP DATA portion of the program do?

A. Reads a data counter and then reads data fields the indicated number of times.

B. Reads data fields the number of times indicated by the switch that was closed.

C. Reads data fields and then transfers them to the next portion of the program.

D. Reads a data counter and then makes the number of moves it indicates.

4. When the program detects a closed switch, it sets a flag. This is done by:

a. Putting a 1 in bit 3 of memory location 49249.

b. Putting the number 127 in the data register.

c. Putting a 0 in bit 0 of the flag location.

d. Putting the number 256 in location 49240.

5. Which line number in the program actually moves the robot?

a. 370
b. 425
c. 440
d. The data fields.

The previous questons have been very specific. To explore some more general topics, please answer the following with short answers.

1. Why is the SKIP DATA portion of the program needed?

2. List the program changes you would need if you wanted to use 4 switches.

3. How would you set up the switches to have the robot measure the size of a block?