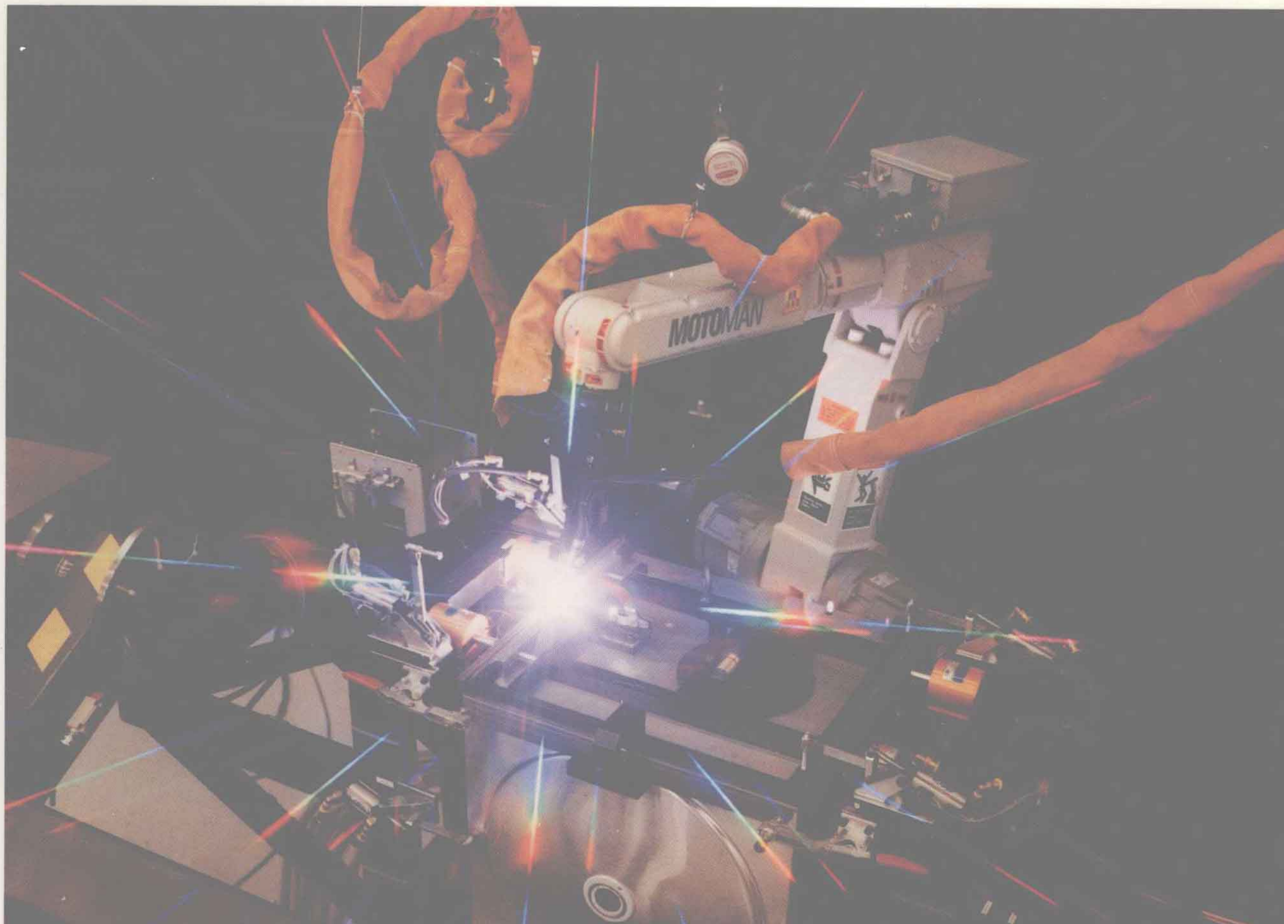
Disk Included

Laboratory Manual

Robotics Technology

James W. Masterson Robert L. Towers Stephen W. Fardo



Laboratory Manual
for

Robotics Technology

by

James W. Masterson
Department of Technology
Eastern Kentucky University
Richmond, KY

Robert L. Towers
Department of Technology
Eastern Kentucky University
Richmond, KY

Stephen W. Fardo
Department of Technology
Eastern Kentucky University
Richmond, KY

Publisher
The Goodheart-Willcox Company, Inc.
Tinley Park, Illinois

International Standard Book Number 1-56637-176-7

2 3 4 5 6 7 8 9 10 96 99 98 97 96

INTRODUCTION

This **Laboratory Manual** and its accompanying **IBM AML Programming Software** is a supplement for the text, **Robotics Technology**. It is intended to help you carry out an in-depth study of robotic systems and the subsystems which comprise them.

Robotics is a very comprehensive, applications-oriented field of study. A complete understanding of this field involves many different technical areas such as electrical principles, electronic devices, digital principles, electromechanical fundamentals, basic programming techniques, hydraulics, pneumatics, and basic manufacturing processes.

The 52 laboratory activities in this manual will help you better understand many of these topics. The activities in Unit I—Principles of Robotics, make use of the IBM AML programming software included with this manual. The laboratory activities in Units II, III, and IV are hands-on activities.

The AML software, which has been adapted for educational use, was originally developed by IBM for use with its robots. The software will provide you with an opportunity to develop and compile simple robot programs, as well as simulate the actions of a robot on a personal computer. You will be working as if you were writing a program away from a robot's workstation (a common occurrence in industry). *A robot is not required; in fact, this version of AML would not be able to operate a robot if one was available.*

This manual, programming software, and **Robotics Technology** textbook, should provide a framework for your training as a robotics technician. You should supplement your knowledge by additional reading about the technical basics of robotics or by enrolling in additional courses as necessary. We are sure that you will find this Laboratory Manual a valuable tool in your learning experiences as a student of robotics.

James W. Masterson
Robert L. Towers
Stephen W. Fardo

TABLE OF CONTENTS

Principles of Robotics

Robot Programming with AML

The computer software provided with this manual is AML *(A Manufacturing Language),* a programming language that was originally developed by IBM for use with its robots. The version that you will use was adapted for educational applications by Dr. Robert Towers, one of the co-authors of **Robotics Technology.**

You will use AML to write *programs,* or step-by-step instructions, for a robot. The program tells the robot everything it needs to know to accomplish a task: which direction to move, how far to move, and what to do at the end of the move (such as open or close a gripper). The programs that you develop as you complete exercises in this manual will be used to operate (in a computer *simulation*) the IBM 7535 Robot. See Figure 1. You will also write some programs for the IBM 7545 Robot, which is identical in shape and size to the 7535, but is more sophisticated. The 7545 is servo-controlled on all four major axes ("servos," or servomechanisms can detect errors and correct for them).

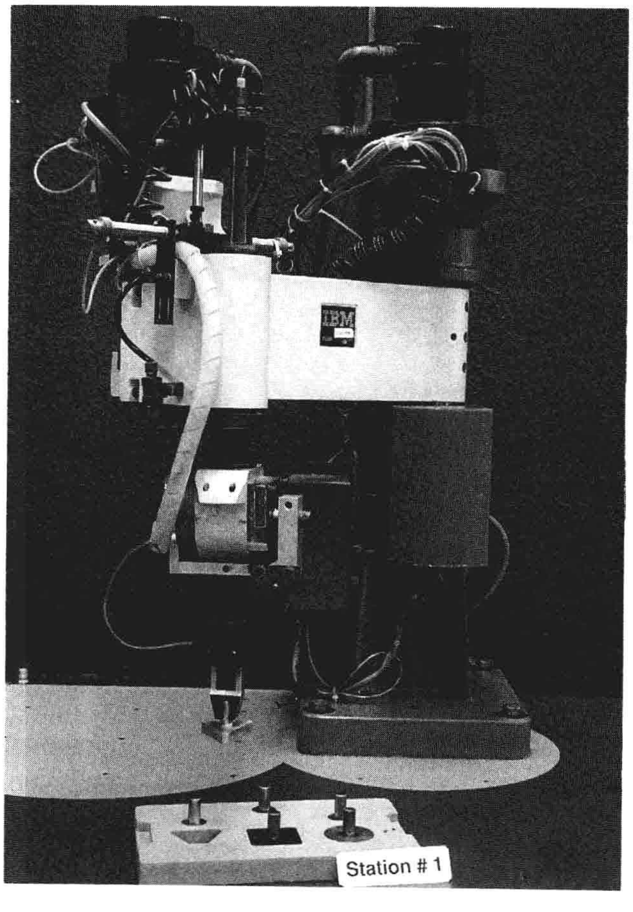


Figure 1 An IBM 7535 robot. The programs you write will be used to operate a simulated version of this robot or the somewhat more-sophisticated 7545 model.

As noted in the textbook, most robot manufacturers have developed *proprietary* languages for use with their robots. Even though these languages often are similar, each has its own set of keywords and commands. ***Keywords*** are words used in programs to identify parts of the program or actual objects, ***commands*** are instructions to ***execute*** (carry out) specific actions. Each program language also has its own syntax that must be properly used to communicate instructions to the robot. ***Syntax*** is the language's "structure," or the way that keywords, commands, and other information must be combined.

The following table shows the statements (keywords and commands) that are used for creating programs in the AML language. Additional information about each statement is available through the Help Menu in the software package.

AML Language Statements

Keywords

Name	**Description**
COUNTER	Identifies a counter.
END	Indicates the end of a program or subroutine.
NEW	Identifies a program constant.
PALLET	Identifies a pallet.
PT	Indicates coordinate values in the work envelope.
STATIC	Reserves controller storage for a counter or pallet.
SUBR	Declares or identifies a subroutine.

Commands

Name	**Description**
BRANCH	Command to go to a line containing a named label.
BREAKPOINT	Can interrupt and restart a program at any point.
DECR	Decreases a named counter by a value of 1.
DELAY	Delays the execution of the next program statement.
DOWN	Lowers the Z-axis (on IBM 7535 robot).
DPMOVE	Moves the manipulator a specific distance.

GETPART	Moves the robot manipulator on the named pallet.
GRASP	Closes the robot's gripper.
INCR	Increases a named counter by value of 1.
ITERATE	Repeats a command or subroutine using new values.
LINEAR	Controls straight line manipulator movement.
NEXTPART	Increases current part count by 1.
PAYLOAD	Controls the tool tip speed.
PREVPART	Decreases current part count by 1.
PMOVE	Moves robot manipulator horizontally.
RELEASE	Opens the robot's gripper.
SETC	Sets a named counter to a specfied value.
TESTC	Tests a named counter for a specified value.
TESTI	Checks a digital input port for a true condition.
TESTP	Compares pallet's current part indicator to a value.
UP	Raises the Z-axis (on IBM 7535 robot).
WAITI	Instruction to wait for a specified digital input.
WRITEO	Opens or closes a digital output port relay.
ZMOVE	Controls Z-axis movement (on IBM 7545 robot).
ZONE	Controls tool tip tolerance.

Subroutines

The program instructions in AML are *subroutine-oriented.* A ***subroutine*** is actually a small program, with its own beginning and ending, that is used *within* the larger program. Subroutines are designed to accomplish some distinct action or purpose, such as picking up an object, moving to a specified point, or waiting for a certain period before carrying out an instruction. The main program can "call" (start) subroutines as needed to perform their tasks. Some subroutines may be used again and again in a program; others only once.

Each subroutine, as noted, has a beginning and an ending. The beginning of the subroutine is its name, commonly known as the ***identifier.*** The identifier is immediately followed by a colon (:), called the ***definition operator.*** Each subroutine terminates with an *end statement,* consisting of the keyword END and a semicolon (;). The semicolon, or ***statement delimiter,*** must be used to end every line in the subroutine, except comments. Figure 2 illustrates the functional parts of a subroutine.

The identifier must meet these criteria:

- Can be up to 72 characters.
- First character must be alphabetic (a letter).
- Remaining characters may be alphabetic, numeric (numerals), or underscores (_) in any combination. No special characters are permitted.

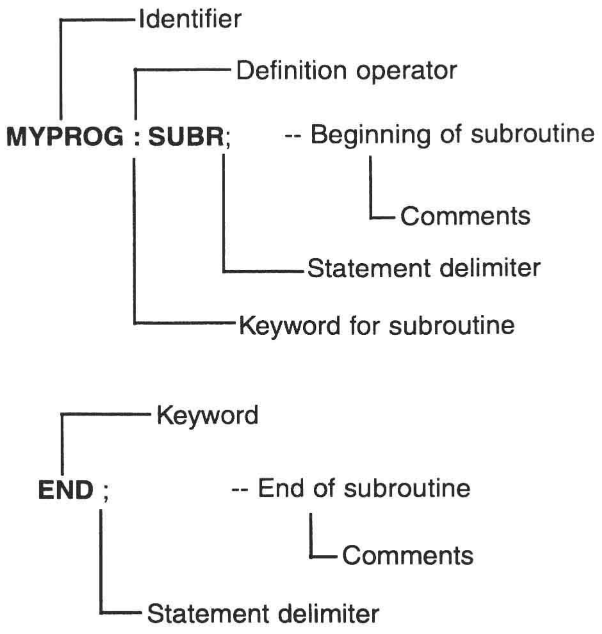


Figure 2 Functional parts of an AML subroutine.

Calling Subroutines

A program generally consists of two or more subroutines, each with a specific task. The main program calls each subroutine as needed. Subroutines can also call other subroutines. There are distinct rules for calling or executing subroutines:

1. A subroutine must be previously declared.
2. The called subroutines must be:
   a. on the same *level* as the calling subroutine, or
   b. owned by the calling subroutine.

Figure 3 illustrates these rules. The *main* subroutine can call both the pickup and putdown subroutines, since it satisfies rule 2b. The *putdown* subroutine can call the pickup subroutine, because it satisfies both rule 1 and rule 2a. The *pickup* subroutine cannot call the putdown subroutine, however, because it does not satisfy rule 1.

Structure of AML Programs

All AML programs have two major sections. One is referred to as the ***Global Declaration Section,*** the other as the ***Executable Section.*** The Global Declaration Section includes information (such as coordinates of locations) that may be applied by various subroutines in the program, while the Executable Section consists of commands to be carried out.

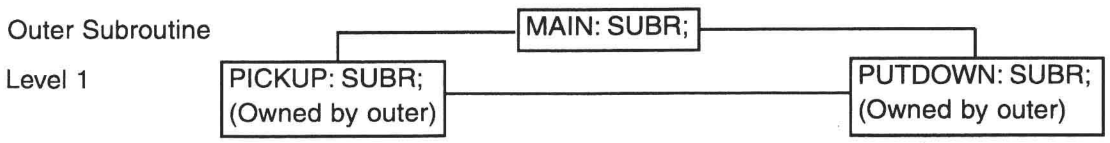

Outer Subroutine
Level 1
MAIN: SUBR;
PICKUP: SUBR; (Owned by outer)
PUTDOWN: SUBR; (Owned by outer)

Figure 3 Flowchart of the MAIN, PICKUP, and PUTDOWN subroutines.

These considerations must be kept in mind when developing AML programs:

- The program name can be up to 8 characters long. It can include letters, numbers, and underscores, but must start with a letter.
- A colon (:) must be used between an identifier (such as “MAIN”) and a keyword (“SUBR”, for example).
- Each line of a program (except a comment) must end with a semicolon (;). Comments can be included by beginning their text with two hyphens (--).

Following is a short sample program that illustrates the preceding considerations. It is used to move the robot’s end effector (hand) to location A, then to location B.

```
EXE_2.AML
-----------------Global Declaration Section------------------
1. A:NEW PT(X,Y,R); --XYR location to be taught
2. B:NEW PT(X,Y,R); --XYR location to be taught

---------------------Program Execution--------------------
4. MAIN:SUBR;
5. PMOVE(A); --move to point A
6. PMOVE(B); --move to point B
7. END;
```

More detailed information about various aspects of AML programming will be provided as you progress through the activities in this unit of the *Robotics Technology Laboratory Manual.* More information regarding the terms used in this example can be found in the Help Menu in the software package.

Installing the IBM AML Programming Language Simulator

System Requirements:

IBM-compatible computer with at least a 286-generation microprocessor, 2MB RAM, 3.5 inch floppy disk and a hard drive with 1.5 MB free space, mouse, VGA color display.

Hard drive installation:

1. Turn on the computer and monitor.
2. When the **C:\>** prompt appears on the screen, type **MKDIR ROBOT** and press ENTER.
3. Type CD\ROBOT and press ENTER to switch to the directory you have just created.
4. Your command prompt should look like this:

   C:\ROBOT>
5. Put the disk containing the AML Programming Simulator into the 3.5 inch floppy disk drive. Type

   COPY A:\*.*C:

   and press ENTER.

   A is the letter typically assigned to the 3.5 inch drive containing the AML Programming Software (this drive could use B or another letter). C is the letter typically assigned to your hard drive.
6. This will copy the AML simulator to your hard disk. To start the simulator, type in AML and press ENTER.

Floppy drive operation:

You can run the AML Simulator from the original disk. The only requirement is that your computer must have two floppy disk drives, one for the program and one for the data disk. You cannot run the AML Simulator from a single floppy disk drive.

Activity 1—Programming Environment

Name ____Date ____________________Score ____________

Objectives:

The purpose of this exercise is to provide hands-on activities in working with the AML Programming Language that has been revised for use with the **Robotics Technology** textbook. The knowledge you gain from using this programming language will help you have a better understanding of other robot programming languages.

After completing this exercise, you will be able to:

- Δ Boot up the computer, call up the AML Programming Environment, and move among the program menus.
- Δ Create a robot program, teach point location positions, successfully compile the robot program, and run it on the program simulator.

Equipment and Materials:

- IBM-compatible computer with mouse and VGA color display.
- Robotics Technology Programming Software package.
- Printer to produce a hard copy of your program.

Procedure:

Note: The procedure that follows assumes that the AML Robotics Technology Programming Software has been installed on the computer's hard disk drive (designated as "C:" on most systems). If your software has not yet been installed on the hard drive, or if you are using a system with two floppy drives, refer to Installing the IBM AML programming Language Simulator on Page 8.

1. Turn on the computer and monitor. Insert a blank, formatted disk in the floppy drive.
2. When the **C:\>** prompt appears on the screen, type CD\ROBOT. Press the ENTER key.
3. Type AML and press ENTER. The Programming Menu will appear on screen, Figure 1-1.

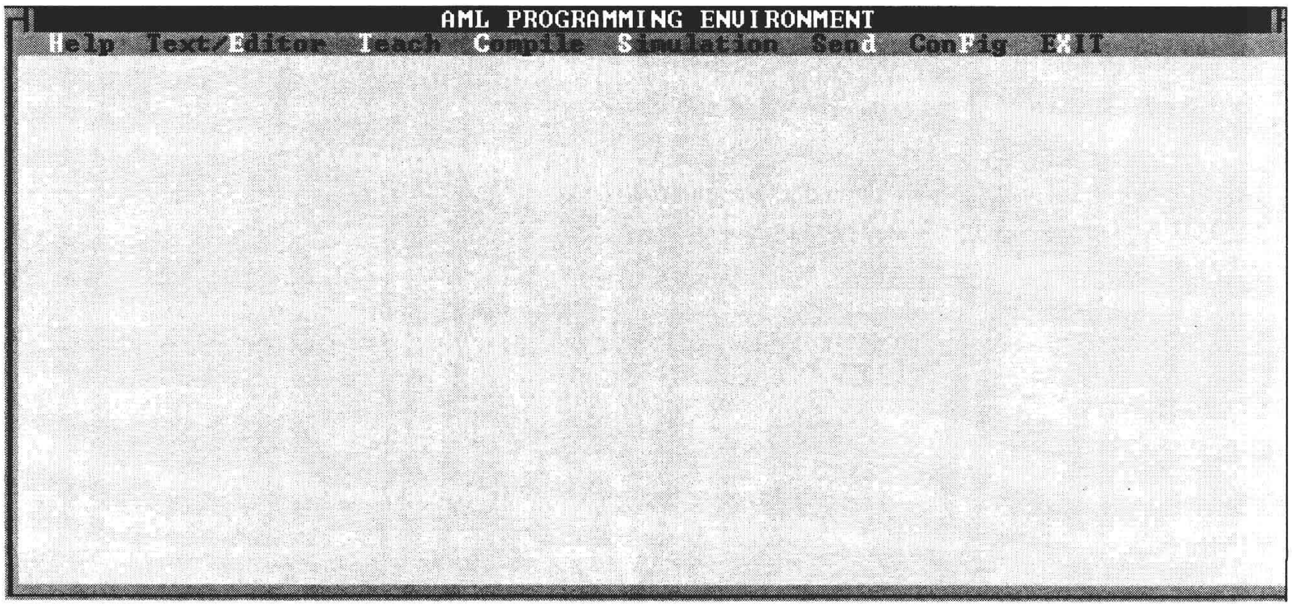


Figure 1-1 Programming Menu Screen.

4. Move the mouse until the cursor (usually, a small blinking square) is on the word Help. Press ("click") the left mouse button to bring up the Help Menu.

5. Select (place the cursor on, then click) the word Logo. The Software Information Menu will then appear. Click on OK to return to the Main Menu.

6. Click on Help again. This time, select System Help. You can use this to find out more information about the menu commands. Select one of the headings listed on the left of screen (for example: Teach) and read the text accompanying it. Click on Exit Help to return to Main Menu.

7. Click on Help again. Select AML/Programming. Information on AML programming commands and keywords can be retrieved by using this menu. Examine several entries, then click on Exit Help to return to Main Menu.

   Now that you are familiar with the help screens you can move on to learning about programming. First, you will type in a short program, then save it. You will also gain experience in using the Teach mode, compiling your program, and running it as a simulation.

8. Select Config from the Main Menu. Set the robot configuration to 7535.

9. Return to Main Menu, and select Text/Editor. Carefully type the following. Use only *CAPITAL* letters and be sure to use the *exact* spacing and punctuation shown.

```
MAIN:SUBR;
 PMOVE(PT(100,200,0));
 DOWN;
 GRASP;
 DELAY(.5);
 UP;
 PMOVE(PT(100.0, 200.0, 0.0));
 DOWN;
 RELEASE;
 DELAY(.5);
 UP;
END:
```

10. Click on File. You will now save the program file to your data disk, which is located in the floppy disk drive, "A" is most common. If your drive is "B," type that letter, instead. To save the file, select Save As, then type in the file name A:EXER1.AML. Press the ENTER. To leave the Text/Editor, select File again and click on Exit.

11. At the Main Menu, select Teach. When prompted for the file name, type in A:EXER1.AML

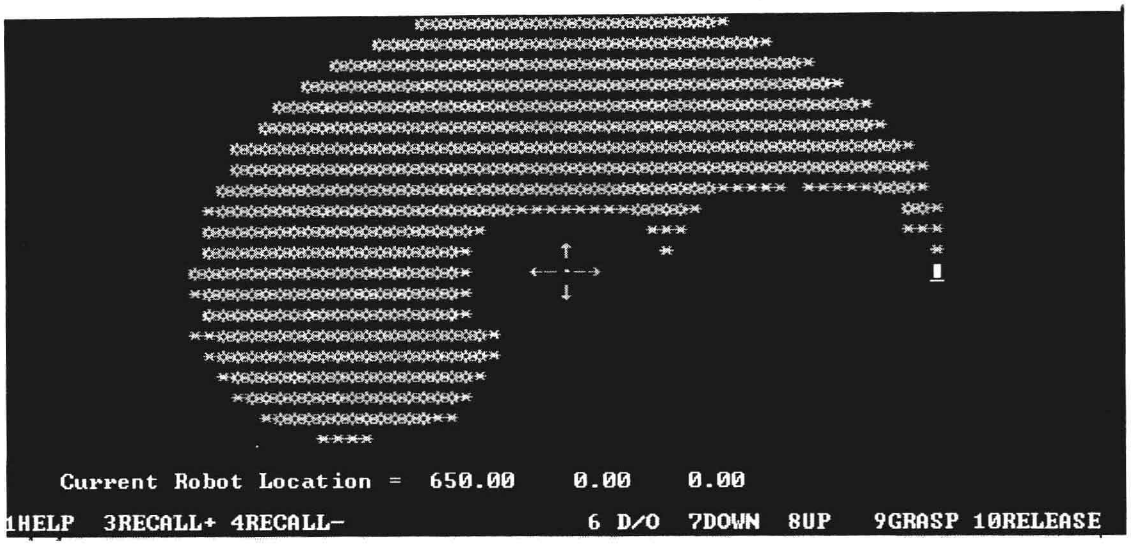
    The Teach mode allows you to edit the point positions in the work envelope. This can be done automatically by the program, or manually by using the keyboard. You will change the points in the program you have just coded; first by entering the points into the program, then by using the keyboard.

12. Press the F6 key. The following message should appear on screen:

```
COMMUNICATIONS ERROR
Check that controller is ON-LINE and cable is attached
Abort teach? (y [yes], n[no], i[ignore])__
```

    If your computer was connected to a robot controller, you could download the program directly to the controller. You will be programming as if you were writing a program away from the robot's work cell. Type the letter I and ENTER. A representation of the robot work envelope will appear on the screen, Figure 1-2.

Name ______________________________



```
Current Robot Location =  650.00     0.00     0.00
1HELP  3RECALL+ 4RECALL-              6 D/O  7DOWN  8UP   9GRASP 10RELEASE
```

Figure 1-2 Robot Teach Screen.

13. Here, you will enter in the new work envelope values and let the program adjust the robot's manipulator. Press the ESC key. Type the number 300 and press ENTER. Type the number 400 and ENTER. Type the number –150 (Note the minus sign), then press ENTER. After you have entered the new values, you will see the cursor move to the robot location of X = 300, Y = 400, R = –150.

14. Press the END key. Move cursor to the second parenthesis in line 2. Press the F7 key to recall the robot point location. (You will need to insert a right parenthesis) before the semicolon in line 2 because one of them will be copied over by the teach program.)

    Now you will learn how to adjust manipulator position manually by using the keyboard.

15. Move the cursor to the second parenthesis in line 7. (PMOVE(PT(100.0, 200.0, 0.0));

16. Press the F6 key. While holding down the Shift key, use the Arrow keys on the numeric key pad to move the robot to X = –400, and Y = –100. For finer control movements, do not press the Shift key. Use the PgUp or PgDn to where R = 25.00 (approx.).

17. Press the End key and press the F7 key to recall the robot point location. Be sure to insert a right parenthesis) before the semicolon in line 7.

    (PMOVE(PT(–400.0, –100.0, 25.00);

18. Press F8 key to save your program changes. Return to the Main Menu.

    Next, you will compile the program. The purpose of the compiler is two-fold. First, it checks for errors in your programing; second it converts the English-like AML statements to computer machine code.

19. Select Compile and type the filename EXER1. Type N after the hard copy, .lst, and .sym prompts.

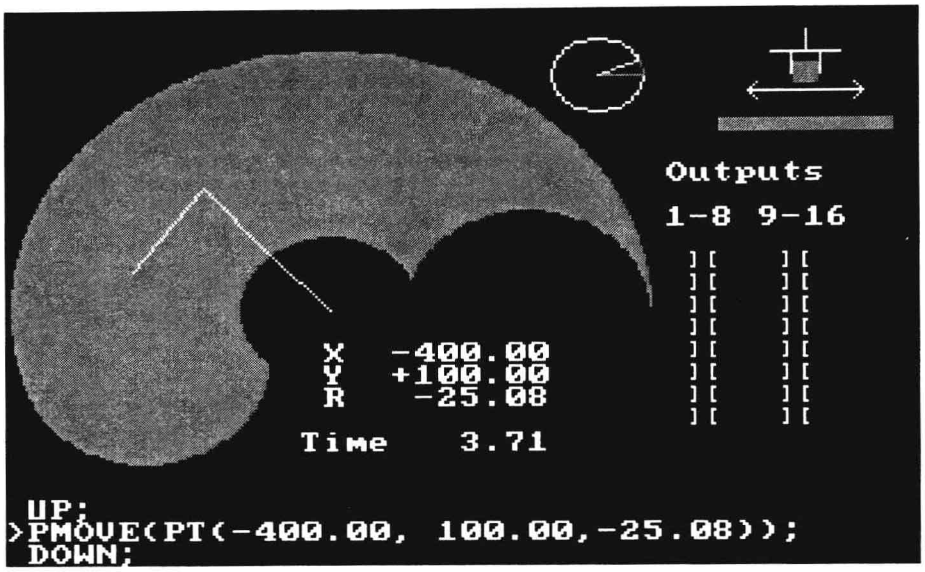
    An error message will appear on the screen concerning the END statement. Ignore this message – press any key to return to the Main Menu.

20. Select Text/Editor. Type in the filename EXER1. Move the cursor to the END statement and change the colon (:) to a semicolon (;). Save the file and exit to Main Menu.

21. Repeat the steps listed in item (20). The message, Successful Compilation, will appear on the screen. Press any key to return to the Main Menu.

22. Select Simulation. You will get a prompt asking you if you want to change the model number of the robot you wish to simulate. Press ENTER at this prompt.

23. Type in EXER1 for file name. Type the letter N after the change input and metric to inches prompt. On the screen, you will see a simulation of a robot carrying out the program you have written, Figure 1-3.



```
X   -400.00
Y   +100.00
R    -25.08
```

```
 UP;
>PMOVE(PT(-400.00, 100.00,-25.08));
 DOWN;
```

Figure 1-3 Robot Simulation Screen.

24. At the End of Program. Repeat [y:n]_ prompt, ENTER Y to run program again. You can print a hard copy of program run times by typing Y at the print timing report prompt. If you do not have a printer connected, type N. You will exit the simulator program.
25. Go back to the Text/Editor and open EXER1. Click on the File menu and select Print. This command will print a hard copy of your program. Click again on File and on Exit to return to the Main Menu.
26. At the Main Menu, select Exit to leave the program. Remove your data disk and turn off the computer.

Analysis:

1. What programming statement (command) causes the robot to move to a point location?

____

2. List the steps needed to run a successfully compiled program on the simulator.

____

____

____

____

____

____

____

____

3. What are two purposes of the compiler?

____

____

Name ____

4. While in the teach mode, what two methods can be used to move the robot to a particular point location?

5. What menu would you choose to find information about a programming command?

6. Which function key is used to make the work envelope appear on the screen in the teach mode?

7. What steps are used to recall the data into the text portion of the program after teaching a point location?

8. How do you bypass the communication error message and make the work envelope appear on the screen?

9. How many seconds does the program take to execute (run) on the simulator?

Activity 2—Movement of Peg Activity

Name ______________________________ Date ____________ Score ________

Objectives:

One of the most common tasks performed by robots in industry involves pick and place activities. The purpose of this exercise is to write a program to move three pegs from one peg holder to another, while at the same time controlling the movement speed of the robot.

After completing this exercise, you will be able to:

- Δ Write a program that will move three pegs from one holder to another while varying the speed of the robot's movements.
- Δ Teach point location positions, compile the robot program, and run program on the program simulator.

Equipment and Materials:

- IBM-compatible computer with mouse and VGA color display.
- Robotics Technology Programming Software package.
- Printer to produce a hard copy of your program.

Note: The illustrations in this activity are meant to give you an idea of the physical relationship between a robot and the supporting material referred to in the activity. The purpose of these exercises is to teach the basics of robot programming. This programming exercise will be done entirely on the computer without any additional laboratory equipment.

Procedure:

1. Turn on the computer and monitor. Insert your program disk in the floppy drive. At the C:\> prompt type CD\ROBOT and press ENTER. Type AML and press ENTER to start the programming menu.
2. Refer to Figures 2-1 and 2-2 for equipment layout in this exercise.

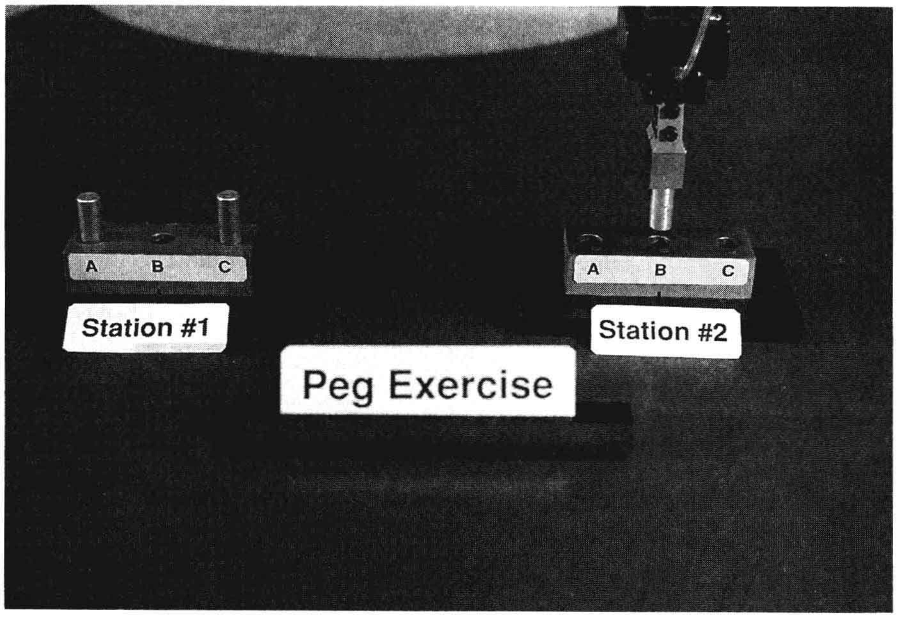


Figure 2-1 Equipment layout for Activity 2.

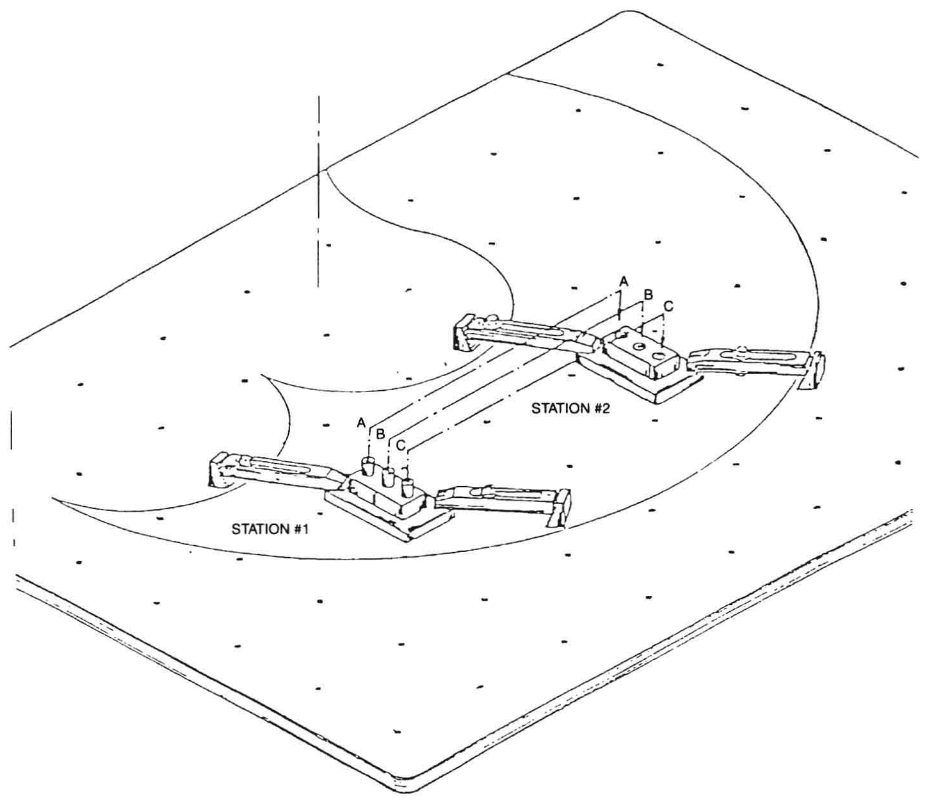


Figure 2-2 Plan view layout of equipment for Activity 2.

3. You will be using a four axis IBM 7535 Robot (see Figure 1 in Principles of Robotics).

The Z axis on the IBM 7535 is a non-servo control axis. Because of this, the robot is only capable of stopping at two positions along the Z axis, either fully up or down. The robot is equipped with an air-actuated gripper. A delay statement will have to be used in your program whenever the gripper is opened or closed. Use the following programming statements and keywords to construct your program. Refer to the Help Menu in the programming software for additional assistance.

AML Commands and Keywords

DELAY(sec.) --delays the execution of next statement (.1- 25.5)
DOWN --lowers the Z-axis
END --keyword required for each subroutine
GRASP --closes gripper
NEW --keyword to identify a constant (a constant may be a number, point, or character string)
PAYLOAD(value) --controls the tool-tip speed (1-10) 1 for slowest, 10 for fastest
PMOVE(point) --moves manipulator horizontally from one location to another
PT --keyword that indicates three value (X,Y,R) name:NEW PT(X,Y,R); or PMOVE(PT(X,Y,R));
RELEASE --opens gripper
SUBR --keyword that declares or identifies a subroutine.
EXAMPLE: id:SUBR; or id:SUBR(parameter);
UP --raises the Z-axis

4. Before constructing your program, click on Config from the Main Menu and select 1 to set the robot configuration. Set the configuration for the 7535 robot and press ENTER. Save your settings by pressing 4 and ENTER.

5. Click on the Text/Editor and begin writing your program using only CAPITAL letters.. On the first few lines, use comment statements to explain the function of the program. Comment statements on each line of the program must be proceeded by a double hyphen (--).