DEVELOPMENT OF AN INTERACTIVE PROGRAMMING SYSTEM FOR THE IBM 7545 ROBOT

by

Radhakrishnan Jayaraman

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Industrial Engineering and Operations Research

APPROVED:

M. P. Deisenroth, Chairman

M. S. Jones

J. W. Roach

June 2, 1986

Blacksburg, Virginia

DEVELOPMENT OF AN INTERACTIVE PROGRAMMING SYSTEM FOR THE IBM 7545 ROBOT

by

Radhakrishnan Jayaraman

M. P. Deisenroth, Chairman

Industrial Engineering and Operations Research

(ABSTRACT)

This thesis presents the development of an interactive programming system for the IBM 7545 robot. Various methods of robot programming are discussed, and the reasons for the development of such an interactive programming system are provided.

The development of this system was divided into five phases, namely, the development of the pseudo-compiler, development of the "system control" program, integration of ASSEMBLY routines, development of the "motion control" program, and the development of test programs. The approach used for each of these five stages are outlined, and a reference to the use of the system is given.

A description of the development of each stage is then given, and the logic associated with all programs are described, and the purpose and operation of all subroutines are also presented.

Some assumptions and limitations of the system are explained, and the operational aspects of the system are described. Additional work needed to improve this system is outlined, and the feasibility of using the concept of this system on other robot programming languages on the IBM 7545 robot are also discussed.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my chairman, Dr. Michael P. Deisenroth, for his constant support and encouragement during my entire program at Virginia Tech. Without his guidance and his vast experience, this research would not have been completed successfully in a timely manner.

I would also like to thank Dr. Marilyn S. Jones and Dr. John W. Roach for their contribution to this research effort, and agreeing to be on my committee.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1.0 CHAPTER 1. INTRODUCTION

Programming of industrial robots, introduced in the early '60s, was done using teach methods. Although these methods were sufficient for simple applications like spray painting and spot welding, where the proportion of teaching time to production time is small, they were not suitable for complex applications like palletizing and operations where hundreds of points must be individually taught. Also, the teach methods were an on-line programming system and hence, the robot had to be taken out of production during the programming phase. Due to these and other factors, a significant amount of research and development efforts have been directed at off-line programming systems.

Current off-line programming systems can be divided into textual languages (like VAL, AML, MCL, etc.) and graphical programming systems (like GRASP, GRPPS, PLACE, etc.). Some of the off-line systems include graphical simulators to verify the accuracy of the program and to allow the user to see a graphically simulated operation of the robot in a work cell, check for collisions and the ability of the robot to reach specified points in the work cell.

Many recent approaches to robot programming seek to provide the power of robot-level languages without requiring programming expertise. An approach, known as task-level programming, requires specifying goals for positions of objects, rather than the motions of the robot needed to

achieve those goals. In particular, a task-level specification is meant to be completely robot independent; no positions or paths that depend on the robot geometry or kinematics are specified by the user. Task-level programming systems require complete geometric models of the environment and of the robot as input; for this reason, they are also referred to as world-modeling systems [11]. Development of comprehensive task-level programming systems is still in the research stages. Partial implementations of task-level systems include AUTOPASS, LAMA and RAPT.

1.1 PROBLEM STATEMENT

Although off-line programming has several advantages over the teach method, the time spent in generating and debugging the program is considerable. Also, with most commercial textual languages, programs are finally debugged using the robot itself. Debugging a robot program is a complex task. It is not unusual for the debugging stage of a robotic application to take more than twice as long as the development stage. Therefore it is critically important for a robotic system to provide powerful debugging aids [3]. Debugging time could be shortened with the use of interactive programming systems. Using such a system programming errors could easily be traced to the line of source code which created the condition and interactive debugging principles could be utilized to speed up program development time. It would also allow partial execution of programs to check for their performance. Such a system would enable programmers to modify their programs and execute them immediately, without spending time going through a compilation phase.

1.2 OBJECTIVE

Currently, the IBM 7545 robot is programmed in the AML/E language using a text editor resident on an IBM PC. A limited graphical simulation package is available for program testing before program compilation. This program is then compiled and downloaded to the robot controller for final test and debugging. The purpose of this research was to develop an interactive programming system for this robot to increase the effectiveness of the programming environment. The reasons for the development of the interactive programming system are as follows:

1. Enable the user to develop and debug programs in an interactive mode from an IBM PC, without having to spend time compiling and downloading programs to the robot controller.

2. Enable the user to interactively debug errors associated with the source program. Certain errors are not detected during compilation, and during the execution process, little information is provided to the user about the specific cause of the error and the statement in the source which created the error.

3. Enable the user to control execution of programs through single-stepping and partial execution.

4. Enable the user to move the robot manipulator using interactive commands entirely on an IBM PC.

The development of this system was broken down into five phases:

1. Development of a pseudo-compiler.
2. Development of an interpreter and "system control" program.
3. Integration of ASSEMBLY subroutines.
4. Development of an AML/E "motion control" program to reside in the controller.
5. Development of test programs.

The pseudo-compiler,for AML/E programs, was developed using Professional FORTRAN (Version 1.14) on the IBM PC/AT. The purpose of the pseudo-compiler was to enable faster execution of the user's program through the generation of an object file, instead of executing it in an interpretive fashion. This program generates an object file, a symbol table, and a list of constants found in the program. This pseudo-code is then interpreted by the FORTRAN program, described in the next paragraph, and these commands are communicated to the robot controller. The pseudo-compiler was designed to handle a subset of AML/E commands.

The next phase involved the development of a Professional FORTRAN program on the IBM PC/AT, which will interpret the user's AML/E program and using communication subroutines, written in ASSEMBLY, will transmit and receive parameters to and from an AML/E "motion control" program resident in the robot controller. The "system control" program on the IBM PC/AT is responsible for computing and keeping track of all variables, counters, DI/DO points and pallet parameters. It also has several error detection

features and the capability of providing the user with specific error messages, not detected by the AML/E compiler. This program was designed with menu features to enable the user to specify execution options, like single-stepping, partial program execution, subroutine execution and program execution with the robot arm moving slowly.

Another phase involved the integration of the ASSEMBLY subroutines, which are responsible for all communications between the host computer and the robot controller, and the FORTRAN program used to interpret the user's AML/E program.

An AML/E "motion control" program was also developed. This program resides in the robot controller and interactively control robot movement relative to the user's AML/E program. This AML/E program reserves locations in the controller memory for variables. The FORTRAN "system control" program transmits values for these variables which are dependent on the commands in the user's AML/E program. These variables are then used, by the AML/E "motion control" program in the controller, for execution.

Test programs were then developed to aid in the debugging process of the interactive programming system. With the help of these test programs, the capabilities and limitations of the system were identified.

2.0 CHAPTER 2. LITERATURE REVIEW

Relevant to the proposed research, literature in the following methods of robot programming were reviewed:

1. Teach programming
2. Off-line programming
3. Interactive programming

2.1 TEACH PROGRAMMING

Teach programming is a means of entering a desired control program into the robot controller. In teach programming the robot is manually led through a desired sequence of motions by an operator who is observing the robot and the robot motions as well as other equipment within the work cell. The teach process involves the teaching, editing and replay of the desired path. The movement information, as well as other necessary data, is recorded by the robot controller as the robot is guided through the desired path during the teach process. The operator can also edit the program to add supplemental data to the motion control program for automatic operation of the robot or associated production equipment. During the teach process the operator may desire to replay various segments of the program for visual verification of the motion or operations. Teach replay features may include both forward and backward replay, single-step operations, and operator-selectable replay motion speeds [6].

2.2 OFF-LINE PROGRAMMING

Off-line programming can be defined as the task of programming through the use of remotely generated point coordinate data, function data, and cycle logic [15]. It eliminates the need for each point to be taught with standard lead-through methods. When off-line programming is used, the robot remains in operation while a new program is being generated. This means more available productive time for both the robot and its associated process equipment.

Most of the commercially available off-line programming systems are manipulator level languages, where the robot movements are specified in terms of world positions of the manipulator attached to the robot structure. Mathematical techniques are used to determine the individual joint values for these positions [16]. An overview of a few off-line programming systems, classified under textual and graphical programming systems, is presented.

2.2.1 Textual Programming Systems

AL, developed at Stanford Artificial Intelligence Laboratory, deals with movements of objects rather than the arm itself. The AL language has an ALGOL-like structure, and provides constructs for control of multiple arms in cooperative motion. It was designed to facilitate programming assembly operations using extensive force sensing capabilities [10].

AML is a high-level structured computer programming language designed by IBM for use with a robot system. It provides the function of a general-purpose computer language, offering a variety of data types and operators, language control structures, display station and input/output control, and system identifiers for program readability. AML also provides additional basic resources, such as motion, sensing, and communications, needed to accomplish a robot task. AML supplies a rich set of program control capabilities so that the programmer has the necessary tools to gain the desired results. Data processing system subroutines are provided to transform the data used with the robotic functions. Program development and debugging facilities are supplied to ease the programming effort [3]. AML is used on the the IBM Series/1 computer for the IBM 7565 Cartesian robot.

AML/E is used on the IBM PC for the IBM 7535 and 7545 robots. AML/E is a subroutine-oriented language. A subroutine is a small unit of a program with a clearly defined beginning and ending. The statements that form a subroutine usually relate logically to achieve a result, such as moving the arm. Within the user subroutines are AML/E statements, which look like English-language commands, which send instructions to the robot controller and the manipulator [1 and 2]. AML/E is suitable for simple assembly operations, such as pick and place and palletizing operations. AML/E allows the use of up to 64 DI/DO points, thus enabling the use of switches, push-buttons, lights, etc. to control the movement of the manipulator by the user or the process itself.

AUTOPASS is a language proposed by IBM to meet task-level programming requirements. AUTOPASS, embedded in a subset of PL/1, is oriented towards object and assembly operations. The user of the AUTOPASS system plans the progress of the assembly operations as a sequence of high level assembly statements, each involving actions such as picking up a bolt or inserting a bolt into a hole. Compilation of AUTOPASS programs needs to be done on an IBM mainframe computer [4].

MCL, developed at McDonnell Douglas, is an extension of APT programming language for numerically controlled tools, and is general purpose in that it is robot independent. It provides statements for robot motion specification, vision system operation, image modeling, real-time conditional logic, and compile time language extensions. MCL was written for the programming of work cells in which a number of devices, including a robot, are under the control of one or more computers [10].

VAL, released by Unimation in 1979, was the the first commercially available robot programming language. VAL was developed by a Stanford alumnus quite familiar with AL, but constrained by the need to use an available minicomputer for implementation. VAL is an extension of the BASIC programming language, and runs in an interpretive mode [10]. VAL was designed as an on-line programming technique, but its language can be used for off-line program development. One of the major disadvantages is that VAL requires a dedicated computer, the LSI-11, to run on.

WAVE, developed in the early '70s at Stanford Artificial Intelligence Laboratory, was one of the first languages for manipulator control. WAVE was developed for research purposes, and was directed at finding the limitations of robotics theory rather than using robots to perform manufacturing tasks [10].

2.2.2 Graphical Programming Systems

GRASP, developed by Derby [8] at Rensselaer Polytechnic Institute, is a graphical off-line program which enables a designer to evaluate the performance of robots in a potential working environment. The designer can base his evaluation on time and motion studies of the performance. The resulting animation in simulated time provides a base for designing new manipulators, for modifying existing designs and for logical work cell layout.

GRPS and GRPPS, developed by Stephanic [14] and Curtis [5] at Michigan Technological University, are graphical techniques for off-line programming of robotic motion. The ASEA IRb-6 and the Rhino robots were modeled, and the programming system closely resembles the teach programming format available on the robots. Wire frame representations of the robot and work cell geometry were used, and visualization aids were available to help the user position the robot correctly. Robot motion can be graphically simulated, once the robot program has been entered.

PLACE, developed by the McAuto Division of McDonnell Douglas, facilitates work cell layout and comparative evaluation of robots and associated tooling. The system can be used to position cell components, dynamically define robot moves, check robot reach limits, build motion sequences, simulate tracking of moving objects and equipment, and analyze cycle times. There are three other modules offered by the company, namely, BUILD for modeling different robot configurations, COMMAND for creating and debugging programs off-line, and ADJUST for off-line work cell calibration [13].

2.3 INTERACTIVE PROGRAMMING

There are very few commercially available interactive programming systems, and the literature available in this area is also very limited. VAL, which is more of an off-line programming language, comes close to being an interactive programming language. This is due to the fact that VAL uses a syntax similar to BASIC, and the statements are interpreted, and not compiled. One major difference between VAL and AML/E is that VAL uses an LSI-11, which is also the controller for the robot, whereas AML/E runs on an IBM PC and the programs are downloaded to the robot controller.

AR-Basic, developed by American Robot Corporation for their Merlin series robots, is one of the commercially available interactive programming systems. This system has five components - core BASIC language, position definition and motion control, device I/O support, text file editing, and file and memory system control. This system's program development tools

are integrated into one system that has a common syntax and style for all its commands. The combination of an interpretive mode of execution, extensive information display and debugging facilities, and a color text editor permits users to exercise control over all stages of the program development process [9].

3.0 CHAPTER 3. METHODOLOGY

This chapter describes the approach used in the development of the interactive programming systems, and provides a reference to the use of the system. The system consists of the main menu, the pseudo-compiler and the system control programs. It also has access to the AML/E editor and compiler to enable the user to edit and compile programs in the environment provided by the AML/E software.

3.1 APPROACH USED IN SYSTEM DEVELOPMENT

The approach used in the development of this system was to make use of batch files, which in turn, calls the various .EXE files. The primary batch file is called INTER.BAT (refer to Figure 1 which summarizes the names and purpose of each file) which is called by typing INTER at the DOS prompt. This batch file calls the main menu program, IRPS1.EXE, which then displays the main menu to the user. The main menu is shown in Figure 2. The main menu program is given in Appendix A.

Function key F2 accesses the AML/E editor, the F3 key accesses the AML/E compiler, the F4 key calls the pseudo-compiler and then the system control program, the F5 key allows the user to set the program name and compiler options, and the F1 key terminates the use of the interactive programming system and return to DOS. If any key, other than these functions are

File name	Purpose
IRPS0	File created with information concerning program options selected by user
IRPS00	File used to pass program name between pseudo-compiler and system control program
INTER.BAT	Primary batch file for interactive execution
IRPS.TMP	File created when user opts to return to DOS
IRPS1.EXE	Main menu program
IRPS2.BAT	File containing commands based on the user's option and control returns to INTER.BAT when this terminates

Figure 1. Files used in the interactive robot programming system.

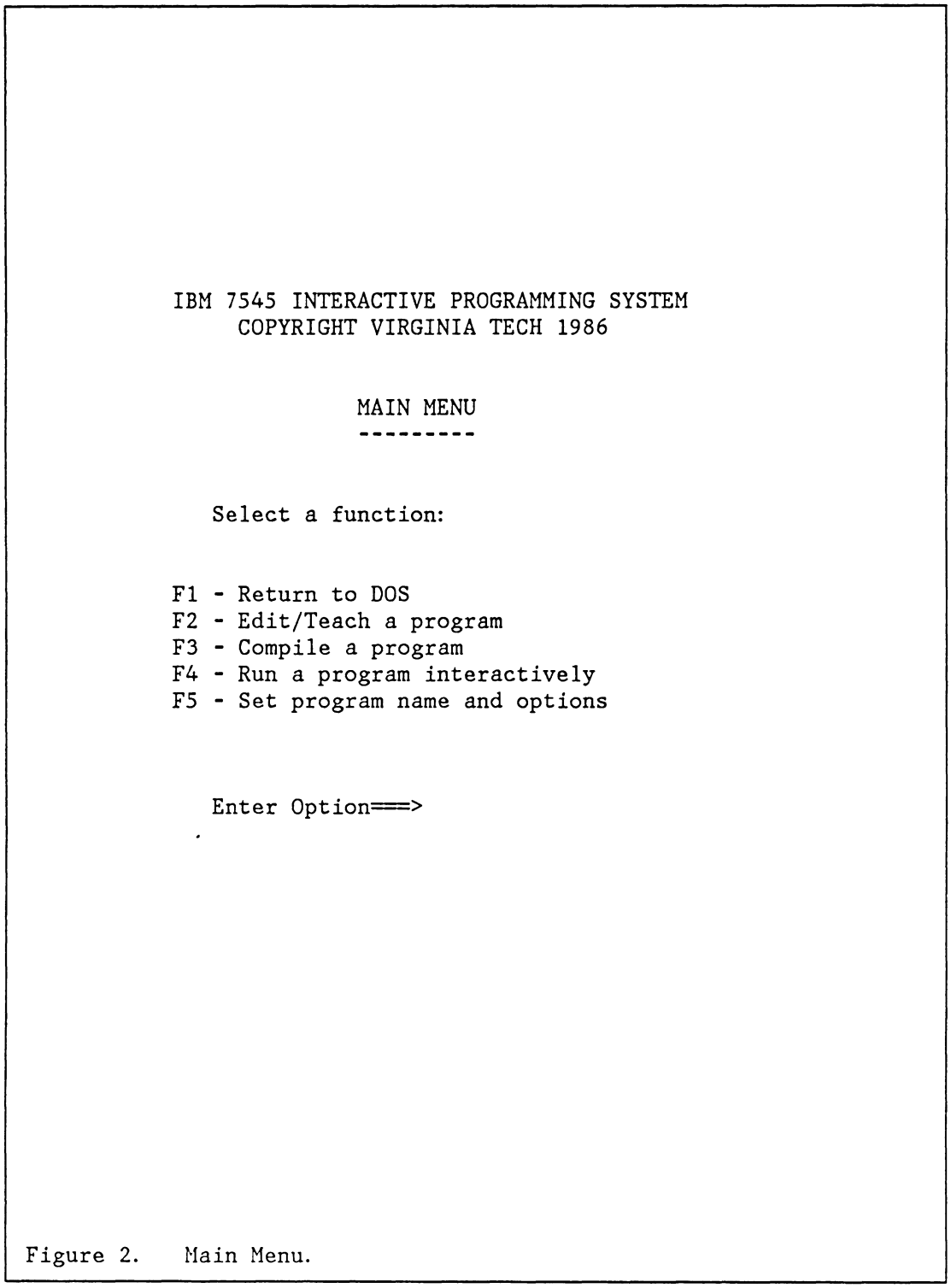

Figure 2. Main Menu.

pressed, the message 'KEY NOT DEFINED -- Hit any key to resume' appears on the bottom left corner of the display.

If the user strikes the F5 key, the program options menu, Figure 3, is displayed. The user can then enter the name of the file, and the file extension is assumed to be .AML. The user can also specify the compiler options. The valid compiler options are /L, for a listing, and /S, for the symbol file. Either one, both or none of the compiler options can be specified. The menu program reads all this information and writes it into a file called IRPS0. This data is retained until the user specifies a different option, or the user returns to DOS. The file IRPS0 is modified if the user specifies a different option, or is deleted if the user returns to DOS.

If the user strikes the F2 key, the AML/E editor is invoked. If the program name has been set, the program reads the file IRPS0 and specifies the file name to be edited. This call to the editor is written into a batch file called IRPS2.BAT. The next statement written to this batch file is INTER, so that when the edit session is complete, the main menu (or the program options menu) is displayed again.

If the user strikes the F3 key, the AML/E compiler is invoked. A call for the execution of the compiler is similar to the call to the editor, in that the batch file IRPS2.BAT is created, which calls the compiler and returns control to the main menu.

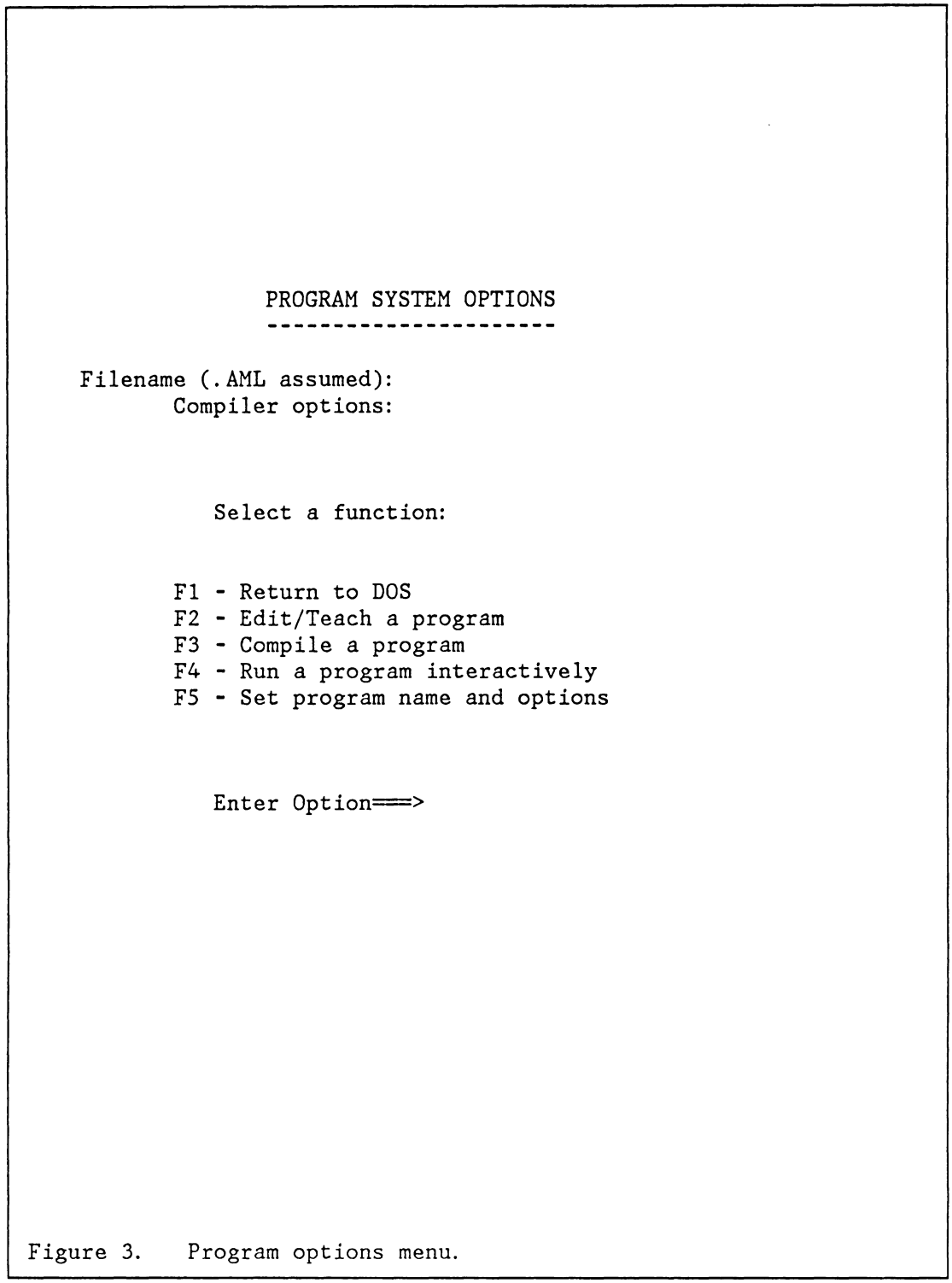

Figure 3. Program options menu.

If the user strikes the F4 key, first the pseudo-compiler is called and then the system control program. If the program name has not been set by the user, the user is prompted for the name. This name is written into a file called IRPS00, which will be used by the system control program. The pseudo-compiler generates five files with extensions .OBJ, .OUT, .VAR, .CN1, and .TAB. These files will then be used by the system program to interactively execute the user's AML/E program. The calls to these two programs are written into the batch file IRPS2.BAT along with the call to the first batch file, INTER, to return control to the main menu.

If the user strikes the F1 key, a temporary file, IRPS.TMP, is created and the program terminates. When control returns to the batch file INTER.BAT, it checks for the existence of the temporary file, and if it does exist, control returns to DOS. Before the batch file terminates, all files created by the programs are deleted.

3.2 INTERACTIVE EXECUTION SPECIFICATIONS

When the user specifies interactive program execution, several messages and menus are displayed for the user. The first message that appears is a warning that the robot is about to go home, and if needed the user may retract the manipulator in the Z direction to avoid colliding with any obstacles. It also tells the user that the manipulator power must be on and the robot controller must be on-line. Once the robot reaches home, the system is ready for interactive program execution.

The first option that is available is whether the user wants to execute the program continuously, or single-step through the program with the user striking the 'Enter' key after every statement is executed. The second option that is available for the user is whether the robot arm should move at a normal speed, or at a slow speed. If the user responds that the robot arm should move slowly, the next prompt requests entry of a payload value to be used for all subsequent motions. The third option is the mode of execution, and the execution options menu, illustrated in Figure 4, is displayed.

If the user strikes the F1 key, the entire program will be executed. If the user strikes the F2 key, the user is prompted for the line numbers at which program execution should start and end. If the user strikes the F3 key, the user is prompted for the subroutine name and the values of formal parameters, if any. The formal parameters may be passed as either variable names or numeric values. If the user strikes the F4 key, execution is aborted, and the main menu is displayed.

If the user selects either partial program execution or subroutine execution, the system will display the menu, illustrated in Figure 5, for setting up the initial conditions prior to interactive execution.

With these setup options, the user may set pallet and counter parameters, the various motion conditions (LINEAR, PAYLOAD and ZONE values), specify values of DO points, and move the robot arm to any desired location. It is advisable to move the Z axis down and set the LINEAR value at the very

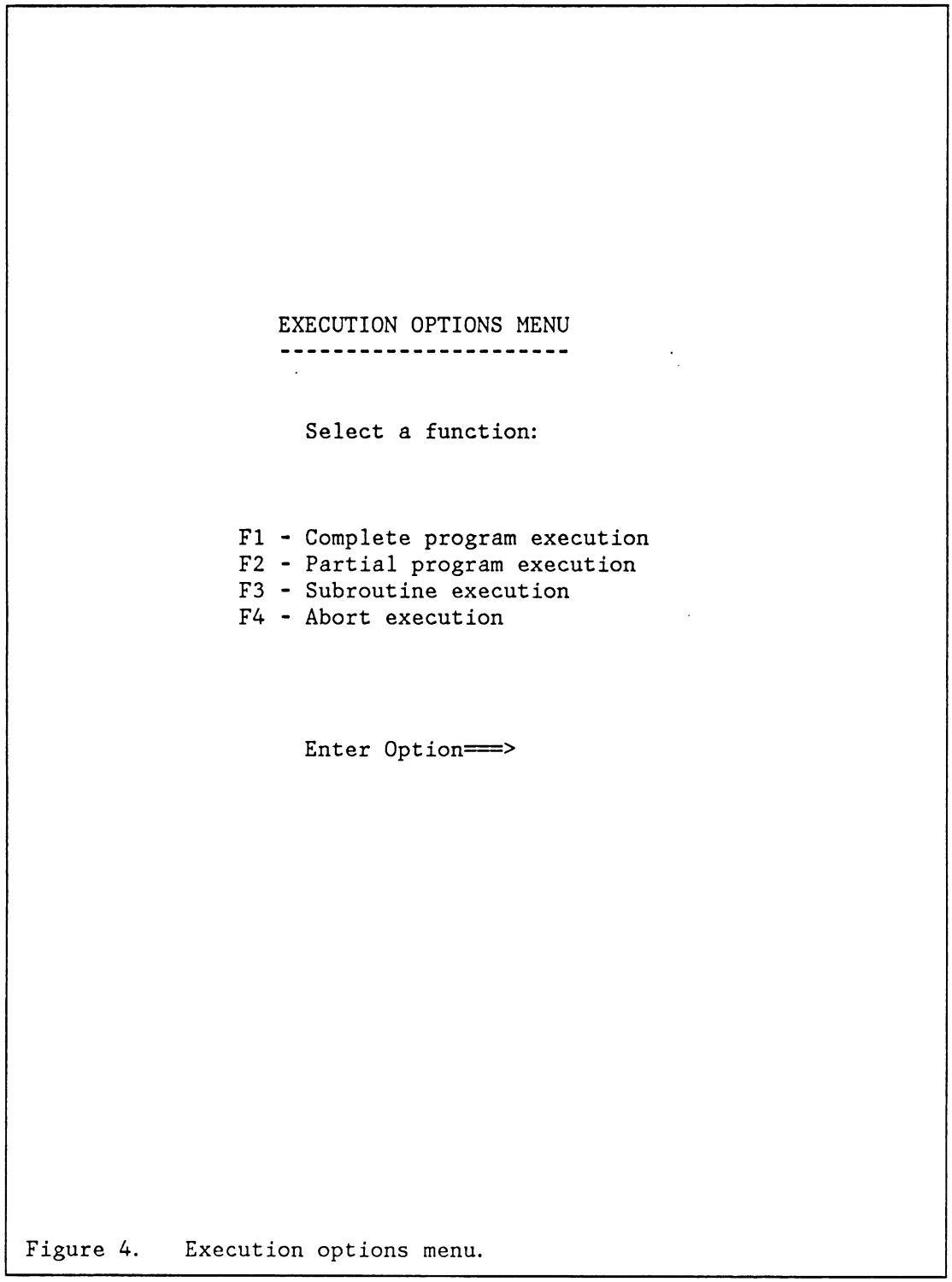

Figure 4. Execution options menu.

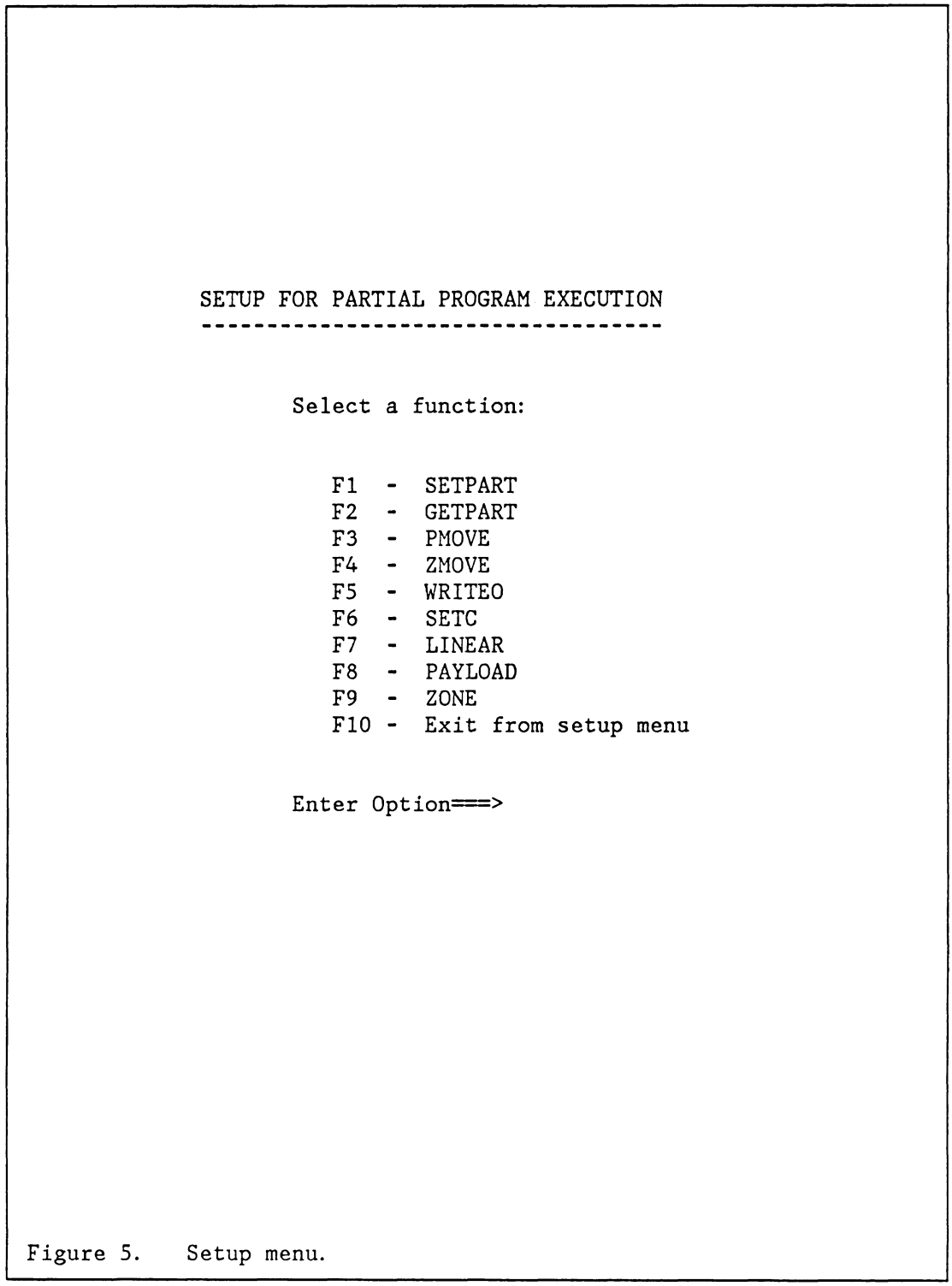

Figure 5. Setup menu.

end, to avoid colliding with obstacles and to ensure that the robot arm is not required to move in a linear fashion in the non-linear region. Once the initial conditions are set, the user exits from this menu and the program execution begins, with the options specified by the user.

During program execution, the user can resume normal speed operation, if slow speed execution is in effect. Also, if single-stepping is in effect, the user can revert to continuous execution. If the user desires, program execution can be aborted at any time. These options can be selected by striking the appropriate function keys which are displayed on the screen. Also displayed on the screen are the current line number being executed and the line number just executed being displayed above that. The name of the file being executed is also displayed, and if any errors are found in the program, program execution is terminated with an error message appearing at the bottom left corner of the display. Once the program execution is completed or terminated, the main menu is displayed.

4.0 CHAPTER 4. DEVELOPMENT OF THE INTERACTIVE PROGRAMMING SYSTEM

This chapter describes the development of each stage of the interactive robot programming system, as summarized in the introductory chapter. The logic associated with all programs in each stage are described, and the purpose and operation of all subroutines are also presented.

4.1 DESCRIPTION OF FILES

The pseudo-compiler basically reads the user's AML/E program (assuming a .AML file extension), and it generates five files, namely, a listing file (.OUT), a variables file (.VAR), a constants file (.CN1), an object file (.OBJ), and a symbol table file (.TAB). The object file and the symbol table file are opened as direct access files, and the others as sequential files. These files are then used by the system control program to interactively execute the user's AML/E program. A description and a sample output of each of the files is given, to better understand the operation of the pseudo-compiler and the system control program.

The listing file, shown in Figure 6, is the same as the user's AML/E program, except that the line number of each line is added in the first four columns of the file. When the program is being executed, this file is read, and the line being executed is displayed on the screen.

```
 1    --  THIS IS A TEST PROGRAM WHICH USES SOME OF THE BASIC
 2    --  MOTION COMMANDS FOR THE PURPOSES OF GENERATING THE
 3    --  PSEUDO COMPILER. THIS PROGRAM HAS BEEN COMPILED
 4    --  USING THE AML/E COMPILER AND VERIFIED USING THE
 5    --  THE AML/E SIMULATOR.
 6
 7    L1 : NEW  PT(-396,5,-200,10);L2:NEW PT(-441,82.9,-200,10);
 8    DP1: NEW <-10,-10,-10.0,-10>;
 9    PAY : NEW 5; LIN : NEW 0 ;
10
11
12    TEST: SUBR;
13          DROP:SUBR;
14             PMOVE(L1);GRASP;PMOVE(L2);RELEASE;
15          END;
16
17       PAYLOAD(PAY);
18       LINEAR ( LIN );
19       PMOVE ( L1); LINEAR (5); PAYLOAD(6);
20       DPMOVE ( DP1);
21       PMOVE(PT(650,0,0,0));
22       DELAY (5.0);
23       PMOVE(L2); LINEAR(0); PAYLOAD(10);
24       DROP; DELAY (PAY);DPMOVE(<-10,-10,-20,-10);
25    END;
```

Figure 6. Sample listing file (.OUT).

NUM	LINENO	STNO	LENGTH	ITYPE	VARNO	Name
1	7	0	2	3	0	L1
2	7	0	2	3	0	L2
3	8	0	3	8	0	DP1
4	9	0	3	2	0	PAY
5	9	0	3	2	0	LIN
6	12	1	4	6	1	TEST
7	13	2	4	6	2	DROP

Figure 7. Sample variables file (.VAR).

NUM1	LINENO	STNO	LENCON	Value
1	7	0	5	-396.
2	7	0	2	5.
3	7	0	5	-200.
4	7	0	3	10.
5	7	0	5	-441.
6	7	0	4	82.9
7	7	0	5	-200.
8	7	0	3	10.
9	8	0	4	-10.
10	8	0	4	-10.
11	8	0	5	-10.0
12	8	0	4	-10.
13	9	0	2	5.
14	9	0	2	0.
15	19	11	2	5.
16	19	12	2	6.
17	21	14	4	650.
18	21	14	2	0.
19	21	14	2	0.
20	21	14	2	0.
21	22	15	3	5.0
22	23	17	2	0.
23	23	18	3	10.
24	24	21	4	-10.
25	24	21	4	-10.
26	24	21	4	-20.
27	24	21	4	-10.

Figure 8. Sample constants file (.CN1).

STNO	LINENO	NTYPE	MTYPE	Par1	Par2
1	12	6	1	22	8
2	13	6	1	7	3
3	14	1	4	1	1
4	14	1	3		
5	14	1	4	1	2
6	14	1	5		
7	15	6	2	2	
8	17	1	8	1	4
9	18	1	7	1	5
10	19	1	4	1	1
11	19	1	7	2	15
12	19	1	8	2	16
13	20	1	2	1	3
14	21	1	4	2	17
15	22	1	1	2	21
16	23	1	4	1	2
17	23	1	7	2	22
18	23	1	8	2	23
19	24	7	2		
20	24	1	1	1	4
21	24	1	2	2	24
22	25	6	2	1	

Figure 9. Sample object file (.OBJ).

NUM	Value1	Value2	Value3	Value4
1	-396.00	5.00	-200.00	10.00
2	-441.00	82.90	-200.00	10.00
3	-10.00	-10.00	-10.00	-10.00
4	5.00			
5	0.00			
6				
7				

Figure 10. Sample symbol table file (.TAB).

The variables file, shown in Figure 7, has seven columns. The first column, represented by the variable NUM, is the number of the variable. The second column specifies the line number, LINENO, at which the variable was found. The third column specifies the statement number, STNO, at which the variable was found. The fourth column specifies the length of the variable name, LENGTH. The fifth defines the type of the variable, ITYPE. The sixth column specifies the subroutine number with which the variable is associated, VARNO. If the variable is a global one, VARNO is zero, otherwise, the variable is associated with a subroutine only. The last column has the name of the variable itself.

The structure of the constants file, shown in Figure 8, is similar to that of the variables file. The first column specifies the number of the constant, NUM1. The second denotes the line number at which the constant was found, LINENO. The third gives the statement number at which the constant was found, STNO. The fourth specifies the length of the constant, LENCON. The fifth column contains the value of the constant.

The object file, shown in Figure 9, has a minimum of four columns. The first column contains the statement number, STNO. The second contains the line number of the statement, LINENO. The third and fourth columns contain the type of command, NTYPE, and the sub-type in that command category, MTYPE, respectively. There may be several other columns of data, depending on the command itself. These additional columns could contain the command parameters, variable numbers of the formal parameters of subroutines, etc.

The symbol table file, shown in Figure 10, has the same number of records as the variables file. Except in the case of pallet definitions, and in the case of a point defined as a set of four counters, this file would either have zero, one or four columns of data. The data could represent counter values, the four coordinates of a point, statement numbers of labels, etc.

4.2 PSEUDO-COMPILER

The file name of the pseudo-compiler on the IBM PC/AT is COMPPC. This section gives an overview of the logic associated with the pseudo-compiler, as shown in Figure 11, and a description of all subroutines in the program. The listing of the program is given in Appendix B.

The main program first reads the name the of the user's AML/E program, from the file IRPS0, which was created by the program options menu. If the user had not made use of this option, the program prompts the user for the AML/E program name. The program assumes a .AML file extension. This name is written into the file IRPS00, which will be used by the system control program for interactive execution. The main program opens all the files. If there is an error in opening the AML/E program file, an error message is issued, and the user is again prompted for the name of the AML/E program file. The main program then calls subroutine PROG.

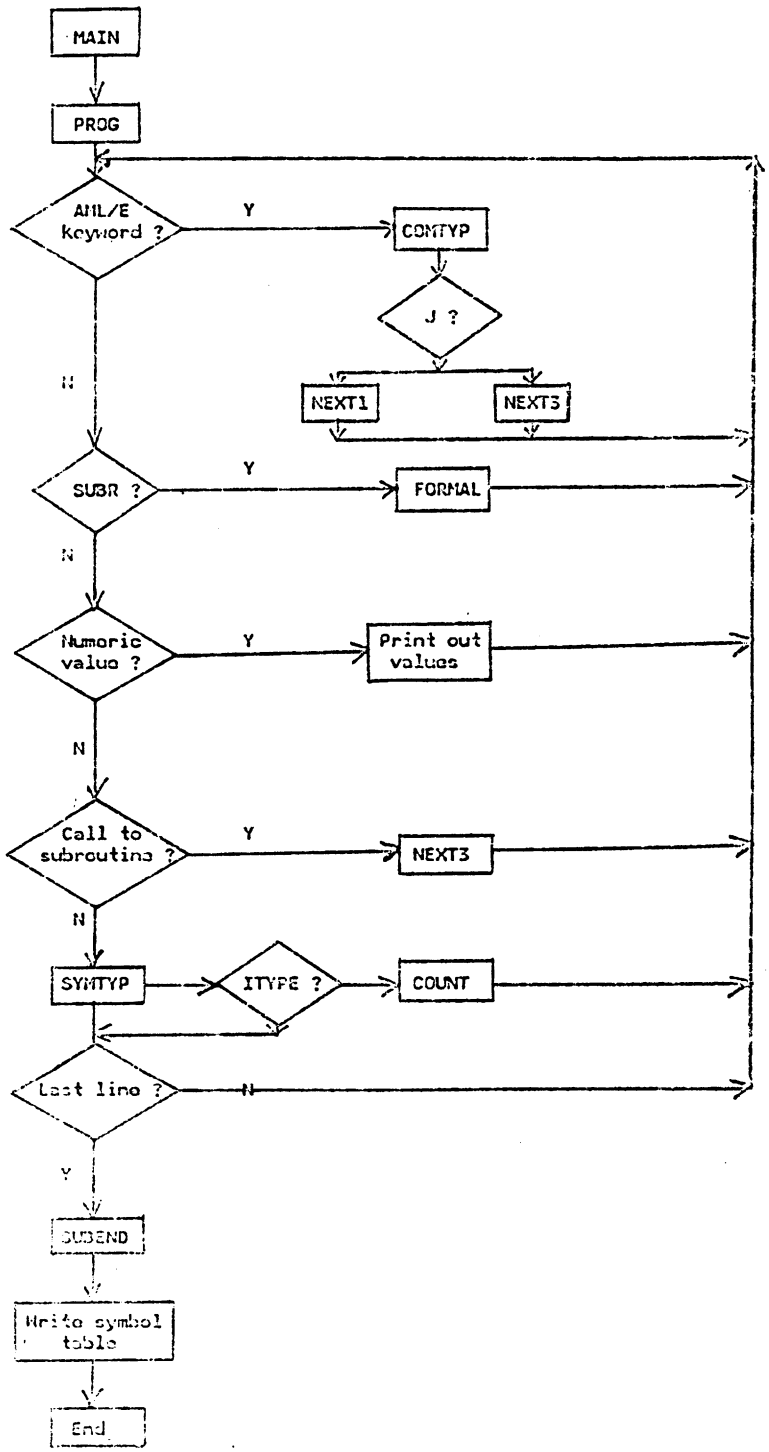

Figure 11. Overview of the pseudo-compiler.

Subroutine PROG initializes all program variables, and reads the user's AML/E program, one line at a time. As every line is read, the program writes the line number, LINENO, and the statement into the listing file, .OUT. The source line is read into a character array, C, and the program then reads this array to check whether it is a blank line or a comment line. If it is not one of the two, this line contains AML/E code, and the program begins to break down the line into AML/E reserved words, variable names and constant values. This is done by reading the characters in the array C, and concatenating it to the previous character till a delimiter is found. Once a delimiter is found, it signifies the end of the symbol, and the program checks for the type of the symbol. The symbol is first checked against the list of AML/E keywords, and if the symbol is 'SUBR', it also checks for formal parameters. If the symbol is one of the first 34 keywords, which are executable commands, the program calls subroutine COMTYP, after incrementing the statement number. If the symbol is not an AML/E keyword, the symbol is checked to see if it is the beginning of a comment. If not, the symbol is checked to see if it is a numeric constant, and if it is, the program then prints out the constant number, the line number, and the value of the constant into the constants file. Also, if the number is an integer, it is converted into a real number by adding a decimal point.

If the symbol does not fall into one of these categories, the program checks to see if the symbol has already been identified. If the symbol is a call to a subroutine, the statement number is incremented, and subroutine NEXT3 is called. If the symbol has not been previously identi-

fied, subroutine SYMTYP is called to determine the type of the symbol. If the symbol is a label, the statement number is incremented and a value of zero is written into the object file for both the variables NTYPE and MTYPE. If the symbol is the definition of a subroutine name, the index for subroutines, NSUB, and the statement number are incremented. NTYPE is assigned a value of six, and MTYPE is assigned a value of one, and these values are written to the object file. If the symbol is the name of a pallet, or if it is the name of a point defined by a set of four counters, it is written into the variables file by the subroutine COUNT. Otherwise, the symbol is written into the variables file by PROG. The subroutine in which this symbol occurs is also written to this file. The symbol just identified is also checked against all the unresolved symbols, and if they match with any of them, subroutine LABELS is called.

Once the type of the symbol is identified, the program goes on to identify other symbols in the same line. If the end of the line is reached, the program increments the line number, and reads the next line in the user's AML/E program. Once all the lines in the user's program are read, subroutine SUBEND is called. Now the program is ready to generate the symbol table file. For every symbol in the variables file, a corresponding line is reserved in the symbol table file. The program reads the variables file, one line at a time. If the variable represents a constant, the constants file is read till the appropriate value is found. Once the value is found, it is written into the symbol table file, at the same record number of the symbol in the variables file. If the symbol represents a point or an aggregate, the constants file is searched, and the

appropriate four values are read in succession, and written to one record in the symbol table file. If the symbol defines a pallet, or a point defined by a set of four counters, nothing is written to the symbol table file, as this has already been done in the subroutine COUNT. If the symbol represents a label, or a subroutine name, or a counter, or a formal parameter, no value for this written into the symbol table file. This process is carried out till all the symbols in the variables file are read.

Subroutine COMTYP is called to classify the type of the AML/E command. The AML/E commands are broken into command categories, identified by the variable NTYPE, and a sub-type within each category, identified by the variable MTYPE. The list of commands, and their corresponding values of NTYPE and MTYPE are shown in Figure 12. The values for NTYPE and MTYPE for all labels are zero. When a subroutine is called in the user's AML/E program, NTYPE is set to 7 and the value of MTYPE is set equal to the statement number at which the subroutine is declared. If there are any command parameters for the AML/E command, the program calls subroutine NEXT1 or NEXT3, depending on the command. If no command parameter exists, the values of NTYPE and MTYPE are written to the object file, along with the line number and statement number of the AML/E instruction. The only instruction where the program does additional checking, is the END command. This command signifies the end of a subroutine, and the program then checks for the statement number at which the subroutine was declared, and this is also written to the object file along with the values of NTYPE, MTYPE, line number and statement number of the END instruction.

Also, the current subroutine number is modified, to take into account the end of this subroutine.

Subroutine NEXT1 is called if the command PMOVE or DPMOVE is encountered in the user's AML/E program. This subroutine checks to see if the command parameter is defined as a variable name or as a set of numbers. The program first checks to see if the first non-blank character after the AML/E command is the character '<'. If it is, the DPMOVE parameter is specified as a set of four constants. If the first two non-blank characters after the AML/E command are 'PT', the PMOVE parameter is specified as a set of four values. If one of these two conditions are met, KTYPE is set to two, and the four values are read one at a time, as described in the subroutine PROG. Each value is written into the constants file, and the values of the statement number, line number, NTYPE, MTYPE, KTYPE and the constant number of the first constant in the set are written to the object file. If the command parameters are not specified by four values, the value of KTYPE is set to one, and the program then reads the name of the variable. This variable is then written into the variables file, and the values of the statement number, line number, NTYPE, MTYPE, KTYPE and the variable number is written into the object file. If the command parameter is specified neither as a set of four constants nor as a variable, they must be specified as a set of four counters or variable names. The program in this case, assigns a value of three to KTYPE, and the four variable names are read one at a time. Each variable name is compared with those names already identified, and the variable number for each is determined. The program then writes the values of the statement

AML/E Command	NTYPE	MTYPE	Category of command
DELAY	1	1	Motion
DPMOVE	1	2	"
GRASP	1	3	"
PMOVE	1	4	"
RELEASE	1	5	"
ZMOVE	1	6	"
LINEAR	1	7	"
PAYLOAD	1	8	"
ZONE	1	9	"
WAITI	2	1	Sensor
WRITEO	2	2	"
BRANCH	3	1	Flow of control
TESTC	3	2	"
TESTI	3	3	"
TESTP	3	4	"
BREAKPOINT	3	5	"
COMPC	4	1	Counter
DECR	4	2	"
INCR	4	3	"
SETC	4	4	"
GETPART	5	1	Pallet
NEXTPART	5	2	"
PREVPART	5	3	"
SETPART	5	4	"
SUBR	6	1	Subroutine
END	6	2	"
WHERE	8	1	-

Figure 12. AML/E command categories.

number, line number, NTYPE, MTYPE, KTYPE and all the four variable numbers into the object file.

Subroutine NEXT3 is called to read the command parameters for the following commands: BRANCH, COMPC, DECR, DELAY, GETPART, INCR, LINEAR, NEXTPART, PAYLOAD, PREVPART, SETC, SETPART, TESTC, TESTI, TESTP, WAITI, WRITEO, ZMOVE, and ZONE. The subroutine is also used to read the formal parameters passed when a subroutine is invoked. The program was designed to handle a maximum of five formal parameters. If any of the labels associated with these commands have not yet been identified, they are categorized as unresolved labels, and will be handled when the labels are read later. The procedure followed in reading the parameters are similar to that described in the previous chapter. Each command parameter has two associated values. The first value is set to one, if the parameter is a variable name, and is set to two if the parameter is a numeric value. The second value specifies either the variable number or the constant number. Any numeric value found is written to the constants file. In commands involving labels, there is only one value associated with them, and that is the statement number where the label is found. These commands include TESTC, TESTI, TESTP, COMPC, and WAITI. In the case of the command COMPC, the name of the first counter has only one associated value with it, which is the variable number of the counter name. The condition, for example, >=, also has one value associated with it. The condition and its associated value in ascending order from a value of one are as follows: <, <=, >, >=, =, and <>. In this manner, all the command parameters

are categorized, and these values along with the statement number, line number, NTYPE and MTYPE are written to the object file.

Subroutine SYMTYP is called to identify the type of symbol which has been concatenated. The symbol must always be followed by a semicolon. If not, it is classified as an undefined symbol, and ITYPE is assigned a value of zero. After the semicolon, if there are no more characters on the line, the symbol is classified as a label, and ITYPE is assigned a value of one. If the first three non-blank characters after the semicolon are 'NEW', the symbol is either a point, aggregate or a constant definition. If the next two non-blank characters are 'PT', the symbol is a point definition. The successive characters are then checked to see if the point is defined as a set of numbers, or as a set of counter names. If they are a set of numbers, ITYPE is assigned a value of 3, otherwise ITYPE is assigned a value of 9 and subroutine COUNT is called. If the symbol does not represent a point, and if the first non-blank character after 'NEW' is the character '<', then the symbol is the name of an aggregate, and ITYPE is assigned a value of 8. If the symbol is not an aggregate, it must be the name of a constant, and ITYPE is assigned a value of 2. If the first six non-blank characters after the semicolon are 'STATIC', the symbol either represents the name of a counter or a pallet. If the symbol represents a counter, ITYPE is assigned a value of 4. If the symbol represents a pallet, ITYPE is assigned a value of 6, and subroutine COUNT is called. If the first four non-blank characters after the semicolon are 'SUBR', the symbol is the name of a subroutine, and ITYPE is assigned a value of 6.

Subroutine COUNT is called to read the four parameters in the definition of a point, when the point is defined as a set of four counters. This subroutine also reads the three points, assumed to be variable names, and the two constants, which may be numeric or symbolic names, which define a pallet. These symbols or numbers are then written either to the variables file or the constants file. The way that this is done is similar to the one described in subroutine PROG, where each symbol is identified separately. The program also writes the number of the symbol or constant into the record location which defines the point or pallet.

Subroutine FORMAL is called to check for formal parameters in a subroutine declaration. If there are no formal parameters, ITYPE is set to zero, and control returns to subroutine PROG. If any formal parameters exist, they are broken down into symbols as described earlier in subroutine PROG, and each symbol is written into the variables file. The subroutine number in which the formal parameter occurs and the value of ITYPE, which is set to 7, are also written to the variables file.

Subroutine LABELS is called when an unresolved label has been identified as a symbol. This subroutine then completes statements in the object file which has this label as a command parameter. Commands which could have unresolved labels include BRANCH, COMPC, TESTC, TESTI, TESTP, and WAITI. The program reads the object file at the statement number which has the unresolved label, and depending on the command, this line is rewritten to include the statement number of the label. Once this is done, control returns to subroutine PROG.

Subroutine SUBEND is called to determine the first executable statement in each subroutine and to pair each 'SUBR' and 'END' statement. This is done by reading the last statement in the object file for the statement number of the first SUBR, and then reading the object file backwards till the previous END statement is encountered. The first executable statement for this SUBR is the statement after the previous END. The statement number of the END statement, and the statement number of the first executable statement are written into the location in the object file where the subroutine is declared. This process is carried on till all subroutines are completed, except the last subroutine and any non-embedded subroutines. For the last subroutine and any non-embedded subroutines, the first executable statement of the subroutine is the first statement after the subroutine declaration. Once all pairs of SUBR and END statements are identified, control returns to subroutine PROG.

The BLOCK DATA simply defines all the valid characters in any AML/E program, and also all the reserved AML/E words. These are all defined in the following three arrays: DELIM, CHARS, and AMLWRD.

There are three utility routines in the pseudo-compiler. The first is subroutine TTOUT, which is used to write a string of characters to the display. The user can select the attributes of the characters to be displayed, and also the location on the screen. It calls two ASSEMBLY utility routines, LOCATE and WRCHAT.

The second utility subroutine is ID, which is called whenever the variable names or numbers needs to be read from the display. The characters that can be read are alphabets, numbers, the minus sign, and the decimal point. All other keyboard characters are not recognized, and if they are struck, an error message is issued.

Subroutine INCREM, the third utility routine, was written to increment a variable by a constant value. This was done to overcome an internal compiler error in the Professional FORTRAN compiler.

4.3 SYSTEM CONTROL PROGRAM

The system control program, called SYS1 on the IBM PC/AT, reads the five files generated by the pseudo-compiler, and transmits information to the controller based on the commands in the user's AML/E program. This section gives an overview of the logic associated with the system control program, shown in Figure 13, and a description of all subroutines in the program. The listing of the program is given in Appendix C.

The main program opens the files generated by the pseudo-compiler, after it reads the name of the user's program from the file IRPS00, which is created by the pseudo-compiler. The constants file, .CN1, is a sequential file, and this is converted into a direct access file, .CON. The object file, the constants file, the symbol table file and the listing file are all opened as direct access files. This program then calls the subroutine

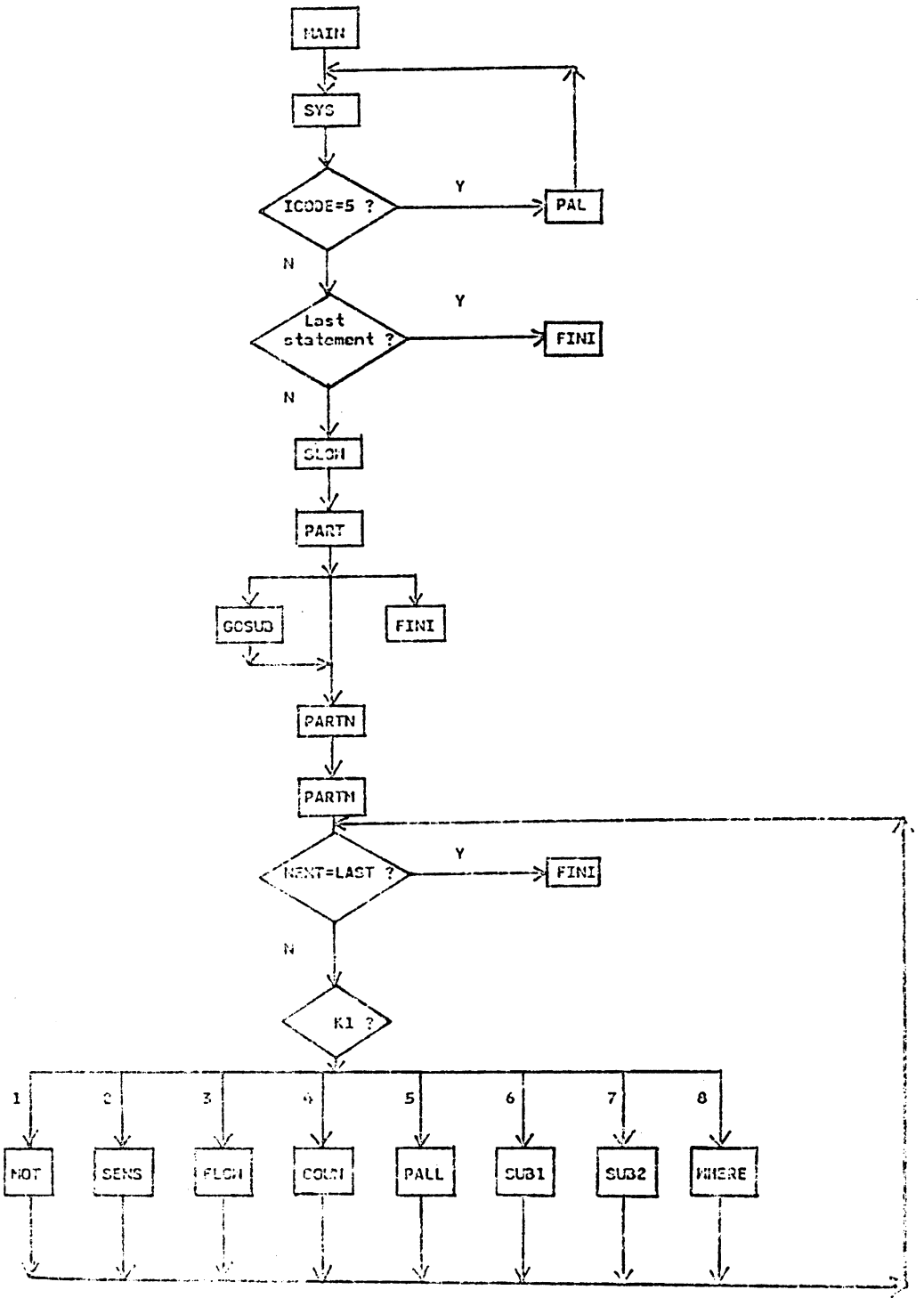

Figure 13. Overview of the system control program.

SYS, and passes the name of the user's program and the length of the name. Once the user's program has been executed, control returns to the main program, which calls subroutine FINI.

Subroutine SYS is called after all the files are opened by the main program. All variables are initialized, and the variables file is read one line at a time. ICODE is the variable which specifies the type of the variable in the file. This variable is equivalent to the variable ITYPE in the pseudo-compiler. For labels, ICODE has a value of one, and the statement number at which the label is found (IC1) is written into the symbol table file. ICODE is equal to 2 for constants, and the value is read from the symbol table, and is assigned to the array CONVAL, and the index NUM1 is incremented. ICODE is equal to 3 for points, and the symbol table is read for the X, Y, Z and R values and are assigned to the arrays XPNT, YPNT, ZPNT and RPNT, and the index NUM2 is incremented. ICODE is equal to 4 for counters. The counters are initialized with a value of zero, and this is written to the symbol table file. If ICODE is equal to 5, the variable is a pallet name, and subroutine PAL is called. ICODE is equal to 6 for subroutine names. The program then checks to see if there are any formal parameters, and if there are any, they are all initialized to zero, and the index NUM5 is incremented. Also, the variable numbers of the formal parameters are written into the object file at the line where the subroutine is defined. ICODE is equal to 8 for aggregates, and the values are read from the symbol table file, and assigned to the arrays XAGG, YAGG, ZAGG and RAGG, and the index NUM4 is incremented. This process is carried on till all the variables have been read.

Once all the variables have been read, the robot is moved to the home position, application five is selected, the controller is put in the auto execution mode, and the start cycle command is issued. Default values for PAYLOAD, LINEAR and ZONE commands are transmitted. The prompt for single-stepping is then displayed, and if this option is selected, the variable, SS, is set to 1. Subroutine SLOW is then called. The first line in the object file is read to determine the first and last executable statements in the user's AML/E program. Subroutine PART is called to check if the user desires partial program execution or subroutine execution. The system is now ready for interactive execution, and it displays the name of the user's program. If slow robot movement is selected, the slow payload value, SLPAY, is transmitted to the controller. The current statement to be executed is denoted by the variable NEXT, and the last statement that needs to be executed is denoted by LAST. The current statement to be executed is read from the object file, and the line from the user's program is read from the listing file. The line is then displayed, and the previously executed line is displayed above this. If the statement to be executed is the last one, program execution terminates.

Variables K1 and K2 which are read from the object file determine the type of command in the user's program. These two variables are equivalent to the variables NTYPE and MTYPE in the pseudo-compiler. K1 has a value of 0 for labels, and the program increments the statement number NEXT. K1 is equal to 1 for motion commands, and the sub-type is specified by K2, and this is then passed to the subroutine MOT as a parameter. K1 is equal to 2 for sensor commands, and K2 is again passed as a parameter to the

subroutine SENS. K1 is equal to 3 for flow of control commands, and subroutine FLOW is called. K1 is equal to 4 for commands associated with counters, and subroutine COUN is called. K1 is equal to 5 for commands associated with pallets, and subroutine PALL is called. K1 is equal to 6 for subroutine declarations or the END statement of a subroutine. The subroutine SUB1 is called with the parameter K2 being passed. K1 is equal to 7 when a subroutine is called in the user's program. Subroutine SUB2 is called, with K2 being passed as a parameter. K1 is equal to 8 when the WHERE command is encountered, and subroutine WHERE is called. The program loops through the object file till the last statement is executed, and then program execution terminates.

Subroutine PAL is called when the variable name in the variables file defines a pallet. This subroutine computes the coordinates of all the points in the pallet, and assigns them to the arrays XPAL, YPAL, ZPAL and RPAL. The current part number is set equal to one, and the index for pallets, NUM6, is incremented. This subroutine assumes that the corner points are defined as variables, and hence reads the coordinates of these points from the symbol table. The parts per row and the number of pallet points, can be either constants, in which case, they are read from the constants file, or they may be variables, in which case, they are read from the symbol table file.

Subroutine MOT handles the following commands and the value of K2 is shown in parenthesis: DELAY(1), DPMOVE(2), GRASP(3), PMOVE(4), RELEASE(5), ZMOVE(6), LINEAR(7), PAYLOAD(8), and ZONE(9). Depending on the type of

motion command, the program reads the the object file for any command parameters. Command parameters can be either constants, in which case they are read from the constants files, or the parameters can be variable names, in which case the arrays CONS, COUNT, PAR, AGG, and PNTVAL are checked to see if the parameter belongs to one of these categories of variables. Once the variable is identified, the value(s) associated with it is also known. Now the program is ready to transmit values to the controller, but prior to that subroutine CHECK is called to ensure that the previous command has been executed. The program also does a limited check to ensure that the command parameters are in the valid range. If the parameters are not valid, subroutine ERROR is called to display the error message, and program execution is aborted. If no error is detected, the values are transmitted to the controller, and the variable CODE is the last value to be transmitted. The controller then executes the command just transmitted. The statement number to be executed, NEXT, is incremented, and control return to subroutine SYS.

The commands DPMOVE and ZMOVE are treated as PMOVE commands, and the host computes the absolute coordinates and transmits them to the controller. Similarly, the commands GRASP AND RELEASE are treated as WRITEO commands, with the DO point being 2, and the value as either 0 or 1. In case the command is PAYLOAD, there is an additional test done to check if the specified payload value would cause the robot arm to move slower than the slow payload value, SLPAY. If it does, the new value is transmitted, and this value is also saved as PAYOLD, so that when the user opts out of slow execution, the most recently specified payload value is transmitted. If

the new payload value does not cause the robot arm to move slower, then the value of SLPAY is transmitted.

Subroutine SENS handles the following commands with the value of K2 shown in parenthesis: WAITI(1) and WRITEO(2). The mode of operation is similar to that described in the previous paragraph. The only difference in the case of the WAITI command, is that after the parameters are transmitted to the controller, there is an additional command to read the value of the RETURN_CODE from the controller to see if a timeout error was encountered.

Subroutine FLOW handles the following flow of control commands: BRANCH(1), TESTC(2), TESTI(3), TESTP(4), and BREAKPOINT(5). The operation of this subroutine is similar to that of MOT. However, these flow of control commands, usually have labels to which control is transferred. So, instead of simply incrementing the statement number to be executed next, the statement number of the label is read from the symbol table file, and is assigned to the variable NEXT. Also, when the command BREAKPOINT is encountered, program execution is suspended, and is resumed only after the user strikes a key. The only communication with the controller in this subroutine is for the TESTI command. The DI point is read using the ASSEMBLY routine RDIDO, and the check for the status of the DI point is done by the host itself.

Subroutine COUN handles the following commands associated with counters: COMPC(1), DECR(2), INCR(3), and SETC(4). All these commands are handled

by the host itself, and there is no communication with the robot controller.

Subroutine PALL handles the following pallet commands: GETPART(1), NEXTPART(2), PREVPART(3), and SETPART(4). The current part number for the pallet is stored in the array PARTNO. Based on the part number, the coordinates of this point are determined from the arrays XPAL, YPAL, ZPAL and RPAL. For the GETPART command, these coordinates are transmitted to the controller, and is treated as a PMOVE command. The other three pallet commands are internal to the host itself, and the variable PARTNO is set accordingly.

Subroutine SUB1 is called whenever a subroutine declaration or the END statement is encountered. K2 is equal to 1 for a subroutine declaration, and the program does nothing in this case. K2 is equal to 2 for the END statement, and in this case, the record of the END statement is read from the object file. This statement has the statement number of the next statement to be executed, which is the statement after the one that invoked the subroutine.

Subroutine SUB2 is called whenever the user's AML/E program invokes a subroutine. This subroutine is designed to handle a maximum of 5 formal parameters. The variable K2 contains the value of the statement number of the subroutine declaration. The program then reads the object file at this statement number and obtains the formal parameter names, if any. The program then writes the statement number of the statement after the

one that invoked the subroutine, into the object file at the line which contains the END statement of this subroutine. This is done so that control returns to the correct statement after the subroutine has ended. If any formal parameters exist, the values passed to the subroutine are read, and are written into the location of the symbol table reserved by the formal parameters. The variable NEXT is assigned to be the first executable statement in the subroutine, and control returns to subroutine SYS.

Subroutine WHERE handles the AML/E command WHERE. The current coordinates of the robot arm, denoted by XVAL, YVAL, ZVAL and RVAL are written into the location of the variable, specified as a command parameter, in the symbol table file. The variable NEXT is incremented, and control returns to subroutine SYS.

Subroutine PART asks the users if they want to execute the the complete program, execute a subroutine or execute only part of the program. This subroutine displays the Execution Options Menu, shown in the previous chapter. If the user opts for complete program execution, control return to subroutine SYS. If the user opts for partial program execution, the program then prompts the user for the line numbers at which the program should begin and end execution. These line numbers are then assigned to variables PART1 and PART2. If the user opts for subroutine execution, subroutine GOSUB is called. The program then displays the Setup Menu, for interactive robot movement and for setting up the initial conditions

prior to partial program or subroutine execution. If the user wishes to setup the initial conditions, subroutine PARTN is called.

Subroutine PARTN executes commands interactively, so that the robot arm is setup prior to subroutine or partial program execution. Depending on the function key pressed by the user, one of the following commands can be executed: SETPART, GETPART, PMOVE, ZMOVE, WRITEO, SETC, LINEAR, PAYLOAD, and ZONE. For each of the interactive commands, the user is also prompted for the parameters, which may be variable names from the program or constants. A limited error check is done on the command parameters, but in this case, program execution is not aborted if any error is found. The error is displayed, and the program returns to the setup menu. The way these commands are executed are similar to the method described earlier. The only difference is that in the previous case, the command parameters were read from the object file, and in this case the program reads them from the user's input.

Subroutine PARTM is used to identify characters typed in by the user to define variables and constants. This subroutine also echoes the key pressed by the user, and displays it on the screen. It also checks to ensure that the variable names and constants are within the valid range, and if they are not, error messages are displayed.

Subroutine GOSUB is called whenever the user opts for subroutine execution. The user is first prompted for the name of the subroutine that needs to be executed. The program reads the variables file to ensure that

the name is a valid one, and if so it reads the object file to determine the statements numbers where the subroutine begins and ends, and assigns them to the variables PART1 and PART2. If the name is invalid, an error message is displayed and the execution options menu is displayed again. The program then checks to see if this subroutine has any formal parameters, and if it does not have any, the setup menu is displayed. If there are formal parameters that need to be passed to the subroutine, the name of the formal parameter is displayed, and the user is asked whether a variable name will be passed for this parameter. If the user desires to pass a variable name, the program prompts the user for the name. The program reads the values associated with this variable name and writes them in the location of the formal parameter in the symbol table file. If the user wants to pass numeric values to this parameter, the program prompts the user for the number of constants to be passed, and the value for each constant. The program reads in all these values, and writes them in the location of the formal parameter in the symbol table file. This process is carried out till all the formal parameters have been given values, and the setup menu is displayed again.

Subroutine SLOW is called to ask the users whether they want to opt for execution of the program with the robot arm moving slowly. If slow movement is desired, the user is prompted for the payload value at which the robot arm should move. This payload value is read into the variable SLPAY. If SLPAY is zero, the robot arm moves at the default speed.

Subroutine CHECK is called prior to every host initiated communication. This is done to ensure that the data in the controller is not overwritten by the data to be transmitted. The variable CODE is read, and data is transmitted only if CODE is equal to zero, which specifies that the controller is waiting for data from the host. If CODE is not equal to zero, the subroutine loops to cause a delay, and reads CODE again.

Subroutine ERROR is called whenever there is a communication error between the host and the robot controller. This subroutine is called after the host tries to communicate, without success, with the controller five times. Once a communication error is detected, program execution is aborted after a message is displayed to the user, and subroutine FINI is called.

Subroutine USER is called whenever there is an error in the user's AML/E program. There are a total of 13 messages in this subroutine, which are displayed depending on the condition encountered. This subroutine is called either when the user's program is being executed, or when the user specifies values during setup prior to partial program execution. When the error occurs during setup, program execution does not terminate, and the user can specify another value. However, when the error occurs during program execution, program execution terminates, and the user needs to edit the AML/E program to correct the error. Of the 13 messages, one of them is not an error message, but is a message to inform the user that the command BREAKPOINT has been found in the program. In this case, program execution resumes after the user strikes a key.

There are three utility routines in the system control program. The first is subroutine TTOUT, which is used to write a string of characters to the display. The user can select the attributes of the characters to be displayed, and also the location on the screen. It calls two ASSEMBLY utility routines, LOCATE and WRCHAT.

The second utility subroutine is ID, which is called whenever the variable names or numbers needs to be read from the display. The characters that can be read are alphabets, numbers, the minus sign, and the decimal point. All other keyboard characters are not recognized, and if they are struck, an error message is issued.

Subroutine FINI, the third utility routine, is called to terminate program execution. It closes all files, and deletes all files created by the pseudo-compiler, except the user's source program (.AML) and the listing file (.OUT).

4.4 INTEGRATION OF ASSEMBLY SUBROUTINES

ASSEMBLY routines were written to enable communication between the host computer and the robot controller. These subroutines were written by DeMeter, and the complete documentation of these subroutines can be found in his thesis [7]. Using these subroutines, one can read and write values of constants, counters, and points into specific memory locations of the robot controller. One can also read the status of DI/DO points at any

instant in time. These values are then used by the motion control program to execute the commands in the user's AML/E program.

The various ASSEMBLY subroutines and their calling sequences are summarized in Appendix D.

4.5 MOTION CONTROL PROGRAM

The motion control program is an AML/E program and is resident in the robot controller. This program is responsible for the motion of the robot arm, based on the parameters transmitted by the host to the controller memory locations. An overview of the motion control program operation is shown in Figure 14. The various memory locations associated with the variables in the motion control program are summarized in Figure 15.

The AML/E commands residing in the controller and the associated CODE value in parenthesis are BREAKPOINT(1), DELAY(2), PMOVE(6), WAITI(7 and 8), and WRITEO(9). All other commands are handled by the host itself.

The host transmits a value for the variable CODE, depending on the command to be executed, and also any associated parameter values, for example, IONUM, VALUE, TIME, POINT1, etc. The variable CODE is the last variable to be transmitted, and the controller constantly checks till it reads a non-zero value for this variable. If CODE is equal to three, the exe-

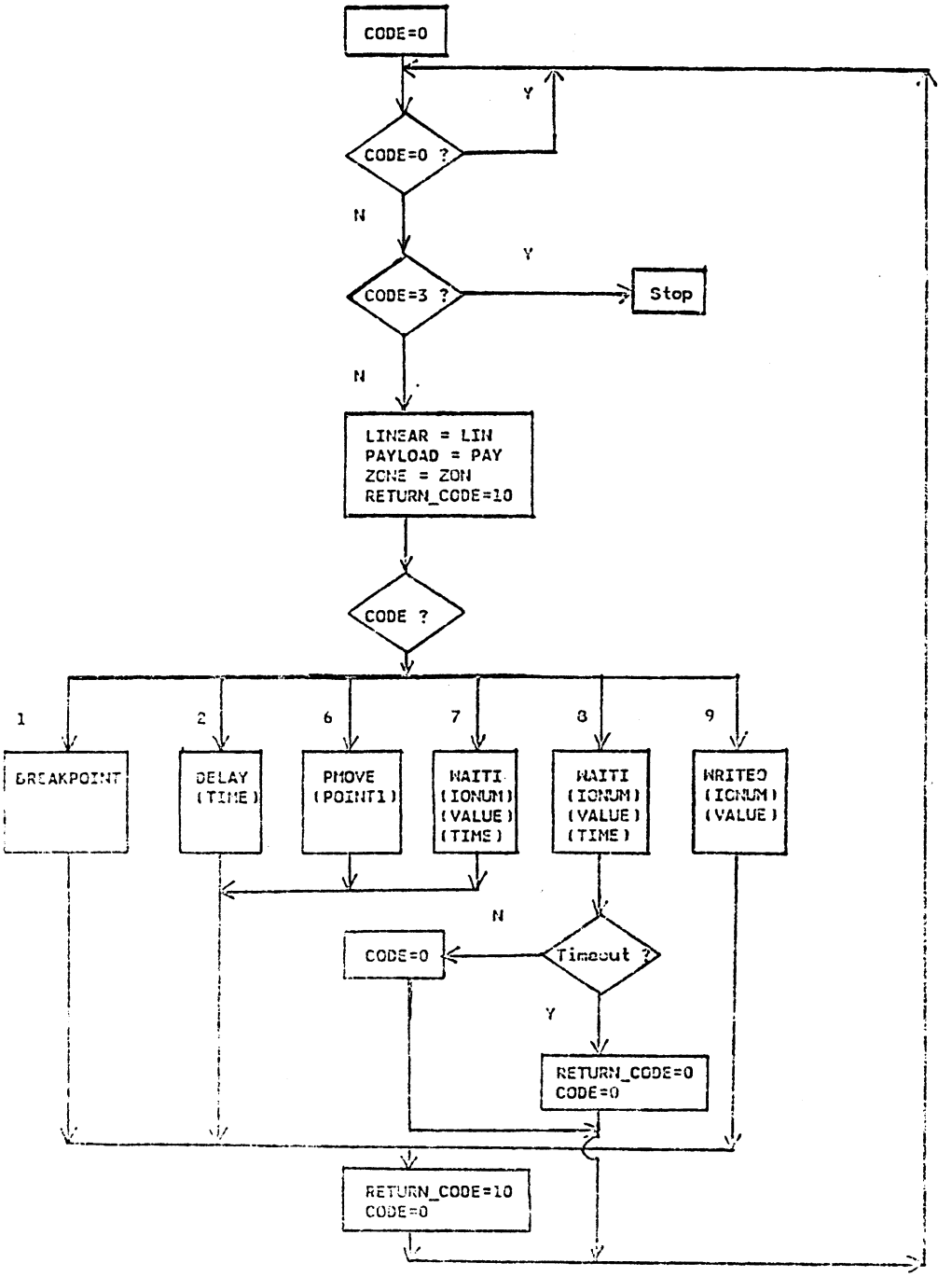

Figure 14. Overview of the motion control program.

VARIABLE	LOCATION	DESCRIPTION
CODE	34	Specifies command type
LIN	35	LINEAR Value
PAY	36	PAYLOAD Value
ZON	37	ZONE Value
VALUE	38	Value of DI/DO ports (0 or 1)
IONUM	39	DI/DO Number
TIME	40	Time value for DELAY/WAITI
RETURN_CODE	41	Code to read after WAITI
POINT1	42-45	X,Y,Z,R values of point

Figure 15. Variables in robot controller memory.

cution of the motion control program terminates. Otherwise, it executes one of the five available commands.

There are two different types of the WAITI command: one which has an indefinite wait, and the other which has a time limit and/or label associated with it. This was done to let the host know if a timeout condition was encountered by setting the variable RETURN_CODE to zero. The value for CODE and RETURN_CODE are then initialized, and the program then loops back and waits for the next CODE to be transmitted by the host.

The commands LINEAR, PAYLOAD, and ZONE are executed every time a value for CODE is transmitted. So every time one of these commands are encountered in the user's AML/E program, the host transmits the associated parameter value to the variable LIN, PAY or ZON, but no CODE value is transmitted.

The listing of the motion control program is given in Appendix E. The compiled version of the motion control program must be resident in the controller partition number five, for interactive program execution.

5.0 CHAPTER 5. ANALYSIS OF THE PROGRAMMING SYSTEM

This chapter discusses some of the assumptions made in the development of the interactive robot programming system, and the limitations of the system. Also discussed are the test programs which were developed to aid in the verification and debugging process of the interactive programming system. Finally, some of the operational aspects of the system are presented.

5.1 ASSUMPTIONS AND LIMITATIONS

The user's AML/E program must be first compiled using the AML/E compiler. This needs to be done, because the error detection features in the pseudo-compiler and the system control program are limited. For example, a statement like "ZMOVE(-200)", would not be detected as an error by the pseudo-compiler, although the statement must be followed by a semicolon. However, a statement like "ZMOVE(-300);" would be detected as an error by the system control program, and the message "Z value out of range" would be displayed. The AML/E compiler can be called from the main menu, so the user does not have to exit from the system to compile programs prior to interactive execution.

The system developed does not handle all the AML/E (Version 4.0) commands. The commands that are not considered are the following: CSTATUS, GET, GROUP, GUARDI, ITERATE, MSTATUS, NOGUARD, PUT, REGION, and XMOVE. The

commands that were considered are shown in Figure 12 in the previous chapter. To include those commands which involve communications, like GET, PUT, etc., would be a complex task. The other commands can be included in the system, with some additional effort.

There are two restrictions on formal parameters in a subroutine. The first is that pallet names cannot be passed to a formal parameter. The second restriction is that a maximum of five formal parameters can be used in a subroutine. If there are more than five, the system just reads the first five and ignores the rest. A valid call to a subroutine, as shown in the second test program in Appendix G, would be:

```
SUB4(UR1,CNT1,CON2,3,CNT4);
```

A point may be defined as a set of four numeric values, or as a set of four counter names, but not as a combination of numeric values and counter names.

The maximum number of lines in the user's AML/E program is 999. This is really not a limitation, as the AML/E editor cannot handle files greater than 800 lines on an IBM PC with 256K memory. The number 999 was chosen, as the line numbers were written to all the files using an I3 format statement.

The system interactively executes the user's program. However, during the execution, there are no help screens provided to assist the user. If they were incorporated, the system would be more user-friendly.

5.2 TEST PROGRAMS

Test programs were developed to aid in verifying and debugging the interactive programming system. These test programs are AML/E programs, which are interactively executed by the system. The test programs were designed on the following criteria:

1. All commands were executed using variables, constants and counters as command parameters.
2. All commands, excluding pallet commands, were used in subroutines, with the command parameters being passed as formal parameters to the subroutines.
3. Subroutines were created with five formal parameters, which was the maximum number that could be handled by the system control program.
4. Limitations in the system control program design were considered, and thus specific test cases were not included.

The first test program was written to test all the AML/E commands, using both variables and numeric values as command parameters. The motion parameters were carefully selected, to ensure that during the testing process, one can visually see the speed of the robot arm motion. Based on this speed, and the point to which the robot moves, one can conclude whether the test program ran successfully or not. Also, the various test conditions in the program were arranged to ensure that the flow of control in the program could be followed with ease. This test program is shown in Appendix F.

The second program tests all the commands, with the command parameters being passed as formal parameters of a subroutine. This program also tests cases when the names of local variables are identical in different subroutines. As the system control program was designed to handle up to five formal parameters, this test program was designed to test subroutines with zero, one and five formal parameters. By testing these cases, it is assumed that the system control program can successfully handle all the cases which are not tested. The success or failure of this program also can be seen visually by the person conducting the test. This test program is shown in Appendix G.

These test programs were then executed interactively, in all modes of operation, namely, single-stepping, slow execution, partial execution, and execution of subroutines alone. Some errors in the system control program logic were identified, and corrected. These test programs proved to be a valuable tool in the verification process, and it is recommended they be used if any future modifications are made to the interactive robot programming system.

5.3 SYSTEM OPERATION

This section compares the speed of the interactive execution process to that when this system is not used. Also, the memory requirement on the host computer is discussed.

To make a comparison on the speed of execution, the two test programs were used. The first test program was compiled, loaded into the robot controller and executed from the operator control panel. The total time taken for these steps was 78 seconds. Then the program was executed in an interactive mode. The time for this process was 130 seconds. Similarly, the second test program took times of 64 seconds and 135 seconds for the two approaches. So, interactive execution is approximately twice as slow as using the AML/E options. However, the flexibility provided by the interactive system, and its aid in the debugging process are valuable.

The interactive system needs an IBM PC with a memory of at least 256K. It is also preferable to run this system on a hard disk, instead of floppy diskettes, to enable faster execution. The size of the executable files and the memory required to store variables are shown in parenthesis for the following files: SYS1 (156K and 13K), COMPPC (112K and 48K), and IRPS1 (45K and 0.5K). For the pseudo-compiler, the memory for variables was restricted, so that the system can be run on an IBM PC with 256K memory. This was achieved by limiting the number of symbolic names and numeric values in a user's program to 100 each. If 512K size memory is available, these array sizes can easily be tripled in value, and the programs can still be executed.

6.0 CHAPTER 6. CONCLUSIONS AND RECOMMENDATIONS

The development of this system provides a demonstration of the use of a host-driven interactive robot programming system. The system, as developed, could be used for program development and debugging purposes associated with basic AML/E programming. This system could serve as a model for future extension to include all AML/E commands, and possibly other robot programming languages also.

This system could be improved by the addition of help menus and more elaborate error detection features. Currently, there are no help menus to assist the user during the interactive execution of the AML/E program. A set of help screens, which the user could call upon during the interactive session, would make this system more user-friendly.

The error detection features in both the pseudo-compiler and the system control program are limited. For this reason it is advised that the user compiles the AML/E program using the AML/E compiler, to ensure that there are no logic and syntax errors. Only if the AML/E compiler detects no errors, the interactive programming system should be used to ensure that program execution is accurate. The errors detected by the system control program causes the interactive execution to be aborted, and the main menu is displayed. The user needs to modify the AML/E program, and execute it interactively again. To be a complete interactive system, runtime errors should be displayed, and the users should be able to modify their

programs, and resume execution at the point where the error was detected. By including such error detection and correction features, the system would be more user-friendly. Also, debugging time would be further reduced, as the user need not go through a compiling phase.

This system currently handles a subset of AML/E (Version 4.0) commands. This was done to simplify the programming effort, and to demonstrate the feasibility of such a system. This system can be expanded to include all the AML/E (Version 4.0) commands, and the additional commands from Version 4.1 can also be included. Commands which do not involve any communication between the host computer and the robot controller, can be added to this system with some effort. However, to include commands which do involve communication, like GET, PUT, etc., is a challenging task. These commands occur when a user's application program is running on the host computer, and data needs to be transmitted to and from the robot controller. So, the system control program needs to incorporate the user's application program in some manner, to be able to execute these commands.

The concept used in the development of this system can be applied to process the AML language, and executed on the IBM 7545 robot. This could be done by transforming the AML commands to equivalent AML/E commands using a pre-processor. This pre-processor would be similar to the pseudo-compiler, and the program could then be executed using the same system control program. The concept of creating a pre-processor could be extended to other robot programming languages, like VAL, MCL, etc.

The only limiting factor would be the transformation of commands from these languages to equivalent commands in the AML/E language.

LIST OF REFERENCES

1. AML/Entry Version 3 User's Guide, IBM Manual 8577125.

2. AML/Entry Version 4 User's Guide, IBM Manual 8577150.

3. Buckley, S. J. and Collins, G. F., "A Structured Programming Robot Language," Handbook of Industrial Robotics, 381-403, John Wiley & Sons (1985).

4. Carayannis, G., "Programming Languages For Intelligent Robots," (Sept. 1983).

5. Curtis, B. L., "Graphical Robotic Planning And Programming System," MS Thesis, Michigan Technological University (1983).

6. Deisenroth, M. P., "Robot Teaching," Handbook of Industrial Robotics, 352-365, John Wiley & Sons (1985).

7. DeMeter, E. C., "The integration of visual and tactile sensing for the definition of regions within a robot workcell," MS Thesis, Virginia Polytechnic and State University (1986).

8. Derby, S. J., "Kinematic Elasto-Dynamic Analysis And Computer Graphics Simulation Of A General Purpose Robot Manipulator," Ph.D. Thesis, Rensselaer Polytechnic Institute (1981).

9. Gilbert, A., Pelton, G., Wang, R., and Motiwalla, S., "Programming a Robot in Basic," Computers in Mechanical Engineering, Vol. 2, No. 6, 28-33 (May 1984).

10. Gruver, W. A., Soroka, B. I., Craig, J. J., and Turner, T. L., "Evaluation of Commercially Available Robot Programming Languages," Proceedings of the 13th International Symposium on Industrial Robots / Robots 7 Conference, (Apr. 1983).

11. Lozano-Perez, T. and Brooks, R. A., "Task-level Manipulator Programming," Handbook of Industrial Robotics, 404-418, John Wiley & Sons (1985).

12. Rembold, U. and Blume, C., "Programming Languages And Systems For Assembly Robots," Computers in Mechanical Engineering, Vol. 2, No. 1, 61-68 (Jan. 1984).

13. Stauffer, R. N., "Robot System Simulation," Robotics Today, Vol. 6, No. 3, 81-90 (June 1984).

14. Stephanic, D. F., "Offline Programming Of Industrial Robots Using Interactive Computer Graphics," MS Thesis, Michigan Technological University (1982).

15. Tarvin, R. L., "An Off-line Programming Approach," Robotics Today, Vol. 3, No. 2, 30-35 (Summer 1981).

16. Yong, Y. F, Gleave, J. A., Green, J. L., and Bonney, M. C., "Off-line Programming Of Robots" Handbook of Industrial Robotics, 366-380, John Wiley & Sons (1985).

APPENDIX A. MAIN MENU PROGRAM

```
C     THIS PROGRAM, IRPS1.FOR, IS THE MAIN MENU PROGRAM.
C     THIS PROGRAM IS CALLED BY THE "INTER.BAT" FILE.
C     THIS PROGRAM PUTS UP THE MAIN MENU FOR THE INTERACTIVE ROBOT
C     PROGRAMMING SYSTEM AND ACCEPTS INPUT FROM THE USER. THE FILENAME
C     AND OPTIONS SPECIFIED BY THE USER ARE WRITTEN INTO A FILE
C     CALLED "IRPS0". A BATCH FILE IS ALSO CREATED - CALLED "IRPS2.BAT"
C     WHICH WILL BE EXECUTED LATER BY THE BATCH FILE "INTER.BAT".


C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             AMLCOM,AMLED,CLRBUF,CLS,CSROFF,CSRON,INKEY,
C             INTER,MENU,TTOUT.

      CHARACTER * 1 BLANK(80),C1,C2,C3,C4
      CHARACTER * 20 NAME
      INTEGER * 2 J,L,KYCOD1,KYCOD2
      COMMON /A/ NAME,BLANK,C1,C2,C3,C4
      COMMON /B/ L

C     VARIABLES C1,C2,C3,C4 REPRESENT THE COMPILER OPTIONS.
C     C1 AND C3 MAY BE '/' OR A BLANK CHARACTER.
C     C2 AND C4 MAY BE 'L','S' OR A BLANK CHARACTER.

      L = 0
      C1 = ' '
      C2 = ' '
      C3 = ' '
      C4 = ' '

      OPEN (1,FILE='IRPS0')
      OPEN (2,FILE='IRPS2.BAT')

C     READ THE FILE IRPS0 FOR PROGRAM OPTIONS, IF ANY.

      READ (1,110,END=10) L,NAME,C1,C2,C3,C4

110   FORMAT(I2,1X,A20,4A1)
10    REWIND (1)

      CALL CLS(31)
      CALL CSRON
      CALL CLRBUF

C     DISPLAY MAIN MENU IF NO FILENAME  EXISTS IN THE FILE "IRPS0",
C     OTHERWISE DISPLAY THE PROGRAM OPTIONS MENU.

      IF (L .LE. 1) THEN
      CALL TTOUT(0,1,21,'IBM 7545 INTERACTIVE PROGRAMMING SYSTEM',39,39,
     *31)
         CALL TTOUT(0,2,26,'COPYRIGHT VIRGINIA TECH 1986',28,28,31)
      CALL TTOUT(0,3,21,'----------------------------------------',39,39,
     *31)
         CALL TTOUT(0,5,35,'MAIN MENU',9,9,31)
         CALL TTOUT(0,6,35,'---------',9,9,31)
      ELSE
         CALL TTOUT(0,1,28,'PROGRAM SYSTEM OPTIONS',22,22,31)
         CALL TTOUT(0,2,28,'----------------------',22,22,31)
         CALL TTOUT(0,4,14,'Filename (.AML assumed):',24,24,26)
         CALL TTOUT(0,5,21,'Compiler options:',17,17,26)
         J = 0

         DO 1 I = 22-L,20
            J = J + 1
```

```
            CALL TTOUT(0,4,38+J,NAME(I:I),L-1,L-1,26)
1        CONTINUE

         IF (C1 .NE. ' ' .AND. C2 .NE. ' ') THEN
            CALL TTOUT(0,5,39,C1,1,1,26)
            CALL TTOUT(0,5,40,C2,1,1,26)
         ENDIF

         IF (C3 .NE. ' ' .AND. C4 .NE. ' ') THEN
            CALL TTOUT(0,5,41,C3,1,1,26)
            CALL TTOUT(0,5,42,C4,1,1,26)
         ENDIF

      ENDIF

      CALL TTOUT(0,9,24,'Select a function:',18,18,27)
      CALL TTOUT(0,12,21,'F1 - Return to DOS',18,18,31)
      CALL TTOUT(0,13,21,'F2 - Edit/Teach a program',25,25,31)
      CALL TTOUT(0,14,21,'F3 - Compile a program',22,22,31)
      CALL TTOUT(0,15,21,'F4 - Run a program interactively',32,32,31)
      CALL TTOUT(0,16,21,'F5 - Set program name and options',33,33,31)
      CALL TTOUT(0,20,24,'Enter Option===>',16,16,27)
11    CALL LOCATE(0,20,41)
      CALL CLRBUF
      CALL INKEY(KYCOD1,KYCOD2)
      IF (KYCOD1 .NE. 0) THEN
         CALL CSROFF
         CALL TTOUT(0,24,1,'KEY NOT DEFINED -- Hit any key to resume',
     *   40,40,28)
         CALL CLRBUF
         CALL INKEY(KYCOD1,KYCOD2)
         CALL CLRBUF
         CALL CSRON
         CALL TTOUT(0,24,1,BLANK,80,80,31)
         GOTO 11
      ENDIF

C     IF F1 IS PRESSED, OPEN A TEMPORARY FILE 'IRPS.TMP'.

      IF (KYCOD2 .EQ. 59) THEN
         CALL CLRBUF
         OPEN (3,FILE='IRPS.TMP')
         CLOSE (3)
         GOTO 100

C     F2 IS PRESSED

      ELSEIF (KYCOD2 .EQ. 60) THEN
         CALL CLRBUF
         CALL AMLED

C     F3 IS PRESSED

      ELSEIF (KYCOD2 .EQ. 61) THEN
         CALL CLRBUF
         CALL AMLCOM

C     F4 IS PRESSED

      ELSEIF (KYCOD2 .EQ. 62) THEN
         CALL CLRBUF
         CALL INTER

C     F5 IS PRESSED

      ELSEIF (KYCOD2 .EQ. 63) THEN
         CALL CLRBUF
         CALL MENU
         GOTO 11
```

```
      ELSE
         CALL CSROFF
         CALL TTOUT(0,24,1,'KEY NOT DEFINED -- Hit any key to resume',
     *   40,40,28)
         CALL CLRBUF
         CALL INKEY(KYCOD1,KYCOD2)
         CALL CLRBUF
         CALL CSRON
         CALL TTOUT(0,24,1,BLANK,80,80,31)
         GOTO 11
      ENDIF

100   CLOSE(1)
      CLOSE(2)
      CALL CLS(0)
      CALL CSRON

      STOP
      END

C     BLOCK DATA

C     THIS BLOCK DEFINES THE ARRAY BLANK, WHICH IS AN ARRAY OF BLANK
C     CHARACTERS.

      BLOCK DATA

      CHARACTER * 1 BLANK(80),C1,C2,C3,C4
      CHARACTER * 20 NAME
      INTEGER L
      COMMON /A/ NAME,BLANK,C1,C2,C3,C4
      COMMON /B/ L

      DATA BLANK/80 * ' '/

      END

C     SUBROUTINE TTOUT

C     THIS SUBROUTINE WRITES A STRING A CHARACTERS AT A SPECIFIED
C     ROW AND COLUMN OF THE DISPLAY.

C     PARAMETER DEFINITION:

C          PAGE      -      PAGE NUMBER
C          ROW       -      ROW NUMBER
C          COLUMN    -      COLUMN NUMBER
C          BUF       -      CHARACTER BUFFER NAME
C          SIZE      -      SIZE OF BUFFER
C          LENGTH    -      LENGTH OF CHARACTER STRING
C          WRATT     -      ATTRIBUTE OF CHARACTER TO BE WRITTEN


C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             LOCATE, WRCHAT.

      SUBROUTINE TTOUT(PAGE,ROW,COLUMN,BUF,LENGTH,SIZE,WRATT)

      INTEGER * 2 PAGE,ROW,COLUMN,SIZE,WRATT
      CHARACTER * 1 BUF(SIZE)
      CHARACTER * 1 BLANK(80),C1,C2,C3,C4
      CHARACTER * 20 NAME
      INTEGER * 2 L
      COMMON /A/ NAME,BLANK,C1,C2,C3,C4
      COMMON /B/ L

      CALL LOCATE(PAGE,ROW,COLUMN)
```

```
      DO 10 I = 1,LENGTH
         CALL WRCHAT(PAGE,BUF(I),WRATT,1)
10    CONTINUE

      RETURN
      END

C     SUBROUTINE AMLED

C     THIS SUBROUTINE CREATES THE BATCH FILE IRPS2.BAT, AND WRITES
C     THE CALL TO THE EDITOR ALONG WITH THE AML/E FILE NAME, IF
C     SPECIFIED, AND ALSO WRITES THE COMMAND, INTER, SO THAT
C     CONTROL RETURN TO THE MAIN MENU PROGRAM WHEN EDITING IS COMPLETE.

      SUBROUTINE AMLED

      CHARACTER * 1 BLANK(80),C1,C2,C3,C4
      CHARACTER * 20 NAME
      INTEGER * 2 L
      COMMON /A/ NAME,BLANK,C1,C2,C3,C4
      COMMON /B/ L

      WRITE (2,10)
      READ (1,100,END=15) L,NAME
      WRITE (2,20) NAME
      WRITE (2,30)
      GOTO 200
15    WRITE (2,20)
      WRITE (2,30)
10    FORMAT('ECHO OFF')
20    FORMAT('EDIT',1X,A20)
30    FORMAT('INTER')
100   FORMAT(I2,1X,A20)

200   REWIND (1)

      RETURN
      END

C     SUBROUTINE TTOUT

C     THIS SUBROUTINE CREATES THE BATCH FILE IRPS2.BAT, AND WRITES
C     THE CALL TO THE AML/E COMPILER ALONG WITH THE AML/E FILE NAME AND
C     OPTIONS, IF SPECIFIED, AND ALSO WRITES THE COMMAND, INTER, SO THAT
C     CONTROL RETURN TO THE MAIN MENU PROGRAM WHEN COMPILING IS COMPLETE.

      SUBROUTINE AMLCOM

      CHARACTER * 1 BLANK(80),C1,C2,C3,C4
      CHARACTER * 20 NAME
      INTEGER * 2 L
      COMMON /A/ NAME,BLANK,C1,C2,C3,C4
      COMMON /B/ L

      WRITE (2,10)
      READ (1,100,END=15) L,NAME,C1,C2,C3,C4
      IF (L .LE. 1) GOTO 15

      IF (C1 .NE. ' ' .AND. C2 .NE. ' ') THEN

         IF (C3 .NE. ' ' .AND. C4 .NE. ' ') THEN
            WRITE (2,20) NAME,C1,C2,C3,C4
         ELSE
            WRITE (2,20) NAME,C1,C2
         ENDIF

      ELSE
         WRITE (2,20) NAME
      ENDIF
```

```
      WRITE (2,30)
      GOTO 200

15    WRITE (2,20)
      WRITE (2,30)

10    FORMAT('ECHO OFF')
20    FORMAT('COMPILER',1X,A20,4A1)
30    FORMAT('INTER')
100   FORMAT(I2,1X,A20,4A1)

200   REWIND (1)

      RETURN
      END

C     SUBROUTINE INTER

C     THIS SUBROUTINE CREATES THE BATCH FILE IRPS2.BAT, AND WRITES
C     THE CALL TO THE PSEUDO-COMPILER, ALONG WITH THE AML/E FILENAME,
C     IF SPECIFIED, AND ALSO THE CALL TO THE SYSTEM CONTROL PROGRAM.
C     THE CALL TO THE BATCH FILE, INTER, IS ALSO WRITTEN SO THAT
C     CONTROL RETURNS TO THE MAIN MENU PROGRAM WHEN THE PROGRAM HAS BEEN
C     EXECUTED INTERACTIVELY.

      SUBROUTINE INTER

      CHARACTER * 1 BLANK(80),C1,C2,C3,C4
      CHARACTER * 20 NAME
      INTEGER * 2 L
      COMMON /A/ NAME,BLANK,C1,C2,C3,C4
      COMMON /B/ L

      WRITE (2,10)
      WRITE (2,20)
      WRITE (2,30)
      WRITE (2,40)

10    FORMAT('ECHO OFF')
20    FORMAT('COMPPC')
30    FORMAT('SYS1')
40    FORMAT('INTER')

      RETURN
      END

C     SUBROUTINE MENU

C     THIS SUBROUTINE DISPLAYS THE PROGRAM OPTIONS MENU AND READS
C     THE USER'S INPUT FOR THE PROGRAM NAME, AND THE COMPILER OPTIONS.

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             CLRBUF,CLS,CSROFF,CSROFF,ID,INKEY,TTOUT.

      SUBROUTINE MENU

      CHARACTER * 1 CHAR,ALPHA(26),NUMER(10)
      INTEGER * 2 J,KYCOD1,KYCOD2
      CHARACTER * 1 BLANK(80),C1,C2,C3,C4
      CHARACTER * 20 NAME
      INTEGER * 2 L
      COMMON /A/ NAME,BLANK,C1,C2,C3,C4
      COMMON /B/ L

      DATA ALPHA/'A','B','C','D','E','F','G','H','I','J','K','L','M',
     *'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'/
      DATA NUMER/'0','1','2','3','4','5','6','7','8','9'/
```

```
      CALL CLS(31)
      CALL CSRON
      CALL TTOUT(0,1,28,'PROGRAM SYSTEM OPTIONS',22,22,31)
      CALL TTOUT(0,2,28,'----------------------',22,22,31)
      CALL TTOUT(0,4,14,'Filename (.AML assumed):',24,24,26)
      CALL TTOUT(0,5,21,'Compiler options:',17,17,26)
      CALL TTOUT(0,9,24,'Select a function:',18,18,27)
      CALL TTOUT(0,12,21,'F1 - Return to DOS',18,18,31)
      CALL TTOUT(0,13,21,'F2 - Edit/Teach a program',25,25,31)
      CALL TTOUT(0,14,21,'F3 - Compile a program',22,22,31)
      CALL TTOUT(0,15,21,'F4 - Run a program interactively',32,32,31)
      CALL TTOUT(0,16,21,'F5 - Set program name and options',33,33,31)
      CALL TTOUT(0,20,24,'Enter Option===>',16,16,27)
      WRITE (1,2)
      REWIND (1)

2     FORMAT('                          ')

C     INITIALIZE NAME TO BLANKS.

      DO 45 I = 1,20
         NAME(I:I) = ' '
45    CONTINUE

      C1 = ' '
      C2 = ' '
      C3 = ' '
      C4 = ' '

      L = 0

C     LOCATE CURSOR, AND DISPLAY VALID INPUTS ON THE SCREEN AND
C     CONCATENATE THE NAME.

      DO 10 I = 1,200
         L = L + 1
         IF (L .GE. 21) GOTO 20
5        J = L
         CALL LOCATE(0,4,38+L)
         CALL CLRBUF
         CALL INKEY(KYCOD1,KYCOD2)

         IF (KYCOD1 .EQ. 13) THEN
            GOTO 20
         ELSE
            CALL ID(KYCOD1,KYCOD2,CHAR,J,ALPHA,NUMER)
            IF (J .EQ. 0) THEN
               CALL CSROFF
           CALL TTOUT(0,24,1,'INVALID CHARACTER - Hit any key to resume'
     *      ,41,41,28)
               CALL CLRBUF
               CALL INKEY(KYCOD1,KYCOD2)
               CALL CLRBUF
               CALL TTOUT(0,24,1,BLANK,80,80,31)
               CALL CSRON
               GOTO 5
            ENDIF

            IF (J .LT. L) THEN
               L = J
               CALL TTOUT(0,4,J+38,BLANK(1),1,1,31)
               GOTO 5
            ELSE
               CALL TTOUT(0,4,38+L,CHAR,1,1,26)

               IF (J .NE. 1) THEN
                  NAME(1:L) = NAME(1:L-1)//CHAR
               ELSE
                  NAME(1:L) = CHAR
```

```
               ENDIF

            ENDIF

         ENDIF

10    CONTINUE

20    CALL CLRBUF

C     POSITION CURSOR TO READ COMPILER OPTIONS.

31    CALL LOCATE(0,5,39)
      CALL INKEY(KYCOD1,KYCOD2)
      IF (KYCOD1 .EQ. 13) GOTO 36

      IF (KYCOD1 .NE. 47) THEN
         CALL CSROFF
         CALL TTOUT(0,24,1,'INVALID CHARACTER - Hit any key to resume'
     *   ,41,41,28)
         CALL CLRBUF
         CALL INKEY(KYCOD1,KYCOD2)
         CALL CLRBUF
         CALL TTOUT(0,24,1,BLANK,80,80,31)
         CALL CSRON
         GOTO 31
      ELSE
         C1 = '/'
         CALL TTOUT(0,5,39,C1,1,1,26)
      ENDIF

32    CALL LOCATE(0,5,40)
      CALL INKEY(KYCOD1,KYCOD2)

      IF (KYCOD1 .EQ. 13) THEN
         C1 = ' '
         GOTO 36
      ENDIF

      CALL ID(KYCOD1,KYCOD2,CHAR,J,ALPHA,NUMER)

      IF (CHAR .NE. 'L' .AND. CHAR .NE. 'S') THEN
         CALL CSROFF
         CALL TTOUT(0,24,1,'INVALID CHARACTER - Hit any key to resume'
     *   ,41,41,28)
         CALL CLRBUF
         CALL INKEY(KYCOD1,KYCOD2)
         CALL CLRBUF
         CALL TTOUT(0,24,1,BLANK,80,80,31)
         CALL CSRON
         GOTO 32
      ELSE
         C2 = CHAR
         CALL TTOUT(0,5,40,C2,1,1,26)
      ENDIF

33    CALL LOCATE(0,5,41)
      CALL INKEY(KYCOD1,KYCOD2)
      IF (KYCOD1 .EQ. 13) GOTO 36

      IF (KYCOD1 .NE. 47) THEN
         CALL CSROFF
         CALL TTOUT(0,24,1,'INVALID CHARACTER - Hit any key to resume'
     *   ,41,41,28)
         CALL CLRBUF
         CALL INKEY(KYCOD1,KYCOD2)
         CALL CLRBUF
         CALL TTOUT(0,24,1,BLANK,80,80,31)
         CALL CSRON
```

```
         GOTO 33
      ELSE
         C3 = '/'
         CALL TTOUT(0,5,41,C3,1,1,26)
      ENDIF

34    CALL LOCATE(0,5,42)
      CALL INKEY(KYCOD1,KYCOD2)

      IF (KYCOD1 .EQ. 13) THEN
         C3 = ' '
         GOTO 36
      ENDIF

      CALL ID(KYCOD1,KYCOD2,CHAR,J,ALPHA,NUMER)

      IF (CHAR .NE. 'L' .AND. CHAR .NE. 'S') THEN
         CALL CSROFF
         CALL TTOUT(0,24,1,'INVALID CHARACTER - Hit any key to resume'
     *   ,41,41,28)
         CALL CLRBUF
         CALL INKEY(KYCOD1,KYCOD2)
         CALL CLRBUF
         CALL TTOUT(0,24,1,BLANK,80,80,31)
         CALL CSRON
         GOTO 34
      ELSE
         C4 = CHAR
         CALL TTOUT(0,5,42,C4,1,1,26)
      ENDIF

36    IF (C2 .EQ. C4) THEN
         C3 = ' '
         C4 = ' '
      ENDIF

C     WRITE THE PROGRAM NAME AND COMPILER OPTIONS INTO THE FILE IRPSO.

      WRITE (1,30) L,NAME(1:L-1),C1,C2,C3,C4
      REWIND (1)

30    FORMAT(I2,1X,A20,4A1)

      RETURN
      END

C     SUBROUTINE ID

C     THIS SUBROUTINE IDENTIFIES THE CHARACTER TYPED IN BY THE USER.
C     VALID CHARACTERS ARE ALL ALPHABETS, NUMBERS, THE BACKSPACE KEY,
C     THE ENTER KEY AND SOME SPECIAL CHARACTERS.

C     PARAMETER DEFINITION:

C          KYCOD1   -      ASCII CODE OF KEY
C          KYCOD2   -      EXTENDED CHARACTER CODE OF KEY
C          CHAR     -      CHARACTER IDENTIFIED
C          J        -      COLUMN POSITION
C          ALPHA    -      ARRAY OF ALPHABETS
C          NUMMER   -      ARRAY OF NUMBERS

      SUBROUTINE ID(KYCOD1,KYCOD2,CHAR,J,ALPHA,NUMER)

      INTEGER * 2 KYCOD1,KYCOD2,J
      CHARACTER * 1 CHAR,ALPHA(26),NUMER(10)
      CHARACTER * 1 BLANK(80),C1,C2,C3,C4
      CHARACTER * 20 NAME
      INTEGER * 2 L
      COMMON /A/ NAME,BLANK,C1,C2,C3,C4
```

```
COMMON /B/ L

IF (KYCOD1 .GE. 48 .AND. KYCOD1 .LE. 57) THEN
   CHAR = NUMER(KYCOD1-47)
ELSEIF (KYCOD1 .GE. 65 .AND. KYCOD1 .LE. 90) THEN
   CHAR = ALPHA(KYCOD1-64)
ELSEIF (KYCOD1 .GE. 97 .AND. KYCOD1 .LE. 122) THEN
   CHAR = ALPHA(KYCOD1-96)
ELSEIF (KYCOD1 .EQ. 29) THEN
   J = J - 1
ELSEIF (KYCOD1 .EQ. 8 .AND. KYCOD2 .EQ. 14) THEN
   J = J - 1
ELSEIF (KYCOD1 .EQ. 127 .AND. KYCOD2 .EQ. 14) THEN
   J = J - 1
ELSEIF (KYCOD1 .EQ. 0 .AND. KYCOD2 .EQ. 75) THEN
   J = J - 1
ELSEIF (KYCOD1 .EQ. 33) THEN
   CHAR = '!'
ELSEIF (KYCOD1 .EQ. 64) THEN
   CHAR = '@'
ELSEIF (KYCOD1 .EQ. 35) THEN
   CHAR = '#'
ELSEIF (KYCOD1 .EQ. 36) THEN
   CHAR = '$'
ELSEIF (KYCOD1 .EQ. 37) THEN
   CHAR = '%'
ELSEIF (KYCOD1 .EQ. 94) THEN
   CHAR = '¢'
ELSEIF (KYCOD1 .EQ. 38) THEN
   CHAR = '&'
ELSEIF (KYCOD1 .EQ. 40) THEN
   CHAR = '('
ELSEIF (KYCOD1 .EQ. 41) THEN
   CHAR = ')'
ELSEIF (KYCOD1 .EQ. 95) THEN
   CHAR = '_'
ELSEIF (KYCOD1 .EQ. 45) THEN
   CHAR = '-'
ELSEIF (KYCOD1 .EQ. 123) THEN
   CHAR = '{'
ELSEIF (KYCOD1 .EQ. 125) THEN
   CHAR = '}'
ELSEIF (KYCOD1 .EQ. 39) THEN
   CHAR = '''
ELSEIF (KYCOD1 .EQ. 58) THEN
   CHAR = ':'
ELSEIF (KYCOD1 .EQ. 47) THEN
   CHAR = '/'
ELSEIF (KYCOD1 .EQ. 92) THEN
   CHAR = '\'
ELSE
   J = 0
ENDIF

RETURN
END
```

APPENDIX B. PSEUDO-COMPILER PROGRAM

```
C     THIS IS THE PSEUDO-COMPILER PROGRAM. THE PROGRAM READS THE USER'S
C     AML/E PROGRAM AND GENERATES FIVE FILES, WHICH WILL THEN BE USED
C     BY THE SYSTEM CONTROL PROGRAM.

C     VARIABLES USED IN THE PSEUDO-COMPILER :

C  ALPHA         =  Array containing alphabetic characters
C  AMLWRD(I)     =  Array containing AML reserved words
C  BLANK         =  Array containing 80 blank characters
C  C(I)          =  Array used to read a line of AML/E program
C  CHARS(I)      =  Array containing alphanumeric and special characters
C  CONS(I)       =  Array containing constants found in AML/E program
C  CONS2(I)      =  Array containing constants found in AML/E program
C  DELIM(I)      =  Array containing delimiters in an AML/E program
C  I             =  Column number pointer
C  INDEX1        =  Starting column number of symbol
C  INDEX2        =  Ending column number of symbol
C  ITYPE         =  Value specifying the type of symbol
C  KTYPE         =  Value specifying whether symbol is a number or variable
C  LENCON(I)     =  Array containing length of constants
C  LENGTH(I)     =  Array containing length of variables
C  LENRES(I)     =  Array containing length of unresolved labels
C  LINENO        =  Index for number of the line in AML/E program
C  MTYPE         =  Value specifying the type of subcommand
C  NUM           =  Index for number of symbol found in AML/E program
C  NUM1          =  Index for number of constant found in AML/E program
C  NUM2          =  Index for number of unresolved symbol found in program
C  NUMER         =  Array containing numeric characters
C  NSUB          =  Index for number of subroutine
C  NTYPE         =  Value specifying the type of command
C  NUMBEG        =  Beginning statement number of subroutine
C  NUMEND        =  Ending statement number of subroutine
C  NUMSUB        =  Statement number of subroutine declaration
C  STNO          =  Statement number of AML/E program
C  SUB(I)        =  Array containing subroutine names
C  SUBNO(I)      =  Array containing first statement number of subroutine
C  SUBNUM        =  Subroutine number
C  SUBOP(I)      =  Array containing index specifying subroutine open/end
C  SUBRNO        =  Number of subroutine into which statement lies
C  SYMBOL(I)     =  Array containing symbols found in AML/E program
C  SYM2(I)       =  Array containing symbols found in AML/E program
C  UNNO(I)       =  Array containing statement number of unresolved labels
C  UNRES(I)      =  Array containing unresolved labels
C  UNVAR(I)      =  Array containing subroutine number of unresolved label
C  VARNO(I)      =  Index specifying whether variable is local or global

C     THE FILES OPENED BY THIS PROGRAM AND THEIR UNIT NUMBERS ARE :

C           FILE EXTENSION      DESCRIPTION             UNIT NUMBER
C           --------------      -----------             -----------
C              .AML             USER'S AML/E PROGRAM        1
C              .OUT             LISTING FILE                2
C              .VAR             VARIABLES FILE              3
C              .CON             CONSTANTS FILE              4
C              .OBJ             OBJECT FILE                 7
C              .TAB             SYMBOL TABLE FILE           8

C     THE OBJECT FILE AND THE SYMBOL TABLE FILE ARE OPENED AS DIRECT
C     ACCESS FILES. ALL OTHER FILES ARE SEQUENTIAL FILES.

C     TWO OTHER FILES ARE OPENED: IRPS0, WITH UNIT NUMBER 9, AND
C     IRPS00, WITH UNIT NUMBER 10. IRPS00 IS USED TO PASS THE NAME
C     OF THE USER'S AML/E PROGRAM NAME TO THE SYSTEM CONTROL
C     PROGRAM. FILE IRPS0 MAY HAVE BEEN CREATED BY THE MAIN MENU
```

```
C     PROGRAM, IF THE USER HAD SPECIFIED THE PROGRAM NAME IN THE
C     PROGRAM OPTIONS MENU.

C     MAIN PROGRAM STARTS

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             CLRBUF,CLS,CSROFF,CSRON,ID,INKEY,LOCATE
C             PROG,TTOUT.

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      CHARACTER * 20 NAME
      CHARACTER * 24 NAMAML,NAMOUT,NAMVAR,NAMOBJ,NAMTAB
      CHARACTER * 24 NA2AML,NA2OUT,NA2VAR,NA2OBJ,NA2TAB
      CHARACTER * 24 NAMCON,NA2CON
      INTEGER * 2 L,KYCOD1,KYCOD2,J
      CHARACTER * 1 BLANK(80),CHAR,ALPHA(26),NUMER(10)
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), P(4)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO

      DATA ALPHA/'A','B','C','D','E','F','G','H','I','J','K','L','M',
     *'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'/
      DATA NUMER/'0','1','2','3','4','5','6','7','8','9'/
      DATA BLANK/80 * ' '/

      CALL CLS(31)
      CALL CLRBUF

C     IF THE USER HAD SET PROGRAM OPTIONS, READ AML/E PROGRAM NAME
C     FROM IRPSO. IF NOT PROMPT USER FOR THE NAME.

      OPEN (9,FILE='IRPSO')
      READ (9,100,END=200) L,NAME
100   FORMAT (I2,1X,A20)
200   CLOSE (9)

210   IF (L .LE. 1) THEN
         CALL TTOUT(0,10,1,'Please enter filename (.AML assumed) :',38,
     *   38,31)

C     THIS LOOP POSITIONS THE CURSOR ON THE DISPLAY, AND ANY VALID
C     CHARACTERS PRESSED BY THE USER ARE DISPLAYED.
C     A MESSAGE APPEARS IF INVALID KEYS ARE PRESSED.
C     THE NAME OF THE FILE IS THEN CONCATENATED.

         L = 0
         DO 10 JJ = 1,200
            L = L + 1
            IF (L .GE. 21) GOTO 20
5           J = L
            CALL LOCATE(0,10,39+L)
            CALL CLRBUF
            CALL INKEY(KYCOD1,KYCOD2)
            IF (KYCOD1 .EQ. 13) THEN
               GOTO 20
            ELSE
               CALL ID(KYCOD1,KYCOD2,CHAR,J,ALPHA,NUMER)
               IF (J .EQ. 0) THEN
                  CALL CSROFF
           CALL TTOUT(0,24,1,'INVALID CHARACTER - Hit any key to resume'
```

```
     *        ,41,41,28)
                 CALL CLRBUF
                 CALL INKEY(KYCOD1,KYCOD2)
                 CALL CLRBUF
                 CALL TTOUT(0,24,1,BLANK,80,80,31)
                 CALL CSRON
                 GOTO 5
              ENDIF

              IF (J .LT. L) THEN
                 L = J
                 CALL TTOUT(0,10,J+39,BLANK(1),1,1,31)
                 GOTO 5
              ELSE
                 CALL TTOUT(0,10,39+L,CHAR,1,1,31)
                 IF (J .NE. 1) THEN
                    NAME(1:L) = NAME(1:L-1)//CHAR
                 ELSE
                    NAME(1:L) = CHAR
                 ENDIF
              ENDIF
           ENDIF
10         CONTINUE

20         IF (L .LE. 1) THEN
              CALL CSROFF
              CALL TTOUT(0,24,1,'INVALID FILENAME - Hit any key to resume'
     *        ,40,40,28)
              CALL CLRBUF
              CALL INKEY(KYCOD1,KYCOD2)
              CALL CLRBUF
              CALL TTOUT(0,24,1,BLANK,80,80,31)
              CALL CSRON
              CALL CLS(31)
              GOTO 210
           ENDIF

C     ALL FILE NAMES ARE DEFINED HERE AND THE USER'S PROGRAM NAME IS
C     WRITTEN TO THE FILE IRPS00.

              NA2AML(1:L+3) = NAME(1:L-1)//'.AML'
              NA2OUT(1:L+3) = NAME(1:L-1)//'.OUT'
              NA2VAR(1:L+3) = NAME(1:L-1)//'.VAR'
              NA2CON(1:L+3) = NAME(1:L-1)//'.CN1'
              NA2OBJ(1:L+3) = NAME(1:L-1)//'.OBJ'
              NA2TAB(1:L+3) = NAME(1:L-1)//'.TAB'

              OPEN (10,FILE='IRPS00')
              WRITE (10,310) L,NAME(1:L-1)
              CLOSE (10)
310           FORMAT(I2,1X,A20)
              GOTO 300
           ENDIF

C     THIS IS EXECUTED IF THE PROGRAM NAME HAS ALREADY BEEN SET

           OPEN (10,FILE='IRPS00')
           WRITE (10,310) L,NAME
           CLOSE (10)

           NAMAML(22-L:24) = NAME(22-L:20)//'.AML'
           NAMOUT(22-L:24) = NAME(22-L:20)//'.OUT'
           NAMVAR(22-L:24) = NAME(22-L:20)//'.VAR'
           NAMCON(22-L:24) = NAME(22-L:20)//'.CN1'
           NAMOBJ(22-L:24) = NAME(22-L:20)//'.OBJ'
           NAMTAB(22-L:24) = NAME(22-L:20)//'.TAB'

           J = 0
           DO 400 JJ = 22-L,24
```

```
         J = J + 1
         NA2AML(J:J) = NAMAML(JJ:JJ)
         NA2OUT(J:J) = NAMOUT(JJ:JJ)
         NA2VAR(J:J) = NAMVAR(JJ:JJ)
         NA2CON(J:J) = NAMCON(JJ:JJ)
         NA2OBJ(J:J) = NAMOBJ(JJ:JJ)
         NA2TAB(J:J) = NAMTAB(JJ:JJ)
400   CONTINUE

300   CALL CSROFF

C     ALL FILES ARE OPENED HERE. IF THERE IS AN ERROR IN OPENING
C     THE USE'S AML/E PROGRAM FILE, A MESSAGE IS DISPLAYED.

      OPEN (1,FILE=NA2AML(1:L+3),STATUS='OLD',ERR=1000)
      OPEN (2,FILE=NA2OUT(1:L+3))
      OPEN (3,FILE=NA2VAR(1:L+3))
      OPEN (4,FILE=NA2CON(1:L+3))
      OPEN (7,FILE=NA2OBJ(1:L+3),ACCESS='DIRECT',FORM='FORMATTED',
     *RECL=80)
      OPEN (8,FILE=NA2TAB(1:L+3),ACCESS='DIRECT',FORM='FORMATTED',
     *RECL=80)
      GOTO 1001

1000  CALL CLS(31)
      CALL CSROFF
      CALL TTOUT(0,10,1,'Error in reading source file..',30,30,28)
      CALL TTOUT(0,10,35,NA2AML(1:L+3),3+L,3+L,28)
      CALL TTOUT(0,24,1,'Press any key to continue....',29,29,28)
      CALL INKEY(KYCOD1,KYCOD2)
      CALL CLS(31)
      L = 0
      CALL CSRON
      GOTO 210

1001  CALL TTOUT(0,24,1,'Reading source file....',23,23,28)
      CALL TTOUT(0,24,27,NA2AML(1:L+3),3+L,3+L,26)

      CALL PROG

      CALL CLS(0)
      CALL CLRBUF
      CALL CSRON
      CLOSE (1)
      CLOSE (2)
      CLOSE (3)
      CLOSE (4)
      CLOSE (7)
      CLOSE (8)
      STOP
      END

C     SUBROUTINE TTOUT

C     THIS SUBROUTINE WRITES A STRING A CHARACTERS AT A SPECIFIED
C     ROW AND COLUMN OF THE DISPLAY.

C     PARAMETER DEFINITION:

C          PAGE      -      PAGE NUMBER
C          ROW       -      ROW NUMBER
C          COLUMN    -      COLUMN NUMBER
C          BUF       -      CHARACTER BUFFER NAME
C          SIZE      -      SIZE OF BUFFER
C          LENG      -      LENGTH OF CHARACTER STRING
C          WRATT     -      ATTRIBUTE OF CHARACTER TO BE WRITTEN
```

```
C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             LOCATE, WRCHAT.

      SUBROUTINE TTOUT(PAGE,ROW,COLUMN,BUF,LENG,SIZE,WRATT)

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      INTEGER * 2 PAGE,ROW,COLUMN,SIZE,WRATT
      CHARACTER * 1 BUF(SIZE)
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), P(4)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO

      CALL LOCATE(PAGE,ROW,COLUMN)

      DO 10 J = 1,LENG
         CALL WRCHAT(PAGE,BUF(J),WRATT,1)
10    CONTINUE

      RETURN
      END

C     SUBROUTINE ID

C     THIS SUBROUTINE IDENTIFIES THE CHARACTER TYPED IN BY THE USER.
C     VALID CHARACTERS ARE ALL ALPHABETS, NUMBERS, THE BACKSPACE KEY,
C     THE ENTER KEY AND SOME SPECIAL CHARACTERS.

C     PARAMETER DEFINITION:

C          KYCOD1   -      ASCII CODE OF KEY
C          KYCOD2   -      EXTENDED CHARACTER CODE OF KEY
C          CHAR     -      CHARACTER IDENTIFIED
C          J        -      COLUMN POSITION
C          ALPHA    -      ARRAY OF ALPHABETS
C          NUMMER   -      ARRAY OF NUMBERS

      SUBROUTINE ID(KYCOD1,KYCOD2,CHAR,J,ALPHA,NUMER)

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      CHARACTER * 1 CHAR,ALPHA(26),NUMER(10)
      INTEGER * 2 KYCOD1,KYCOD2,J
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), P(4)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO

      IF (KYCOD1 .GE. 48 .AND. KYCOD1 .LE. 57) THEN
         CHAR = NUMER(KYCOD1-47)
      ELSEIF (KYCOD1 .GE. 65 .AND. KYCOD1 .LE. 90) THEN
         CHAR = ALPHA(KYCOD1-64)
      ELSEIF (KYCOD1 .GE. 97 .AND. KYCOD1 .LE. 122) THEN
         CHAR = ALPHA(KYCOD1-96)
```

```
      ELSEIF (KYCOD1 .EQ. 29) THEN
         J = J - 1
      ELSEIF (KYCOD1 .EQ. 8 .AND. KYCOD2 .EQ. 14) THEN
         J = J - 1
      ELSEIF (KYCOD1 .EQ. 127 .AND. KYCOD2 .EQ. 14) THEN
         J = J - 1
      ELSEIF (KYCOD1 .EQ. 0 .AND. KYCOD2 .EQ. 75) THEN
         J = J - 1
         CHAR = '!'
      ELSEIF (KYCOD1 .EQ. 64) THEN
         CHAR = 'ə'
      ELSEIF (KYCOD1 .EQ. 35) THEN
         CHAR = '#'
      ELSEIF (KYCOD1 .EQ. 36) THEN
         CHAR = '$'
      ELSEIF (KYCOD1 .EQ. 37) THEN
         CHAR = '%'
      ELSEIF (KYCOD1 .EQ. 94) THEN
         CHAR = '¢'
      ELSEIF (KYCOD1 .EQ. 38) THEN
         CHAR = '&'
      ELSEIF (KYCOD1 .EQ. 40) THEN
         CHAR = '('
      ELSEIF (KYCOD1 .EQ. 41) THEN
         CHAR = ')'
      ELSEIF (KYCOD1 .EQ. 95) THEN
         CHAR = '_'
      ELSEIF (KYCOD1 .EQ. 45) THEN
         CHAR = '-'
      ELSEIF (KYCOD1 .EQ. 123) THEN
         CHAR = '{'
      ELSEIF (KYCOD1 .EQ. 125) THEN
         CHAR = '}'
      ELSEIF (KYCOD1 .EQ. 39) THEN
         CHAR = ''''
      ELSEIF (KYCOD1 .EQ. 58) THEN
         CHAR = ':'
      ELSEIF (KYCOD1 .EQ. 47) THEN
         CHAR = '/'
      ELSEIF (KYCOD1 .EQ. 92) THEN
         CHAR = '\'
      ELSE
         J = 0
      ENDIF

      RETURN
      END

C     SUBROUTINE PROG

C     THIS SUBROUTINE INITIALIZES ALL VARIABLES AND READS THE AML/E
C     PROGRAM ONE LINE AT A TIME, AND STORES IT INTO THE CHARACTER
C     ARRAY C. VARIBALE I IS USED AS A POINTER INTO THIS ARRAY.

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             COMTYP,FORMAL,LABELS,NEXT3,SUBEND,SYMTYP.

      SUBROUTINE PROG

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), P(4)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
```

```
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO

      LINENO = 0
      NUM = 1
      NUM1 = 1
      NUM2 = 1
      NUMER = 0
      STNO = 0
      SUBRNO = 0
      NSUB = 1
      J1 = 0
      J2 = 0
      J3 = 0
      J4 = 0
      K1 = 0
      K2 = 0
      K3 = 0
      K4 = 0
      K5 = 0
      C5 = 0.0
      P(1) = 0.0
      P(2) = 0.0
      P(3) = 0.0
      P(4) = 0.0

      DO 4 J = 1,100
         SUBOP(J) = 0
         VARNO(J) = 0
         LENGTH(J) = 0
         LENCON(J) = 0
4     CONTINUE

      DO 5 J = 1,100
         UNVAR(J) = 0
5     CONTINUE

10    LINENO = LINENO + 1

C     READ THE AML/E PROGRAM LINE, AND WRITE TO THE OUTPUT FILE WITH
C     THE LINE NUMBER.

      READ (1,100,END=1000) C
      WRITE (2,200) LINENO, C

100   FORMAT(A72)
200   FORMAT(1X,I3,4X,A72)

C     CHECK FOR BLANK LINE

      I = 0
20    I = I + 1
      IF (I .GT. 72) GOTO 10
      IF (C(I:I) .EQ. ' ') GOTO 20

C     CHECK FOR COMMENT LINE

      IF (C(I:I) .EQ. '-') GOTO 10

23    INDEX1 = I
      INDEX2 = I
      SYMBOL(NUM)(1:1) = C(I:I)
34    I = I + 1
      IF (I .GT. 72) GOTO 10
      INDEX2 = INDEX2 + 1

C     CHECK NEXT CHARACTER FOR DELIMITERS
```

```
      DO 35 J = 1,10

         IF (C(I:I) .EQ. DELIM(J)) THEN
            LENGTH(NUM) = INDEX2 - INDEX1
            SYMBOL(NUM)(1:LENGTH(NUM)) = C(INDEX1:INDEX2 - 1)
            GOTO 37
         ENDIF

35    CONTINUE

C     NEXT CHARACTER IS NOT A DELIMITER. CONCATENATE IT TO THE PARTIALLY
C     CONSTRUCTED SYMBOL AND THEN READ THE NEXT CHARACTER

      SYMBOL(NUM)(1:INDEX2 - INDEX1 + 1) = SYMBOL(NUM)(1:INDEX2 - INDEX1
     *) // C(I:I)
      GOTO 34

C     CHECK TO SEE IF SYMBOL IS AN AML KEYWORD. IF THE KEYWORD IS 'SUBR'
C     ALSO CHECK FOR FORMAL PARAMETERS. IF SYMBOL IS ONE OF THE FIRST
C     34 KEYWORDS(EXECUTABLE STATEMENTS) DETERMINE WHAT TYPE OF
C     COMMAND IT IS AND PRINT IT OUT IN THE OBJECT FILE. ALSO,
C     INCREMENT THE STATEMENT NUMBER

37    VARNO(NUM) = SUBRNO
      DO 36 J = 1,34

         IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. AMLWRD(J)) THEN
            STNO = STNO + 1
            CALL COMTYP(J)
            GOTO 50
         ENDIF

36    CONTINUE

      DO 38 J = 35,42

         IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. AMLWRD(J) .AND. J .NE. 42)
     *   GOTO 50
         IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. 'SUBR') THEN
            CALL FORMAL
            GOTO 50
         ENDIF

38    CONTINUE

C     CHECK TO SEE IF SYMBOL IS THE BEGINNING OF COMMENT OR A DELIMITER

      DO 39 J = 1,10

         IF (SYMBOL(NUM)(1:1) .EQ. '-' .AND. C(INDEX2:INDEX2) .EQ. '-')
     *   GOTO 10
         IF (SYMBOL(NUM)(1:1) .EQ. DELIM(J) .AND. J .NE. 5) GOTO 50

39    CONTINUE

C     CHECK TO SEE IF SYMBOL IS A NUMERIC CONSTANT. IF IT IS PRINT OUT
C     THE CONSTANT NUMBER, THE LINE NUMBER AND THE VALUE OF THE CONSTANT
C     CONVERT INTEGER NUMBERS TO REAL NUMBERS BY ADDING A DECIMAL POINT

      DO 41 J = 27,36
         IF (SYMBOL(NUM)(1:1) .EQ. CHARS(J)) NUMER = 1
41    CONTINUE

      IF (SYMBOL(NUM)(1:1) .EQ. '-' .OR. NUMER .EQ. 1) THEN
         CONS(NUM1)(1:INDEX2 - INDEX1 + 1) = SYMBOL(NUM)(1:INDEX2 - INDE
     *   X1 + 1)

         DO 40 JJ = 1,LENGTH(NUM)
```

```
            IF (CONS(NUM1)(JJ:JJ) .EQ. '.') GOTO 95
40        CONTINUE

          LENGTH(NUM) = LENGTH(NUM) + 1
          CONS(NUM1)(LENGTH(NUM):LENGTH(NUM)) = '.'
95        LENCON(NUM1) = LENGTH(NUM)
          WRITE(4,49) NUM1, LINENO, STNO, LENCON(NUM1),(CONS2(K,NUM1),K=1
     *,LENCON(NUM1))
          NUM1 = NUM1 + 1
          NUMER = 0
          GOTO 50

        ENDIF

C       THE SYMBOL IS NOT A AML WORD OR A DELIMITER OR A NUMBER
C       PRINT OUT THE SYMBOL, LINE AND STATEMENT NUMBERS AND THE SYMBOL
C       AFTER CHECKING TO SEE IF THE SYMBOL HAS ALREADY BEEN IDENTIFIED
C       IF THE SYMBOL IS A CALL TO A SUBROUTINE, INCREMENT STATEMENT NO.

        DO 42 J = 1, NSUB - 1

           IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. SUB(J)) THEN
              STNO = STNO + 1
              NTYPE = 7
              MTYPE = SUBNO(J)
              CALL NEXT3(NTYPE,MTYPE)
              GOTO 50
           ENDIF

42      CONTINUE

        CALL SYMTYP
        IF (ITYPE .EQ. 0) GOTO 50

        DO 47 J = 1, NUM - 1

         IF (VARNO(J) .EQ. 0) THEN
           IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. SYMBOL(J)(1:LENGTH(J)) .AND
     *    . LENGTH(NUM) .EQ. LENGTH(J)) GOTO 50
          ELSE
           IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. SYMBOL(J)(1:LENGTH(J)) .AND
     *    . LENGTH(NUM) .EQ. LENGTH(J) .AND. VARNO(NUM) .EQ. VARNO(J))
     *    GOTO 50
          ENDIF

47      CONTINUE

C       IF THE SYMBOL IS A LABEL, INCREMENT AND PRINT STATEMENT NUMBER
C       IF THE SYMBOL IS A SUBROUTINE NAME, INCREMENT THE NUMBER FOR THE
C       SUBROUTINE AND THE STATEMENT NUMBER

        IF (ITYPE .EQ. 1) THEN
           STNO = STNO + 1
           NTYPE = 0
           MTYPE = 0
           WRITE (7,59,REC=STNO) STNO,LINENO,NTYPE,MTYPE
        ELSEIF (ITYPE .EQ. 6) THEN
           SUB(NSUB) = SYMBOL(NUM)(1:LENGTH(NUM))
           STNO = STNO + 1
           NTYPE = 6
           MTYPE = 1
           SUBNO(NSUB)= STNO
           SUBRNO = NSUB
           NSUB = NSUB + 1
           WRITE (7,59,REC=STNO) STNO,LINENO,NTYPE,MTYPE
59         FORMAT(1X,I4,1X,I4,1X,I1,1X,I4)
        ENDIF

        IF (ITYPE .EQ. 5 .OR. ITYPE .EQ. 9) GOTO 101
```

```
      VARNO(NUM) = SUBRNO
      WRITE (3,46) NUM,LINENO,STNO,LENGTH(NUM),ITYPE,VARNO(NUM),
     *(SYM2(K,NUM),K =1,LENGTH(NUM))

      NUM = NUM + 1
      VARNO(NUM) = SUBRNO
101   DO 92 J = 1,NUM2-1

         IF (SYMBOL(NUM-1)(1:LENGTH(NUM-1)) .EQ. UNRES(J)(1:LENRES(J))
     *   .AND. LENGTH(NUM-1) .EQ. LENRES(J) .AND. VARNO(NUM-1) .EQ.
     *   UNVAR(J)) CALL LABELS(J)

92    CONTINUE

46    FORMAT(1X,I3,2X,I3,2X,I4,2X,I3,2X,I1,2X,I2,2X,72A1)
49    FORMAT(1X,I3,2X,I3,2X,I4,2X,I3,2X,72A1)

50    I = I + 1
      IF (I .GT. 72) GOTO 10

C     CHECK NEXT CHARACTER FOR DELIMITERS

      DO 67 J = 1,10
         IF (C(I:I) .EQ. DELIM(J) .AND. J .NE. 5) GOTO 50
67    CONTINUE

      GOTO 23

C     PAIR ALL "SUBR" AND "END" STATEMENTS

1000  CALL SUBEND

      REWIND (3)
      REWIND (4)

C     READ THE VARIABLES FILE AND THE CONSTANTS FILE AND GENERATE THE
C     THE SYMBOL TABLE FILE.

      DO 90 MM = 1,NUM-1
99       READ (3,46) K1,K2,K3,K4,K5

         IF (K5 .EQ. 2) THEN
74          READ(4,69) J1,J2,J3,J4,C5
            IF (K2 .EQ. J2 .AND. K3 .EQ. J3) THEN
               WRITE (8,71,REC=K1) K1,C5
            ELSE
               GOTO 74
            ENDIF

         ELSEIF (K5 .EQ. 3 .OR. K5 .EQ. 8) THEN
75          READ(4,69) J1,J2,J3,J4,P(1)
            IF (K2 .EQ. J2 .AND. K3 .EQ. J3) THEN

               DO 76 J = 2,4
                  READ(4,69) J1,J2,J3,J4,P(J)
76             CONTINUE

               WRITE (8,77,REC=K1) K1, (P(J),J=1,4)
            ELSE
               GOTO 75
            ENDIF

         ELSEIF (K5 .EQ. 5 .OR. K5 .EQ. 9) THEN
            GOTO 90
         ELSE
            WRITE (8,71,REC=K1) K1
         ENDIF

90    CONTINUE
```

```
69    FORMAT (1X,I3,2X,I3,2X,I4,2X,I3,2X,F8.2)
71    FORMAT (1X,I3,2X,F8.2)
77    FORMAT (1X,I3,4(2X,F8.2))

      RETURN
      END

C     BLOCK DATA

C     THIS BLOCK DEFINES ALL THE DELIMITERS, ALPHABETIC AND NUMERIC
C     DATA. THE AML KEYWORDS ARE ALSO DEFINED HERE.

      BLOCK DATA

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), P(4)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO

      DATA DELIM/' ',';',':',',','-','(',')','<','>','='/

      DATA CHARS/'A','B','C','D','E','F','G','H','I','J','K','L','M','N'
     *,'O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','
     *4','5','6','7','8','9',' ',';',':',',','-','(',')','<','>','_','.'
     *,'+','''','='/

      DATA AMLWRD/'BRANCH','BREAKPOINT','COMPC','CSTATUS','DECR','DELAY'
     *,'DPMOVE','END','GET','GETPART','GRASP','GUARDI','INCR','ITERATE',
     *'LINEAR','MSTATUS','NEXTPART','NOGUARD','PAYLOAD','PREVPART','PMOV
     *E','PUT','RELEASE','SETC','SETPART','TESTC','TESTI','TESTP','WAITI
     *','WHERE','WRITEO','XMOVE','ZMOVE','ZONE','COUNTER','GROUP','NEW',
     *'PALLET','PT','REGION','STATIC','SUBR'/

      END

C     SUBROUTINE SYMTYP

C     THIS SUBROUTINE IS USED TO CLASSIFY THE SYMBOL JUST IDENTIFIED.

C     VALUE FOR ITYPE ARE:

C        UNDEFINED                      - 0
C        LABELS                         - 1
C        CONSTANTS                      - 2
C        POINTS DEFINED WITH NUMBERS    - 3
C        COUNTERS                       - 4
C        PALLETS                        - 5
C        SUBROUTINE NAMES               - 6
C        FORMAL PARAMETERS              - 7
C        AGGREGATES                     - 8
C        POINTS DEFINED WITH COUNTERS   - 9

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             COUNT, INCREM.

      SUBROUTINE SYMTYP

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
```

```
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), P(4)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO

      II = I - 1
10    II = II + 1

      IF (II .GT. 72) THEN
         ITYPE = 0
         RETURN
      ENDIF

      IF (C(II:II) .EQ. ' ') GOTO 10

      IF (C(II:II) .NE. ':') THEN
         ITYPE = 0
         RETURN
      ENDIF

20    II = II + 1

      IF (II .GT. 72) THEN
         ITYPE = 1
         RETURN
      ENDIF

      IF (C(II:II) .EQ. ' ') GOTO 20

      IF (C(II:II+1) .EQ. '--') THEN
         ITYPE=1
         RETURN
      ENDIF

      IF (C(II:II+3) .EQ. 'NEW ') THEN

C     BECAUSE OF ERROR IN COMPILING, CALLING SUBROUTINE

         CALL INCREM(II,3)

30       II = II + 1

         IF (II .GT. 72) THEN
            ITYPE = 0
         ENDIF

         IF (C(II:II) .EQ. ' ') GOTO 30

         IF (C(II:II+2) .EQ. 'PT ' .OR. C(II:II+2) .EQ. 'PT(') THEN
            II = II + 2
35          II = II + 1
            IF (C(II:II) .EQ. ' ' .OR. C(II:II) .EQ. '(') GOTO 35
            NUMER = 0

            DO 37 J = 27,36
               IF(C(II:II) .EQ. CHARS(J)) NUMER = 1
37          CONTINUE

            IF (C(II:II) .EQ. '-' .OR. NUMER .EQ. 1) THEN
               ITYPE = 3
               NUMER = 0
            ELSE
```

```
               ITYPE = 9
               NUM = NUM + 1
               VARNO(NUM) = SUBRNO
               CALL COUNT(II)
            ENDIF

         ELSEIF (C(II:II) .EQ. '<') THEN
            ITYPE = 8
         ELSE
            ITYPE = 2
         ENDIF

      ELSEIF (C(II:II+6) .EQ. 'STATIC ') THEN

C     BECAUSE OF ERROR IN COMPILING, CALLING SUBROUTINE

         CALL INCREM(II,6)

40       II = II + 1

         IF (II .GT. 72) THEN
            ITYPE = 0
         ENDIF

         IF (C(II:II) .EQ. ' ') GOTO 40
         IF (C(II:II+7) .EQ. 'COUNTER ' .OR. C(II:II+7) .EQ. 'COUNTER;')
     *   THEN
            ITYPE = 4
         ELSEIF (C(II:II+6) .EQ. 'PALLET ' .OR. C(II:II+6) .EQ. 'PALLET(
     *') THEN
            ITYPE = 5
            II = II + 6
            NUM = NUM + 1
            VARNO(NUM) = SUBRNO
            CALL COUNT(II)
         ELSE
            ITYPE = 0
         ENDIF

      ELSEIF (C(II:II+4) .EQ. 'SUBR ' .OR. C(II:II+4) .EQ. 'SUBR(' .OR.
     *        C(II:II+4) .EQ. 'SUBR;') THEN
         ITYPE = 6
      ELSE
         ITYPE = 1
      ENDIF

      RETURN
      END

C     SUBROUTINE FORMAL

C     SUBROUTINE TO CHECK FOR FORMAL PARAMETERS IN A SUBROUTINE AND
C     PRINT THEM OUT IF THEY EXIST

      SUBROUTINE FORMAL

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), P(4)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO
```

```
      II = I - 1
10    II = II + 1

      IF (II .GT. 72) THEN
         ITYPE = 0
         RETURN
      ENDIF

      IF (C(II:II) .EQ. ' ') GOTO 10

      IF (C(II:II) .EQ. ';') THEN
         ITYPE = 0
         RETURN
      ENDIF

20    II = II + 1

      IF (II .GT. 72) THEN
         ITYPE = 0
         RETURN
      ENDIF

      IF (C(II:II) .EQ. ' ') GOTO 20

23    INDEX1 = II
      INDEX2 = II
      SYMBOL(NUM)(1:1) = C(II:II)
34    II = II + 1

      IF (II .GT. 72) THEN
         ITYPE = 0
         RETURN
      ENDIF

      INDEX2 = INDEX2 + 1

C     CHECK NEXT CHARACTER FOR END OF FORMAL PARAMETER

      IF (C(II:II) .EQ. ')' .OR. C(II:II) .EQ. ',' .OR. C(II:II) .EQ.
     *    ' ') THEN
         LENGTH(NUM) = INDEX2 - INDEX1
         SYMBOL(NUM)(1:LENGTH(NUM)) = C(INDEX1:INDEX2 - 1)
         ITYPE = 7
         GOTO 37
      ENDIF

C     NEXT CHARACTER IS NOT A DELIMITER. CONCATENATE IT TO THE PARTIALLY
C     CONSTRUCTED SYMBOL AND THEN READ THE NEXT CHARACTER

      SYMBOL(NUM)(1:INDEX2 - INDEX1 + 1) = SYMBOL(NUM)(1:INDEX2 - INDEX1
     *) // C(II:II)
      GOTO 34

C     WRITE THE FORMAL PARAMETER INTO THE VARIABLES FILE.

37    VARNO(NUM) = SUBRNO
      WRITE (3,46) NUM,LINENO,STNO,LENGTH(NUM),ITYPE,VARNO(NUM),
     *(SYM2(K,NUM),K =1,LENGTH(NUM))

46    FORMAT(1X,I3,2X,I3,2X,I4,2X,I3,2X,I1,2X,I2,2X,72A1)

      NUM = NUM + 1
      VARNO(NUM) = SUBRNO
      IF (C(II:II) .EQ. ',') GOTO 20
      I = II

      RETURN
      END
```

```
C     SUBROUTINE COMTYP

C     SUBROUTINE TO CLASSIFY TYPE OF COMMAND AND WRITE TO OBJECT FILE.
C     THE VALUES OF NTYPE AND MTYPE ARE ESTABLISHED BASED ON THE
C     COMMAND TYPE AND THE SUBTYPE OF THE COMMAND.
C     DEPENDING ON THE COMMAND SUBROUTINE NEXT1 OR NEXT3 IS CALLED
C     TO READ ALL THE COMMAND PARAMETERS, IF ANY. THE VALUES OF
C     THE STATEMENT NUMBER, LINE NUMBER, NTYPE AND MTYPE ARE
C     WRITTEN TO THE OBJECT FILE.

C     PARAMETER DEFINITION:

C          J         -       POINTER TO THE AML/E COMMAND WORD

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             NEXT1,NEXT3.

      SUBROUTINE COMTYP(J)

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), P(4)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO

      IF (J .LE. 17) THEN
         GOTO 10
      ELSE
         GOTO 20
      ENDIF

C     "BRANCH"

10    IF (J .EQ. 1) THEN
         NTYPE = 3
         MTYPE = 1
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "BREAKPOINT"

      ELSEIF (J .EQ. 2) THEN
         NTYPE = 3
         MTYPE = 5

C     "COMPC"

      ELSEIF (J .EQ. 3) THEN
         NTYPE = 4
         MTYPE = 1
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "CSTATUS"

      ELSEIF (J .EQ. 4) THEN
         NTYPE = 0
         MTYPE = 0

C     "DECR"
```

```
      ELSEIF (J .EQ. 5) THEN
         NTYPE = 4
         MTYPE = 2
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "DELAY"

      ELSEIF (J .EQ. 6) THEN
         NTYPE = 1
         MTYPE = 1
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "DPMOVE"

      ELSEIF (J .EQ. 7) THEN
         NTYPE = 1
         MTYPE = 2
         CALL NEXT1(NTYPE,MTYPE)
         RETURN

C     "END"

      ELSEIF (J .EQ. 8) THEN

         NTYPE = 6
         MTYPE = 2

         DO 15 JJ = NSUB-1,1,-1

            IF (SUBOP(JJ) .EQ. 0) THEN
               SUBOP(JJ) = 1
               KTYPE = SUBNO(JJ)
               WRITE (7,100,REC=STNO) STNO,LINENO,NTYPE,MTYPE,KTYPE

               DO 16 JJJ = JJ-1,1,-1

                  IF (SUBOP(JJJ) .EQ. 0) THEN
                     SUBRNO = JJJ
                     RETURN
                  ENDIF

16             CONTINUE

               RETURN

            ENDIF

15       CONTINUE

C     "GET"

      ELSEIF (J .EQ. 9) THEN
         NTYPE = 0
         MTYPE = 0

C     "GETPART"

      ELSEIF (J .EQ. 10) THEN
         NTYPE = 5
         MTYPE = 1
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "GRASP"

      ELSEIF (J .EQ. 11) THEN
```

```
            NTYPE = 1
            MTYPE = 3

C       "GUARDI"

        ELSEIF (J .EQ. 12) THEN
            NTYPE = 0
            MTYPE = 0

C       "INCR"

        ELSEIF (J .EQ. 13) THEN
            NTYPE = 4
            MTYPE = 3
            CALL NEXT3(NTYPE,MTYPE)
            RETURN

C       "ITERATE"

        ELSEIF (J .EQ. 14) THEN
            NTYPE = 3
            MTYPE = 6

C       "LINEAR"

        ELSEIF (J .EQ. 15) THEN
            NTYPE = 1
            MTYPE = 7
            CALL NEXT3(NTYPE,MTYPE)
            RETURN

C       "MSTATUS"

        ELSEIF (J .EQ. 16) THEN
            NTYPE = 0
            MTYPE = 0

C       "NEXTPART"

        ELSEIF (J .EQ. 17) THEN
            NTYPE = 5
            MTYPE = 2
            CALL NEXT3(NTYPE,MTYPE)
            RETURN
        ENDIF

C       "NOGUARD"

20      IF (J .EQ. 18) THEN
            NTYPE = 0
            MTYPE = 0

C       "PAYLOAD"

        ELSEIF (J .EQ. 19) THEN
            NTYPE = 1
            MTYPE = 8
            CALL NEXT3(NTYPE,MTYPE)
            RETURN

C       "PREVPART"

        ELSEIF (J .EQ. 20) THEN
            NTYPE = 5
            MTYPE = 3
            CALL NEXT3(NTYPE,MTYPE)
            RETURN

C       "PMOVE"
```

```
      ELSEIF (J .EQ. 21) THEN
         NTYPE = 1
         MTYPE = 4
         CALL NEXT1(NTYPE,MTYPE)
         RETURN

C     "PUT"

      ELSEIF (J .EQ. 22) THEN
         NTYPE = 0
         MTYPE = 0

C     "RELEASE"

      ELSEIF (J .EQ. 23) THEN
         NTYPE = 1
         MTYPE = 5

C     "SETC"

      ELSEIF (J .EQ. 24) THEN
         NTYPE = 4
         MTYPE = 4
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "SETPART"

      ELSEIF (J .EQ. 25) THEN
         NTYPE = 5
         MTYPE = 4
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "TESTC"

      ELSEIF (J .EQ. 26) THEN
         NTYPE = 3
         MTYPE = 2
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "TESTI"

      ELSEIF (J .EQ. 27) THEN
         NTYPE = 3
         MTYPE = 3
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "TESTP"

      ELSEIF (J .EQ. 28) THEN
         NTYPE = 3
         MTYPE = 4
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "WAITI"

      ELSEIF (J .EQ. 29) THEN
         NTYPE = 2
         MTYPE = 1
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "WHERE"
```

```
      ELSEIF (J .EQ. 30) THEN
         NTYPE = 8
         MTYPE = 1
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "WRITEO"

      ELSEIF (J .EQ. 31) THEN
         NTYPE = 2
         MTYPE = 2
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "XMOVE"

      ELSEIF (J .EQ. 32) THEN
         NTYPE = 0
         MTYPE = 0

C     "ZMOVE"

      ELSEIF (J .EQ. 33) THEN
         NTYPE = 1
         MTYPE = 6
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

C     "ZONE"

      ELSEIF (J .EQ. 34) THEN

         NTYPE = 1
         MTYPE = 9
         CALL NEXT3(NTYPE,MTYPE)
         RETURN

      ENDIF

      WRITE (7,100,REC=STNO) STNO,LINENO,NTYPE,MTYPE
100   FORMAT(1X,I4,1X,I4,1X,I1,1X,I4,1X,I4)

      RETURN
      END

C     SUBROUTINE SUBEND

C     SUBROUTINE TO DETERMINE FIRST EXECUTABLE STATEMENT IN EACH
C     SUBROUTINE AND TO PAIR EACH "SUBR" AND "END" STATEMENT

      SUBROUTINE SUBEND

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), P(4)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO


C     READ THE LAST STATEMENT ("END") FOR THE STNO OF FIRST "SUBR".
C     THEN READ BACKWARDS TILL THE PREVIOUS "END" STATEMENT SHOWS UP.
```

```
C     THE FIRST EXECUTABLE STATEMENT FOR THIS SUBR IS THE STATEMENT
C     AFTER THE PREVIOUS "END". WRITE THE STNO OF THE "END" STATEMENT
C     AND THE FIRST EXECUTABLE STNO IN THE RECORD OF THE "SUBR" STNO.
C     CONTINUE THIS PROCESS TILL ALL THE SUBROUTINES ARE COMPLETED.

      NUMEND = STNO
      SUBNUM = 1

10    READ (7,100,REC=NUMEND) NUMSUB
      IF (SUBNUM .EQ. NSUB) GOTO 30

      DO 20 J = NUMEND-1,NUMSUB+3,-1
         READ (7,200,REC=J) NTYPE,MTYPE

         IF (NTYPE .EQ. 6 .AND. MTYPE .EQ. 2) THEN
            NUMBEG = J + 1
            READ (7,300,REC=NUMSUB) K1,K2,K3,K4
            WRITE (7,300,REC=NUMSUB) K1,K2,K3,K4,NUMEND,NUMBEG
            NUMEND = J
            SUBNUM = SUBNUM + 1
            GOTO 10
         ENDIF

20    CONTINUE

C     THE FINAL "SUBR" AND ANY NON-IMBEDDED "SUBR" IS COMPLETED HERE.

30    NUMBEG = NUMSUB + 1
      READ (7,300,REC=NUMSUB) K1,K2,K3,K4
      WRITE (7,300,REC=NUMSUB) K1,K2,K3,K4,NUMEND,NUMBEG
      SUBNUM = SUBNUM + 1
      NUMEND = NUMSUB

      IF (SUBNUM .LT. NSUB) THEN

35       NUMEND = NUMEND - 1
         READ (7,200,REC=NUMEND) K1,K2

         IF (K1 .EQ. 6 .AND. K2 .EQ. 2) THEN
            GOTO 10
         ELSE
            GOTO 35
         ENDIF

      ELSE

         RETURN

      ENDIF

100   FORMAT(18X,I4)
200   FORMAT(11X,I1,1X,I4)
300   FORMAT(1X,I4,1X,I4,1X,I4,1X,I1,1X,I4,1X,I4,1X,I4,1X,I4)

      RETURN
      END

C     SUBROUTINE NEXT1

C     SUBROUTINE TO CHECK IF NEXT SYMBOL IS A LABEL OR A SERIES OF
C     NUMBERS AND  WRITE TO THE OBJECT FILE AND, IF NECESSARY TO THE
C     CONSTANTS FILE. THIS SUBROUTINE IS CALLED IF THE COMMAND IS
C     "PMOVE" OR "DPMOVE".

C     PARAMETER DEFINITION:

C          NTYPE     -      COMMAND TYPE
C          MTYPE     -      SUBTYPE OF COMMAND
```

```
      SUBROUTINE NEXT1(NTYPE,MTYPE)

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), MM(4)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO

      NUMER = 0
      KKK = 0
      KTYPE = 1
10    I = I + 1
      IF (I .GT. 72) RETURN

C     CHECK IF THE FIRST CHARACTER IS A DELIMITER

      DO 15 J = 1,10
         IF (C(I:I) .EQ. DELIM(J) .AND. J .NE. 8 .AND. J .NE. 5) GOTO 10
15    CONTINUE

      IF (C(I:I) .EQ. '<') GOTO 55
      IF (C(I:I+2) .EQ. 'PT ' .OR. C(I:I+2) .EQ. 'PT(') GOTO 56

23    INDEX1 = I
      INDEX2 = I
      SYMBOL(NUM)(1:1) = C(I:I)
34    I = I + 1
      IF (I .GT. 72) RETURN
      INDEX2 = INDEX2 + 1

C     CHECK NEXT CHARACTER FOR DELIMITERS

      DO 35 J = 1,10

         IF (C(I:I) .EQ. DELIM(J) .AND. J .NE. 5) THEN
            LENGTH(NUM) = INDEX2 - INDEX1
            SYMBOL(NUM)(1:LENGTH(NUM)) = C(INDEX1:INDEX2 - 1)
            GOTO 37
         ENDIF

35    CONTINUE

C     NEXT CHARACTER IS NOT A DELIMITER. CONCATENATE IT TO THE PARTIALLY
C     CONSTRUCTED SYMBOL AND THEN READ THE NEXT CHARACTER

      SYMBOL(NUM)(1:INDEX2 - INDEX1 + 1) = SYMBOL(NUM)(1:INDEX2 - INDEX1
     *) // C(I:I)
      GOTO 34

C     FIND THE STNO OF THE SYMBOL AND WRITE TO THE OBJECT FILE.

37    IF (KTYPE .GT. 1) GOTO 57

      DO 47 J = 1, NUM - 1

       IF (VARNO(J) .EQ. 0) THEN

         IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. SYMBOL(J)(1:LENGTH(J)) .AND
     *    . LENGTH(NUM) .EQ. LENGTH(J)) GOTO 48

       ELSE
```

```
         IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. SYMBOL(J)(1:LENGTH(J)) .AND
     *    . LENGTH(NUM) .EQ. LENGTH(J) .AND. VARNO(NUM) .EQ. VARNO(J))
     *    GOTO 48

         ENDIF

         GOTO 47
48       KTYPE = 1
         WRITE (7,46,REC=STNO) STNO,LINENO,NTYPE,MTYPE,KTYPE,J
         RETURN

47       CONTINUE

55       I = I + 1
         IF (I .GT. 72) RETURN
         IF (C(I:I) .EQ. ' ') GOTO 55
         KTYPE = 2
         GOTO 23

56       I = I + 2
99       I = I + 1
         IF (I .GT. 72) RETURN
         IF (C(I:I) .EQ. ' ' .OR. C(I:I) .EQ. '(') GOTO 99
         KTYPE = 2
         GOTO 23

57       NUMER = 0
         DO 58 J = 27,36
            IF (SYMBOL(NUM)(1:1) .EQ. CHARS(J)) NUMER = 1
58       CONTINUE

C        VALUES ARE ALL NUMBERS. WRITE TO THE OBJECT FILE AND CONSTANTS
C        FILE.

         IF (SYMBOL(NUM)(1:1) .EQ. '-' .OR. NUMER .EQ. 1) THEN
            NUMER  = 0
            KTYPE = 2
            KKK = KKK + 1
            CONS(NUM1)(1:INDEX2 - INDEX1 + 1) = SYMBOL(NUM)(1:INDEX2 - INDE
     *      X1 + 1)
            DO 40 JJ = 1,LENGTH(NUM)
               IF (CONS(NUM1)(JJ:JJ) .EQ. '.') GOTO 95
40          CONTINUE
            LENGTH(NUM) = LENGTH(NUM) + 1
            CONS(NUM1)(LENGTH(NUM):LENGTH(NUM)) = '.'
95          LENCON(NUM1) = LENGTH(NUM)
            WRITE(4,49) NUM1, LINENO, STNO, LENCON(NUM1),(CONS2(K,NUM1),K=1
     *,LENCON(NUM1))

            IF (KKK .LT. 4) THEN
               NUM1 = NUM1 + 1
               NUMER = 0
               GOTO 55
            ENDIF

            WRITE (7,46,REC=STNO) STNO,LINENO,NTYPE,MTYPE,KTYPE,NUM1-3
            NUM1 = NUM1 + 1
            NUMER = 0
            RETURN

         ENDIF

C        VAULES ARE SPECIFIED AS COUNTERS/LABELS. WRITE THE FOUR SYMBOL
C        NUMBERS INTO THE OBJECT FILE.

         DO 78 J = 1, NUM - 1

          IF (VARNO(J) .EQ. 0) THEN
```

```
         IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. SYMBOL(J)(1:LENGTH(J)) .AND
     *    . LENGTH(NUM) .EQ. LENGTH(J)) GOTO 79

         ELSE

         IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. SYMBOL(J)(1:LENGTH(J)) .AND
     *    . LENGTH(NUM) .EQ. LENGTH(J) .AND. VARNO(NUM) .EQ. VARNO(J))
     *      GOTO 79

         ENDIF

         GOTO 78

79           KTYPE = 3
             KKK = KKK + 1
             MM(KKK) = J

             IF (KKK .LT. 4) THEN
                GOTO 55
             ENDIF

             WRITE (7,46,REC=STNO) STNO,LINENO,NTYPE,MTYPE,KTYPE,(MM(K),
     *       K = 1,KKK)
             RETURN

78       CONTINUE

46       FORMAT(1X,I4,1X,I4,1X,I1,1X,I4,1X,I4,1X,I4,1X,I4,1X,I4,1X,I4,1X,I4)
49       FORMAT(1X,I3,2X,I3,2X,I4,2X,I3,2X,72A1)

50       RETURN
         END

C        SUBROUTINE NEXT3

C        SUBROUTINE TO READ THE THREE PARAMETERS IN THE FOLLOWING
C        COMMANDS - "TESTC", "TESTI" AND "TESTP" AND THE FOUR PARAMETERS
C        IN THE "WAITI" AND "COMPC" COMMANDS AND THE TWO PARAMETERS IN THE
C        COMMANDS - "SETC", "SETPART" AND "WRITEO", AND WRITE TO THE
C        OBJECT FILE. THIS SUBROUTINE ALSO READS ALL THE FORMAL PARAMETERS
C        PASSED WHEN A CALL TO A SUBROUTINE IS INVOKED.IT IS ASSUMED THAT
C        THERE ARE A MAXIMUM OF FIVE FORMAL PARAMETERS.
C        THIS SUBROUTINE IS ALSO CALLED IF THE COMMAND IS "ZMOVE",
C        "DELAY", "LINEAR", "PAYLOAD", "ZONE", "DECR", "GETPART", "INCR",
C        "NEXTPART", OR "PREVPART". IT IS ALSO USED FOR
C        "BRANCH", BUT IF THE LABEL HAS NOT BEEN DEFINED, IT IS PLACED
C        AS AN UNRESOLVED LABEL.

C        PARAMETER DEFINITION:

C             NTYPE     -       COMMAND TYPE
C             MTYPE     -       SUBTYPE OF COMMAND

C        SUBROUTINES CALLED BY THIS PROGRAM ARE :
C                INCREM.

         SUBROUTINE NEXT3(NTYPE,MTYPE)

         CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
         CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
         CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
         INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
         INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
         DIMENSION LENGTH(100),LENCON(100), LENRES(100), MM(20)
         EQUIVALENCE (SYMBOL(1),SYM2(1,1))
         EQUIVALENCE (CONS(1),CONS2(1,1))

         COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
```

```
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO

      NUMER = 0
      KKK = 0
      NEND = 0
      IF (NTYPE .EQ. 7) I = I - 1

10    I = I + 1
      IF (I .GT. 72) RETURN

C     CHECK IF THE FIRST CHARACTER IS A DELIMITER

      DO 15 J = 1,10
         IF (NTYPE .EQ. 7) THEN
            IF (C(I:I) .EQ. ';') THEN
               WRITE (7,98,REC=STNO) STNO,LINENO,NTYPE,MTYPE
               RETURN
            ENDIF
         ENDIF
         IF (C(I:I) .EQ. DELIM(J) .AND. J .NE. 5) GOTO 10
15    CONTINUE

23    INDEX1 = I
      INDEX2 = I
      SYMBOL(NUM)(1:1) = C(I:I)
34    I = I + 1
      IF (I .GT. 72) RETURN
      INDEX2 = INDEX2 + 1

C     CHECK NEXT CHARACTER FOR DELIMITERS

      IF (NTYPE .EQ. 7) THEN
         IF (C(I:I) .EQ. ')') NEND = 1
      ENDIF

      DO 35 J = 1,10
         IF (C(I:I) .EQ. DELIM(J) .AND. J .NE. 5) THEN
            LENGTH(NUM) = INDEX2 - INDEX1
            SYMBOL(NUM)(1:LENGTH(NUM)) = C(INDEX1:INDEX2 - 1)
            KKK = KKK + 1
            GOTO 37
         ENDIF
35    CONTINUE

C     NEXT CHARACTER IS NOT A DELIMITER. CONCATENATE IT TO THE PARTIALLY
C     CONSTRUCTED SYMBOL AND THEN READ THE NEXT CHARACTER

      SYMBOL(NUM)(1:INDEX2 - INDEX1 + 1) = SYMBOL(NUM)(1:INDEX2 - INDEX1
     *) // C(I:I)
      GOTO 34

C     CHECK TO SEE IF THE SYMBOL IS A NUMBER. IF IT IS WRITE THE
C     CONSTANT NUMBER INTO THE OBJECT FILE.

37    DO 57 J = 27,36
         IF (SYMBOL(NUM)(1:1) .EQ. CHARS(J)) NUMER =  1
57    CONTINUE

      IF (SYMBOL(NUM)(1:1) .EQ. '-' .OR.  NUMER .EQ. 1) THEN

         NUMER = 0
         MM(KKK) = 2
         KKK = KKK + 1
         MM(KKK) = NUM1
         CONS(NUM1)(1:INDEX2 - INDEX1 + 1) = SYMBOL(NUM)(1:INDEX2 - INDE
     *   X1 + 1)
         DO 40 JJ = 1,LENGTH(NUM)
```

```
            IF (CONS(NUM1)(JJ:JJ) .EQ. '.') GOTO 95
40          CONTINUE
            LENGTH(NUM) = LENGTH(NUM) + 1
            CONS(NUM1)(LENGTH(NUM):LENGTH(NUM)) = '.'
95          LENCON(NUM1) = LENGTH(NUM)
            WRITE(4,49) NUM1, LINENO, STNO, LENCON(NUM1),(CONS2(K,NUM1),K=1
     *,LENCON(NUM1))
            NUM1 = NUM1 + 1

C       THE MAXIMUM VALUE OF KKK IS DETERMINED BY THE ACTUAL COMMAND.

            IF (NTYPE .EQ. 2 .AND. MTYPE .EQ. 1) THEN
               IF (KKK .LT. 7) GOTO 55
            ELSEIF (NTYPE .EQ. 3 .AND. MTYPE .EQ. 1) THEN
               IF (KKK .LT. 2) GOTO 55
            ELSEIF (NTYPE .EQ. 3) THEN
               IF (KKK .LT. 5) GOTO 55
            ELSEIF (NTYPE .EQ. 4 .AND. MTYPE .EQ. 1) THEN
               IF (KKK .LT. 5) GOTO 55
            ELSEIF (NTYPE .EQ. 7) THEN
               IF (NEND .EQ. 0) GOTO 55
            ELSEIF (NTYPE .EQ. 1 .AND. MTYPE .EQ. 1) THEN
               IF (KKK .LT. 2) GOTO 55
            ELSEIF (NTYPE .EQ. 1 .AND. MTYPE .EQ. 6) THEN
               IF (KKK .LT. 2) GOTO 55
            ELSEIF (NTYPE .EQ. 1 .AND. MTYPE .EQ. 7) THEN
               IF (KKK .LT. 2) GOTO 55
            ELSEIF (NTYPE .EQ. 1 .AND. MTYPE .EQ. 8) THEN
               IF (KKK .LT. 2) GOTO 55
            ELSEIF (NTYPE .EQ. 1 .AND. MTYPE .EQ. 9) THEN
               IF (KKK .LT. 2) GOTO 55
            ELSEIF (NTYPE .EQ. 4 .AND. MTYPE .EQ. 2) THEN
               IF (KKK .LT. 2) GOTO 55
            ELSEIF (NTYPE .EQ. 4 .AND. MTYPE .EQ. 3) THEN
               IF (KKK .LT. 2) GOTO 55
            ELSEIF (NTYPE .EQ. 5 .AND. MTYPE .EQ. 1) THEN
               IF (KKK .LT. 2) GOTO 55
            ELSEIF (NTYPE .EQ. 5 .AND. MTYPE .EQ. 2) THEN
               IF (KKK .LT. 2) GOTO 55
            ELSEIF (NTYPE .EQ. 5 .AND. MTYPE .EQ. 3) THEN
               IF (KKK .LT. 2) GOTO 55
            ELSEIF (NTYPE .EQ. 8 .AND. MTYPE .EQ. 1) THEN
               IF (KKK .LT. 2) GOTO 55
            ELSE
               IF (KKK .LT. 4) GOTO 55
            ENDIF

            GOTO 99

          ENDIF

C       FIND THE STNO OF THE SYMBOL AND WRITE TO THE OBJECT FILE.

          DO 47 J = 1, NUM - 1

           IF (VARNO(J) .EQ. 0) THEN
             IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. SYMBOL(J)(1:LENGTH(J)) .AND
     *    . LENGTH(NUM) .EQ. LENGTH(J)) GOTO 48

           ELSE

             IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. SYMBOL(J)(1:LENGTH(J)) .AND
     *    . LENGTH(NUM) .EQ. LENGTH(J) .AND. VARNO(NUM) .EQ. VARNO(J))
     *      GOTO 48

           ENDIF

           GOTO 47
```

```
48     IF (NTYPE .EQ. 4 .AND. MTYPE .EQ. 1) THEN

            IF (KKK .EQ. 1) THEN

               MM(KKK) = J
               KKK = KKK + 1

            ELSE

               MM(KKK) = 1
               KKK = KKK + 1
               MM(KKK) = J

            ENDIF

            IF (KKK .LT. 5) GOTO 55

            GOTO 99

       ELSE

            MM(KKK) = 1
            KKK = KKK + 1
            MM(KKK) = J

         IF (NTYPE .EQ. 2 .AND. MTYPE .EQ. 1) THEN
            IF (KKK .LT. 7) GOTO 55
         ELSEIF (NTYPE .EQ. 3 .AND. MTYPE .EQ. 1) THEN
            IF (KKK .LT. 2) GOTO 55
         ELSEIF (NTYPE .EQ. 3) THEN
            IF (KKK .LT. 5) GOTO 55
         ELSEIF (NTYPE .EQ. 4 .AND. MTYPE .EQ. 1) THEN
            IF (KKK .LT. 5) GOTO 55
         ELSEIF (NTYPE .EQ. 7) THEN
            IF (NEND .EQ. 0) GOTO 55
         ELSEIF (NTYPE .EQ. 1 .AND. MTYPE .EQ. 1) THEN
            IF (KKK .LT. 2) GOTO 55
         ELSEIF (NTYPE .EQ. 1 .AND. MTYPE .EQ. 6) THEN
            IF (KKK .LT. 2) GOTO 55
         ELSEIF (NTYPE .EQ. 1 .AND. MTYPE .EQ. 7) THEN
            IF (KKK .LT. 2) GOTO 55
         ELSEIF (NTYPE .EQ. 1 .AND. MTYPE .EQ. 8) THEN
            IF (KKK .LT. 2) GOTO 55
         ELSEIF (NTYPE .EQ. 1 .AND. MTYPE .EQ. 9) THEN
            IF (KKK .LT. 2) GOTO 55
         ELSEIF (NTYPE .EQ. 4 .AND. MTYPE .EQ. 2) THEN
            IF (KKK .LT. 2) GOTO 55
         ELSEIF (NTYPE .EQ. 4 .AND. MTYPE .EQ. 3) THEN
            IF (KKK .LT. 2) GOTO 55
         ELSEIF (NTYPE .EQ. 5 .AND. MTYPE .EQ. 1) THEN
            IF (KKK .LT. 2) GOTO 55
         ELSEIF (NTYPE .EQ. 5 .AND. MTYPE .EQ. 2) THEN
            IF (KKK .LT. 2) GOTO 55
         ELSEIF (NTYPE .EQ. 5 .AND. MTYPE .EQ. 3) THEN
            IF (KKK .LT. 2) GOTO 55
         ELSEIF (NTYPE .EQ. 8 .AND. MTYPE .EQ. 1) THEN
            IF (KKK .LT. 2) GOTO 55
         ELSE
            IF (KKK .LT. 4) GOTO 55
         ENDIF

            GOTO 99

       ENDIF

47     CONTINUE

C      LABEL HAS NOT BEEN DEFINED. PLACE IT AS AN UNRESOLVED LABEL.
```

```
      LENRES(NUM2) = LENGTH(NUM)
      UNNO(NUM2) = STNO
      UNVAR(NUM2) = SUBRNO
      UNRES(NUM2)(1:LENRES(NUM2)) = SYMBOL(NUM)(1:LENGTH(NUM))
      NUM2 = NUM2 + 1

      IF (NTYPE .NE. 2) THEN
         MM(5) = 0
      ELSE
         MM(7) = 0
      ENDIF

      GOTO 99

C     IF COMMAND IS "COMPC", CHECK THE CONDITION AT STATEMENT 65

55    IF (KKK .EQ. 2 .AND. NTYPE .EQ. 4 .AND. MTYPE .EQ. 1) GOTO 65
      I = I + 1
      IF (I .GT. 72) RETURN
      IF (C(I:I) .EQ. ')') GOTO 99
      IF (C(I:I) .EQ. ' ' .OR. C(I:I) .EQ. ',') GOTO 55
      GOTO 23

C     CONDITION IN COMMAND IS CHECKED HERE AND THE VALUE OF MM(2) IS
C     ASSIGNED A VALUE DEPENDING ON THE CONDITION.

65    IF (C(I:I) .EQ. ' ' .OR. C(I:I) .EQ. ',') I = I + 1
      IF (I .GT. 72) RETURN
      IF (C(I:I) .EQ. ' ' .OR. C(I:I) .EQ. ',') GOTO 65

      IF (C(I:I+1) .EQ. '<=') THEN
         MM(KKK) = 2

C     BECAUSE OF ERROR IN COMPILING, CALLING SUBROUTINE

         CALL INCREM(I,1)

      ELSEIF (C(I:I+1) .EQ. '<>') THEN
         MM(KKK) = 6

C     BECAUSE OF ERROR IN COMPILING, CALLING SUBROUTINE

         CALL INCREM(I,1)

      ELSEIF (C(I:I+1) .EQ. '>=') THEN
         MM(KKK) = 4

C     BECAUSE OF ERROR IN COMPILING, CALLING SUBROUTINE

         CALL INCREM(I,1)

      ELSEIF (C(I:I) .EQ. '=') THEN
         MM(KKK) = 5

      ELSEIF (C(I:I) .EQ. '>') THEN
         MM(KKK) = 3

      ELSEIF (C(I:I) .EQ. '<') THEN
         MM(KKK) = 1

      ENDIF

      GOTO 10

99    WRITE (7,98,REC=STNO) STNO,LINENO,NTYPE,MTYPE,(MM(K),K=1,KKK)
98    FORMAT(1X,I4,1X,I4,1X,I1,1X,I4,20(1X,I4))
49    FORMAT(1X,I3,2X,I3,2X,I4,2X,I3,2X,72A1)

      RETURN
```

```
      END

C     SUBROUTINE LABELS

C     SUBROUTINE TO COMPLETE STATEMENTS IN THE OBJECT FILES WHICH HAVE
C     UNRESOLVED LABELS ASSOCIATED WITH THEM. COMMANDS INCLUDE
C     "BRANCH", "COMPC", "TESTC", "TESTI", "TESTP" AND "WAITI".

C     PARAMETER DEFINITION:

C          J        -      VARIABLE NUMBER OF UNRESOLVED LABEL

      SUBROUTINE LABELS(J)

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), MM(4)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO

      READ (7,100,REC=UNNO(J)) K1,K2,K3,K4,K5,K6,K7,K8,K9,K10

      IF (K3 .EQ. 3) THEN

C     "BRANCH" COMMAND

         IF (K4 .EQ. 1) THEN

            K5 = 1
            K6 = NUM - 1
            WRITE (7,100,REC=UNNO(J)) K1,K2,K3,K4,K5,K6

C     "TESTC","TESTI","TESTP" COMMANDS.

         ELSEIF (K4 .EQ. 2 .OR. K4 .EQ. 3 .OR. K4 .EQ. 4) THEN

            K9 = NUM - 1
            WRITE (7,100,REC=UNNO(J)) K1,K2,K3,K4,K5,K6,K7,K8,K9

         ENDIF

      ENDIF

C     "WAITI" COMMAND

      IF (K3 .EQ. 2 .AND. K4 .EQ. 1) THEN
            K11 = NUM - 1
            WRITE (7,100,REC=UNNO(J)) K1,K2,K3,K4,K5,K6,K7,K8,K9,K10,K11
      ENDIF

C     "COMPC' COMMAND

      IF (K3 .EQ. 4 .AND. K4 .EQ. 1) THEN
            K9 = NUM - 1
            WRITE (7,100,REC=UNNO(J)) K1,K2,K3,K4,K5,K6,K7,K8,K9
      ENDIF

100   FORMAT(1X,I4,1X,I4,1X,I1,1X,I4,1X,I4,1X,I4,1X,I4,1X,I4,1X,I4,1X,I4,1X,I4
     *,1X,I4)

      RETURN
```

```
      END

C     SUBROUTINE COUNT

C     THIS SUBROUTINE READS THE FOUR PARAMETERS IN THE"PT" DEFINITION
C     OF A POINT. THE PARAMETERS CAN BE CONSTANTS OR LABELS/COUNTERS.
C     THIS SUBROUTINE ALSO READS THE THREE POINTS, ASSUMED TO BE
C     LABELS AND THE TWO CONSTANTS WHICH MAY BE LABELS OR NUMBERS
C     WHICH DEFINE A PALLET. THEN THESE VALUES ARE WRITTEN TO BOTH
C     THE CONSTANTS FILE AND THE SYMTAB FILE.

C     PARAMETER DEFINITION:

C          II        -       CURRENT INDEX NUMBER IN ARRAY C

      SUBROUTINE COUNT(II)

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), MM(10)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO

      NUMER = 0
      KKK = 0
      I = II - 1

10    I = I + 1
      IF (I .GT. 72) RETURN

C     CHECK IF THE FIRST CHARACTER IS A DELIMITER

      DO 15 J = 1,10
         IF (C(I:I) .EQ. DELIM(J) .AND. J .NE. 5) GOTO 10
15    CONTINUE

23    INDEX1 = I
      INDEX2 = I
      SYMBOL(NUM)(1:1) = C(I:I)
34    I = I + 1
      IF (I .GT. 72) RETURN
      INDEX2 = INDEX2 + 1

C     CHECK NEXT CHARACTER FOR DELIMITERS

      DO 35 J = 1,10
         IF (C(I:I) .EQ. DELIM(J) .AND. J .NE. 5) THEN
            LENGTH(NUM) = INDEX2 - INDEX1
            SYMBOL(NUM)(1:LENGTH(NUM)) = C(INDEX1:INDEX2 - 1)
            KKK = KKK + 1
            GOTO 37
         ENDIF
35    CONTINUE

C     NEXT CHARACTER IS NOT A DELIMITER. CONCATENATE IT TO THE PARTIALLY
C     CONSTRUCTED SYMBOL AND THEN READ THE NEXT CHARACTER

      SYMBOL(NUM)(1:INDEX2 - INDEX1 + 1) = SYMBOL(NUM)(1:INDEX2 - INDEX1
     *) // C(I:I)
      GOTO 34
```

```
C     CHECK TO SEE IF THE SYMBOL IS A NUMBER. IF IT IS WRITE THE
C     CONSTANT NUMBER INTO THE OBJECT FILE.

37    DO 57 J = 27,36
         IF (SYMBOL(NUM)(1:1) .EQ. CHARS(J)) NUMER =  1
57    CONTINUE

      IF (SYMBOL(NUM)(1:1) .EQ. '-' .OR.  NUMER .EQ. 1) THEN
         NUMER = 0
         MM(KKK) = 2
         KKK = KKK + 1
         MM(KKK) = NUM1
         CONS(NUM1)(1:INDEX2 - INDEX1 + 1) = SYMBOL(NUM)(1:INDEX2 - INDE
     *   X1 + 1)
         DO 40 JJ = 1,LENGTH(NUM)
            IF (CONS(NUM1)(JJ:JJ) .EQ. '.') GOTO 95
40       CONTINUE
         LENGTH(NUM) = LENGTH(NUM) + 1
         CONS(NUM1)(LENGTH(NUM):LENGTH(NUM)) = '.'
95       LENCON(NUM1) = LENGTH(NUM)
         WRITE(4,49) NUM1, LINENO, STNO, LENCON(NUM1),(CONS2(K,NUM1),K=1
     *,LENCON(NUM1))
         NUM1 = NUM1 + 1

C     KKK HAS A MAXIMUM VALUE OF 8 FOR THE DEFINITION OF A POINT,
C     AND A VALUE OF 10 FOR A PALLET DEFINITION.

         IF (ITYPE .EQ. 9) THEN
            IF (KKK .LT. 8) GOTO 55
         ELSEIF (ITYPE .EQ. 5) THEN
            IF (KKK .LT. 10) GOTO 55
         ENDIF

         GOTO 99
      ENDIF

C     FIND THE STNO OF THE SYMBOL AND WRITE TO THE OBJECT FILE.

      DO 47 J = 1, NUM - 1

       IF (VARNO(J) .EQ. 0) THEN

         IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. SYMBOL(J)(1:LENGTH(J)) .AND
     *   . LENGTH(NUM) .EQ. LENGTH(J)) GOTO 48
       ELSE

         IF (SYMBOL(NUM)(1:LENGTH(NUM)) .EQ. SYMBOL(J)(1:LENGTH(J)) .AND
     *   . LENGTH(NUM) .EQ. LENGTH(J) .AND. VARNO(NUM) .EQ. VARNO(J))
     *     GOTO 48

       ENDIF

       GOTO 47

48          MM(KKK) = 1
            KKK = KKK + 1
            MM(KKK) = J

            IF (ITYPE .EQ. 9) THEN

               IF (KKK .LT. 8) GOTO 55

            ELSEIF (ITYPE .EQ. 5) THEN

               IF (KKK .LT. 10) GOTO 55

            ENDIF

            GOTO 99
```

```
47    CONTINUE

55    I = I + 1
      IF (I .GT. 72) RETURN
      IF (C(I:I) .EQ. ' ' .OR. C(I:I) .EQ. ',') GOTO 55
      GOTO 23

C     WRITE THE VARIABLE NUMBERS INTO THE SYMBOL TABLE FILE.
C     THEN WRITE THE SYMBOL INTO THE VARIABLES FILE.

99    WRITE (8,98,REC=NUM-1) NUM-1,(MM(K),K=1,KKK)
      VARNO(NUM-1) = SUBRNO
      WRITE (3,46) NUM-1,LINENO,STNO,LENGTH(NUM-1),ITYPE,VARNO(NUM-1),
     *(SYM2(K,NUM-1),K =1,LENGTH(NUM-1))

46    FORMAT(1X,I3,2X,I3,2X,I4,2X,I3,2X,I1,2X,I2,2X,72A1)
49    FORMAT(1X,I3,2X,I3,2X,I4,2X,I3,2X,72A1)
98    FORMAT(1X,I3,10X,10(1X,I4))

      RETURN
      END

C     SUBROUTINE INCREM

C     THIS SUBROUTINE JUST INCREMENTS A VARIABLE BY A VALUE.
C     THIS WAS DONE TO RESOLVE AN INTERNAL COMPILER ERROR.

C     PARAMETER DEFINITION:

C          KK        -       VARIABLE TO BE INCREMENTED
C          KKK       -       VALUE BY WHICH VARIABLE IS INCREMENTED

      SUBROUTINE INCREM(KK,KKK)

      CHARACTER LINE * 72, AMLWRD(42) * 10, DELIM(10), CHARS(50)
      CHARACTER SYMBOL(100) * 72, C* 72, SYM2(72,100), CONS(100) * 72
      CHARACTER CONS2(72,100), SUB(100) * 72, UNRES(100) * 72
      INTEGER * 4 STNO, NSUB, SUBNO(100), SUBOP(100), UNNO(100)
      INTEGER * 4 VARNO(100), UNVAR(100), SUBRNO
      DIMENSION LENGTH(100),LENCON(100), LENRES(100), P(4)
      EQUIVALENCE (SYMBOL(1),SYM2(1,1))
      EQUIVALENCE (CONS(1),CONS2(1,1))

      COMMON/ A/ AMLWRD,DELIM,CHARS,SYMBOL,C,CONS,SUB,UNRES
      COMMON/ B/ LENGTH,LENCON,ITYPE,I,LINENO,NUM,NUM1,INDEX1,INDEX2
      COMMON/ C/ STNO,NSUB,SUBNO,SUBOP,NUM2,UNNO,LENRES
      COMMON/ D/ VARNO,UNVAR,SUBRNO

      KK = KK + KKK

      RETURN
      END
```

APPENDIX C. SYSTEM CONTROL PROGRAM

```
C     THIS IS THE "SYSTEM CONTROL" PROGRAM RESIDENT ON THE IBM PC
C     WHICH READS THE OBJECT FILE GENERATED BY THE COMPILER AND
C     TRANSMITS PARAMETERS TO THE ROBOT CONTROLLER.


C  VARIABLES USED IN THE SYSTEM CONTROL PROGRAM

C  AGG(I)       =  Array containing aggregate number
C  ALIN         =  LINEAR value
C  CNTVAL(I)    =  Array containing counter value
C  CONS(I)      =  Array containing constant number
C  CONVAL(I)    =  Array containing constant value
C  COUNT(I)     =  Array containing counter number
C  IONUM(I)     =  Array containing DI/DO value
C  LAST         =  Last statement number to be executed
C  LASUB        =  Last statement number of subroutine
C  NEXT         =  Next statement number to be executed
C  NREC         =  Number of records in constants file
C  NUM1         =  Index for constants
C  NUM2         =  Index for points
C  NUM3         =  Index for counters
C  NUM4         =  Index for aggregates
C  NUM5         =  Index for parameters
C  NUM6         =  Index for pallets
C  PALLET(I)    =  Array containing pallet number
C  PALPTS(I)    =  Array containing number of points in pallet
C  PAR(I)       =  Array containing parameter number
C  PART1        =  First statement number to be executed (partial execution)
C  PART2        =  Last statement number to be executed (partial execution)
C  PARTNO(I)    =  Array containing current part number of pallet
C  PAY          =  PAYLOAD value
C  PAYOLD       =  Old PAYLOAD value
C  PNTVAL(I)    =  Array containing point number
C  RAGG         =  R coordinate of aggregate
C  RPAL         =  R coordinate of pallet point
C  RPAR         =  R coordinate of parameter
C  RPNT         =  R coordinate of point
C  RVAL         =  Current R coordinate of robot arm
C  SLPAY        =  PAYLOAD value for slow robot arm motion
C  SS           =  Index specifying whether single-stepping or not
C  STAT         =  Array containing line of AML/E program
C  XAGG         =  X coordinate of aggregate
C  XPAL         =  X coordinate of pallet point
C  XPAR         =  X coordinate of parameter
C  XPNT         =  X coordinate of point
C  XVAL         =  Current X coordinate of robot arm
C  YAGG         =  Y coordinate of aggregate
C  YPAL         =  Y coordinate of pallet point
C  YPAR         =  Y coordinate of parameter
C  YPNT         =  Y coordinate of point
C  YVAL         =  Current Y coordinate of robot arm
C  ZAGG         =  Z coordinate of aggregate
C  ZPAL         =  Z coordinate of pallet point
C  ZPAR         =  Z coordinate of parameter
C  ZPNT         =  Z coordinate of point
C  ZVAL         =  Current Z coordinate of robot arm
C  ZON          =  ZONE value

C     THE MAIN PROGRAM OPEN ALL FILES AND CALLS SUBROUTINE SYS
C     TO START INTERACTIVE EXECUTION.

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             CLS,CSROFF,FINI,INKEY,SYS,TTOUT.

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
```

```
      INTEGER SS,SLPAY,PART1,PART2
      CHARACTER * 20 NAME
      CHARACTER * 24 NAMCON,NAMOUT,NAMVAR,NAMOBJ,NAMTAB
      CHARACTER * 24 NA2CON,NA2OUT,NA2VAR,NA2OBJ,NA2TAB
      CHARACTER * 24 NAMCN2,NA3CON,NA2AML,NAMAML
      INTEGER * 2 LL,KYCOD1,KYCOD2

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

C     THE USER'S AML/E PROGRAM NAME IS READ FROM THE FILE IRPS00.
C     THIS FILE IS CREATED BY THE PSEUDO-COMPILER.
C     THE PROGRAM THEN CONCATENATES THE EXTENSIONS FOR THE OTHER
C     FILES CREATED BY THE COMPILER, AND OPENS THEM UP.

      OPEN (9,FILE='IRPS00')
      READ (9,100) LL,NAME
100   FORMAT (I2,1X,A20)
      CLOSE (9,STATUS='DELETE')

      NAMAML(22-LL:24) = NAME(22-LL:20)//'.AML'
      NAMCN2(22-LL:24) = NAME(22-LL:20)//'.CN1'
      NAMOUT(22-LL:24) = NAME(22-LL:20)//'.OUT'
      NAMVAR(22-LL:24) = NAME(22-LL:20)//'.VAR'
      NAMCON(22-LL:24) = NAME(22-LL:20)//'.CON'
      NAMOBJ(22-LL:24) = NAME(22-LL:20)//'.OBJ'
      NAMTAB(22-LL:24) = NAME(22-LL:20)//'.TAB'

      J = 0

      DO 400 JJ = 22-LL,24
         J = J + 1
         NA2AML(J:J) = NAMAML(JJ:JJ)
         NA3CON(J:J) = NAMCN2(JJ:JJ)
         NA2OUT(J:J) = NAMOUT(JJ:JJ)
         NA2VAR(J:J) = NAMVAR(JJ:JJ)
         NA2CON(J:J) = NAMCON(JJ:JJ)
         NA2OBJ(J:J) = NAMOBJ(JJ:JJ)
         NA2TAB(J:J) = NAMTAB(JJ:JJ)
400   CONTINUE

C     THE CONSTANTS FILE .CN1, IS READ AND CONVERTED INTO A DIRECT ACCESS
C     FILE .CON.

      OPEN (1,FILE=NA3CON(1:LL+3))
      OPEN (2,FILE=NA2CON(1:LL+3))
      NREC = 0

10    READ (1,*,END=52) I1,I2,I3,I4,T
      WRITE(2,51) I1,I2,I3,I4,T
      NREC = NREC + 1
      GOTO 10

51    FORMAT(1X,I3,2X,I3,2X,I4,2X,I3,2X,F8.2,50(' '))
52    CLOSE (1,STATUS='DELETE')
      CLOSE (2)

      OPEN (2,FILE=NA2OUT(1:LL+3),ACCESS='DIRECT',FORM='FORMATTED',
```

```
     *RECL=80)
      OPEN (3,FILE=NA2VAR(1:LL+3))
      OPEN (4,FILE=NA2CON(1:LL+3),ACCESS='DIRECT',FORM='FORMATTED',
     *RECL=80)
      OPEN (7,FILE=NA2OBJ(1:LL+3),ACCESS='DIRECT',FORM='FORMATTED',
     *RECL=80)
      OPEN (8,FILE=NA2TAB(1:LL+3),ACCESS='DIRECT',FORM='FORMATTED',
     *RECL=80)

      CALL SYS(NA2AML,LL)

      CALL CLS(31)
      CALL CSROFF
      CALL TTOUT(0,24,1,'Execution completed...',22,22,28)
      CALL TTOUT(0,24,24,'Hit any key to terminate program.',33,33,31)
      CALL INKEY(KYCOD1,KYCOD2)

      CALL FINI

      END

C     SUBROUTINE TTOUT

C     THIS SUBROUTINE WRITES A STRING A CHARACTERS AT A SPECIFIED
C     ROW AND COLUMN OF THE DISPLAY.

C     PARAMETER DEFINITION:

C          PAGE      -      PAGE NUMBER
C          ROW       -      ROW NUMBER
C          COLUMN    -      COLUMN NUMBER
C          BUF       -      CHARACTER BUFFER NAME
C          SIZE      -      SIZE OF BUFFER
C          LENG      -      LENGTH OF CHARACTER STRING
C          WRATT     -      ATTRIBUTE OF CHARACTER TO BE WRITTEN


C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             LOCATE, WRCHAT.

      SUBROUTINE TTOUT(PAGE,ROW,COLUMN,BUF,LENG,SIZE,WRATT)

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2
      INTEGER * 2 PAGE,ROW,COLUMN,SIZE,WRATT,LENG
      CHARACTER * 1 BUF(SIZE)

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

      CALL LOCATE(PAGE,ROW,COLUMN)

      DO 10 J = 1,LENG
         CALL WRCHAT(PAGE,BUF(J),WRATT,1)
10    CONTINUE

      RETURN
```

```
      END

C     SUBROUTINE FINI

C     THIS SUBROUTINE DOES ALL THE HOUSEKEEPING BY CLOSING FILES
C     BEFORE PROGRAM EXECUTION TERMINATES.

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             CLRBUF,CLS,CSRON.

      SUBROUTINE FINI

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

      CALL CLS(0)
      CALL CLRBUF
      CALL CSRON

      CLOSE (2)
      CLOSE (3,STATUS='DELETE')
      CLOSE (4,STATUS='DELETE')
      CLOSE (7,STATUS='DELETE')
      CLOSE (8,STATUS='DELETE')

      STOP
      END

C     SUBROUTINE SYS

C     THIS SUBROUTINE WRITES A STRING A CHARACTERS AT A SPECIFIED
C     ROW AND COLUMN OF THE DISPLAY.

C     PARAMETER DEFINITION:

C          NA2AML    -       USER'S AML/E PROGRAM NAME
C          LL        -       LENGTH OF PROGRAM NAME

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             CHECK,CLRBUF,CLS,CNTRL,CSRON,CSROFF,COUN,EXEC,FLOW,IFKEY,INKEY,
C             LOCATE,MOT,PAL,PALL,PART,SENS,SLOW,SUB1,SUB2,TTOUT,WHERE,WRITER.

      SUBROUTINE SYS(NA2AML,LL)

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2
      INTEGER * 2 KYCOD1,KYCOD2,LL
      CHARACTER * 80 STAT
      CHARACTER * 24 NA2AML
      CHARACTER * 1 BLANK(80)

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
```

```
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

      DATA BLANK/80 * ' '/

C     INITIALIZE ALL PROGRAM VARIABLES.

      DO 1001 JJJJ = 1,80
         STAT(JJJJ:JJJJ) = BLANK(1)
1001  CONTINUE

      SS = 0
      SLPAY = 0
      PAYOLD = 0.0
      PART1 = 1
      PART2 = 1000
      IC = 1
      ALIN = 0.0
      PAY = 0.0
      ZON = 0.0
      XVAL = 650.0
      YVAL = 0.0
      ZVAL = 0.0
      RVAL = 0.0
      NUM = 1
      NUM1 = 1
      NUM2 = 1
      NUM3 = 1
      NUM4 = 1
      NUM5 = 1
      NUM6 = 1

      DO 1 I = 1,32
         IONUM(I) = 0
1     CONTINUE

C     READ EACH LINE OF THE VARIABLES FILE, AND ASSIGN VALUES
C     READ FROM THE SYMBOL TABLE FILE, TO VARIABLE NAMES.

10    READ (3,100,END=1000) IC1,ICODE

C     IF THE VARIABLE NAME IS A LABEL, ASSIGN THE STATEMENT NUMBER AS THE
C     VALUE FOR THE LABEL, AND WRITE INTO THE SYMBOL TABLE FILE.

11    IF (ICODE .EQ. 1) THEN
         CC = IC1
         WRITE (8,400,REC=NUM) NUM,CC

C     IF THE VARIABLE IS A CONSTANT, ASSIGN VALUE TO ARRAY CONVAL.

      ELSEIF (ICODE .EQ. 2) THEN
         READ (8,200,REC=NUM) CODE
         CONS(NUM1) = NUM
         CONVAL(NUM1) = CODE
         NUM1 = NUM1 + 1

C     IF THE VARIABLE IS A POINT, ASSIGN VALUES TO ARRAYS XPNT,YPNT,RPNT,
C     AND ZPNT.

      ELSEIF (ICODE .EQ. 3) THEN
```

```
         READ (8,300,REC=NUM) X,Y,Z,R
         PNTVAL(NUM2) = NUM
         XPNT(NUM2) = X
         YPNT(NUM2) = Y
         ZPNT(NUM2) = Z
         RPNT(NUM2) = R
         NUM2 = NUM2 + 1

C     IF THE VARIABLE IS A COUNTER, INITIALIZE COUNTER VALUE TO ZERO, AND
C     WRITE TO THE SYMBOL TABLE FILE.

      ELSEIF (ICODE .EQ. 4) THEN
         COUNT(NUM3) = NUM
         CNTVAL(NUM3) = 0
         CC = CNTVAL(NUM3)
         WRITE (8,400,REC=NUM) NUM,CC
         NUM3 = NUM3 + 1

C     IF THE VARIABLE IS A PALLET, CALL SUBROUTINE PAL.

      ELSEIF (ICODE .EQ. 5) THEN
         CALL PAL(NUM)

C     IF THE VARIABLE IS A SUBROUTINE NAME, FIRST CHECK FOR FORMAL PARAMETERS.
C     IF THERE ARE ANY FORMAL PARAMETERS, INITIALIZE VALUES TO ZEROES, AND
C     WRITE THE VARIABLE NUMBER OF THE FORMAL PARAMETER TO THE OBJECT FILE.

      ELSEIF (ICODE .EQ. 6) THEN
         NN = 0
12       NUM = NUM + 1
         NN = NN + 1
         READ (3,100,END=1000) IC1,ICODE

         IF (ICODE .EQ. 7) THEN
            PAR(NUM5) = NUM
            XPAR(NUM5) = 0.0
            YPAR(NUM5) = 0.0
            ZPAR(NUM5) = 0.0
            RPAR(NUM5) = 0.0
            NUM5 = NUM5 + 1

            READ (7,700,REC=IC1) L1,L2,L3,L4,L5,L6,(L(I),I=1,5)
            L(NN) = NUM
            WRITE (7,700,REC=IC1) L1,L2,L3,L4,L5,L6,(L(I),I=1,5)
            GOTO 12
         ENDIF

         GOTO 11

C     IF THE VARIABLE IS AN AGGREGATE, READ IN THE FOUR VALUES, AND
C     ASSIGN THEM TO THE ARRAYS XAGG,YAGG,ZAGG,RAGG.

      ELSEIF (ICODE .EQ. 8) THEN
         READ (8,300,REC=NUM) X,Y,Z,R
         AGG(NUM4) = NUM
         XAGG(NUM4) = X
         YAGG(NUM4) = Y
         ZAGG(NUM4) = Z
         RAGG(NUM4) = R
         NUM4 = NUM4 + 1

      ENDIF

C     INCREMENT THE VARIABLE NUMBER AND READ THE NEXT ONE.

      NUM = NUM + 1
      GOTO 10

1000  REWIND (3)
```

```
C     ONCE ALL VARIABLES HAVE BEEN READ, DISPLAY MESSAGE TO USER.

      CALL CLS(31)
      CALL TTOUT(0,10,15,'Please make sure manipulator power is on.',41
     *,41,28)
      CALL TTOUT(0,11,15,'Please make sure controller is online.',38,38
     *,28)
      CALL TTOUT(0,13,15,'The robot is about to go home.',30,30,28)
      CALL TTOUT(0,14,15,'Please make sure the gripper ',29,29,28)
      CALL TTOUT(0,14,45,'clears any obstructions.',24,24,28)
      CALL TTOUT(0,24,1,'Hit any key when ready....',26,26,28)
      CALL CSROFF
      CALL INKEY(KYCOD1,KYCOD2)
      CALL CLS(31)
      IER = 0
      TRY = 1.0

C     RESET CONTROLLER.

551   CALL CNTRL(20,IER)

      IF (IER .NE. 0) THEN
         TRY = TRY + 1.0
         IF (TRY .GE. 5.0) CALL ERROR
         IER = 0
         GOTO 551
      ENDIF

      IER = 0
      TRY = 1.0

C     RETURN HOME.

552   CALL EXEC(11,IER)

      IF (IER .NE. 0) THEN
         TRY = TRY + 1.0
         IF (TRY .GE. 5.0) CALL ERROR
         IER = 0
         GOTO 552
      ENDIF

      IER = 0
      TRY = 1.0

C     SELECT APPLICATION 5.

553   CALL EXEC(35,IER)

      IF (IER .NE. 0) THEN
         TRY = TRY + 1.0
         IF (TRY .GE. 5.0) CALL ERROR
         IER = 0
         GOTO 553
      ENDIF

      IER = 0
      TRY = 1.0

C     SET IN AUTO EXECUTION MODE.

554   CALL EXEC(20,IER)

      IF (IER .NE. 0) THEN
         TRY = TRY + 1.0
         IF (TRY .GE. 5.0) CALL ERROR
         IER = 0
         GOTO 554
```

```
      ENDIF

      IER = 0
      TRY = 1.0

C     START CYCLE.

555   CALL EXEC(22,IER)

      IF (IER .NE. 0) THEN
         TRY = TRY + 1.0
         IF (TRY .GE. 5.0) CALL ERROR
         IER = 0
         GOTO 555
      ENDIF


      IER = 0
      TRY = 1.0

C     SET A PAYLOAD VALUE OF 0.

556   CALL WRITER(36,PAY,IER)

      IF (IER .NE. 0) THEN
         TRY = TRY + 1.0
         IF (TRY .GE. 5.0) CALL ERROR
         IER = 0
         GOTO 556
      ENDIF


      IER = 0
      TRY = 1.0

C     SET A LINEAR VALUE OF 0.

557   CALL WRITER(35,ALIN,IER)

      IF (IER .NE. 0) THEN
         TRY = TRY + 1.0
         IF (TRY .GE. 5.0) CALL ERROR
         IER = 0
         GOTO 557
      ENDIF

      IER = 0
      TRY = 1.0

C     SET A ZONE VALUE OF 0.

558   CALL WRITER(37,ZON,IER)

      IF (IER .NE. 0) THEN
         TRY = TRY + 1.0
         IF (TRY .GE. 5.0) CALL ERROR
         IER = 0
         GOTO 558
      ENDIF

      IER = 0

C     PROMPT THE USER FOR SINGLE-STEPPING OR CONTINUOUS OPERATION.
C     IF TEH USER OPTS FOR SINGLE-STEPPIN, SET SS TO 1.

      CALL CSRON
      CALL TTOUT(0,10,1,'Do you want to single step',26,26,31)
      CALL TTOUT(0,11,1,'through the program (Y/N) ?',27,27,31)
      CALL LOCATE(0,11,29)
```

```
      CALL INKEY(KYCOD1,KYCOD2)
      IF (KYCOD1 .EQ. 89 .OR. KYCOD1 .EQ. 121) SS = 1

C     CALL SUBROUTINE SLOW, WHICH ASKS THE USER IF THE ROBOT ARM NEEDS
C     TO MOVE SLOWLY.

      CALL SLOW

C     READ THE FIRST RECORD IN THE OBJECT FILE FOR THE FIRST
C     EXECUTABLE STATEMENT NUMBER AND THE STATEMENT NUMBER OF THE
C     LAST "END"STATEMENT.

      READ (7,500,REC=1) K1,K2

C     CALL SUBROUTINE PART WHICH ASKS THE USER IF THEY WOULD LIKE
C     TO EXECUTE THE COMPLETE PROGRAM, PART OF THE PROGRAM OR
C     EXECUTE A SUBROUTINE.

      CALL PART

C     BASED ON THE USER INPUTS, THE VARIABLES LAST AND NEXT ARE SET.
C     NEXT CONTAINS THE NEXT STATEMENT NUMBER TO BE EXECUTED, AND LAST
C     CONTAINS THE LAST STATEMENT NUMBER TO BE EXECUTED.

      IF (PART1 .EQ. 1 .AND. PART2 .EQ. 1000) THEN
         LAST = K1
         NEXT = K2
      ELSE
         LAST = PART2
         NEXT = PART1
      ENDIF

C     DISPLAY NAME OF PROGRAM BEING EXECUTED, AND THE USE OF FUNCTION
C     KEYS WHICH COULD BE PRESSED TO EITHER ABORT PROGRAM EXECUTION,
C     OR RESUME NORMAL SPEED EXECUTION.

      CALL CLS(31)
      CALL TTOUT(0,24,1,'Executing program....',21,21,28)
      CALL TTOUT(0,24,27,NA2AML(1:LL+3),3+LL,3+LL,26)

C     THIS MESSAGE IS DISPLAYED ONLY IF SINGLE-STEPPING IS IN EFFECT.

      IF (SS .EQ. 0) THEN
         CALL TTOUT(0,21,1,'Press F1 key to abort execution.',32,
     *   32,26)
      ENDIF

C     THIS MESSAGE IS DISPLAYED ONLY IF SLOW MOVEMENT IS IN EFFECT,
C     AND CONTINUOUS PROGRAM EXECUTION IS IN EFFECT.

      IF (SLPAY .NE. 0 .AND. SS .EQ. 0) THEN
          CALL TTOUT(0,22,1,'Press F2 key for normal speed execution.',
     *    40,40,26)
      ENDIF

C     IF SLOW MOVEMENT IS IN EFFECT, A MESSAGE IS DISPLAYED, AND THE
C     PAYLOAD VALUE SPECIFIED BY THE USER IS TRANSMITTED TO THE CONTROLLER.

      IF (SLPAY .NE. 0) THEN
         CALL TTOUT(0,24,75,'SLOW',4,4,156)
         IER = 0
         PAYLD = SLPAY
         TRY = 1.0

559      CALL WRITER(36,PAYLD,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
```

```
            IER = 0
            GOTO 559
         ENDIF

         IER = 0

      ENDIF

C     IF THE NEXT STATEMENT TO BE EXECUTED IS THE LAST ONE, STOP.
C     OTHERWISE, READ THE NEXT STATEMENT FROM THE OBJECT FILE, AND
C     THE LINE FROM THE LISTING FILE, AND DISPLAY ON THE SCREEN.

20    IF (NEXT .EQ. LAST) GOTO 2000
      READ (7,600,REC=NEXT) K54,K1,K2
      CALL TTOUT(0,11,1,STAT,80,80,26)
      READ (2,22,REC=K54) STAT
22    FORMAT(A80)
      CALL TTOUT(0,12,1,STAT,80,80,31)

C     DISPLAY FUNCTION KEY USAGE IF SINGLE-STEPPING IS IN EFFECT.

      IF (SS .EQ. 1) THEN
         CALL CSRON
         CALL TTOUT(0,1,5,'Press <Enter> key to continue single step.',
     *   42,42,26)
         CALL TTOUT(0,3,5,'Press F1 key to abort execution.',32,
     *   32,26)
         CALL TTOUT(0,5,5,'Press F2 key for continuous execution.',38,
     *   38,26)

C     DISPLAY MESSAGE IF SLOW EXECUTION IS IN EFFECT.

         IF (SLPAY .NE. 0) THEN
            CALL TTOUT(0,7,5,'Press F3 key for normal speed execution.',
     *      40,40,26)
            CALL LOCATE(0,7,47)
            GOTO 21
         ENDIF

C     WAIT FOR USER INPUT. IF USER STRIKES "ENTER" KEY, CONTINUE
C     WITH SINGLE-STEPPING. IF USER HITS F1, STOP PROGRAM EXECUTION,
C     AND IF USER STRIKES F2, EXECUTE PROGRAM CONTINUOUSLY.
C     IF SLOW MOVEMENT IS IN EFFECT, AND USER STRIKES F3, RESUME
C     NORMAL SPEED EXECUTION, AND TRANSMIT THE OLD PAYLOAD VALUE
C     TO THE CONTROLLER.

         CALL LOCATE(0,5,45)
21       CALL INKEY(KYCOD1,KYCOD2)

         IF (KYCOD1 .EQ. 13) THEN
            SS = 1
         ELSEIF (KYCOD1 .EQ. 0 .AND. KYCOD2 .EQ. 59) THEN
            CALL FINI
         ELSEIF (KYCOD1 .EQ. 0 .AND. KYCOD2 .EQ. 60) THEN
            SS = 0
            CALL TTOUT(0,21,1,'Press F1 key to abort execution.',32,
     *      32,26)
         ELSEIF (KYCOD1 .EQ. 0 .AND. KYCOD2 .EQ. 61 .AND. SLPAY .NE. 0)
     *   THEN
            SLPAY = 0
82          IER = 0
            TRY = 1.0
            CALL CHECK

560         CALL WRITER(36,PAYOLD,IER)

            IF (IER .NE. 0) THEN
               TRY = TRY + 1.0
               IF (TRY .GE. 5.0) CALL ERROR
```

```
               IER = 0
               GOTO 560
            ENDIF

            IER = 0
            CALL TTOUT(0,7,1,BLANK,80,80,31)
            CALL TTOUT(0,22,1,BLANK,80,80,31)
            CALL TTOUT(0,24,75,'    ',4,4,31)
            CALL LOCATE(0,5,45)
            GOTO 21
         ELSE
            GOTO 21
         ENDIF

         CALL CSROFF
         CALL TTOUT(0,1,5,BLANK,80,80,31)
         CALL TTOUT(0,3,5,BLANK,80,80,31)
         CALL TTOUT(0,5,5,BLANK,80,80,31)
         CALL TTOUT(0,7,5,BLANK,80,80,31)

         IF (SLPAY .NE. 0 .AND. SS .EQ. 0) THEN
            CALL TTOUT(0,24,75,'SLOW',4,4,156)
          CALL TTOUT(0,22,1,'Press F2 key for normal speed execution.',
     *    40,40,26)
         ENDIF

      ENDIF

C     READ USER INPUT AFTER EVERY STATEMENT IS EXECUTED.
C     IF SLOW MOVEMENT IS IN EFFECT, AND USER STRIKES F2, RESUME
C     NORMAL SPEED EXECUTION AND TRANSMIT OLD PAYLOAD VALUE TO
C     CONTROLLER.

      IF (SLPAY .NE. 0) THEN
         CALL IFKEY(KYCOD1,KYCOD2)

         IF (KYCOD1 .EQ. 0 .AND. KYCOD2 .EQ. 60) THEN
            SLPAY = 0
81          IER = 0
            TRY = 1.0
            CALL CHECK

561         CALL WRITER(36,PAYOLD,IER)

            IF (IER .NE. 0) THEN
               TRY = TRY + 1.0
               IF (TRY .GE. 5.0) CALL ERROR
               IER = 0
               GOTO 561
            ENDIF

            IER = 0
            CALL TTOUT(0,22,1,BLANK,80,80,31)
            CALL TTOUT(0,24,75,'    ',4,4,31)

         ENDIF

      ENDIF

C     READ USER INPUT AFTER EVERY STATEMENT IS EXECUTED. IF THE
C     USER STRIKES F1, STOP PROGRAM EXECUTION.

      CALL IFKEY(KYCOD1,KYCOD2)
      IF (KYCOD1 .EQ. 0 .AND. KYCOD2 .EQ. 59) CALL FINI
      CALL CLRBUF

C     CHECK THE COMMAND TYPE. K1 IS 0 FOR LABELS, 1 FOR MOTION
C     COMMANDS, 2 FOR SENSOR COMMANDS, 3 FOR FLOW OF CONTROL
C     COMMANDS, 4 FOR COUNTER COMMANDS, 5 FOR PALLET COMMANDS,
```

```
C     6 FOR SUBROUTINE COMMANDS, 7 FOR CALLS TO SUBROUTINES,
C     AND 8 FOR THE COMMAND "WHERE". BASED ON THE COMMAND TYPE
C     CALL THE APPROPRIATE SUBROUTINE.

      IF (K1 .EQ. 0) THEN
         NEXT = NEXT + 1
      ELSEIF (K1. EQ. 1) THEN
         CALL MOT(K2)
      ELSEIF (K1. EQ. 2) THEN
         CALL SENS(K2)
      ELSEIF (K1. EQ. 3) THEN
         CALL FLOW(K2)
      ELSEIF (K1. EQ. 4) THEN
         CALL COUN(K2)
      ELSEIF (K1. EQ. 5) THEN
         CALL PALL(K2)
      ELSEIF (K1. EQ. 6) THEN
         CALL SUB1(K2)
      ELSEIF (K1. EQ. 7) THEN
         CALL SUB2(K2)
      ELSEIF (K1. EQ. 8) THEN
         CALL WHERE
      ENDIF

C     GO READ THE NEXT STATEMENT TO BE EXECUTED.

      GOTO 20

100   FORMAT(11X,I4,7X,I1)
200   FORMAT(6X,F8.2)
300   FORMAT(4X,4(2X,F8.2))
400   FORMAT(1X,I3,2X,F8.2)
500   FORMAT(18X,I4,1X,I4)
600   FORMAT(6X,I4,1X,I1,1X,I4)
700   FORMAT(1X,I4,1X,I4,1X,I1,8(1X,I4))

2000  RETURN
      END

C     SUBROUTINE PAL

C     THIS SUBROUTINE COMPUTES THE POINTS OF EVERY PALLET POSITION.

C     PARAMETER DEFINITION:

C          NUM       -       VARIABLE NUMBER OF PALLET

      SUBROUTINE PAL(NUM)

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

C     READ THE VARIABLE / CONSTANT NUMBERS OF PARAMETERS
```

```
      READ (8,100,REC=NUM) L1,L2,L3,L4,L5,L6,L7,L8,L9,L10

C     READ THE VALUES OF THE THREE CORNER POINTS FROM THE SYMBOL
C     TEABLE FILE.

      READ (8,200,REC=L2) X1,Y1,Z1,R1
      READ (8,200,REC=L4) X2,Y2,Z2,R2
      READ (8,200,REC=L6) X3,Y3,Z3,R3
      PALLET(NUM6) = NUM

C     READ IN THE VALUE OF PART PER ROW, IPPR, FROM THE SYMBOL TABLE
C     FILE OR CONSTANTS FILE.

      IF (L7 .EQ. 1) THEN
         READ (8,200,REC=L8) CC
         IPPR = CC
      ELSEIF (L7 .EQ. 2) THEN
         READ (4,300,REC=L8) CC
         IPPR = CC
      ENDIF

C     READ IN THE VALUE OF NUMBER OF PALLET POINTS, PALPTS, FROM THE SYMBOL
C     TABLE FILE OR CONSTANTS FILE.

      IF (L9 .EQ. 1) THEN
         READ (8,200,REC=L10) CC
         PALPTS(NUM6) = CC
      ELSEIF (L9 .EQ. 2) THEN
         READ (4,300,REC=L10) CC
         PALPTS(NUM6) = CC
      ENDIF

C     THIS SECTION COMPUTES THE COORDINATES OF EACH PALLET POINT, BASED
C     ON THE COORDINATES OF THE CORNER POINTS, NUMBER OF PALLET POINTS,
C     AND THE PARTS PER ROW.

      XTOTX = X2 - X1
      XTOTY = X3 - X2
      YTOTX = Y2 - Y1
      YTOTY = Y3 - Y2
      IPPC = PALPTS(NUM6) / IPPR

      IF (IPPR .NE. 1) THEN
         XINCX = XTOTX / (IPPR-1)
         YINCX = YTOTX / (IPPR-1)
      ELSE
         XINCX = XTOTX
         YINCX = YTOTX
      ENDIF

      IF (IPPC .NE. 1) THEN
         XINCY = XTOTY / (IPPC-1)
         YINCY = YTOTY / (IPPC-1)
      ELSE
         XINCY = XTOTY
         YINCY = YTOTY
      ENDIF
      XPAL(NUM6,1) = X1
      YPAL(NUM6,1) = Y1
      K = 1

      DO 10 I = 1,PALPTS(NUM6),IPPR
         XPAL(NUM6,I) = XPAL(NUM6,1) + (XINCY * (K - 1))
         YPAL(NUM6,I) = YPAL(NUM6,1) + (YINCY * (K - 1))

         DO 20 J = 1,IPPR
            XPAL(NUM6,I+J-1) = XPAL(NUM6,I) + ((J - 1) * XINCX)
            YPAL(NUM6,I+J-1) = YPAL(NUM6,I) + ((J - 1) * YINCX)
20       CONTINUE
```

```
         K = K + 1
10     CONTINUE

C      THE Z VALUE FOR ALL PALLET POINTS ARE SET TO ZERO.
C      THE R VALUE FOR ALL PALLET POINTS IS SET TO THE R VALUE
C      OF THE LOWER LEFT CORNER POINT.

       DO 30 I = 1,PALPTS(NUM6)
          ZPAL(NUM6,I) = 0.0
          RPAL(NUM6,I) = R1
30     CONTINUE

C      THE INITIAL PART NUMBER FOR THE PALLET IS SET TO ONE.

       PARTNO(NUM6) = 1
       NUM6 = NUM6 + 1

100    FORMAT(14X,10(1X,I4))
200    FORMAT(4X,4(2X,F8.2))
300    FORMAT(22X,F8.2)

       RETURN
       END

C      SUBROUTINE MOT

C      THIS SUBROUTINE HANDLES ALL THE MOTION COMMANDS.

C      PARAMETER DEFINITION:

C           K2        -       SUBTYPE OF COMMAND
C                               1 - DELAY
C                               2 - DPMOVE
C                               3 - GRASP
C                               4 - PMOVE
C                               5 - RELEASE
C                               6 - ZMOVE
C                               7 - LINEAR
C                               8 - PAYLOAD
C                               9 - ZONE

C      SUBROUTINES CALLED BY THIS PROGRAM ARE :
C              CHECK,ERROR,USER,WRITEP,WRITER.

       SUBROUTINE MOT(K2)

       INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
       INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
       INTEGER SS,SLPAY,PART1,PART2

       DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
       DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
       DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
       DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
       DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

       COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
       COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
       COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
       COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
       COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
       COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
       COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

C      DELAY

C      READ THE OBJECT FILE FOR THE COMMAND PARAMETER. L1 IS 1 IF THE
C      PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A NUMBER. L2 IS THE
```

```
C     VARIABLE / CONSTANT NUMBER.
C     L1 AND L2 REPRESENT THE TIME VALUE.

      IF (K2 .EQ. 1) THEN
         READ (7,100,REC=NEXT) L1,L2

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO TIME.

         IF (L1 .EQ. 1) THEN

            DO 10 I = 1,NUM1-1
               IF (L2 .EQ. CONS(I)) THEN
                  TIME = CONVAL(I)
                  GOTO 11
               ENDIF

10          CONTINUE

            DO 12 I = 1,NUM3-1

               IF (L2 .EQ. COUNT(I)) THEN
                  TIME = CNTVAL(I)
                  GOTO 11
               ENDIF

12          CONTINUE

            DO 19 I = 1,NUM5-1

               IF (L2 .EQ. PAR(I)) THEN
                  READ (8,500,REC=L2) TIME
                  GOTO 11
               ENDIF

19          CONTINUE

C     IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C     FILE, AND ASSIGNED TO TIME.

         ELSEIF (L1 .EQ. 2) THEN
            READ (4,200,REC=L2) TIME
         ENDIF

C     THE VALUE IS CHECKED TO BE WITHIN VALID LIMITS, AND IF NOT
C     AN ERROR MESSAGE IS DISPLAYED.

11       IF (TIME .LT. 0.0 .OR. TIME .GT. 25.5) CALL USER(1)
         CODE = 2.0
         IER = 0
         TRY = 1.0
         CALL CHECK

C     THE VALUE IS TRANSMITTED TO THE CONTROLLER ALONG WITH A CODE
C     VALUE OF 2.

551      CALL WRITER(40,TIME,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 551
         ENDIF

         IER = 0
         TRY = 1.0
```

```
552      CALL WRITER(34,CODE,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 552
         ENDIF

         IER = 0

C     DPMOVE

C     READ THE OBJECT FILE FOR THE COMMAND PARAMETERS. L1 IS 1 IF THE
C     PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A NUMBER. L2 IS THE
C     VARIABLE / CONSTANT NUMBER.
C     L1 AND L2 REPRESENT THE AGGREGATE VALUE.

      ELSEIF (K2 .EQ. 2) THEN
         READ (7,100,REC=NEXT) L1,L2

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF AGGREGATES
C     OR PARAMETERS, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUES ARE ADDED TO XVAL,YVAL,RVAL, AND ZVAL.

         IF (L1 .EQ. 1) THEN

            DO 20 I = 1,NUM4-1

               IF (L2 .EQ. AGG(I)) THEN
                  XVAL = XVAL + XAGG(I)
                  YVAL = YVAL + YAGG(I)
                  ZVAL = ZVAL + ZAGG(I)
                  RVAL = RVAL + RAGG(I)
                  GOTO 21
               ENDIF

20          CONTINUE

               DO 29 I = 1,NUM5-1

                  IF (L2 .EQ. PAR(I)) THEN
                     READ (8,500,REC=L2) XAGG(I),YAGG(I),ZAGG(I),RAGG(I)
                     XVAL = XVAL + XAGG(I)
                     YVAL = YVAL + YAGG(I)
                     ZVAL = ZVAL + ZAGG(I)
                     RVAL = RVAL + RAGG(I)
                     GOTO 21
                  ENDIF

29             CONTINUE

C     IF THE VALUE IS A NUMBER, THAT VALUE AND THE NEXT THREE VALUES
C     ARE READ FROM THE OBJECT FILE, AND ADDED TO XVAL,YVAL,ZVAL, AND
C     RVAL.

         ELSEIF (L1 .EQ. 2) THEN
            READ (4,200,REC=L2) X
            READ (4,200,REC=L2+1) Y
            READ (4,200,REC=L2+2) Z
            READ (4,200,REC=L2+3) R
            XVAL = XVAL + X
            YVAL = YVAL + Y
            ZVAL = ZVAL + Z
            RVAL = RVAL + R
            GOTO 21
```

```
C     IF THE VALUES ARE SPECIFIED AS A SET OF COUNTERS, THE FOUR COUNTER
C     VALUES ARE READ FROM THE SYMBOL TABLE FILE AND ADDED TO THE
C     VARIABLES XVAL,YVAL,ZVAL, AND RVAL.

         ELSEIF (L1 .EQ. 3) THEN
            READ (7,600,REC=NEXT) L3,L4,L5
            READ (8,500,REC=L2) X
            READ (8,500,REC=L3) Y
            READ (8,500,REC=L4) Z
            READ (8,500,REC=L5) R
            XVAL = XVAL + X
            YVAL = YVAL + Y
            ZVAL = ZVAL + Z
            RVAL = RVAL + R
         ENDIF

C     THE R AND Z VALUES ARE CHECKED TO BE WITHIN VALID LIMITS,
C     AND IF NOT, AN ERROR MESSAGE IS DISPLAYED.

21       CODE = 6.0
         IER = 0
         TRY = 1.0
         IF (RVAL .LT. -180.0 .OR. RVAL .GT. 180.0) CALL USER(2)
         IF (ZVAL .LT. -250.0 .OR. ZVAL .GT. 0.0) CALL USER(3)
         CALL CHECK

C     THE COORDINATES OF THE POINT ARE THEN TRANSMITTED AFTER
C     CHECKING THAT THE CONTROLLER IS READY TO ACCEPT DATA. THE
C     CODE VALUE OF 6.0 IS ALSO TRANSMITTED.

553      CALL WRITEP(42,XVAL,YVAL,ZVAL,RVAL,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 553
         ENDIF

         IER = 0
         TRY = 1.0
554      CALL WRITER(34,CODE,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 554
         ENDIF

         IER = 0

C     GRASP

C     THE DO PORT IS SET TO 2, AND ITS VALUE TO 1. THESE VALUES ALONG
C     WITH A CODE VALUE OF 9.0 IS TRANSMITTED TO THE CONTROLLER.

      ELSEIF (K2 .EQ. 3) THEN
         IONUM(18) = 1
         CODE = 9.0
         IER = 0
         TRY = 1.0
         CALL CHECK
555      CALL WRITER(39,2.0,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
```

```
            GOTO 555
         ENDIF

         IER = 0
         TRY = 1.0

556      CALL WRITER(38,1.0,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 556
         ENDIF

         IER = 0
         TRY  = 1.0

557      CALL WRITER(34,CODE,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 557
         ENDIF

         IER = 0

C     PMOVE

C     READ THE OBJECT FILE FOR THE COMMAND PARAMETERS. L1 IS 1 IF THE
C     PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A NUMBER. L2 IS THE
C     VARIABLE / CONSTANT NUMBER.
C     L1 AND L2 REPRESENT THE POINT VALUE.

      ELSEIF (K2 .EQ. 4) THEN
         READ (7,100,REC=NEXT) L1,L2

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF POINTS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUES ARE ASSIGNED TO XVAL,YVAL,ZVAL, AND RVAL.

         IF (L1 .EQ. 1) THEN

            DO 40 I = 1,NUM2-1

               IF (L2 .EQ. PNTVAL(I)) THEN
                  XVAL = XPNT(I)
                  YVAL = YPNT(I)
                  ZVAL = ZPNT(I)
                  RVAL = RPNT(I)
                  GOTO 41
               ENDIF

40          CONTINUE

               DO 49 I = 1,NUM5-1

                  IF (L2 .EQ. PAR(I)) THEN
                     READ (8,500,REC=L2) XVAL,YVAL,ZVAL,RVAL
                     GOTO 41
                  ENDIF

49             CONTINUE

C     IF THE VALUE IS A SET OF NUMBERS, THEIR VALUES ARE READ FROM THE
C     CONSTANTS FILE, AND ASSIGNED TO XVAL,YVAL,ZVAL, AND RVAL.
```

```
         ELSEIF (L1 .EQ. 2) THEN
            READ (4,200,REC=L2) XVAL
            READ (4,200,REC=L2+1) YVAL
            READ (4,200,REC=L2+2) ZVAL
            READ (4,200,REC=L2+3) RVAL
            GOTO 41

C     IF THE VALUES ARE SPECIFIED AS A SET OF COUNTERS, THE FOUR COUNTER
C     VALUES ARE READ FROM THE SYMBOL TABLE FILE AND ASSIGNED TO THE
C     VARIABLES XVAL,YVAL,ZVAL, AND RVAL.

         ELSEIF (L1 .EQ. 3) THEN
            READ (7,600,REC=NEXT) L3,L4,L5
            READ (8,500,REC=L2) XVAL
            READ (8,500,REC=L3) YVAL
            READ (8,500,REC=L4) ZVAL
            READ (8,500,REC=L5) RVAL
         ENDIF

41       CODE = 6.0
         IER = 0
         TRY = 1.0

C     THE R AND Z VALUES ARE CHECKED TO BE WITHIN VALID LIMITS,
C     AND IF NOT, AN ERROR MESSAGE IS DISPLAYED.

         IF (RVAL .LT. -180.0 .OR. RVAL .GT. 180.0) CALL USER(2)
         IF (ZVAL .LT. -250.0 .OR. ZVAL .GT. 0.0) CALL USER(3)
         CALL CHECK

C     THE COORDINATES OF THE POINT ARE THEN TRANSMITTED AFTER
C     CHECKING THAT THE CONTROLLER IS READY TO ACCEPT DATA. THE
C     CODE VALUE OF 6.0 IS ALSO TRANSMITTED.

558      CALL WRITEP(42,XVAL,YVAL,ZVAL,RVAL,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 558
         ENDIF

         IER = 0
         TRY  = 1.0

559      CALL WRITER(34,CODE,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 559
         ENDIF

         IER = 0

C     RELEASE

C     THE DO PORT IS SET TO 2, AND ITS VALUE TO 0. THESE VALUES ALONG
C     WITH A CODE VALUE OF 9.0 IS TRANSMITTED TO THE CONTROLLER.

      ELSEIF (K2 .EQ. 5) THEN
         IONUM(18) = 0
         CODE = 9.0
         IER = 0
         TRY = 1.0
         CALL CHECK
```

```
560      CALL WRITER(39,2.0,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 560
         ENDIF

         IER = 0
         TRY = 1.0

561      CALL WRITER(38,0.0,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 561
         ENDIF

         IER = 0
         TRY = 1.0

562      CALL WRITER(34,CODE,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 562
         ENDIF

         IER = 0

C     ZMOVE

C     READ THE OBJECT FILE FOR THE COMMAND PARAMETERS. L1 IS 1 IF THE
C     PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A NUMBER. L2 IS THE
C     VARIABLE / CONSTANT NUMBER.
C     L1 AND L2 REPRESENT THE Z VALUE.

      ELSEIF (K2 .EQ. 6) THEN
         READ (7,100,REC=NEXT) L1,L2

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO ZVAL.

         IF (L1 .EQ. 1) THEN

            DO 60 I = 1,NUM1-1

               IF (L2 .EQ. CONS(I)) THEN
                  ZVAL = CONVAL(I)
                  GOTO 61
               ENDIF

60          CONTINUE

            DO 62 I = 1,NUM3-1

               IF (L2 .EQ. COUNT(I)) THEN
                  ZVAL = CNTVAL(I)
                  GOTO 61
               ENDIF

62          CONTINUE
```

```
            DO 69 I = 1,NUM5-1

               IF (L2 .EQ. PAR(I)) THEN
                  READ (8,500,REC=L2) ZVAL
                  GOTO 61
               ENDIF

69          CONTINUE

C     IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C     FILE, AND ASSIGNED TO ZVAL.

            ELSEIF (L1 .EQ. 2) THEN
               READ (4,200,REC=L2) ZVAL
            ENDIF

61          CODE = 6.0
            IER = 0
            TRY = 1.0

C     THE R AND Z VALUES ARE CHECKED TO BE WITHIN VALID LIMITS,
C     AND IF NOT, AN ERROR MESSAGE IS DISPLAYED.

            IF (RVAL .LT. -180.0 .OR. RVAL .GT. 180.0) CALL USER(2)
            IF (ZVAL .LT. -250.0 .OR. ZVAL .GT. 0.0) CALL USER(3)
            CALL CHECK

C     THE COORDINATES OF THE POINT ARE THEN TRANSMITTED AFTER
C     CHECKING THAT THE CONTROLLER IS READY TO ACCEPT DATA. THE
C     CODE VALUE OF 6.0 IS ALSO TRANSMITTED.

563         CALL WRITEP(42,XVAL,YVAL,ZVAL,RVAL,IER)

            IF (IER .NE. 0) THEN
               TRY = TRY + 1.0
               IF (TRY .GE. 5.0) CALL ERROR
               IER = 0
               GOTO 563
            ENDIF

            IER = 0
            TRY = 1.0

564         CALL WRITER(34,CODE,IER)

            IF (IER .NE. 0) THEN
               TRY = TRY + 1.0
               IF (TRY .GE. 5.0) CALL ERROR
               IER = 0
               GOTO 564
            ENDIF

            IER = 0

C     LINEAR

C     READ THE OBJECT FILE FOR THE COMMAND PARAMETERS. L1 IS 1 IF THE
C     PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A NUMBER. L2 IS THE
C     VARIABLE / CONSTANT NUMBER.
C     L1 AND L2 REPRESENT THE LINEAR VALUE.

         ELSEIF (K2 .EQ. 7) THEN
            READ (7,100,REC=NEXT) L1,L2

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO ALIN.

            IF (L1 .EQ. 1) THEN
```

```
            DO 70 I = 1,NUM1-1

               IF (L2 .EQ. CONS(I)) THEN
                  ALIN = CONVAL(I)
                  GOTO 71
               ENDIF

70          CONTINUE

            DO 72 I = 1,NUM3-1

               IF (L2 .EQ. COUNT(I)) THEN
                  ALIN = CNTVAL(I)
                  GOTO 71
               ENDIF

72          CONTINUE

            DO 79 I = 1,NUM5-1

               IF (L2 .EQ. PAR(I)) THEN
                  READ (8,500,REC=L2) ALIN
                  GOTO 71
               ENDIF

79          CONTINUE

C     IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C     FILE, AND ASSIGNED TO ALIN.

            ELSEIF (L1 .EQ. 2) THEN
               READ (4,200,REC=L2) ALIN
            ENDIF

71          IER = 0
            TRY = 1.0

C     THE LINEAR VALUE IS CHECKED TO BE WITHIN VALID LIMITS,
C     AND IF NOT, AN ERROR MESSAGE IS DISPLAYED.

            IF (ALIN .LT. 0.0 .OR. ALIN .GT. 50.0) CALL USER(4)
            CALL CHECK

C     THE LINEAR VALUE IS THEN TRANSMITTED TO THE CONTROLLER AFTER
C     ENSURING THAT THE CONTROLLER IS READY TO RECEIVE DATA.

565         CALL WRITER(35,ALIN,IER)

            IF (IER .NE. 0) THEN
               TRY = TRY + 1.0
               IF (TRY .GE. 5.0) CALL ERROR
               IER = 0
               GOTO 565
            ENDIF

            IER = 0

C     PAYLOAD

C     READ THE OBJECT FILE FOR THE COMMAND PARAMETERS. L1 IS 1 IF THE
C     PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A NUMBER. L2 IS THE
C     VARIABLE / CONSTANT NUMBER.
C     L1 AND L2 REPRESENT THE PAYLOAD VALUE.

         ELSEIF (K2 .EQ. 8) THEN
            READ (7,100,REC=NEXT) L1,L2
```

```
C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO PAY.

         IF (L1 .EQ. 1) THEN

            DO 80 I = 1,NUM1-1

               IF (L2 .EQ. CONS(I)) THEN
                  PAY = CONVAL(I)
                  GOTO 81
               ENDIF

80          CONTINUE

            DO 82 I = 1,NUM3-1

               IF (L2 .EQ. COUNT(I)) THEN
                  PAY = CNTVAL(I)
                  GOTO 81
               ENDIF

82          CONTINUE

            DO 89 I = 1,NUM5-1

               IF (L2 .EQ. PAR(I)) THEN
                  READ (8,500,REC=L2) PAY
                  GOTO 81
               ENDIF

89          CONTINUE

C     IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C     FILE, AND ASSIGNED TO PAY.

         ELSEIF (L1 .EQ. 2) THEN
            READ (4,200,REC=L2) PAY
         ENDIF

81       IER = 0
         TRY = 1.0

C     THE PAYLOAD VALUE IS CHECKED TO BE WITHIN VALID LIMITS,
C     AND IF NOT, AN ERROR MESSAGE IS DISPLAYED.
C     THE VARIABLE PAYOLD IS SET TO PAY. IF SLOW ROBOT
C     ARM MOVEMENT IS IN EFFECT, THE VARIABLE PAY IS COMPARED
C     TO SLPAY, AND THE ONE THAT GIVES SLOWER MOTION IS
C     TRANSMITTED.

         IF (PAY .LT. 0.0 .OR. PAY .GT. 19.0) CALL USER(5)
         PAYOLD = PAY

         IF (SLPAY .GT. 0) THEN

            IF (PAY .GT. SLPAY .AND. PAY .LE. 10.0) THEN
               PAY = SLPAY
            ENDIF

         ENDIF

         CALL CHECK

C     THE PAYLOAD VALUE IS THEN TRANSMITTED TO THE CONTROLLER AFTER
C     ENSURING THAT THE CONTROLLER IS READY TO RECEIVE DATA.

566      CALL WRITER(36,PAY,IER)

         IF (IER .NE. 0) THEN
```

```
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 566
         ENDIF

         IER = 0

C     ZONE

C     READ THE OBJECT FILE FOR THE COMMAND PARAMETERS. L1 IS 1 IF THE
C     PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A NUMBER. L2 IS THE
C     VARIABLE / CONSTANT NUMBER.
C     L1 AND L2 REPRESENT THE ZONE VALUE.

      ELSEIF (K2 .EQ. 9) THEN
         READ (7,100,REC=NEXT) L1,L2

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO ZON.

         IF (L1 .EQ. 1) THEN

            DO 90 I = 1,NUM1-1

               IF (L2 .EQ. CONS(I)) THEN
                  ZON = CONVAL(I)
                  GOTO 91
               ENDIF

90          CONTINUE

            DO 92 I = 1,NUM3-1

               IF (L2 .EQ. COUNT(I)) THEN
                  ZON = CNTVAL(I)
                  GOTO 91
               ENDIF

92          CONTINUE

            DO 99 I = 1,NUM5-1

               IF (L2 .EQ. PAR(I)) THEN
                  READ (8,500,REC=L2) ZON
                  GOTO 91
               ENDIF

99          CONTINUE

C     IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C     FILE, AND ASSIGNED TO ZON.

         ELSEIF (L1 .EQ. 2) THEN
            READ (4,200,REC=L2) ZON
         ENDIF

91       IER = 0
         TRY = 1.0

C     THE ZONE VALUE IS CHECKED TO BE WITHIN VALID LIMITS,
C     AND IF NOT, AN ERROR MESSAGE IS DISPLAYED.

         IF (ZON .LT. 0.0 .OR. ZON .GT. 15.0) CALL USER(6)
         CALL CHECK

C     THE ZONE VALUE IS THEN TRANSMITTED TO THE CONTROLLER AFTER
C     ENSURING THAT THE CONTROLLER IS READY TO RECEIVE DATA.
```

```
567      CALL WRITER(37,ZON,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 567
         ENDIF

         IER = 0

      ENDIF

C     THE NEXT STATEMENT TO BE EXECUTED IS ASSIGNED, AND CONTROL RETURNS
C     TO SUBROUTINE SYS.

      NEXT = NEXT + 1

100   FORMAT(17X,2(1X,I4))
200   FORMAT(22X,F8.2)
300   FORMAT(2X,I1,4(2X,F8.2))
400   FORMAT(2X,I1,2X,I8)
500   FORMAT(4X,4(2X,F8.2))
600   FORMAT(27X,3(1X,I4))

      RETURN
      END

C     SUBROUTINE SENS

C     THIS SUBROUTINE HANDLES ALL THE SENSOR COMMANDS.

C     PARAMETER DEFINITION:

C          K2        -       SUBTYPE OF COMMAND
C                              1 - WAITI
C                              2 - WRITEO


C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             CHECK,ERROR,READR,USER,WRITER.

      SUBROUTINE SENS(K2)

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

C     WAITI

C     READ THE OBJECT FILE FOR THE COMMAND PARAMETERS. L1 IS 1 IF THE
C     PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A NUMBER. L2 IS THE
C     VARIABLE / CONSTANT NUMBER. L3 IS 1 IF THE PARAMETER IS A VARIABLE
C     NAME AND 2 IF IT IS A NUMBER. L4 IS THE VARIABLE / CONSTANT NUMBER.
```

```
C     L5 IS 1 IF THE PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A
C     NUMBER. L6 IS THE VARIABLE / CONSTANT NUMBER.
C     L7 IS THE VARIABLE NUMBER OF THE LABEL, WHICH IS OPTIONAL.
C     L1 AND L2 REPRESENT THE DI PORT, L3 AND L4 REPRESENT THE
C     DI VALUE, L5 AND L6 REPRESENT THE TIME LIMIT.

      IF (K2 .EQ. 1) THEN
         READ (7,200,REC=NEXT) L1,L2,L3,L4,L5,L6,L7
         L8 = 0

C     THE SYMBOL TABLE FILE IS READ FOR THE STATEMENT NUMBER OF THE LABEL,
C     AND THIS VALUE IS ASSIGNED TO L8. THIS HAPPENS ONLY IF THERE IS A
C     LABEL IN THE STATEMENT., OTHERWISE L8 IS SET TO ZERO.

         IF (L7 .GT. 0) THEN
            READ (8,210,REC=L7) CC
            L8 = CC
         ENDIF

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO L2.

         IF (L1 .EQ. 1) THEN

            DO 10 I = 1,NUM1-1

               IF (L2 .EQ. CONS(I)) THEN
                  L2 = CONVAL(I)
                  GOTO 11
               ENDIF

10          CONTINUE

            DO 12 I = 1,NUM3-1

               IF (L2 .EQ. COUNT(I)) THEN
                  L2 = CNTVAL(I)
                  GOTO 11
               ENDIF

12          CONTINUE

            DO 19 I = 1,NUM5-1

               IF (L2 .EQ. PAR(I)) THEN
                  READ (8,500,REC=L2) T
                  L2 = T
                  GOTO 11
               ENDIF

19          CONTINUE

C     IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C     FILE, AND ASSIGNED TO L2.

         ELSEIF (L1 .EQ. 2) THEN
           READ (4,300,REC=L2) T
           L2 = T
           GOTO 11
         ENDIF

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO L4.

11       IF (L3 .EQ. 1) THEN

            DO 13 I = 1,NUM1-1
```

```
                IF (L4 .EQ. CONS(I)) THEN
                   L4 = CONVAL(I)
                   GOTO 14
                ENDIF

13           CONTINUE

             DO 15 I = 1,NUM3-1

                IF (L4 .EQ. COUNT(I)) THEN
                   L4 = CNTVAL(I)
                   GOTO 14
                ENDIF

15           CONTINUE

             DO 29 I = 1,NUM5-1

                IF (L2 .EQ. PAR(I)) THEN
                   READ (8,500,REC=L4) T
                   L4 = T
                   GOTO 14
                ENDIF

29           CONTINUE

C      IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C      FILE, AND ASSIGNED TO L4.

           ELSEIF (L1 .EQ. 2) THEN
             READ (4,300,REC=L4) T
             L4 = T
             GOTO 14
           ENDIF

C      THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C      AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C      VALUE IS ASSIGNED TO THE VARIABLE TIME.

14         IF (L5 .EQ. 1) THEN

             DO 16 I = 1,NUM1-1

                IF (L6 .EQ. CONS(I)) THEN
                   TIME = CONVAL(I)
                   GOTO 17
                ENDIF

16           CONTINUE

             DO 18 I = 1,NUM3-1

                IF (L6 .EQ. COUNT(I)) THEN
                   TIME = CNTVAL(I)
                   GOTO 17
                ENDIF

18           CONTINUE

             DO 39 I = 1,NUM5-1

                IF (L6 .EQ. PAR(I)) THEN
                   READ (8,500,REC=L6) T
                   TIME = T
                   GOTO 17
                ENDIF

39           CONTINUE
```

```
C     IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C     FILE, AND ASSIGNED TO TIME.

         ELSEIF (L5 .EQ. 2) THEN
           READ (4,300,REC=L6) TIME
           GOTO 17
         ENDIF

C     IF THE WAITI COMMAND HAS AN INDEFINITE WAIT, THE CODE VALUE IS
C     SET TO 7.0. IF THERE IS A SPECIFIC TIME LIMIT THE CODE VALUE
C     IS SET TO 8.0.

17       IF (TIME .EQ. 0.0) THEN
            CODE = 7.0
         ELSE
            CODE = 8.0
         ENDIF

C     THE DI NUMBER IS CHECKED TO BE WITHIN VALID LIMITS, AND ITS VALUE
C     TO BE ZERO OR ONE, AND THE TIME LIMIT TO BE WITHIN THE VALID RANGE,
C     OTHERWISE ERROR MESSAGES ARE DISPLAYED.
C     THE VALUES OF THE DI NUMBER, DI VALUE AND THE TIME VALUE ARE
C     TRANSMITTED TOT THE CONTROLLER, ALONG WITH THE CODE VALUE.

         IER = 0
         TRY = 1.0
         IF (L2 .LT. 1 .OR. L2 .GT. 16) CALL USER(7)
         VAL1 = L2
         IF (L4 .NE. 0 .AND. L4 .NE. 1) CALL USER(8)
         VAL2 = L4
         IF (TIME .LT. 0.0 .OR. TIME .GT. 25.5) CALL USER(1)
         CALL CHECK
551      CALL WRITER(39,VAL1,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 551
         ENDIF

         IER = 0
         TRY = 1.0
552      CALL WRITER(38,VAL2,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 552
         ENDIF

         IER = 0
         TRY = 1.0
553      CALL WRITER(40,TIME,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 553
         ENDIF

         IER = 0
         TRY = 1.0
554      CALL WRITER(34,CODE,IER)

         IF (IER .NE. 0) THEN
```

```
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 554
         ENDIF

C     ONCE THE COMMAND HAS BEEN EXECUTED, THE CONTROLLER IS READ TO
C     DETERMINE THE OUTCOME OF THE WAITI COMMAND, BY READING THE
C     VARIABLE RETVAL.

         IER = 0
         TRY = 1.0
         CALL CHECK
556      CALL READR(41,RETVAL,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 556
         ENDIF

         IER = 0

C     RETVAL TAKES A VALUE OF 10.0, IF THE DI PORT MET THE CONDITION IN
C     THE TIME LIMIT, OTHERWISE IT IS SET TO 0. IF THE CONDITION IS NOT
C     MET, AND THERE IS A LABEL TO BRANCH TO, CONTROL TRANSFERS TO THE
C     STATEMENT NUMBER OF THE LABEL. OTHERWISE, AND ERROR MESSAGE IS
C     DISPLAYED.

         IF (RETVAL .EQ. 10.0) THEN
            NEXT = NEXT + 1
         ELSEIF (RETVAL .EQ. 0.0) THEN

            IF (L8 .NE. 0) THEN
               NEXT = L8
            ELSE
               CALL USER(9)
            ENDIF

         ENDIF

C     WRITEO

C     READ THE OBJECT FILE FOR THE COMMAND PARAMETERS. L1 IS 1 IF THE
C     PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A NUMBER. L2 IS THE
C     VARIABLE / CONSTANT NUMBER. L3 IS 1 IF THE PARAMETER IS A VARIABLE
C     NAME AND 2 IF IT IS A NUMBER. L4 IS THE VARIABLE / CONSTANT NUMBER.
C     L1 AND L2 REPRESENT THE DO PORT, L3 AND L4 REPRESENT THE
C     DO VALUE.

      ELSEIF (K2 .EQ. 2) THEN
         READ (7,200,REC=NEXT) L1,L2,L3,L4

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO L2.

         IF (L1 .EQ. 1) THEN

            DO 20 I = 1,NUM1-1

               IF (L2 .EQ. CONS(I)) THEN
                  L2 = CONVAL(I)
                  GOTO 21
               ENDIF

20          CONTINUE
```

```
            DO 22 I = 1,NUM3-1

               IF (L2 .EQ. COUNT(I)) THEN
                  L2 = CNTVAL(I)
                  GOTO 21
               ENDIF

22          CONTINUE

            DO 49 I = 1,NUM5-1

               IF (L2 .EQ. PAR(I)) THEN
                  READ (8,500,REC=L2) T
                  L2 = T
                  GOTO 21
               ENDIF

49          CONTINUE

C     IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C     FILE, AND ASSIGNED TO L2.

         ELSEIF (L1 .EQ. 2) THEN
            READ (4,300,REC=L2) T
            L2 = T
         ENDIF

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO L4.

21       IF (L3 .EQ. 1) THEN

            DO 23 I = 1,NUM1-1

               IF (L4 .EQ. CONS(I)) THEN
                  L4 = CONVAL(I)
                  GOTO 25
               ENDIF

23          CONTINUE

            DO 24 I = 1,NUM3-1

               IF (L4 .EQ. COUNT(I)) THEN
                  L4 = CNTVAL(I)
                  GOTO 25
               ENDIF

24          CONTINUE

            DO 59 I = 1,NUM5-1

               IF (L4 .EQ. PAR(I)) THEN
                  READ (8,500,REC=L4) T
                  L4 = T
                  GOTO 25
               ENDIF

59          CONTINUE

C     IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C     FILE, AND ASSIGNED TO L4.

         ELSEIF (L3 .EQ. 2) THEN
            READ (4,300,REC=L4) T
            L4 = T
         ENDIF
```

```
25       L2 = L2 + 16
         IONUM(L2) = L4
         IF (L2 .LT. 16 .OR. L2 .GT. 32) CALL USER(10)
         VAL1 = L2 - 16
         IF (L4 .NE. 0 .AND. L4 .NE. 1) CALL USER(8)
         VAL2 = L4
         CODE = 9.0
         IER = 0
         TRY = 1.0
         CALL CHECK
557      CALL WRITER(39,VAL1,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 557
         ENDIF

         IER = 0
         TRY = 1.0
558      CALL WRITER(38,VAL2,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 558
         ENDIF

         IER = 0
         TRY = 1.0
559      CALL WRITER(34,CODE,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 559
         ENDIF

         IER = 0

         NEXT = NEXT + 1

      ENDIF

200   FORMAT(17X,7(1X,I4))
210   FORMAT(6X,F8.2)
300   FORMAT(22X,F8.2)
400   FORMAT(2X,I1,2(2X,I8))
500   FORMAT(4X,4(2X,F8.2))
900   FORMAT(2X,I1,2X,I8,2X,I8,2X,F8.2,2X,I8)

      RETURN
      END

C     SUBROUTINE FLOW

C     THIS SUBROUTINE HANDLES ALL THE FLOW OF CONTROL COMMANDS.

C     PARAMETER DEFINITION:

C          K2       -      SUBTYPE OF COMMAND
C                            1 - BRANCH
C                            2 - TESTC
C                            3 - TESTI
C                            4 - TESTP
C                            5 - BREAKPOINT
```

```
C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             CHECK,ERROR,RDIDO,USER,WRITER.

      SUBROUTINE FLOW(K2)

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

C     BRANCH

C     READ THE OBJECT FILE FOR THE VARIABLE NUMBER OF THE LABEL. READ THE
C     SYMBOL TABLE FILE FOR THE STATEMENT NUMBER OF THE LABEL, AND
C     ASSIGN IT TO THE VARIABLES L3 AND NEXT, WHICH IS THE NEXT
C     STATEMENT NUMBER TO BE EXECUTED.
C     L1 AND L2 REPRESENT THE LABEL.

      IF (K2 .EQ. 1) THEN
         READ (7,100,REC=NEXT) L1,L2
         READ (8,110,REC=L2) CC
         L3 = CC
         NEXT = L3

C     TESTC

C     READ THE OBJECT FILE FOR THE COMMAND PARAMETERS. L1 IS 1 IF THE
C     PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A NUMBER. L2 IS THE
C     VARIABLE / CONSTANT NUMBER. L3 IS 1 IF THE PARAMETER IS A VARIABLE
C     NAME AND 2 IF IT IS A NUMBER. L4 IS THE VARIABLE / CONSTANT NUMBER.
C     L5 IS THE VARIABLE NUMBER OF THE LABEL.
C     THE SYMBOL TABLE FILE IS READ FOR THE STATEMENT NUMBER OF THE LABEL,
C     AND THIS VALUE IS ASSIGNED TO L6.
C     L1 AND L2 REPRESENT THE COUNTER NAME, L3 AND L4 REPRESENT THE
C     COUNTER VALUE, L5 REPRESENTS THE LABEL.


      ELSEIF (K2 .EQ. 2) THEN
         READ (7,100,REC=NEXT) L1,L2,L3,L4,L5
         READ (8,110,REC=L5) CC
         L6 = CC

C     THE COUNTER NAME IS COMPARED AGAINST THE NAMES OF COUNTERS AND
C     PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO L2.

         DO 20 I = 1,NUM3-1

            IF (L2 .EQ. COUNT(I)) THEN
               L2 = CNTVAL(I)
               GOTO 21
            ENDIF

20       CONTINUE
```

```
            DO 27 I = 1,NUM5-1

               IF (L2 .EQ. PAR(I)) THEN
                  READ (8,500,REC=L2) T
                  L2 = T
                  GOTO 21
               ENDIF

27          CONTINUE

C        IF THE VALUE TO BE COMPARED TO IS A VARIABLE NAME, IT IS CHECKED
C        CHECKED AGAINST NAMES OF CONSTANTS, COUNTERS, AND PARAMETERS TILL
C        A MATCH IS FOUND. THEN ITS VALUE IS ASSIGNED TO L4.

21          IF (L3 .EQ. 1) THEN

               DO 22 I = 1,NUM1-1

                  IF (L4 .EQ. CONS(I)) THEN
                     L4 = CONVAL(I)
                     GOTO 23
                  ENDIF

22             CONTINUE

               DO 24 I = 1,NUM3-1

                  IF (L4 .EQ. COUNT(I)) THEN
                     L4 = CNTVAL(I)
                     GOTO 23
                  ENDIF

24             CONTINUE

               DO 28 I = 1,NUM5-1

                  IF (L4 .EQ. PAR(I)) THEN
                     READ (8,500,REC=L4) T
                     L4 = T
                     GOTO 23
                  ENDIF

28             CONTINUE

C        IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C        FILE, AND ASSIGNED TO L4.

            ELSEIF (L3 .EQ. 2) THEN
               READ (4,200,REC=L4) T
               L4 = T
            ENDIF

C        THE VALUE IS CHECKED TO BE WITHIN VALID LIMITS, AND IF NOT, AN
C        ERROR MESSAGE IS DISPLAYED. OTHERWISE THE TEST IS DONE, AND
C        DEPENDING ON THE OUTCOME, THE NEXT STATEMENT TO BE EXECUTED
C        IS DETERMINED.

23          IF (L4 .GT. 32767 .OR. L4 .LT. -32767) CALL USER(11)

            IF (L2 .EQ. L4) THEN
               NEXT = L6
            ELSE
               NEXT = NEXT + 1
            ENDIF

C        TESTI

C        READ THE OBJECT FILE FOR THE COMMAND PARAMETERS. L1 IS 1 IF THE
C        PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A NUMBER. L2 IS THE
```

```
C     VARIABLE / CONSTANT NUMBER. L3 IS 1 IF THE PARAMETER IS A VARIABLE
C     NAME AND 2 IF IT IS A NUMBER. L4 IS THE VARIABLE / CONSTANT NUMBER.
C     L5 IS THE VARIABLE NUMBER OF THE LABEL.
C     THE SYMBOL TABLE FILE IS READ FOR THE STATEMENT NUMBER OF THE LABEL,
C     AND THIS VALUE IS ASSIGNED TO L6.
C     L1 AND L2 REPRESENT THE DI PORT, L3 AND L4 REPRESENT THE
C     DI VALUE, L5 REPRSENTS THE LABEL.


      ELSEIF (K2 .EQ. 3) THEN
         READ (7,100,REC=NEXT) L1,L2,L3,L4,L5
         READ (8,110,REC=L5) CC
         L6 = CC

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO L2.

         IF (L1 .EQ. 1) THEN

            DO 30 I = 1,NUM1-1

               IF (L2 .EQ. CONS(I)) THEN
                  L2 = CONVAL(I)
                  GOTO 31
               ENDIF

30          CONTINUE

            DO 37 I = 1,NUM3-1

               IF (L2 .EQ. COUNT(I)) THEN
                  L2 = CNTVAL(I)
                  GOTO 31
               ENDIF

37          CONTINUE

            DO 38 I = 1,NUM5-1

               IF (L2 .EQ. PAR(I)) THEN
                  READ (8,500,REC=L2) T
                  L2 = T
                  GOTO 31
               ENDIF

38          CONTINUE

C     IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C     FILE, AND ASSIGNED TO L2.

         ELSEIF (L1 .EQ. 2) THEN
            READ (4,200,REC=L2) T
            L2 = T
         ENDIF

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO L4.

31       IF (L3 .EQ. 1) THEN

            DO 32 I = 1,NUM1-1

               IF (L4 .EQ. CONS(I)) THEN
                  L4 = CONVAL(I)
                  GOTO 33
               ENDIF
```

```
32          CONTINUE

            DO 34 I = 1,NUM3-1

               IF (L4 .EQ. COUNT(I)) THEN
                  L4 = CNTVAL(I)
                  GOTO 33
               ENDIF

34          CONTINUE

            DO 39 I = 1,NUM5-1

               IF (L4 .EQ. PAR(I)) THEN
                  READ (8,500,REC=L4) T
                  L4 = T
                  GOTO 33
               ENDIF

39          CONTINUE

C     IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C     FILE, AND ASSIGNED TO L4.

         ELSEIF (L3 .EQ. 2) THEN
            READ (4,200,REC=L4) T
            L4 = T
         ENDIF

C     THE DI PORT IS CHECKED TO BE WITHIN LIMITS, OTHERWISE AN ERROR
C     MESSAGE IS DISPLAYED.
C     THE VALUE OF THE DI PORT IS READ FROM THE CONTROLLER, AND ITS
C     VALUE IS RETURNED IN THE ARRAY IONUM.

33       IER = 0
         TRY = 1.0
         IF (L2 .LT. 1 .OR. L2 .GT. 16) CALL USER(7)
         CALL CHECK
551      CALL RDIDO(L2,IONUM(L2),IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 551
         ENDIF

C     THE DI VALUE IS CHECKED TO BE A 0 OR A 1, OTHERWISE AN ERROR
C     MESSAGE IS DISPLAYED.

         IER = 0
         IF (L4 .NE. 0 .AND. L4 .NE. 1) CALL USER(8)

C     THE DI VALUE IS THEN CHECKED AGAINST THE USER'S VALUE, AND BASED
C     ON THE OUTCOME, THE NEXT STATEMENT TO BE EXECUTED IS DETERMINED.

         IF (IONUM(L2) .EQ. L4) THEN
            NEXT = L6
         ELSE
            NEXT = NEXT + 1
         ENDIF

C     TESTP

C     READ THE OBJECT FILE FOR THE COMMAND PARAMETERS. L1 IS 1 IF THE
C     PARAMETER IS A VARIABLE NAME, AND 2 IF IT IS A NUMBER. L2 IS THE
C     VARIABLE / CONSTANT NUMBER. L3 IS 1 IF THE PARAMETER IS A VARIABLE
C     NAME AND 2 IF IT IS A NUMBER. L4 IS THE VARIABLE / CONSTANT NUMBER.
C     L5 IS THE VARIABLE NUMBER OF THE LABEL.
```

```
C     THE SYMBOL TABLE FILE IS READ FOR THE STATEMENT NUMBER OF THE LABEL,
C     AND THIS VALUE IS ASSIGNED TO L6.
C     L1 AND L2 REPRESENT THE PALLET NAME, L3 AND L4 REPRESENT THE
C     PALLET VALUE, L5 REPRESENTS THE LABEL.


      ELSEIF (K2 .EQ. 4) THEN
         READ (7,100,REC=NEXT) L1,L2,L3,L4,L5
         READ (8,110,REC=L5) CC
         L6 = CC

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF PALLETS
C     IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO L2.

         IF (L1 .EQ. 1) THEN

            DO 40 I = 1,NUM6-1

               IF (L2 .EQ. PALLET(I)) THEN
                  L2 = PARTNO(I)
                  GOTO 41
               ENDIF

40          CONTINUE

         ENDIF

C     THE VARIABLE NAME IS COMPARED AGAINST THE NAMES OF CONSTANTS, COUNTERS
C     AND PARAMETERS IN THE PROGRAM, AND ONCE THE MATCH HAS BEEN FOUND, ITS
C     VALUE IS ASSIGNED TO L4.

41       IF (L3 .EQ. 1) THEN

            DO 42 I = 1,NUM1-1

               IF (L4 .EQ. CONS(I)) THEN
                  L4 = CONVAL(I)
                  GOTO 43
               ENDIF

42          CONTINUE

            DO 44 I = 1,NUM3-1

               IF (L4 .EQ. COUNT(I)) THEN
                  L4 = CNTVAL(I)
                  GOTO 43
               ENDIF

44          CONTINUE

            DO 49 I = 1,NUM5-1

               IF (L4 .EQ. PAR(I)) THEN
                  READ (8,500,REC=L4) T
                  L4 = T
                  GOTO 43
               ENDIF

49          CONTINUE

C     IF THE VALUE IS A NUMBER, ITS VALUE IS READ FROM THE CONSTANTS
C     FILE, AND ASSIGNED TO L4.

         ELSEIF (L3 .EQ. 2) THEN
            READ (4,200,REC=L4) T
            L4 = T
         ENDIF
```

```
C     THE COMPARISON IS THEN MADE, AND BASED ON THE OUTCOME, THE NEXT
C     STATEMENT TO BE EXECUTED IS DETERMINED.

43       IF (L2 .EQ. L4) THEN
            NEXT = L6
         ELSE
            NEXT = NEXT + 1
         ENDIF

C     BREAKPOINT

C     WHEN THIS COMMAND IS ENCOUNTERD, THE PROGRAM TRANSMITS A CODE VALUE
C     OF 1 TO THE CONTROLLER, AND DISPLAYS A MESSAGE TO THE USER.

      ELSEIF (K2 .EQ. 5) THEN
         CODE = 1.0
         IER = 0
         TRY = 1.0
         CALL CHECK
552      CALL WRITER(34,CODE,IER)

         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 552
         ENDIF

         IER = 0

         CALL USER(12)
         NEXT = NEXT + 1

      ENDIF

99    FORMAT(2X,I1,3(2X,I8))
100   FORMAT(17X,5(1X,I4))
110   FORMAT(6X,F8.2)
200   FORMAT(22X,F8.2)
500   FORMAT(4X,4(2X,F8.2))

      RETURN
      END

C     SUBROUTINE COUN

C     THIS SUBROUTINE HANDLES ALL THE COUNTER COMMANDS.

C     PARAMETER DEFINITION:

C          K2        -       SUBTYPE OF COMMAND
C                              1 - COMPC
C                              2 - DECR
C                              3 - INCR
C                              4 - SETC

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             USER.

      SUBROUTINE COUN(K2)

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
```

```
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD


C     COMPC

C     READ THE COMMAND PARAMETER NUMBERS. L1 IS THE VARIABLE NUMBER OF
C     OF THE COUNTER. L2 IS THE CONDITION IN THE STATEMENT. L3 IS 2
C     IF THE VALUE IS A NUMBER AND 1 IF THE VALUE IS SPECIFIED AS A
C     COUNTER OR A CONSTANT. L5 IS THE LABEL NUMBER. THE SYMBOL TABLE
C     FILE IS READ AR RECORD L5, TO READ THE STATEMENT NUMBER OF THE
C     LABEL, AND L6 IS ASSIGNED THE STATEMENT NUMBER.
C     L1 REPRESENTS COUNTER NAME, L2 REPRESENTS THE CONDITION,,
C     L3 AND L4 REPRESENT THE COUNTER VALUE, L5 REPRESENTS THE LABEL.


      IF (K2 .EQ. 1) THEN
         READ (7,300,REC=NEXT) L1,L2,L3,L4,L5
         READ (8,310,REC=L5) CC
         L6 = CC

C     THE COUNTER TO BE COMPARED IS CHECKED AGAINST THE COUNTER NAMES
C     AND PARAMETER NAMES, AND ITS VALUE IS ASSIGNED TO VARIABLE J1.

            DO 10 I = 1,NUM3-1

               IF (L1 .EQ. COUNT(I)) THEN
                  J1 = CNTVAL(I)
                  GOTO 11
               ENDIF

10          CONTINUE

            DO 17 I = 1,NUM5-1

               IF (L1 .EQ. PAR(I)) THEN
                  READ (8,600,REC=L1) T
                  J1 = T
                  GOTO 11
               ENDIF

17          CONTINUE

C     THE VARIABLE NAME IS COMPARED AGAINST CONSTATNTS, COUNTERS, AND
C     PARAMETER NAMES, AND ITS VALUE IS ASSIGNED TO J2.

11          IF (L3 .EQ. 1) THEN

               DO 14 I = 1,NUM1-1

                  IF (L4 .EQ. CONS(I)) THEN
                     J2 = CONVAL(I)
                     GOTO 13
                  ENDIF

14             CONTINUE

               DO 12 I = 1,NUM3-1

                  IF (L4 .EQ. COUNT(I)) THEN
                     J2 = CNTVAL(I)
```

```
                  GOTO 13
               ENDIF

12          CONTINUE

         DO 18 I = 1,NUM5-1

            IF (L4 .EQ. PAR(I)) THEN
               READ (8,600,REC=L4) T
               J2 = T
               GOTO 13
            ENDIF

18       CONTINUE

C     IF THE VALUE IS A NUMBER, IT IS READ FROM THE CONSTANTS FILE, AND
C     ASSIGNED TO J2.

         ELSEIF (L3 .EQ. 2) THEN
            READ (4,400,REC=L4) T
               J2 = T
               GOTO 13
         ENDIF

C     THE CONDITIONS ARE COMPARED HERE. L2 IS 1 IF THE CONDITION IS
C     "<", 2 FOR "<=", 3 FOR ">", 4 FOR "=>", 5 FOR "=", AND 6 FOR "<>".
C     THE NEXT STATEMENT TO BE EXECUTED IS ASSIGNED TO THE VARIABLE NEXT,
C     DEPENDING ON THE OUTCOME OF THE COMPARISON.

13       IF (L2 .EQ. 1 .AND. J1 .LT. J2) THEN
            NEXT = L6
         ELSEIF (L2 .EQ. 2 .AND. J1 .LE. J2) THEN
            NEXT = L6
         ELSEIF (L2 .EQ. 3 .AND. J1 .GT. J2) THEN
            NEXT = L6
         ELSEIF (L2 .EQ. 4 .AND. J1 .GE. J2) THEN
            NEXT = L6
         ELSEIF (L2 .EQ. 5 .AND. J1 .EQ. J2) THEN
            NEXT = L6
         ELSEIF (L2 .EQ. 6 .AND. J1 .NE. J2) THEN
            NEXT = L6
         ELSE
            NEXT = NEXT + 1
         ENDIF

         GOTO 910

C     DECR

C     THE OBJECT FILE IS READ FOR THE COUNTER VARIABLE NUMBER.
C     L1 AND L2 REPRESENT THE COUNTER NAME.

      ELSEIF (K2 .EQ. 2) THEN
         READ (7,300,REC=NEXT) L1,L2

C     THE COUNTER IS COMPARED TO THE COUNTER NAMES IN THE PROGRAM, AND
C     ONCE THE MATCH HAS BEEN FOUND, THE VALUE IS DECREMENTED, AND ITS
C     VALUE IS WRITTEN TO THE SYMBOL TABLE FILE.

         DO 20 I = 1,NUM3-1

            IF (L2 .EQ. COUNT(I)) THEN
               CNTVAL(I) = CNTVAL(I) - 1
               CC = CNTVAL(I)
               WRITE (8,200,REC=L2) L2,CC
               NEXT = NEXT + 1
               GOTO 910
            ENDIF
```

```
20       CONTINUE

C      INCR

C      THE OBJECT FILE IS READ FOR THE COUNTER VARIABLE NUMBER.
C      L1 AND L2 REPRESENT THE COUNTER NAME.

       ELSEIF (K2 .EQ. 3) THEN
          READ (7,300,REC=NEXT) L1,L2

C      THE COUNTER IS COMPARED TO THE COUNTER NAMES IN THE PROGRAM, AND
C      ONCE THE MATCH HAS BEEN FOUND, THE VALUE IS INCREMENTED, AND ITS
C      VALUE IS WRITTEN TO THE SYMBOL TABLE FILE.

          DO 30 I = 1,NUM3-1

             IF (L2 .EQ. COUNT(I)) THEN
                CNTVAL(I) = CNTVAL(I) + 1
                CC = CNTVAL(I)
                WRITE (8,200,REC=L2) L2,CC
                NEXT = NEXT + 1
                GOTO 910
             ENDIF

30        CONTINUE

C      SETC

C      THE OBJECT FILE IS READ FOR THE COUNTER VARIABLE NUMBER AND THE
C      CONSTANT / VARIABLE NUMBER TO BE ASSIGNED TO IT.
C      L1 AND L2 REPRESENT THE COUNTER NAME, L3 AND L4 REPRESENT THE
C      COUNTER VALUE.


       ELSEIF (K2 .EQ. 4) THEN
          READ (7,300,REC=NEXT) L1,L2,L3,L4

C      THE COUNTER NAME IS COMPARED TO THE NAMES IN THE PROGRAM. ONCE THE
C      MATCH AHS BEEN FOUND, THE INDEX FOR THE COUNTER IS ASSIGNED TO L5.

          DO 40 I = 1,NUM3-1

             IF (L2 .EQ. COUNT(I)) THEN
                L5 = I
                GOTO 999
             ENDIF

40        CONTINUE

C      THE VARIABLE NAME FOR THE COUNTER VALUE IS CHECKED AGAINST
C      CONSTANTS, COUNTERS, AND PARAMETERS NAMES, AND THE VALUE IS
C      ASSIGNED TO THE ARRAY CNTVAL.

999       IF (L3 .EQ. 1) THEN

             DO 41 I = 1,NUM1-1

                IF (L4 .EQ. CONS(I)) THEN
                   CNTVAL(L5) = CONVAL(I)
                   NEXT = NEXT + 1
                   GOTO 990
                ENDIF

41           CONTINUE

             DO 42 I = 1,NUM3-1

                IF (L4 .EQ. COUNT(I)) THEN
                   CNTVAL(L5) = CNTVAL(I)
```

```
                  NEXT = NEXT + 1
                  GOTO 990
               ENDIF

42          CONTINUE

            DO 43 I = 1,NUM5-1

               IF (L4 .EQ. PAR(I)) THEN
                  READ(8,600,REC=L4) T
                  CNTVAL(L5) = T
                  NEXT = NEXT + 1
                  GOTO 990
               ENDIF

43          CONTINUE

C     IF THE VALUE IS SPECIFIED AS A NUMBER, THE CONSTANTS FILE IS
C     READ AND THE VALUE IS ASSIGNED TO THE ARRAY CNTVAL.

            ELSEIF (L3 .EQ. 2) THEN
               READ (4,400,REC=L4) T
               CNTVAL(L5) = T
               NEXT = NEXT + 1
            ENDIF

C     THE COUNTER VALUE IS CHECKED TO BE WITHIN LIMITS, AND IF NOT, AN
C     ERROR MESSAGE IS DISPLAYED. OTHERWISE, THE VALUE IS WRITTEN
C     TO THE SYMBOL TABLE FILE.

990         CC = CNTVAL(L5)
            IF (CC .GT. 32767 .OR. CC .LT. -32767) CALL USER(11)
            WRITE (8,200,REC=L2) L2,CC

         ENDIF

200      FORMAT(1X,I3,2X,F8.2)
300      FORMAT(17X,5(1X,I4))
310      FORMAT(6X,F8.2)
400      FORMAT(22X,F8.2)
500      FORMAT(2X,I1,5(2X,I8))
600      FORMAT(4X,4(2X,F8.2))

910      RETURN
         END

C        SUBROUTINE PALL

C        THIS SUBROUTINE HANDLES ALL THE PALLET COMMANDS.

C        PARAMETER DEFINITION:

C             K2        -       SUBTYPE OF COMMAND
C                                 1 - GETPART
C                                 2 - NEXTPART
C                                 3 - PREVPART
C                                 4 - SETPART

C        SUBROUTINES CALLED BY THIS PROGRAM ARE :
C                CHECK,ERROR,USER,WRITEP,WRITER.

         SUBROUTINE PALL(K2)

         INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
         INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
         INTEGER SS,SLPAY,PART1,PART2

         DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
         DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
```

```
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

C     GETPART

C     READ THE COMMAND PARAMETER VARIABLE NUMBER.
C     L1 AND L2 REPRESENT THE PALLET NAME.

      IF (K2 .EQ. 1) THEN
         READ (7,100,REC=NEXT) L1,L2

C     CHECK THE VARIABLE NUMBER AGAINST PALLET NUMBERS. TRANSMIT THE
C     COORDINATES OF THE PALLET POINT OT THE CONTROLLER, ALONG WITH A CODE
C     VALUE OF 6, AFTER CHECKING THAT THE ROBOT CONTROLLER IS AVAILABLE.

         IF (L1 .EQ. 1) THEN

            DO 10 I = 1,NUM6-1

               IF (L2 .EQ. PALLET(I)) THEN
                  XVAL = XPAL(I,PARTNO(I))
                  YVAL = YPAL(I,PARTNO(I))
                  RVAL = RPAL(I,PARTNO(I))
                  CODE = 6.0
                  IER = 0
                  TRY = 1.0
                  CALL CHECK
551               CALL WRITEP(42,XVAL,YVAL,ZVAL,RVAL,IER)

                  IF (IER .NE. 0) THEN
                     TRY = TRY + 1.0
                     IF (TRY .GE. 5.0) CALL ERROR
                     IER = 0
                     GOTO 551
                  ENDIF

                  IER = 0
                  TRY = 1.0
552               CALL WRITER(34,CODE,IER)

                  IF (IER .NE. 0) THEN
                     TRY = TRY + 1.0
                     IF (TRY .GE. 5.0) CALL ERROR
                     IER = 0
                     GOTO 552
                  ENDIF

                  IER = 0

                  GOTO 900
               ENDIF

10          CONTINUE

         ENDIF

C     NEXTPART

C     READ THE COMMAND PARAMETER.
C     L1 AND L2 REPRESENT THE PALLET NAME.
```

```
      ELSEIF (K2 .EQ. 2) THEN
         READ (7,100,REC=NEXT) L1,L2

C     CHECK THE VARIABLE NUMBER AGAINST PALLET NUMBERS. ONCE THE PALLET
C     NAME HAS BEEN FOUND, INCREMENT THE ARRAY PARTNO, IF ITS VALUE
C     IS LESS THAN THE TOTAL NUMBER OF POINTS. OTHERWISE SET PARTNO TO 1.
C     THE Z VALUE IS SET TO THE CURRENT Z VALUE OF THE ARM.

         IF (L1 .EQ. 1) THEN

            DO 20 I = 1,NUM6-1

               IF (L2 .EQ. PALLET(I)) THEN

                  IF (PARTNO(I) .LT. PALPTS(I)) THEN
                     PARTNO(I) = PARTNO(I) + 1
                  ELSE
                     PARTNO(I) = 1
                  ENDIF

                  ZPAL(I,PARTNO(I)) = ZVAL
                  GOTO 900

               ENDIF

20          CONTINUE

         ENDIF

C     PREVPART

C     READ THE COMMAND PARAMETER.
C     L1 AND L2 REPRESENT THE PALLET NAME.

      ELSEIF (K2 .EQ. 3) THEN
         READ (7,100,REC=NEXT) L1,L2

C     CHECK THE VARIABLE NUMBER AGAINST PALLET NUMBERS. ONCE THE PALLET
C     NAME HAS BEEN FOUND, DECREMENT THE ARRAY PARTNO, IF ITS VALUE
C     IS GREATER THAN 1. OTHERWISE SET PARTNO TO THE MAXIMUM VALUE OF PARTS.
C     THE Z VALUE IS SET TO THE CURRENT Z VALUE OF THE ARM.

         IF (L1 .EQ. 1) THEN

            DO 30 I = 1,NUM6-1

               IF (L2 .EQ. PALLET(I)) THEN

                  IF (PARTNO(I) .GT. 1) THEN
                     PARTNO(I) = PARTNO(I) - 1
                  ELSE
                     PARTNO(I) = PALPTS(I)
                  ENDIF

                  ZPAL(I,PARTNO(I)) = ZVAL
                  GOTO 900
               ENDIF

30          CONTINUE

         ENDIF

C     SETPART

C     READ THE COMMAND PARAMETERS.
C     L1 AND L2 REPRESENT THE PALLETNAME, L3 AND L4 REPRESENT THE
C     PALLET VALUE.
```

```
      ELSEIF (K2 .EQ. 4) THEN
         READ (7,100,REC=NEXT) L1,L2,L3,L4

C     SEE IF THE PALLET VALUE IS DEFINED AS A CONSTANT, COUNTER OR AS A
C     PARAMETER AND DETERMINE ITS VALUE AND ASSIGN IT TO L4.

         IF (L3 .EQ. 1) THEN

            DO 40 I = 1,NUM1-1

               IF (L4 .EQ. CONS(I)) THEN
                  L4 = CONVAL(I)
                  GOTO 41
               ENDIF

40          CONTINUE

            DO 42 I = NUM3-1

               IF (L4 .EQ. COUNT(I)) THEN
                  L4 = CNTVAL(I)
                  GOTO 41
               ENDIF

42          CONTINUE

            DO 43 I = 1,NUM5-1

               IF (L4 .EQ. PAR(I)) THEN
                  READ (8,200,REC=L4) T
                  L4 = T
                  GOTO 41
               ENDIF

43          CONTINUE

C     IF THE PALLET VALUE IS A NUMBER, READ FROM THE CONSTANTS FILE AND
C     ASSIGN IT TO THE VARIABLE L4.

         ELSEIF (L3 .EQ. 2) THEN
            READ (4,300,REC=L4) T
            L4 = T
         ENDIF

C     CHECK THE VARIABLE NUMBER OF THE PALLET AGAINST THE PALLET NUMBERS.
C     ONCE THE MATCH HAS BEEN FOUND, SET THE ARRAY PARTNO TO L4, IF L4
C     IS IN THE VALID RANGE, OTHERWISE, DISPLAY AN ERROR MESSAGE.
C     ALSO SET THE Z VALUE OF THE POINT EQUAL TO THE CURRENT Z VALUE.

41       IF (L1 .EQ. 1) THEN

            DO 44 I = 1,NUM6-1

               IF (L2 .EQ. PALLET(I)) THEN

                  IF (L4 .LE. PALPTS(I)) THEN
                     PARTNO(I) = L4
                  ELSE
                     CALL USER(13)
                  ENDIF

                  ZPAL(I,PARTNO(I)) = ZVAL
                  GOTO 900
               ENDIF

44          CONTINUE

         ENDIF
```

```
      ENDIF

99    FORMAT(2X,I1,4(2X,F8.2))
100   FORMAT(17X,7(1X,I4))
200   FORMAT(6X,F8.2)
300   FORMAT(22X,F8.2)

C     SET THE NEXT STATEMENT NUMBER TO BE EXECUTED, AND RETURN.

900   NEXT = NEXT + 1

      RETURN
      END

C     SUBROUTINE SUB1

C     THIS SUBROUTINE HANDLES SUBROUTINE COMMANDS.

C     PARAMETER DEFINITION:

C          K2       -      SUBTYPE OF COMMAND
C                             1 - SUBR
C                             2 - END

      SUBROUTINE SUB1(K2)

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

C     IF THE COMMAND IS "END", READ THE OBJECT FILE TO FIGURE OUT THE
C     THE NEXT STATEMENT TO BE EXECUTED, AND ASSIGN IT TO THE VARIABLE, NEXT.

      IF (K2 .EQ. 2) THEN
         READ (7,100,REC=NEXT) L2
         NEXT = L2
      ENDIF

99    FORMAT(2X,I1,2X,I8)
100   FORMAT(23X,I4)

      RETURN
      END

C     SUBROUTINE SUB2

C     THIS SUBROUTINE HANDLES CALLS TO SUBROUTINES.
C     THIS HANDLES A MAXIMUM OF 5 FORMAL PARAMETERS.

C     PARAMETER DEFINITION:

C          K2       -      STATEMENT NUMBER WHERE SUBROUTINE IS DEFINED

      SUBROUTINE SUB2(K2)
```

```
      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

      NN = 1

C     READ THE STATEMENT WHERE SUBROUTINE IS DEFINED FOR THE STATEMENT
C     NUMBER OF THE "END" STATEMENT OF SUBROUTINE, L1, FIRST EXECUTABLE
C     STATEMENT OF SUBROUTINE, L2, AND THE FIVE FORMAL PARAMETER NUMBERS,
C     L(1) THROUGH L(5).

      READ (7,100,REC=K2) L1,L2,(L(I),I=1,5)

C     READ THE "END" STATEMENT OF SUBROUTINE, AND WRITE THE SAME
C     INFORMATION ALONG WITH THE NEXT STATEMENT NUMBER TO BE EXECUTED
C     ONCE THE EXECUTION OF THE SUBROUTINE IS COMPLTE.

      READ (7,110,REC=L1) N1,N2,N3,N4,N5
      WRITE (7,110,REC=L1) N1,N2,N3,N4,N5,NEXT+1

C     READ THE STATEMENT WHERE THE CALL TO THE SUBROUTINE IS MADE, FOR
C     THE VARIABLE / CONSTANT NUMBERS OF THE FIVE FORMAL PARAMETERS.

      READ (7,120,REC=NEXT) (M(I),I=1,10)

C     CHECK IF THERE ARE ANY FORMAL PARAMETERS. IF THERE ARE ANY, SEE
C     IF THEY ARE PASSED AS NUMBERS OR AS VARIABLE NAMES.

      DO 10 I = 1,9,2
         IF (M(I) .EQ. 0) GOTO 999

C     IF THE PARAMETER IS A VARIABLE NAME, SEE IF THE VARIABLE IS A
C     CONSTANT, OR A POINT, OR A COUNTER, OR AN AGGREGATE.
C     DEPENDING ON THE TYPE OF VARIABLE, EITHER ONE OR FOUR VALUES
C     ARE ASSIGNED TO THE ARRAY P, AND VARIABLE NN IS SET TO 1 OR 4.

         IF (M(I) .EQ. 1) THEN
            DO 11 J = 1,NUM1-1

               IF (M(I+1) .EQ. CONS(J)) THEN
                  P(1) = CONVAL(J)
                  NN = 1
                 GOTO 9
               ENDIF

11          CONTINUE

            DO 12 J = 1,NUM2-1

               IF (M(I+1) .EQ. PNTVAL(J)) THEN
                  P(1) = XPNT(J)
                  P(2) = YPNT(J)
                  P(3) = ZPNT(J)
                  P(4) = RPNT(J)
                  NN = 4
```

```
                  GOTO 9
                ENDIF

12          CONTINUE

            DO 13 J = 1,NUM3-1

               IF (M(I+1) .EQ. COUNT(J)) THEN
                  P(1) = CNTVAL(J)
                  NN = 1
                 GOTO 9
               ENDIF

13          CONTINUE

            DO 14 J = 1,NUM4-1

               IF (M(I+1) .EQ. AGG(J)) THEN
                  P(1) = XAGG(J)
                  P(2) = YAGG(J)
                  P(3) = ZAGG(J)
                  P(4) = RAGG(J)
                  NN = 4
                 GOTO 9
               ENDIF

14          CONTINUE

C     IF THE PARAMETER IS A CONSTANT, READ THE CONSTANTS FILE FOR THE
C     VALUE, AND SET NN EQUAL TO 1.

         ELSEIF (M(I) .EQ. 2) THEN
            READ (4,300,REC=M(I+1)) IC1,P(1)
            NN = 1
         ENDIF

9        II = (I+1) / 2

C     WRITE THE VALUE PASSED INTO THE LOCATION OF THE FORMAL PARAMETER
C     IN THE SYMBOL TABLE FILE.

         IF (L(II) .GT. 0) THEN
            WRITE (8,400,REC=L(II)) L(II),(P(K),K=1,NN)
         ENDIF

10    CONTINUE

C     ONCE ALL THE FORMAL PARAMETERS HAVE BEEN IDENTIFIED, SET THE
C     NEXT STATEMENT TO BE EXECUTED TO BE THE FIRST EXECUTABLE
C     STATEMENT OF THE SUBROUTINE.

999   NEXT = L2

99    FORMAT(2X,I1,2X,I8)
100   FORMAT(17X,7(1X,I4))
110   FORMAT(1X,I4,1X,I4,1X,I1,3(1X,I4))
120   FORMAT(17X,10(1X,I4))
300   FORMAT(11X,I4,7X,F8.2)
400   FORMAT(1X,I3,4(2X,F8.2))

      RETURN
      END

C     SUBROUTINE WHERE

C     THIS SUBROUTINE HANDLES THE "WHERE" COMMAND.

      SUBROUTINE WHERE
```

```
      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

C     THE PROGRAM READS THE VARIABLE NUMBER OF THE COMMAND PARAMETER.
C     L1 AND L2 REPRESENT THE VARIABLE NAME.

      READ (7,100,REC=NEXT) L1,L2

      IF (L1 .EQ. 1) THEN

C     CHECK IF THE VARIABLE IS A POINT. IF SO, SET THE CURRENT VALUE
C     OF THE COORDINATES OF THE ROBOT ARM TO THE ARRAYS, XPNT,YPNT,
C     ZPNT AND RPNT.

         DO 10 I = 1,NUM2-1

            IF (L2 .EQ. PNTVAL(I)) THEN
               XPNT(I) = XVAL
               YPNT(I) = YVAL
               ZPNT(I) = ZVAL
               RPNT(I) = RVAL
               GOTO 11
            ENDIF

10       CONTINUE

C     IF THE VARIABLE IS A FORMAL PARAMETER, ASSIGN THE COORDINATES TO
C     THE ARRAYS XPAR,YPAR,ZPAR, AND RPAR.

         DO 12 I = 1,NUM5-1

            IF (L2 .EQ. PAR(I)) THEN
               XPAR(I) = XVAL
               YPAR(I) = YVAL
               ZPAR(I) = ZVAL
               RPAR(I) = RVAL
               GOTO 11
            ENDIF

12       CONTINUE
      ENDIF

C     INCREMENT THE STATEMENT NUMBER TO BE EXECUTED, AND WRITE THE
C     THE CURRENT VALUE OF THE ROBOT ARM COORDINATES INTO THE LOCATION
C     OF THE VARIABLE IN THE SYMBOL TABLE FILE.

11    NEXT = NEXT + 1
      WRITE (8,200,REC=L2) L2,XVAL,YVAL,ZVAL,RVAL

100   FORMAT(17X,2(1X,I4))
200   FORMAT(1X,I3,4(2X,F8.2))
300   FORMAT(2X,I1,4(2X,F8.2))

      RETURN
```

```
      END

C     SUBROUTINE ERROR

C     THIS SUBROUTINE REPORTS CONTROLLER COMMUNICATION ERRORS, AND
C     PROGRAM EXECUTION TERMINATES.

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             FINI,INKEY,TTOUT.

      SUBROUTINE ERROR

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

      CALL TTOUT(0,20,1,'Communication error detected.Program aborted.',
     *45,45,28)
      CALL TTOUT(0,24,1,'Hit any to key to terminate program...',38,38,
     *28)
      CALL INKEY(KYCOD1,KYCOD2)

      CALL FINI

      END

C     SUBROUTINE USER

C     THIS SUBROUTINE REPORTS USER ERRORS IN THE AML/E PROGRAM AND
C     ERRORS DURING USER INPUT FOR INTERACTIVE EXECUTION.

C     PARAMETER DEFINITION:

C          NNNN      -       ERROR NUMBER

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             CSROFF,CSRON,FINI,INKEY,TTOUT.

      SUBROUTINE USER(NNNN)

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2
      INTEGER * 2 KYCOD1,KYCOD2
      CHARACTER * 1 BLANK(80)

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
```

```
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

      DATA BLANK/80 * ' '/

C     THE ERROR NUMBERS ARE LESS THAN 20 FOR ERRORS IN THE USER'S
C     PROGRAM. THE SAME ERRORS DURING USER INPUT FOR INTERACTIVE
C     EXECUTION HAVE A VALUE OF 20 ADDED TO THEM. IF THE ERROR
C     IS GREATER THAN 20, ISET IS SET TO 1, AND PROGRAM EXECUTION
C     DOES NOT TERMINATE. HOWEVER, IF THE ERROR IS IN THE USER'S
C     PROGRAM, PROGRAM EXECUTION TERMINATES.

      CALL CSROFF
      ISET = 0

      IF (NNNN .GT. 20) THEN
         ISET = 1
         NN = NNNN - 20
      ELSE
         NN = NNNN
      ENDIF

C     MESSAGE IS DISPLAYED, AND PROGRAM EXECUTION DOES NOT TERMINATE WHEN
C     THE COMMAND "BREAKPOINT" IS FOUND IN THE USER'S PROGRAM.

      IF (NN .EQ. 12) THEN
      CALL TTOUT(0,20,1,'BREAKPOINT found. Hit any key to resume...',
     *42,42,28)
         CALL INKEY(KYCOD1,KYCOD2)
         CALL TTOUT(0,20,1,BLANK,80,80,31)
         RETURN
      ENDIF

      IF (NN .EQ. 1) THEN
      CALL TTOUT(0,20,1,'Time value out of range. Program aborted.',
     *41,41,28)

      ELSEIF (NN .EQ. 2) THEN
      CALL TTOUT(0,20,1,'R value out of range. Program aborted.',
     *38,38,28)

      ELSEIF (NN .EQ. 3) THEN
      CALL TTOUT(0,20,1,'Z value out of range. Program aborted.',
     *38,38,28)

      ELSEIF (NN .EQ. 4) THEN
      CALL TTOUT(0,20,1,'LINEAR value out of range. Program aborted.',
     *43,43,28)

      ELSEIF (NN .EQ. 5) THEN
      CALL TTOUT(0,20,1,'PAYLOAD value out of range. Program aborted.',
     *44,44,28)

      ELSEIF (NN .EQ. 6) THEN
      CALL TTOUT(0,20,1,'ZONE value out of range. Program aborted.',
     *41,41,28)

      ELSEIF (NN .EQ. 7) THEN
      CALL TTOUT(0,20,1,'DI port out of range. Program aborted.',
     *38,38,28)

      ELSEIF (NN .EQ. 8) THEN
      CALL TTOUT(0,20,1,'DI/DO value out of range. Program aborted.',
     *42,42,28)

      ELSEIF (NN .EQ. 9) THEN
      CALL TTOUT(0,20,1,'Timeout error in WAITI. Program aborted.',
```

```
     *40,40,28)

      ELSEIF (NN .EQ. 10) THEN
      CALL TTOUT(0,20,1,'DO port out of range. Program aborted.',
     *38,38,28)

      ELSEIF (NN .EQ. 11) THEN
      CALL TTOUT(0,20,1,'Counter value out of range. Program aborted.',
     *44,44,28)

      ELSEIF (NN .EQ. 13) THEN
      CALL TTOUT(0,20,1,'Pallet value out of range. Program aborted.',
     *43,43,28)

      ENDIF

      IF (ISET .EQ. 1) THEN
         CALL TTOUT(0,24,1,'Press any key to continue....',29,29,28)
         CALL INKEY(KYCOD1,KYCOD2)
         CALL TTOUT(0,20,1,BLANK,80,80,31)
         CALL TTOUT(0,24,1,BLANK,80,80,31)
         CALL CSRON
         RETURN

      ELSE

      CALL TTOUT(0,24,1,'Hit any to key to terminate program...',38,38,
     *28)
         CALL INKEY(KYCOD1,KYCOD2)
         CALL CSRON
         CALL FINI

      ENDIF

      END

C     SUBROUTINE CHECK

C     THIS SUBROUTINE CHECKS TO ENSURE THE DATA IN THE CONTROLLER IS
C     NOT OVERWRITTEN BEFORE THE PREVIOUS STATEMENT IS EXECUTED.

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             ERROR,READR.

      SUBROUTINE CHECK

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

C     THE CODE VALUE IS READ FORM THE CONTROLLER.

      IER = 0
      TRY = 1.0
```

```
10    CALL READR(34,CODE,IER)
      IF (IER .NE. 0) THEN
         TRY = TRY + 1.0
         IF (TRY .GE. 5.0) CALL ERROR
         IER = 0
         GOTO 10
      ENDIF

      IER = 0

C     IF THE CODE IS NOT 0, THE VALUE IS READ AGAIN AFTER
C     A DELAY BY THIS DUMMY LOOP.

      IF (CODE .NE. 0.0) THEN

         DO 20 I = 1,1000
            DUM = 0.0
20       CONTINUE

         TRY = 1.0
         GOTO 10

      ENDIF

      RETURN
      END

C     SUBROUTINE PART

C     THIS SUBROUTINE ASKS THE USERS IF THEY WANT TO EXECUTE THE
C     COMPLETE PROGRAM OR ONLY A PARTIAL PROGRAM.

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             CLS,CSROFF,CSRON,FINI,GOSUB,INKEY,LOCATE,PARTN,TTOUT.

      SUBROUTINE PART

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2
      INTEGER * 2 KYCOD1,KYCOD2,JJ1(4),JJ2(4)
      CHARACTER * 1 BLANK(80),C(4)

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

      DATA BLANK/80 * ' '/

*     DISPLAY THE EXECUTION OPTIONS MENU AND READ USER'S INPUT.

33    CALL CLS(31)
      CALL CSRON
      CALL TTOUT(0,1,29,'EXECUTION OPTIONS MENU',22,22,31)
      CALL TTOUT(0,2,29,'----------------------',22,22,31)
      CALL TTOUT(0,5,31,'Select a function:',18,18,27)
      CALL TTOUT(0,9,26,'F1  -  Complete program execution',33,33,31)
      CALL TTOUT(0,10,26,'F2  -  Partial program execution',32,32,31)
      CALL TTOUT(0,11,26,'F3  -  Subroutine execution',27,27,31)
```

```
      CALL TTOUT(0,12,26,'F4  -  Abort execution',22,22,31)
      CALL TTOUT(0,20,31,'Enter option===>',16,16,27)
36    CALL LOCATE(0,20,48)
      CALL INKEY(KYCOD1,KYCOD2)

C     IF THE USER'S INPUT IS INVALID, DISPLAY ERROR MESSAGE.

      IF (KYCOD1 .NE. 0) THEN
         CALL CSROFF
         CALL TTOUT(0,24,1,'KEY NOT DEFINED -- Hit any key to resume',
     *   40,40,28)
         CALL INKEY(KYCOD1,KYCOD2)
         CALL CSRON
         CALL TTOUT(0,24,1,BLANK,80,80,31)
         GOTO 36
      ENDIF

C     IF THE USER STRIKES THE F1 KEY, RETURN AND EXECUTE THE
C     COMPLETE PROGRAM.

      IF (KYCOD2 .EQ. 59) THEN
         RETURN

C     IF THE USER STRIKES THE F2 KEY, PROMPT THE USER FOR THE LINE
C     NUMBERS AT WHICH EXECUTION SHOULD START AND END.

      ELSEIF (KYCOD2 .EQ. 60) THEN
         GOTO 37

C     IF THE USER STRIKES THE F3 KEY, CALL THE SUBROUTINE GOSUB.

      ELSEIF (KYCOD2 .EQ. 61) THEN
         CALL GOSUB(IER)

C     IF THERE ARE ANY ERRORS FROM THE SUBROUTINE GOSUB, THEN
C     PROMPT USER FOR SETUP OPTIONS, OTHERWISE
C     RETURN TO EXECUTION OPTIONS MENU.

         IF (IER .EQ. 0) THEN
            GOTO 1110
         ELSE
            GOTO 33
         ENDIF

C     IF THE USER STRIKES THE F4 KEY, CALL THE SUBROUTINE FINI.

      ELSEIF (KYCOD2 .EQ. 62) THEN
         CALL FINI

C     DISPLAY ERROR MESSAGE IF INVALID KEY IS STRUCK.

      ELSE
         CALL CSROFF
         CALL TTOUT(0,24,1,'KEY NOT DEFINED -- Hit any key to resume',
     *   40,40,28)
         CALL INKEY(KYCOD1,KYCOD2)
         CALL CSRON
         CALL TTOUT(0,24,1,BLANK,80,80,31)
         GOTO 36
      ENDIF

C     IF USER SPECIFIES PARTIAL PROGRAM EXECUTION, PROMPT THE USER FOR
C     STARTING LINE NUMBER. READ THE USER'S INPUT. VARIABLE PART1 IS
C     SET TO THE STARTING LINE NUMBER.

37    CALL CLS(31)
      CALL TTOUT(0,10,1,'Please specify line number at which',35,35,31)
      CALL TTOUT(0,10,37,'you want the program to start.',30,30,31)
```

```
      JJJ = 1
34    CALL LOCATE(0,10,68+JJJ)
      CALL INKEY(KYCOD1,KYCOD2)

C     IF THE USER TYPES THE RETURN KEY, AND THE CURSOR IS AT THE FIRST
C     CHARACTER POSITION, PROGRAM EXECUTION STARTS AT THE FIRST LINE.
C     THEN THE USER IS PROMPTED FOR THE ENDING LINE NUMBER.
C     THE FOUR CHARACTERS TYPED IN ARE STORED IN THE VARIABLES, JJ1(1),
C     JJ1(2),JJ1(3),AND JJ1(4).
C     IF A NUMBER IS STRUCK, IT IS DECODED, AND DISPLAYED ON THE SCREEN
C     AND ASSIGNED TO ONE OF THESE VARIABLES. THE INDEX, JJJ, IS THEN
C     INCREMENTED.

      IF (KYCOD1 .EQ. 13 .AND. JJJ .EQ. 1) THEN
         PART1 = 1
         GOTO 1000
      ELSEIF (KYCOD1 .EQ. 13 .AND. JJJ .EQ. 2) THEN
         PART1 = JJ1(1)
         GOTO 1000
      ELSEIF (KYCOD1 .EQ. 13 .AND. JJJ .EQ. 3) THEN
         PART1 = JJ1(1) * 10 + JJ1(2)
         GOTO 1000
      ELSEIF (KYCOD1 .EQ. 13 .AND. JJJ .EQ. 4) THEN
         PART1 = JJ1(1) * 100 + JJ1(2) * 10 + JJ1(3)
         GOTO 1000
      ELSEIF (KYCOD1 .EQ. 48) THEN
         C(JJJ) = '0'
         JJ1(JJJ) = 0
         CALL TTOUT(0,10,68+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 49) THEN
         C(JJJ) = '1'
         JJ1(JJJ) = 1
         CALL TTOUT(0,10,68+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 50) THEN
         C(JJJ) = '2'
         JJ1(JJJ) = 2
         CALL TTOUT(0,10,68+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 51) THEN
         C(JJJ) = '3'
         JJ1(JJJ) = 3
         CALL TTOUT(0,10,68+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 52) THEN
         C(JJJ) = '4'
         JJ1(JJJ) = 4
         CALL TTOUT(0,10,68+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 53) THEN
         C(JJJ) = '5'
         JJ1(JJJ) = 5
         CALL TTOUT(0,10,68+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 54) THEN
         C(JJJ) = '6'
         JJ1(JJJ) = 6
         CALL TTOUT(0,10,68+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 55) THEN
         C(JJJ) = '7'
         JJ1(JJJ) = 7
         CALL TTOUT(0,10,68+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 56) THEN
         C(JJJ) = '8'
         JJ1(JJJ) = 8
         CALL TTOUT(0,10,68+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 57) THEN
         C(JJJ) = '9'
         JJ1(JJJ) = 9
         CALL TTOUT(0,10,68+JJJ,C(JJJ),1,1,31)
      ELSE
         CALL CSROFF
         CALL TTOUT(0,24,1,'INVALID CHARACTER - Hit any key to resume'
     *   ,41,41,28)
```

```
         CALL INKEY(KYCOD1,KYCOD2)
         CALL TTOUT(0,24,1,BLANK,80,80,31)
         CALL CSRON
         GOTO 34
      ENDIF

      JJJ = JJJ + 1
      IF (JJJ .LE. 4) GOTO 34
      PART1 = JJ1(1) * 1000 + JJ1(2) * 100 + JJ1(3) * 10 + JJ1(4)

C     THE USER IS NOW PROMPTED FOR THE END LINE NUMBER. THE END
C     LINE NUMBER IS STORED IN THE VARIABLE PART2.

1000  CALL CLS(31)
      CALL TTOUT(0,10,1,'Please specify line number at which',35,35,31)
      CALL TTOUT(0,10,37,'you want the program to end.',28,28,31)
      JJJ = 1
35    CALL LOCATE(0,10,66+JJJ)
      CALL INKEY(KYCOD1,KYCOD2)

C     IF THE USER TYPES THE RETURN KEY, AND THE CURSOR IS AT THE FIRST
C     CHARACTER POSITION, PROGRAM EXECUTION ENDS AT THE LAST LINE.
C     THE FOUR CHARACTERS TYPED IN ARE STORED IN THE VARIABLES, JJ2(1),
C     JJ2(2),JJ2(3),AND JJ2(4).
C     IF A NUMBER IS STRUCK, IT IS DECODED, AND DISPLAYED ON THE SCREEN
C     AND ASSIGNED TO ONE OF THESE VARIABLES. THE INDEX, JJJ, IS THEN
C     INCREMENTED.

      IF (KYCOD1 .EQ. 13 .AND. JJJ .EQ. 1) THEN
         PART2 = 1000
         GOTO 1001
      ELSEIF (KYCOD1 .EQ. 13 .AND. JJJ .EQ. 2) THEN
         PART2 = JJ2(1)
         GOTO 1001
      ELSEIF (KYCOD1 .EQ. 13 .AND. JJJ .EQ. 3) THEN
         PART2 = JJ2(1) * 10 + JJ2(2)
         GOTO 1001
      ELSEIF (KYCOD1 .EQ. 13 .AND. JJJ .EQ. 4) THEN
         PART2 = JJ2(1) * 100 + JJ2(2) * 10 + JJ2(3)
         GOTO 1001
      ELSEIF (KYCOD1 .EQ. 48) THEN
         C(JJJ) = '0'
         JJ2(JJJ) = 0
         CALL TTOUT(0,10,66+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 49) THEN
         C(JJJ) = '1'
         JJ2(JJJ) = 1
         CALL TTOUT(0,10,66+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 50) THEN
         C(JJJ) = '2'
         JJ2(JJJ) = 2
         CALL TTOUT(0,10,66+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 51) THEN
         C(JJJ) = '3'
         JJ2(JJJ) = 3
         CALL TTOUT(0,10,66+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 52) THEN
         C(JJJ) = '4'
         JJ2(JJJ) = 4
         CALL TTOUT(0,10,66+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 53) THEN
         C(JJJ) = '5'
         JJ2(JJJ) = 5
         CALL TTOUT(0,10,66+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 54) THEN
         C(JJJ) = '6'
         JJ2(JJJ) = 6
         CALL TTOUT(0,10,66+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 55) THEN
```

```
         C(JJJ) = '7'
         JJ2(JJJ) = 7
         CALL TTOUT(0,10,66+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 56) THEN
         C(JJJ) = '8'
         JJ2(JJJ) = 8
         CALL TTOUT(0,10,66+JJJ,C(JJJ),1,1,31)
      ELSEIF (KYCOD1 .EQ. 57) THEN
         C(JJJ) = '9'
         JJ2(JJJ) = 9
         CALL TTOUT(0,10,66+JJJ,C(JJJ),1,1,31)
      ELSE
         CALL CSROFF
         CALL TTOUT(0,24,1,'INVALID CHARACTER - Hit any key to resume'
     *   ,41,41,28)
         CALL INKEY(KYCOD1,KYCOD2)
         CALL TTOUT(0,24,1,BLANK,80,80,31)
         CALL CSRON
         GOTO 35
      ENDIF

      JJJ = JJJ + 1
      IF (JJJ .LE. 4) GOTO 35
      PART2 = JJ2(1) * 1000 + JJ2(2) * 100 + JJ2(3) * 10 + JJ2(4)

C     IF THE END LINE NUMBER IS SMALLER THAN THE START LINE NUMBER,
C     AN ERROR MESSAGE IS DISPLAYED.

1001  IF (PART2 .LT. PART1) THEN
         CALL CLS(31)
         CALL CSROFF
         CALL TTOUT(0,24,1,'End line number smaller than',28,28,28)
         CALL TTOUT(0,24,30,'starting line number.',21,21,28)
         CALL TTOUT(0,24,52,'Press any Key to continue..',27,27,28)
         CALL INKEY(KYCOD1,KYCOD2)
         GOTO 33
      ENDIF

C     THE FIRST RECORD OF THE OBJECT FILE IS READ TO DETERMINE THE
C     THE FIRST EXECUTABLE STATEMENT NUMBER AND THE LAST STATEMENT
C     NUMBER IN THE PROGRAM.

      READ (7,100,REC=1) K1,K2

C     IF THE USER WANTS PROGRAM EXECUTION TO START AT THE BEGINNING,
C     THE VARIABLE PART1 IS SET TO THE FIRST EXECUTABLE STATEMENT NUMBER.

      IF (PART1 .EQ. 1) THEN
         PART1 = K2
         GOTO 1004
      ENDIF

C     THE OBJECT FILE IS READ FOR THE LINE NUMBERS OF EACH STATEMENT.

      JJJ = 1
1002  READ (7,101,REC=JJJ) JJJJ
100   FORMAT(18X,I4,1X,I4)
101   FORMAT(6X,I4,8X,I4,1X,I4)

C     IF THE LINE NUMBER IS LESS THAN PART1, READ THE NEXT STATEMENT
C     IN THE OBJECT FILE.

      IF (JJJJ .LT. PART1) THEN
         JJJ = JJJ + 1

C     IF THE STATEMENT NUMBER IS GREATER THAN THE LAST
C     STATEMENT NUMBER, DISPLAY AN ERROR MESSAGE.

         IF (JJJ .GE. K1) THEN
```

```
            CALL CLS(31)
            CALL CSROFF
            CALL TTOUT(0,24,1,'Invalid start line number specified.',36,
     *      36,28)
            CALL TTOUT(0,24,38,'Press any key to continue....',29,29,28)
            CALL INKEY(KYCOD1,KYCOD2)
            GOTO 33
         ENDIF

         GOTO 1002

      ENDIF

C     THE PROGRAM EXECUTION STARTS AT THE STATEMENT NUMBER DENOTED BY JJJ.

      PART1 = JJJ

C     THIS SECTION CHECKS FOR THE LAST STATEMENT NUMBER.

1004  JJJ = PART1

C     IF THE USER DOES NOT SPECIFY ANY NUMBER FOR THE END LINE, PART2 IS
C     EQUAL TO 1000, AND PROGRAM EXECUTION ENDS AT THE LAST STATEMENT
C     NUMBER, K2.

      IF (PART2 .EQ. 1000) THEN
         PART2 = K1
         GOTO 1110
      ENDIF

C     THE OBJECT FILE IS READ FOR THE LINE NUMBERS, AND COMPARED TO THE
C     USER'S INPUT. IF THE LINE NUMBER IS LESS THAN PART2, THE NEXT LINE
C     IN THE OBJECT FILE IS READ.

1003  READ (7,101,REC=JJJ) JJJJ

      IF (JJJJ .LT. PART2) THEN
         JJJ = JJJ + 1

C     IF THE LAST STATEMENT NUMBER IS REACHED, AN ERROR MESSAGE IS
C     DISPLAYED.

         IF (JJJ .GE. K1) THEN
            CALL CLS(31)
            CALL CSROFF
            CALL TTOUT(0,24,1,'Invalid end line number specified.',34,
     *      34,28)
            CALL TTOUT(0,24,36,'Press any key to continue....',29,29,28)
            CALL INKEY(KYCOD1,KYCOD2)
            GOTO 33
         ENDIF

         GOTO 1003

      ENDIF

C     IF THE END STATEMENT NUMBER IS THE LAST STATEMENT OF THE PROGRAM,
C     PART2 IS SET TO THE LAST STATEMENT NUMBER, OTHERWISE, IT IS SET
C     TO THE END STATEMENT NUMBER PLUS 1.

      IF (JJJ .EQ. K1) THEN
         PART2 = JJJ
      ELSE
         PART2 = JJJ + 1
      ENDIF

C     DISPLAY MESSAGE TO THE USER.

1110  CALL CLS(31)
```

```
      CALL CSROFF
      CALL TTOUT(0,5,1,'Please move the robot arm to desired position',
     *45,45,31)
      CALL TTOUT(0,5,47,'before partial execution begins.',32,32,31)
      CALL TTOUT(0,6,1,'The next menu will show some of the',35,35,31)
      CALL TTOUT(0,6,37,'interactive commands allowed to move',36,36,31)
      CALL TTOUT(0,7,1,'the robot arm. Please try to use the ZMOVE',42,
     *42,31)
      CALL TTOUT(0,7,44,'and the LINEAR commands at the very',35,35,31)
      CALL TTOUT(0,8,1,'end to ensure that the arm does not hit',39,39,
     *31)
      CALL TTOUT(0,8,41,'any obstructions, and that the arm does',39,39
     *,31)
      CALL TTOUT(0,9,1,'not move in a linear fashion in the',35,35,31)
      CALL TTOUT(0,9,37,'non-linear region.',18,18,31)
      CALL TTOUT(0,12,1,'You are allowed to use names of variables',41,
     *41,31)
      CALL TTOUT(0,12,43,'from your program to define points,',35,35,31)
      CALL TTOUT(0,13,1,'values or counters.',19,19,31)
      CALL TTOUT(0,24,1,'Press any key to continue....',29,29,28)
      CALL INKEY(KYCOD1,KYCOD2)

C     DISPLAY SETUP MENU, AND READ USER'S INPUT.

1112  CALL CLS(31)
      CALL CSRON
      CALL TTOUT(0,1,23,'SETUP FOR PARTIAL PROGRAM EXECUTION',35,35,31)
      CALL TTOUT(0,2,23,'--------------------------------',35,35,31)
      CALL TTOUT(0,5,30,'Select a function:',18,18,27)
      CALL TTOUT(0,8,33,'F1  -  SETPART',14,14,31)
      CALL TTOUT(0,9,33,'F2  -  GETPART',14,14,31)
      CALL TTOUT(0,10,33,'F3  -  PMOVE',12,12,31)
      CALL TTOUT(0,11,33,'F4  -  ZMOVE',12,12,31)
      CALL TTOUT(0,12,33,'F5  -  WRITEO',13,13,31)
      CALL TTOUT(0,13,33,'F6  -  SETC',11,11,31)
      CALL TTOUT(0,14,33,'F7  -  LINEAR',13,13,31)
      CALL TTOUT(0,15,33,'F8  -  PAYLOAD',14,14,31)
      CALL TTOUT(0,16,33,'F9  -  ZONE',11,11,31)
      CALL TTOUT(0,17,33,'F10 -  Exit from setup menu',27,27,31)
      CALL TTOUT(0,20,30,'Enter option===>',16,16,27)
1111  CALL LOCATE(0,20,47)
      CALL INKEY(KYCOD1,KYCOD2)

C     DISPLAY ERROR MESSAGE IF USER STRIKES AN INVALID KEY.

      IF (KYCOD1 .NE. 0 .OR. KYCOD2 .LT. 59 .OR. KYCOD2 .GT. 68) THEN
         CALL CSROFF
         CALL TTOUT(0,24,1,'Key not defined -- Hit any key to resume',
     *   40,40,28)
         CALL INKEY(KYCOD1,KYCOD2)
         CALL CSRON
         CALL TTOUT(0,24,1,BLANK,80,80,31)
         GOTO 1111
      ENDIF

C     IF USER STRIKES F10 KEY, EXIT FROM SETUP MENU, OTHERWISE CALL
C     SUBROUTINE PARTN.

      IF (KYCOD2 .EQ. 68) GOTO 2000

      CALL PARTN(KYCOD2)

      GOTO 1112

2000  RETURN
      END

C     SUBROUTINE PARTN
```

```
C     THIS SUBROUTINE EXECUTES COMMANDS INTERACTIVELY, SO THAT THE
C     ROBOT ARM IS SETUP PRIOR TO PARTIAL PROGRAM EXECUTION.

C     PARAMETER DEFINITION:

C          KCOD2     -       FUNCTION KEY SELECTED BY USER


C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             CHECK,CLS,CSROFF,CSRON,INKEY,LOCATE,PARTM,TTOUT,
C             USER,WRITEP,WRITER.

      SUBROUTINE PARTN(KCOD2)

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2
      INTEGER * 2 KYCOD1,KYCOD2,KCOD2
      CHARACTER * 1 BLANK(80)
      CHARACTER * 72 CHAR,CNT

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

      DATA BLANK/80 * ' '/

C     OPEN A TEMPORARY FILE TO WRITE AND READ THE USER'S INPUT.

      OPEN (9,FILE='IRPS00')
      CALL CLS(31)
      CALL CSRON

C     SETPART

C     IF THE USER HAD STRUCK F1 KEY, PROMPT THE USER FOR THE PALLET
C     NAME. READ THE INPUT, AND WRITE THE NAME INTO THE FILE IRPS00.
C     COMPARE THIS NAME TO THE VARIABLE NAMES IN THE VARIABLES FILE.
C     IF NAME DOES NOT EXIST, DISPLAY ERROR MESSAGE. IF PALLET NAME
C     HAS BEEN FOUND, FIND THE INDEX NUMBER FOR THE ARRAY "PALLET".

      IF (KCOD2 .EQ. 59) THEN
198      CALL TTOUT(0,10,37,'SETPART',7,7,31)
         CALL TTOUT(0,11,37,'-------',7,7,31)
         CALL TTOUT(0,14,25,'Pallet name  = ',15,15,31)
         CALL PARTM(1,14,40,CHAR,LL,2)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
199      READ (3,300,END=201) IC2,NNN,ICODE,CNT
         IF (ICODE .NE. 5) GOTO 199
         IF (CNT(1:NNN) .NE. CHAR(1:LL)) GOTO 199
         DO 203 I = 1,NUM6 - 1
            IF (IC2 .EQ. PALLET(I)) THEN
               L5 = I
               GOTO 202
            ENDIF
203      CONTINUE

201      CALL CSROFF
```

```
         CALL TTOUT(0,24,1,'Pallet undefined -- Hit any key to resume',
     *   41,41,28)
         CALL INKEY(KYCOD1,KYCOD2)
         REWIND (3)
         GOTO 1000

C     PROMPT THE USER FOR THE PALLET VALUE. READ THE VALUE, AND CHECK
C     THAT THE VALUE LIES BETWEEN 1 AND THE MAXIMUM NUMBER OF POINTS
C     FOR THE PALLET. IF NOT, DISPLAY ERROR MESSAGE. IF THE NUMBER IS
C     IS VALID, SET THE VALUE OF THE ARRAY PARTNO EQUAL TO THE VALUE,
C     AND THE Z VALUE OF THE PALLET POINT TO BE EQUAL TO THE CURRENT
C     Z VALUE OF THE ROBOT ARM.

202      CALL TTOUT(0,15,25,'Pallet value = ',15,15,31)
         CALL PARTM(1,15,40,CHAR,LL,1)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
         READ (9,200) VAL1
         REWIND (9)
         IF (VAL1 .GT. PALPTS(L5) .OR. VAL1 .LT. 1.0) THEN
            PARTNO(L5) = 1
            CALL USER(33)
            REWIND (3)
            GOTO 1000
         ELSE
            PARTNO(L5) = VAL1
            ZPAL(L5,PARTNO(L5)) = ZVAL
         ENDIF

         REWIND (3)

C     GETPART

C     IF THE USER HAD STRUCK THE F2 KEY, PROMPT THE USER FOR THE
C     PALLET NAME. CHECK THAT PALLET NAME IS VALID, AS DESCRIBED
C     ABOVE.

      ELSEIF (KCOD2 .EQ. 60) THEN
498      CALL TTOUT(0,10,37,'GETPART',7,7,31)
         CALL TTOUT(0,11,37,'-------',7,7,31)
         CALL TTOUT(0,14,25,'Pallet name  = ',15,15,31)
         CALL PARTM(2,14,40,CHAR,LL,2)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
499      READ (3,300,END=501) IC2,NNN,ICODE,CNT
         IF (ICODE .NE. 5) GOTO 499
         IF (CNT(1:NNN) .NE. CHAR(1:LL)) GOTO 499
         DO 503 I = 1,NUM6 - 1
            IF (IC2 .EQ. PALLET(I)) THEN
               L5 = I
               GOTO 502
            ENDIF
503      CONTINUE

501      CALL CSROFF
         CALL TTOUT(0,24,1,'Pallet undefined -- Hit any key to resume',
     *   41,41,28)
         CALL INKEY(KYCOD1,KYCOD2)
         REWIND (3)
         GOTO 1000

C     SET THE CURRENT VALUES OF THE ROBOT ARM POSITION TO THE COORDINATES
C     OF THE POINT AS DEFINED BY THE PALLET POSITION.

502      XVAL = XPAL(L5,PARTNO(L5))
         YVAL = YPAL(L5,PARTNO(L5))
         RVAL = RPAL(L5,PARTNO(L5))
```

```
C     CHECK THAT THE ROBOT CONTROLLER IS READY TO RECEIVE DATA, AND
C     TRANSMIT THE COORDINATES TO THE CONTROLLER, ALONG WITH THE
C     CODE VALUE OF 6, REPRESENTING THE PMOVE COMMAND.

         CALL CHECK
         CODE = 6.0
         IER = 0
         TRY = 1.0

505      CALL WRITEP(42,XVAL,YVAL,ZVAL,RVAL,IER)
         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 505
         ENDIF

         IER = 0
         TRY = 1.0
506      CALL WRITER(34,CODE,IER)
         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 506
         ENDIF

         REWIND (3)

C     PMOVE

C     IF THE USER STRUCK THE F3 KEY, ASK THE USER WHETHER THE COMMAND
C     PARAMETER IS A VARIABLE NAME OR A SET OF NUMBERS. IF THE PARAMETER
C     IS A VARIABLE NAME, PROMPT THE USER FOR THE NAME.

      ELSEIF (KCOD2 .EQ. 61) THEN
598      CALL TTOUT(0,10,38,'PMOVE',5,5,31)
         CALL TTOUT(0,11,38,'-----',5,5,31)
         CALL TTOUT(0,14,20,'Is the point a variable name (Y/N) ?',36,
     *   36,31)
         CALL LOCATE(0,14,57)
         CALL INKEY(KYCOD1,KYCOD2)
         IF (KYCOD1 .NE. 89 .AND. KYCOD1 .NE. 121) THEN
            CALL TTOUT(0,14,1,BLANK,80,80,31)
            GOTO 507
         ENDIF

C     IF THE USER WANTS TO PASS A VARIABLE NAME, READ THE INPUT
C     AND CHECK FOR THE VALIDITY OF THE NAME AS BEFORE.

         CALL TTOUT(0,14,1,BLANK,80,80,31)
         CALL TTOUT(0,14,25,'Point name = ',13,13,31)
         CALL PARTM(3,14,38,CHAR,LL,2)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
599      READ (3,300,END=601) IC2,NNN,ICODE,CNT
         IF (ICODE .NE. 3) GOTO 599
         IF (CNT(1:NNN) .NE. CHAR(1:LL)) GOTO 599
         DO 603 I = 1,NUM2 - 1
            IF (IC2 .EQ. PNTVAL(I)) THEN
               L5 = I
               GOTO 602
            ENDIF
603      CONTINUE

601      CALL CSROFF
         CALL TTOUT(0,24,1,'Point undefined -- Hit any key to resume',
     *   40,40,28)
         CALL INKEY(KYCOD1,KYCOD2)
```

```
         REWIND (3)
         GOTO 1000

C     ASSIGN THE VALUES OF THE POINT TO THE FOUR VARIABLES, VAL1,
C     VAL2, VAL3, AND VAL4.

602      VAL1 = XPNT(L5)
         VAL2 = YPNT(L5)
         VAL3 = ZPNT(L5)
         VAL4 = RPNT(L5)

         GOTO 607

C     IF THE USER WANTS TO PASS NUMBERS AS THE PARAMETER, PROMPT THE
C     USER FOR THE X,Y,Z, AND R VALUES, ONE AT A TIME. READ THE USER'S
C     INPUT FOR EACH VALUE.

507      CALL TTOUT(0,14,25,'X value = ',10,10,31)
         CALL PARTM(3,14,35,CHAR,LL,1)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
         IF (LL .LE. 1) GOTO 1
         READ (9,200) VAL1
         REWIND (9)
1        CALL TTOUT(0,15,25,'Y value = ',10,10,31)
         CALL PARTM(3,15,35,CHAR,LL,1)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
         IF (LL .LE. 1) GOTO 2
         READ (9,200) VAL2
         REWIND (9)
2        CALL TTOUT(0,16,25,'Z value = ',10,10,31)
         CALL PARTM(3,16,35,CHAR,LL,1)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
         IF (LL .LE. 1) GOTO 3
         READ (9,200) VAL3
         REWIND (9)
3        CALL TTOUT(0,17,25,'R value = ',10,10,31)
         CALL PARTM(3,17,35,CHAR,LL,1)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
         IF (LL .LE. 1) GOTO 607
         READ (9,200) VAL4
         REWIND (9)

C     CHECK FOR THE VALIDITY OF THE R AND Z VALUES. IF THEY ARE
C     INVALID, DISPLAY AN ERROR MESSAGE.

607      CALL CHECK
         IF (VAL4 .LT. -180.0 .OR. VAL4 .GT. 180.0) THEN
            CALL USER(22)
            REWIND (3)
            GOTO 1000
         ENDIF

         IF (VAL3 .LT. -250.0 .OR. VAL3 .GT. 0.0) THEN
            CALL USER(23)
            REWIND (3)
            GOTO 1000
         ENDIF

C     ASSIGN THE CURRENT POSITION OF THE ROBOT ARM TO THE VALUES
C     VAL1, VAL2, VAL3 AND VAL4. TRANSMIT THESE VALUES TO THE
C     ROBOT CONTROLLER, ALONG WITH A CODE VALUE OF 6, REPRESENTING
C     A PMOVE COMMAND.

         XVAL = VAL1
         YVAL = VAL2
```

```
         ZVAL = VAL3
         RVAL = VAL4
         CODE = 6.0
         IER = 0
         TRY = 1.0

551      CALL WRITEP(42,XVAL,YVAL,ZVAL,RVAL,IER)
         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 551
         ENDIF

         IER = 0
         TRY = 1.0
552      CALL WRITER(34,CODE,IER)
         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 552
         ENDIF

         REWIND (3)

C        ZMOVE

C        IF THE USER STRUCK THE F4 KEY, PROMT THE USE FOR THE Z VALUE.

         ELSEIF (KCOD2 .EQ. 62) THEN
            CALL TTOUT(0,10,38,'ZMOVE',5,5,31)
            CALL TTOUT(0,11,38,'-----',5,5,31)
            CALL TTOUT(0,14,25,'Z value = ',10,10,31)
            CALL PARTM(4,14,35,CHAR,LL,1)
            WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
            REWIND (9)
            READ (9,200) VAL1
            REWIND (9)

C        CHECK THAT THE Z VALUE IS VALID, AND IF SO, SET THE CURRENT
C        Z VALUE, ZVAL, TO THE USER'S INPUT. TRANSMIT THE CURRENT POINT
C        VALUE TO THE CONTROLLER, ALONG WITH A CODE VALUE OF 6.

            IF (VAL1 .LT. -250.0 .OR. VAL1 .GT. 0.0) THEN
               CALL USER(23)
               GOTO 1000
            ENDIF

            ZVAL = VAL1
            CALL CHECK
            CODE = 6.0
            IER = 0
            TRY = 1.0

553         CALL WRITEP(42,XVAL,YVAL,ZVAL,RVAL,IER)
            IF (IER .NE. 0) THEN
               TRY = TRY + 1.0
               IF (TRY .GE. 5.0) CALL ERROR
               IER = 0
               GOTO 553
            ENDIF

            IER = 0
            TRY = 1.0
554         CALL WRITER(34,CODE,IER)
            IF (IER .NE. 0) THEN
               TRY = TRY + 1.0
               IF (TRY .GE. 5.0) CALL ERROR
```

```
            IER = 0
            GOTO 554
         ENDIF

C     WRITEO

C     IF THE USER STRUCK THE F5 KEY, PROMPT THE USER FOR THE DO
C     NUMBER. THE INPUT MUST BE FROM 1 THROUGH 16, OTHERWISE AN
C     ERROR MESSAGE IS DISPLAYED.

      ELSEIF (KCOD2 .EQ. 63) THEN
         CALL TTOUT(0,10,37,'WRITEO',6,6,31)
         CALL TTOUT(0,11,37,'------',6,6,31)
         CALL TTOUT(0,14,25,'DO number = ',12,12,31)
         CALL PARTM(5,14,37,CHAR,LL,1)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
         READ (9,200) VAL1
         REWIND (9)

         IF (VAL1 .LT. 1.0 .OR. VAL1 .GT. 16.0) THEN
            CALL USER(30)
            GOTO 1000
         ENDIF

C     THEN PROMPT THE USER FOR THE VALUE, WHICH MUST BE A ZERO OR
C     A ONE.

         CALL TTOUT(0,15,25,'DO value  = ',12,12,31)
         CALL PARTM(5,15,37,CHAR,LL,1)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
         READ (9,200) VAL2
         REWIND (9)

         IF (VAL2 .NE. 0.0 .AND. VAL2 .NE. 1.0) THEN
            CALL USER(28)
            GOTO 1000
         ENDIF

C     THE DO NUMBER, THE VALUE AND A CODE VALUE OF 9, ARE TRANSMITTED
C     TO THE CONTROLLER, TO EXECUTE THIS COMMAND.

         CALL CHECK
         CODE = 9.0
         IER = 0
         TRY = 1.0

555      CALL WRITER(39,VAL1,IER)
         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 555
         ENDIF

         IER = 0
         TRY = 1.0

556      CALL WRITER(38,VAL2,IER)
         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 556
         ENDIF

         IER = 0
         TRY = 1.0
```

```
557      CALL WRITER(34,CODE,IER)
         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 557
         ENDIF

C     SETC

C     IF THE USER STRUCK THE F6 KEY, THE USER IS PROMPTED FOR THE
C     COUNTER NAME. THE NAME IS CHECKED FOR ITS EXISTENCE, AS
C     BEFORE.

      ELSEIF (KCOD2 .EQ. 64) THEN
298      CALL TTOUT(0,10,39,'SETC',4,4,31)
         CALL TTOUT(0,11,39,'----',4,4,31)
         CALL TTOUT(0,14,25,'Counter name  = ',16,16,31)
         CALL PARTM(6,14,41,CHAR,LL,2)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
299      READ (3,300,END=301) IC2,NNN,ICODE,CNT
         IF (ICODE .NE. 4) GOTO 299
         IF (CNT(1:NNN) .NE. CHAR(1:LL)) GOTO 299

         DO 303 I = 1,NUM3 - 1

            IF (IC2 .EQ. COUNT(I)) THEN
               L5 = I
               GOTO 302
            ENDIF

303      CONTINUE

301      CALL CSROFF
         CALL TTOUT(0,24,1,'Counter undefined -- Hit any key to resume',
     *   42,42,28)
         CALL INKEY(KYCOD1,KYCOD2)
         REWIND (3)
         GOTO 1000

C     THE USER IS THEN PROMPTED FOR THE VALUE. IF THE VALUE IS OUT OF
C     THE VALID RANGE, AN ERROR MESSAGE IS DISPLAYED.

302      CALL TTOUT(0,15,25,'Counter value = ',16,16,31)
         CALL PARTM(6,15,41,CHAR,LL,1)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
         READ (9,200) VAL1
         REWIND (9)

         IF (VAL1 .LT. -32767.0 .OR. VAL1 .GT. 32767.0) THEN
            CALL USER(31)
            REWIND (3)
            GOTO 1000
         ENDIF

C     THE COUNTER VALUE IS THEN SET TO THE VALUE KEYED IN BY THE USER.
C     THIS VALUE IS ALSO WRITTEN TO THE SYMBOL TABLE FILE.

         CNTVAL(L5) = VAL1
         WRITE(8,400,REC=IC2) IC2,VAL1
         REWIND (3)

C     LINEAR
```

```
C     IF THE USER STRUCK THE F7 KEY, THE USER IS PROMPTED FOR THE
C     LINEAR VALUE. THIS VALUE IS CHECKED TO BE WITHIN THE VALID
C     LIMITS.

      ELSEIF (KCOD2 .EQ. 65) THEN
         CALL TTOUT(0,12,25,'LINEAR value = ',15,15,31)
         CALL PARTM(7,12,40,CHAR,LL,1)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
         READ (9,200) VAL1
         REWIND (9)

         IF (VAL1 .LT. 0.0 .OR. VAL1 .GT. 50.0) THEN
            CALL USER(24)
            GOTO 1000
         ENDIF

C     THIS VALUE IS THEN TRANSMITTED TO THE CONTROLLER.

         CALL CHECK
         IER = 0
         TRY = 0.0

561      CALL WRITER(35,VAL1,IER)
         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 561
         ENDIF

C     PAYLOAD

C     IF THE USER STRUCK THE F8 KEY, THE USER IS PROMPTED FOR THE
C     PAYLOAD VALUE. THIS VALUE IS CHECKED TO BE WITHIN THE VALID
C     LIMITS.

      ELSEIF (KCOD2 .EQ. 66) THEN
         CALL TTOUT(0,12,25,'PAYLOAD value = ',16,16,31)
         CALL PARTM(8,12,41,CHAR,LL,1)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
         READ (9,200) VAL1
         REWIND (9)

         IF (VAL1 .LT. 0.0 .OR. VAL1 .GT. 19.0) THEN
            CALL USER(25)
            GOTO 1000
         ENDIF

C     THIS VALUE IS THEN TRANSMITTED TO THE CONTROLLER.

         CALL CHECK
         IER = 0
         TRY = 0.0

562      CALL WRITER(36,VAL1,IER)
         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 562
         ENDIF

C     ZONE

C     IF THE USER STRUCK THE F9 KEY, THE USER IS PROMPTED FOR THE
C     ZONE VALUE. THIS VALUE IS CHECKED TO BE WITHIN THE VALID
C     LIMITS.
```

```
      ELSEIF (KCOD2 .EQ. 67) THEN
         CALL TTOUT(0,12,25,'ZONE value = ',13,13,31)
         CALL PARTM(9,12,38,CHAR,LL,1)
         WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
         REWIND (9)
         READ (9,200) VAL1
         REWIND (9)
         IF (VAL1 .LT. 0.0 .OR. VAL1 .GT. 15.0) THEN
            CALL USER(26)
            GOTO 1000
         ENDIF

C     THIS VALUE IS THEN TRANSMITTED TO THE CONTROLLER.

         CALL CHECK
         IER = 0
         TRY = 0.0

563      CALL WRITER(37,VAL1,IER)
         IF (IER .NE. 0) THEN
            TRY = TRY + 1.0
            IF (TRY .GE. 5.0) CALL ERROR
            IER = 0
            GOTO 563
         ENDIF

      ENDIF

1000  CLOSE (9,STATUS='DELETE')

100   FORMAT(I2,1X,72A1)
200   FORMAT(3X,F8.2)
300   FORMAT(1X,I3,13X,I3,2X,I1,6X,A72)
400   FORMAT(1X,I3,2X,F8.2)

      RETURN
      END

C     SUBROUTINE PARTM

C     SUBROUTINE TO IDENTIFY CHARACTERS TYPED IN FOR VARIABLES PRIOR
C     TO PARTIAL EXECUTION.

C     PARAMETER DEFINITION:

C          KCODE     -      CODE PASSED DEPENDING ON COMMAND TYPE
C          ROW       -      ROW NUMBER
C          COLM      -      COLUMN NUMBER
C          CHAR      -      CHARACTER TYPED BY USER
C          LL        -      INCREMNTAL COLUMN NUMBER
C          MM        -      INDICATES WHETHER INPUT SHOULD BE
C                           NUMERIC (VALUE IS 1) OR ALPHANUMERIC (VALUE IS 2)


C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             CSROFF,CSRON,ID,INKEY,LOCATE,TTOUT.

      SUBROUTINE PARTM(KCODE,ROW,COLM,CHAR,LL,MM)

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2
      INTEGER * 2 KYCOD1,KYCOD2,J
      INTEGER ROW,COLM
      CHARACTER * 1 BLANK(80)
      CHARACTER * 1 CHAR1,ALPHA(26),NUMER(10)
      CHARACTER * 72 CHAR
```

```
      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

      DATA BLANK/80 * ' '/

      DATA ALPHA/'A','B','C','D','E','F','G','H','I','J','K','L','M',
     *'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'/
      DATA NUMER/'0','1','2','3','4','5','6','7','8','9'/

1     DO 10 MMM = 1,72
         CHAR(MMM:MMM) = ' '
10    CONTINUE

C     LOCATE THE CURSOR AND READ USER'S INPUT AND CHECK FOR VALID KEYS.

      LL = 1
5     CALL LOCATE(0,ROW,COLM+LL)
      J = LL
      CALL INKEY(KYCOD1,KYCOD2)

      IF (KYCOD1 .EQ. 13) THEN
         GOTO 20
      ELSE
         CALL ID(KYCOD1,KYCOD2,CHAR1,J,ALPHA,NUMER)

C     IF USER'S INPUT IS INVLAID, DISPLAY ERROR MESSAGE.

         IF (J .EQ. 0) THEN
            CALL CSROFF
           CALL TTOUT(0,24,1,'INVALID CHARACTER - Hit any key to resume'
     *       ,41,41,28)
            CALL INKEY(KYCOD1,KYCOD2)
            CALL TTOUT(0,24,1,BLANK,80,80,31)
            CALL CSRON
            GOTO 5
         ENDIF

C     IF THE USER STRIKES THE BACKSPACE KEY, ERASE THE PREVIOUS
C     CHARACTER FROM THE SCREEN, OTHERWISE DISPLAY THE CHARACTER
C     THE USER TYPES.

         IF (J .LT. LL) THEN
            LL = J
            CALL TTOUT(0,ROW,COLM+J,BLANK(1),1,1,31)
            GOTO 5
         ELSE
            CALL TTOUT(0,ROW,COLM+LL,CHAR1,1,1,26)

C     CONCATENATE THE STRING OF CHARACTERS TYPED IN BY THE USER, AND
C     INCREMENT THE CURSOR POSITION.

            IF (J .NE. 1) THEN
               CHAR(1:LL) = CHAR(1:LL-1)//CHAR1
            ELSE
               CHAR(1:LL) = CHAR1
            ENDIF

            LL = LL + 1
```

```
            GOTO 5

         ENDIF

      ENDIF

C     IF THE INPUT IS NUMERIC, CHECK FOR A DECIMAL POINT. IF THERE
C     IS NO DECIMAL POINT, ADD ONE TO THE STRING OF CHARACTERS.

20    IF (MM .EQ. 1) THEN
         LL = LL - 1

         DO 30 MMM = 1,LL
            IF (CHAR(MMM:MMM) .EQ. '.') GOTO 40
30       CONTINUE

         LL = LL + 1
         CHAR(LL:LL) = '.'

C     CHECK EACH CHARACTER TO BE A NUMBER, OR A DECIMAL POINT, OR
C     THE MINUS SIGN. IF ANY CHARACTER IS NOT ONE OF THESE, THE
C     NUMBER IS INVALID, AND THE ERROR CODE, IER, IS SET TO 1.

40       IER = 0

         DO 43 MMM = 1,LL

            DO 44 MMMM = 1,10
               IF (CHAR(MMM:MMM) .EQ. NUMER(MMMM)) GOTO 43
               IF (CHAR(MMM:MMM) .EQ. '-' .AND. MMM .EQ. 1) GOTO 43
               IF (CHAR(MMM:MMM) .EQ. '.') GOTO 43
44          CONTINUE

            IER = 1
43       CONTINUE

         ISET = 0

C     IF THERE IS MORE THAN ONE DECIMAL POINT, IER IS SET TO 1.

         DO 59 MMM = 1,LL
            IF (CHAR(MMM:MMM) .EQ. '.') ISET = ISET + 1
            IF (ISET .GT. 1) IER = 1
59       CONTINUE

C     IF THE LENGTH OF CHARACTERS IN THE NUMBER IS GREATER THAN 8,
C     OR IF IER IS 1, DISPLAY ERROR MESSAGE. THEN BLANK OUT THE
C     THE USER'S INPUT, AND PROMPT AGAIN.

         IF (LL .GT. 8 .OR. IER .NE. 0) THEN
            CALL CSROFF
            CALL TTOUT(0,24,1,'Invalid number -- Hit any key to resume',
     *      39,39,28)
            CALL INKEY(KYCOD1,KYCOD2)
            CALL TTOUT(0,24,1,BLANK,80,80,31)

            DO 52 MMM = 1,LL
               CALL TTOUT(0,ROW,COLM+MMM,' ',1,1,31)
52          CONTINUE

            IER = 0
            CALL CSRON
            GOTO 1
         ENDIF

C     IF THE INPUT IS ALPHANUMERIC, CHECK THAT THE FIRST CHARACTER
C     IS ALPHABETIC. IF NOT, DISPLAY ERROR MESSAGE AND PROMPT USER AGAIN.

      ELSE
```

```
         IER = 0
         LL = LL - 1

         DO 45 MMM = 1,26
            IF (CHAR(1:1) .EQ. ALPHA(MMM)) GOTO 46
45       CONTINUE

48       CALL CSROFF
         CALL TTOUT(0,24,1,'Invalid name -- Hit any key to resume',
     *   37,37,28)
         CALL INKEY(KYCOD1,KYCOD2)
         CALL TTOUT(0,24,1,BLANK,80,80,31)

         DO 53 MMM = 1,LL
            CALL TTOUT(0,ROW,COLM+MMM,' ',1,1,31)
53       CONTINUE

         IER = 0
         CALL CSRON
         GOTO 1

C     IF ANY CHARACTER IS THE DECIMAL POINT OR THE MINUS SIGN,
C     DISPLAY ERROR MESSAGE.

46       DO 47 MMM = 1,LL
            IF (CHAR(MMM:MMM) .EQ. '.') GOTO 48
            IF (CHAR(MMM:MMM) .EQ. '-') GOTO 48
47       CONTINUE

      ENDIF

      RETURN
      END

C     SUBROUTINE ID

C     THIS SUBROUTINE IDENTIFIES THE CHARACTER TYPED IN BY THE USER.
C     VALID CHARACTERS ARE ALL ALPHABETS, NUMBERS, THE BACKSPACE KEY,
C     THE ENTER KEY AND SOME SPECIAL CHARACTERS.

C     PARAMETER DEFINITION:

C          KYCOD1   -      ASCII CODE OF KEY
C          KYCOD2   -      EXTENDED CHARACTER CODE OF KEY
C          CHAR     -      CHARACTER IDENTIFIED
C          J        -      COLUMN POSITION
C          ALPHA    -      ARRAY OF ALPHABETS
C          NUMBER   -      ARRAY OF NUMBERS

      SUBROUTINE ID(KYCOD1,KYCOD2,CHAR1,J,ALPHA,NUMER)

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2
      INTEGER * 2 KYCOD1,KYCOD2,J
      CHARACTER * 1 CHAR1,ALPHA(26),NUMER(10)

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
```

```
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

      IF (KYCOD1 .GE. 48 .AND. KYCOD1 .LE. 57) THEN
         CHAR1 = NUMER(KYCOD1-47)
      ELSEIF (KYCOD1 .GE. 65 .AND. KYCOD1 .LE. 90) THEN
         CHAR1 = ALPHA(KYCOD1-64)
      ELSEIF (KYCOD1 .GE. 97 .AND. KYCOD1 .LE. 122) THEN
         CHAR1 = ALPHA(KYCOD1-96)
      ELSEIF (KYCOD1 .EQ. 29) THEN
         J = J - 1
      ELSEIF (KYCOD1 .EQ. 8 .AND. KYCOD2 .EQ. 14) THEN
         J = J - 1
      ELSEIF (KYCOD1 .EQ. 127 .AND. KYCOD2 .EQ. 14) THEN
         J = J - 1
      ELSEIF (KYCOD1 .EQ. 0 .AND. KYCOD2 .EQ. 75) THEN
         J = J - 1
      ELSEIF (KYCOD1 .EQ. 45) THEN
         CHAR1 = '-'
      ELSEIF (KYCOD1 .EQ. 46) THEN
         CHAR1 = '.'
      ELSEIF (KYCOD1 .EQ. 95) THEN
         CHAR1 = '_'
      ELSE
         J = 0
      ENDIF

      RETURN
      END

C     SUBROUTINE GOSUB

C     THIS SUBROUTINE INTERACTIVELY EXECUTES A SUBROUTINE.

C     PARAMETER DEFINITION:

C          IER      -      ERROR CODE DUE TO USER INPUT
C                          0 WHEN NO ERROR DETECTED
C                          1 WHEN THERE IS AN ERROR

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C          CLS,CSROFF,CSRON,LOCATE,INKEY,PARTM,TTOUT.

      SUBROUTINE GOSUB(IER)

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2
      INTEGER * 2 KYCOD1,KYCOD2
      CHARACTER * 1 BLANK(80),C
      CHARACTER * 72 CHAR,CNT,CNT1(5)

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)
      DIMENSION NNN1(5),IC21(5),VAL(4)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

      DATA BLANK/80 * ' '/

      IER = 0
```

```
      OPEN (9,FILE='IRPS00')
      CALL CLS(31)
      CALL CSRON

      DO 597 I = 1,5
         IC21(I) = 0
         NNN1(I) = 0
597   CONTINUE

C     PROMT THE USER FOR THE SUBROUTINE NAME. READ THE CHARACTERS
C     KEYED IN AND WRITE TO THE TEMPORARY FILE, IRPS00, AND READ
C     THE VARIABLES FILE AND COMPARE ANY SUBROUTINE NAME WITH
C     THE NAME TYPED IN BY THE USER.

598   CALL TTOUT(0,3,31,'SUBROUTINE EXECUTION',20,20,31)
      CALL TTOUT(0,4,31,'--------------------',20,20,31)
      CALL TTOUT(0,7,22,'Subroutine name = ',18,18,31)
      CALL PARTM(0,7,40,CHAR,LL,2)
      WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
      REWIND (9)

599   READ (3,300,END=601) IC2,IC3,NNN,ICODE,CNT
      IF (ICODE .NE. 6) GOTO 599
      IF (CNT(1:NNN) .NE. CHAR(1:LL)) GOTO 599
      GOTO 605

C     THE SUBROUTINE NAME DOES NOT MATCH ANY NAMES IN THE PROGRAM.
C     DISPLAY AN ERROR MESSAGE AND EXIT FROM THIS SUBROUTINE.

601   CALL CSROFF
      CALL TTOUT(0,24,1,'Name undefined -- Hit any key to resume',
     *39,39,28)
      CALL INKEY(KYCOD1,KYCOD2)
      REWIND (3)
      CALL CSRON
      IER = 1
      GOTO 1000

C     READ THE OBJECT FILE FOR THE FIRST AND LAST EXECUTABLE
C     STATEMENT NUMBERS FOR THIS SUBROUTINE AND ASSIGN THESE
C     STATEMENT NUMBERS TO VARIABLES PART1 AND PART2.

605   READ (7,400,REC=IC3) K1,K2
      PART1 = K2
      PART2 = K1

C     READ THE VARIABLES FILE TO CHECK FOR ANY FORMAL PARAMETERS,
C     AND IF THERE ARE ANY, ASSIGN THE VARIABLE NAMES TO THE
C     ARRAY CNT1. VARIABLE IC21(I) IS SET TO ZERO, IF THERE ARE
C     ONLY I-1 FORMAL PARAMETERS.

      DO 700 I = 1,5
606      READ (3,300,END=701) IC21(I),IC3,NNN1(I),ICODE,CNT1(I)
         IF (ICODE .EQ. 7) GOTO 700
         IC21(I) = 0
         GOTO 701
700   CONTINUE

701   REWIND(3)

C     PROMPT THE USER FOR VALUES FOR ANY FORMAL PARAMETERS.
C     THE USER CAN EITHER PASS A VARIABLE NAME TO THE PARAMETER,
C     OR CAN PASS A SET OF VALUES.

      DO 702 I = 1,5

         IF (IC21(I) .NE. 0) THEN
            CALL TTOUT(0,7,1,BLANK,80,80,31)
            CALL TTOUT(0,7,1,'Formal parameter :',18,18,31)
```

```
            CALL TTOUT(0,7,20,CNT1(I)(1:NNN1(I)),NNN1(I),NNN1(I),28)
            CALL TTOUT(0,9,1,BLANK,80,80,31)
            CALL TTOUT(0,10,1,BLANK,80,80,31)
            CALL TTOUT(0,12,1,BLANK,80,80,31)
            CALL TTOUT(0,13,1,BLANK,80,80,31)
            CALL TTOUT(0,14,1,BLANK,80,80,31)
            CALL TTOUT(0,15,1,BLANK,80,80,31)
            CALL TTOUT(0,9,10,'Do you want to pass a variable name',35,
     *      35,31)
            CALL TTOUT(0,10,10,'for this parameter (Y/N) ?',26,26,31)
            CALL LOCATE(0,10,46)
            CALL INKEY(KYCOD1,KYCOD2)

C     IF THE USER WANTS TO PASS A VARIABLE NAME, THE USER IS THEN
C     PROMPTED FOR THE VARIABLE NAME THAT NEEDS TO BE PASSED.
C     THE NAME TYPED IN IS READ, AND COMPARED WITH VARIABLE NAMES READ
C     FROM THE VARIABLES FILE.

            IF (KYCOD1 .EQ. 89 .OR. KYCOD1 .EQ. 121) THEN
               CALL TTOUT(0,9,1,BLANK,80,80,31)
               CALL TTOUT(0,10,1,BLANK,80,80,31)
709            CALL TTOUT(0,9,20,'Variable name = ',16,16,31)
               CALL PARTM(0,9,36,CHAR,LL,2)
704            READ (3,300,END=703) IC2,IC3,NNN,ICODE,CNT
               IF (CNT(1:NNN) .NE. CHAR(1:LL)) GOTO 704

C     IF THE VARIABLE NAME IS A CONSTANT OR A COUNTER, ITS VALUE IS READ
C     FROM THE SYMBOL TABLE FILE, AND WRITTEN TO THE LOCATION OF THE
C     FORMAL PARAMTER IN THE SYMBOL TABLE FILE.

               IF (ICODE .EQ. 2 .OR. ICODE .EQ. 4) THEN
                  READ (8,501,REC=IC2) CC1
                  WRITE (8,500,REC=IC21(I)) IC21(I),CC1

C     IF THE VARIABLE NAME IS A POINT OR AN AGGREGATE, THE FOUR VALUES
C     ARE READ, AND WRITTEN TO THE LOCATION OF THE FORMAL PARAMETER.

               ELSEIF (ICODE .EQ. 3 .OR. ICODE .EQ. 7 .OR. ICODE .EQ. 8)
     *         THEN
                  READ (8,501,REC=IC2) CC1,CC2,CC3,CC4
                  WRITE (8,500,REC=IC21(I)) IC21(I),CC1,CC2,CC3,CC4

C     IF THE VARIABLE NAME IS A POINT DEFINED AS A SET OF COUNTERS,
C     THE COUNTER NUMBERS ARE READ AND WRITTEN TO THE LOCATION OF THE
C     OF THE FORMAL PARAMETER.

               ELSEIF (ICODE .EQ. 9) THEN
                  READ (8,699,REC=IC2) NC1,NC2,NC3,NC4,NC5,NC6,NC7,NC8
                  WRITE (8,600,REC=IC21(I)) IC21(I),NC1,NC2,NC3,NC4,NC5,
     *            NC6,NC7,NC8
               ENDIF

               REWIND (3)
               GOTO 702

C     IF THE VARIABLE NAME IS NOT FOUND, AN ERROR MESSAGE IS DISPLAYED,
C     AND THE EXECUTIONS OPTIONS MENU IS DISPLAYED AGAIN.

703            CALL CSROFF
          CALL TTOUT(0,24,1,'Variable not defined -- Hit any ',32,32,28)
               CALL TTOUT(0,24,33,'key to resume....',17,17,28)
               CALL INKEY(KYCOD1,KYCOD2)
               REWIND (3)
               CALL CSRON
               IER = 1
               GOTO 1000
            ENDIF
```

```
C     IF THE USER WANTS TO PASS NUMERIC VALUES TO THE FORMAL
C     PARAMETER, THE PROGRAM PROMPTS THE USER FOR THE NUMBER OF
C     OF VALUES TO BE PASSED FOR IT. THE MAXIMUM NUMBER THAT CAN
C     BE PASSED IS FOUR, FOR EACH PARAMETER.

           CALL TTOUT(0,9,1,BLANK,80,80,31)
           CALL TTOUT(0,10,1,BLANK,80,80,31)
           CALL CSRON
           CALL TTOUT(0,9,1,'Please specify the number of values ',36,
     *     36,31)
           CALL TTOUT(0,9,37,'for this parameter and then please',34,
     *     34,31)
720       CALL TTOUT(0,10,1,'enter them in the proper order.',31,31,31)
           CALL PARTM(0,10,32,CHAR,LL,1)
           WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
           REWIND (9)
           READ (9,800) VAL1
           REWIND (9)
           IVAL1 = VAL1

           IF (IVAL1 .GT. 4) THEN
              CALL CSROFF
           CALL TTOUT(0,24,1,'Too may values -- Hit any key ',30,30,28)
              CALL TTOUT(0,24,31,'to resume....',13,13,28)
              CALL INKEY(KYCOD1,KYCOD2)
              CALL CSRON
              IER = 1
              GOTO 1000
           ENDIF

           IF (IVAL1 .LT. 1) THEN
              CALL CSROFF
           CALL TTOUT(0,24,1,'Too few values -- Hit any key ',30,30,28)
              CALL TTOUT(0,24,31,'to resume....',13,13,28)
              CALL INKEY(KYCOD1,KYCOD2)
              CALL CSRON
              IER = 1
              GOTO 1000
           ENDIF

C     THE PROGRAM THEN PROMPTS THE USER FOR EACH VALUE FOR EVERY PARAMETER.
C     THESE VALUES ARE THEN WRITTEN TO THE LOCATION OF THE FORMAL
C     PARAMETER IN THE SYMBOL TABLE FILE.

           DO 730 JJJJ = 1,IVAL1
              IF (JJJJ .EQ. 1) C(1:1) = '1'
              IF (JJJJ .EQ. 2) C(1:1) = '2'
              IF (JJJJ .EQ. 3) C(1:1) = '3'
              IF (JJJJ .EQ. 4) C(1:1) = '4'
              CALL TTOUT(0,11+JJJJ,25,'Value ',6,6,31)
              CALL TTOUT(0,11+JJJJ,31,C(1:1),1,1,31)
              CALL TTOUT(0,11+JJJJ,33,'= ',2,2,31)
              CALL PARTM(0,11+JJJJ,35,CHAR,LL,1)
              WRITE (9,100) LL,(CHAR(MM:MM),MM=1,LL)
              REWIND (9)
              READ (9,800) VAL(JJJJ)
              REWIND (9)

730        CONTINUE

           WRITE (8,500,REC=IC21(I)) IC21(I),(VAL(JJJJ),JJJJ=1,IVAL1)
           GOTO 702

        ENDIF

        GOTO 1000

702   CONTINUE
```

```
100   FORMAT(I2,1X,72A1)
300   FORMAT(1X,I3,7X,I4,2X,I3,2X,I1,6X,A72)
400   FORMAT(18X,I4,1X,I4)
500   FORMAT(1X,I3,4(2X,F8.2))
501   FORMAT(4X,4(2X,F8.2))
600   FORMAT(1X,I3,8(1X,I4))
699   FORMAT(4X,8(1X,I4))
800   FORMAT(3X,F8.2)

1000  CLOSE (9,STATUS='DELETE')

      RETURN
      END

C     SUBROUTINE SLOW

C     THIS SUBROUTINE ASKS USERS IF THEY WANT TO OPT FOR SLOW EXECUTION.

C     SUBROUTINES CALLED BY THIS PROGRAM ARE :
C             CLS,CSROFF,CSROFF,INKEY,LOCATE,TTOUT,USER.

      SUBROUTINE SLOW

      INTEGER COUNT(50),CNTVAL(50),PALLET(10),PALPTS(10),PARTNO(10)
      INTEGER CONS(100),AGG(20),PAR(50),PNTVAL(100)
      INTEGER SS,SLPAY,PART1,PART2
      INTEGER * 2 KYCOD1,KYCOD2,JJ1(2)
      CHARACTER * 1 C(2),BLANK(80)

      DIMENSION LASUB(10),IONUM(32),CONVAL(100),L(5),M(10),P(4)
      DIMENSION XPAL(10,50),YPAL(10,50),ZPAL(10,50),RPAL(10,50)
      DIMENSION XPNT(100),YPNT(100),ZPNT(100),RPNT(100)
      DIMENSION XAGG(20),YAGG(20),ZAGG(20),RAGG(20)
      DIMENSION XPAR(50),YPAR(50),ZPAR(50),RPAR(50)

      COMMON/A/COUNT,CNTVAL,PALLET,PALPTS,PARTNO
      COMMON/B/CONS,CONVAL,PAR,PNTVAL,AGG,NREC
      COMMON/C/LASUB,IONUM,LAST,NEXT,ALIN,PAY,ZON,IC
      COMMON/D/XPAL,YPAL,ZPAL,RPAL,XVAL,YVAL,ZVAL,RVAL
      COMMON/E/XPNT,YPNT,ZPNT,RPNT,XAGG,YAGG,ZAGG,RAGG
      COMMON/F/NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,XPAR,YPAR,ZPAR,RPAR
      COMMON/G/SS,SLPAY,PART1,PART2,PAYOLD

      DATA BLANK/80 * ' '/

      CALL CLS(31)
      CALL CSRON
      CALL TTOUT(0,10,1,'Do you want the robot arm to move slowly',
     *40,40,31)
      CALL TTOUT(0,10,42,'during program execution (Y/N) ?',32,32,31)
      CALL LOCATE(0,10,76)
      CALL INKEY(KYCOD1,KYCOD2)

C     IF THE USER OPTS FOR SLOW ROBOT ARM MOVEMENT, PROMPT FOR
C     THE PAYLOAD VALUE, AND READ THE VALUE FROM THE USER'S INPUT.
C     THE VALID KEYS THAT THE USER CAN STRIKE ARE ONLY NUMERIC
C     CHARACTERS AND THE ENTER KEY.
C     THE FIRST CHARACTER TYPED IN IS STORED IN JJ1(1) AND THE
C     SECOND CHARACTER IS STORED IN JJ1(2). THE PAYLOAD VALUE
C     IS THEN COMPUTED AND STORED IN THE VARIABLE SLPAY.

      IF (KYCOD1 .EQ. 89 .OR. KYCOD1 .EQ. 121) THEN
         CALL CLS(31)
      CALL TTOUT(0,10,1,'Please specify a payload value at ',34,34,31)
33    CALL TTOUT(0,11,1,'which you want the program run.',31,31,31)
         JJJ = 1
34       CALL LOCATE(0,11,32+JJJ)
         CALL INKEY(KYCOD1,KYCOD2)
```

```
            IF (KYCOD1 .EQ. 13 .AND. JJJ .EQ. 1) THEN
               SLPAY = 0
               RETURN
            ELSEIF (KYCOD1 .EQ. 13 .AND. JJJ .EQ. 2) THEN
               SLPAY = JJ1(1)
               RETURN
            ELSEIF (KYCOD1 .EQ. 48) THEN
               C(JJJ) = '0'
               JJ1(JJJ) = 0
               CALL TTOUT(0,11,32+JJJ,C(JJJ),1,1,31)
            ELSEIF (KYCOD1 .EQ. 49) THEN
               C(JJJ) = '1'
               JJ1(JJJ) = 1
               CALL TTOUT(0,11,32+JJJ,C(JJJ),1,1,31)
            ELSEIF (KYCOD1 .EQ. 50) THEN
               C(JJJ) = '2'
               JJ1(JJJ) = 2
               CALL TTOUT(0,11,32+JJJ,C(JJJ),1,1,31)
            ELSEIF (KYCOD1 .EQ. 51) THEN
               C(JJJ) = '3'
               JJ1(JJJ) = 3
               CALL TTOUT(0,11,32+JJJ,C(JJJ),1,1,31)
            ELSEIF (KYCOD1 .EQ. 52) THEN
               C(JJJ) = '4'
               JJ1(JJJ) = 4
               CALL TTOUT(0,11,32+JJJ,C(JJJ),1,1,31)
            ELSEIF (KYCOD1 .EQ. 53) THEN
               C(JJJ) = '5'
               JJ1(JJJ) = 5
               CALL TTOUT(0,11,32+JJJ,C(JJJ),1,1,31)
            ELSEIF (KYCOD1 .EQ. 54) THEN
               C(JJJ) = '6'
               JJ1(JJJ) = 6
               CALL TTOUT(0,11,32+JJJ,C(JJJ),1,1,31)
            ELSEIF (KYCOD1 .EQ. 55) THEN
               C(JJJ) = '7'
               JJ1(JJJ) = 7
               CALL TTOUT(0,11,32+JJJ,C(JJJ),1,1,31)
            ELSEIF (KYCOD1 .EQ. 56) THEN
               C(JJJ) = '8'
               JJ1(JJJ) = 8
               CALL TTOUT(0,11,32+JJJ,C(JJJ),1,1,31)
            ELSEIF (KYCOD1 .EQ. 57) THEN
               C(JJJ) = '9'
               JJ1(JJJ) = 9
               CALL TTOUT(0,11,32+JJJ,C(JJJ),1,1,31)
            ELSE
               CALL CSROFF
            CALL TTOUT(0,24,1,'INVALID CHARACTER - Hit any key to resume'
          *  ,41,41,28)
               CALL INKEY(KYCOD1,KYCOD2)
               CALL TTOUT(0,24,1,BLANK,80,80,31)
               CALL CSRON
               GOTO 34
            ENDIF

            JJJ = JJJ + 1
            IF (JJJ .LE. 2) GOTO 34
            SLPAY = JJ1(1) * 10 + JJ1(2)

C     IF THE USER SPECIFIES A PAYLOAD VALUE GREATER THAN 19, DISPLAY
C     AN ERROR MESSAGE AND PROMPT THE USER AGAIN.

            IF (SLPAY .GT. 19) THEN
               SLPAY = 0
               CALL USER(25)
               GOTO 33
            ENDIF
```

```
ENDIF

RETURN
END
```

APPENDIX D. ASSEMBLY ROUTINES AND THEIR CALLING SEQUENCES

[NOTE: All integer variable should be declared as INTEGER*2] CNTRL

This subroutine controls execution of an application program.

The calling sequence is:

```
CALL CNTRL(ICODE,IER)
```

where,

ICODE, an integer variable, could be one of the following:

1 - Suspend program execution

2 - Restart program execution

4 - Execute until next terminator encountered

20 - Reset controller

IER is an integer variable containing one of the following error return codes:

0 - Communication was successfully completed

1 - An EOT was encountered, communication was not successful

2 - A NAK was encountered, communication was not successful

EXEC

This subroutine performs a remote function related to the operator control panel.

The calling sequence is:

```
CALL EXEC(ICODE,IER)
```

where,

ICODE, an integer variable, could be one of the following:

11 - Return Home

12 - Recall Memory

13 - Reset Error

20 - Auto

22 - Start Cycle

23 - Stop Cycle

24 - Stop and Mem

25 - Step

31 - Select Application 1

32 - Select Application 2

33 - Select Application 3

34 - Select Application 4

35 - Select Application 5

IER is the error return code. Same as described earlier.

PUTWAIT

This subroutine waits for the execution of a PUT command by the controller.

The calling sequence is:

```
CALL PUTWAIT(IER)
```

where,

IER is the error return code. Same as described earlier.

RDIDO

This subroutine reads the status of any one of the 16 DI or DO ports from the controller.

The calling sequence is:

```
CALL RDIDO(IONUM,IOSTAT,IER)
```

where,

IONUM is the integer value of the desired port number:

1-16 for DI ports 1-16

17-32 for DO ports 1-16

IOSTAT is the integer value specifying the status of the port:

0 - Off

1 - On

IER is the error return code. Same as described earlier.

READP

This subroutine reads the X, Y, Z and R values of a point from four successive controller memory locations.

The calling sequence is:

CALL READP(INUM,PX,PY,PZ,PR,IER) or,

CALL READP(INUM,P,IER) where P is a four dimensional array.

where,

INUM is the integer value of the starting variable number which is the X value of the point.

PX (or P(1)) - X value of the point

PY (or P(2)) - Y value of the point

PZ (or P(3)) - Z value of the point

PR (or P(4)) - R value of the point

IER is the error return code. Same as described earlier.

READR

This subroutine reads a real value (+/-32767.0) from the controller memory.

The calling sequence is:

CALL READR(INUM,VAL,IER)

where,

INUM is the integer value of the starting variable number;

VAL is the real value to be read from the controller;

IER is the error return code. Same as described earlier.

RSTATUS

This subroutine reads the reject and machine status.

The calling sequence is:

CALL RSTATUS(ICODE,IVAL1,IVAL2,IER) or,

CALL RSTATUS(ICODE,IVAL,IER) where IVAL is a two dimensional array

where,

ICODE an integer variable, could be one of the following:

1 - Read machine status

2 - Read reject status (why the controller sent the EOT)

8 - Read current instruction address

IVAL1 (or IVAL(1)) is an integer return value which could be one of the following if ICODE is 1:

128 - Servo failure

64 - Power failure

32 - Overrun

16 - Over Time

8 - Transmission error

6 - AML/E error

4 - Data error

If IVAL1 is 6, then IVAL2 (or IVAL(2)), an integer variable, contains additional information:

10 - Part number too small for pallet

11 - Part number too large for pallet

20 - Invalid index for a group

30 - Communications not established

If IVAL1 is 4, then IVAL2 contains additional information:

0 - No data error present

1 - Bus error

2 - Memory test error

17 - Arithmetic error

18 - Programming error

19 - Invalid op code

20 - Invalid data

21 - Invalid port number

23 - Stack error

24 - Address error

64 - Point out of workspace

If ICODE is 2, then IVAL1 could be:

16 - Record format error

21 - Invalid port number

32 - Undefined record

48 - Unacceptable condition (Improper application startup sequence)

49 - Unacceptable condition (C record, but no application)

64 - Point out of workspace

80 - Insufficient memory

81 - Invalid robot type

82 - Double select error

83 - Invalid application number

84 - C record format error

96 - Invalid identifier sent before N record

112 - Xoff time out (30 secs)

116 - Invalid data

117 - Too much data

118 - Too little data

128 - Manipulator power off

160 - Xoff time out State

161 - Cannot accept command (Xon State)

162 - Communication operation not allowed (Xto or Wxo mode)

170 - Controller has gone off-line

IER is the error return code. Same as described earlier.

WRITEP

This subroutine writes the X, Y, Z and R values of a point into four successive controller memory locations.

The calling sequence is:

CALL WRITEP(INUM,PX,PY,PZ,PR,IER)

CALL WRITEP(INUM,P,IER) where P is a four dimensional array

where,

INUM is the integer value of the starting variable number which is the X value of the point.

PX (or P(1)) - X value of the point

PY (or P(2)) - Y value of the point

PZ (or P(3)) - Z value of the point

PR (or P(4)) - R value of the point

IER is the error return code. Same as described earlier.

WRITER

This subroutine writes a real value (+/-32767.0) into the controller memory.

The calling sequence is:

```
CALL WRITER(INUM,VAL,IER)
```

where,

INUM is the integer value of the starting variable number;

VAL is the real value to be transmitted to the controller;

IER is the error return code. Same as described earlier.

APPENDIX E. MOTION CONTROL PROGRAM

```
1
2
3     --     THIS AML/E PROGRAM RESIDES IN THE ROBOT CONTROLLER AND
4     --     IS RESPONSIBLE FOR EXECUTION OF THE COMMANDS TRANSMITTED
5     --     FROM THE HOST COMPUTER. DATA IS WRITTEN TO THE MEMORY
6     --     OF THE CONTROLLER, SPECIFYING THE VALUES OF LINEAR, PAYLOAD,
7     --     ZONE, THE TYPE OF COMMAND TO BE EXECUTED AND THE
8     --     PARMETERS NECESSARY TO EXECUTE THE COMMAND. THE
9     --     CONTROLLER SENDS DATA TO THE HOST, IF NEEDED, USING "PUT"
10    --     COMMANDS  AND THE TYPES OF DATA THAT CAN BE TRANSFERRED ARE
11    --     INTEGER COUNTERS.
12
13
14    --     DEFINITION OF VARIABLES USED IN THE PROGRAM
15
16    CODE          : STATIC COUNTER;          -- INTEGER COUNTER
17    LIN           : STATIC COUNTER;          -- INTEGER COUNTER
18    PAY           : STATIC COUNTER;          -- INTEGER COUNTER
19    ZON           : STATIC COUNTER;          -- INTEGER COUNTER
20    VALUE         : STATIC COUNTER;          -- INTEGER COUNTER
21    IONUM         : STATIC COUNTER;          -- INTEGER COUNTER
22    TIME          : STATIC COUNTER;          -- INTEGER COUNTER
23    RETURN_CODE   : STATIC COUNTER;          -- INTEGER COUNTER
24    POINT1        : NEW PT(0,0,0,0);         -- POINT
25
26    --     BEGINNING OF PROGRAM
27
28    RADHA : SUBR;
29
30     EXEC : SUBR;
31
32
33       LINEAR(LIN);
34       PAYLOAD(PAY);
35       ZONE(ZON);
36
37       SETC (RETURN_CODE,10);
38       TESTC(CODE,1,L_BREAK);
39       TESTC(CODE,2,L_DELAY);
40       TESTC(CODE,6,L_PMOVE);
41       TESTC(CODE,7,L_WAITI1);
42       TESTC(CODE,8,L_WAITI2);
43       TESTC(CODE,9,L_WRITE0);
44
45
46      L_BREAK : BREAKPOINT;
47                BRANCH(FINI);
48
49      L_DELAY : DELAY(TIME);
50                BRANCH(FINI);
51
52      L_PMOVE : PMOVE(POINT1);
53                BRANCH(FINI);
54
55      L_WAITI1: WAITI(IONUM,VALUE,TIME);
56                BRANCH(FINI);
57
58      L_WAITI2: WAITI(IONUM,VALUE,TIME,TIMOUT);
59                SETC(CODE,0);
60                BRANCH(FINI2);
61
62        TIMOUT: SETC(RETURN_CODE,0);
63                SETC(CODE,0);
```

```
64                BRANCH(FINI2);
65
66      L_WRITEO: WRITEO(IONUM,VALUE);
67
68
69      FINI:
70                SETC(RETURN_CODE,10);
71                SETC(CODE,0);
72
73      FINI2:
74
75     END;
76
77    --    START OF EXECUTABLE CODE
78
79    SETC(CODE,0);
80
81    START :
82
83            COMPC(CODE = 0, START);
84            COMPC(CODE = 3, FINISH);
85
86    CALL  : EXEC;
87
88            BRANCH(START);
89
90    FINISH :
91
92    END;
```

APPENDIX F. TEST PROGRAM NUMBER 1

```
1     --    TEST PROGRAM NO. 1
2
3     --    THIS PROGRAM TESTS ALL THE AML/E COMMANDS, USING BOTH
4     --    VARIABLES AND NUMBERS. THIS PROGRAM IS USED TO
5     --    VERIFY THE "SYSTEM CONTROL" PROGRAM ON THE IBM PC.
6
7     --    PROGRAM WRITTEN BY RADHAKRISHNAN JAYARAMAN  (MARCH 1986)
8
9     CNT1 : STATIC COUNTER;
10    CNT2 : STATIC COUNTER;
11    CNT3 : STATIC COUNTER;
12    CNT4 : STATIC COUNTER;
13
14    CON0 : NEW 0;
15    CON1 : NEW 1;
16    CON2 : NEW 2;
17    CON3 : NEW 3;
18    CON5 : NEW -10;
19    CON6 : NEW -10;
20    CON7 : NEW -10;
21    CON8 : NEW -10;
22
23    DP1  : NEW <-10,-10,-10,-10>;
24
25    LL1 : NEW PT(-500,-55.00,-200.,180.0);
26    LR1 : NEW PT(-396.07,5,-200,180);
27    UR1 : NEW PT(-441.07,82.94,-200,180);
28    PPR1: NEW 3;
29    NUM : NEW 6;
30
31    PAL1 : STATIC PALLET(LL1,LR1,UR1,PPR1,NUM);
32
33    MAIN : SUBR;
34
35          SETC(CNT1,CON1);
36          SETC(CNT2,CNT1);
37          SETC(CNT3,-100);
38          SETC(CNT4,0);
39
40          ZONE(CON2);
41          PAYLOAD(CON1);
42          PMOVE(LL1);
43          LINEAR(CON1);
44          DPMOVE(DP1);
45          LINEAR(0);
46          PAYLOAD(0);
47          ZONE(0);
48          PMOVE(PT(-396.07,5,-20,180));
49          LINEAR(CNT1);
50          DPMOVE(<-10,-10,-10,-10>);
51          ZMOVE(CNT4);
52          GRASP;
53          DELAY(CNT1);
54          PAYLOAD(CNT1);
55          ZMOVE(-30);
56          ZMOVE(CON0);
57          SETC(CNT1,76);
58          SETC(CNT2,490);
59          PMOVE(PT(CNT1,CNT2,CNT3,CNT4));
60          RELEASE;
61          DELAY(2.0);
62          SETC(CNT1,-10);
63          SETC(CNT2,-10);
64          SETC(CNT3,10);
65          DPMOVE(<CON5,CON6,CON7,CON8>);
66          ZMOVE(CON0);
```

```
 67          DELAY(CON1);
 68
 69          SETC(CNT1,1);
 70          SETC(CNT2,2);
 71          SETC(CNT3,3);
 72          SETC(CNT4,4);
 73
 74          TESTC(CNT1,1,LAB1);
 75          ZMOVE(-100);
 76
 77     LAB1: TESTC(CNT1,CNT2,FINI);
 78          TESTC(CNT1,CON3,FINI);
 79          TESTC(CNT1,CON1,LAB2);
 80          BREAKPOINT;
 81
 82     LAB2: INCR(CNT1);
 83          TESTC(CNT1,1,FINI);
 84          TESTC(CNT1,CNT2,LAB3);
 85          BREAKPOINT;
 86
 87     LAB3: COMPC(CNT1 = 0,FINI);
 88          COMPC(CNT1 <= 0,FINI);
 89          COMPC(CNT1 <> 2,FINI);
 90          COMPC(CNT1 >= CNT4,FINI);
 91          COMPC(CNT1 < 0,FINI);
 92          COMPC(CNT1 > CON3,FINI);
 93          COMPC(CNT1 = CON2,LAB4);
 94          BREAKPOINT;
 95
 96     LAB4: COMPC(CNT4 > CON3,LAB5);
 97          BREAKPOINT;
 98
 99     LAB5: COMPC(CNT4 <> 0,LAB6);
100          BREAKPOINT;
101
102     LAB6: COMPC(CNT3 <= CNT4,LAB7);
103          BREAKPOINT;
104
105     LAB7: COMPC(CNT2 < CNT4,LAB8);
106          BREAKPOINT;
107
108     LAB8: DECR(CNT4);
109          COMPC(CNT1 = CNT2,LAB9);
110          BREAKPOINT;
111
112     LAB9: SETPART(PAL1,CON1);
113          GETPART(PAL1);
114          NEXTPART(PAL1);
115          TESTP(PAL1,2,LAB10);
116          BREAKPOINT;
117
118     LAB10:
119          LINEAR(0);
120          PAYLOAD(0);
121          PMOVE(PT(650,0,0,0));
122          SETPART(PAL1,CNT1);
123          GETPART(PAL1);
124          PREVPART(PAL1);
125          GETPART(PAL1);
126          NEXTPART(PAL1);
127          GETPART(PAL1);
128          SETPART(PAL1,4);
129          GETPART(PAL1);
130          BRANCH(LAB11);
131          BREAKPOINT;
132
133     LAB11:
134          WRITEO(CON2,CON1);
135          SETC(CNT1,1);
```

```
136         WRITEO(15,CNT1);
137         BREAKPOINT;
138         SETC(CNT4,15);
139         WRITEO(CNT4,0);
140         TESTI(CON3,1,LAB12);
141         BREAKPOINT;
142
143   LAB12:
144         TESTI(CNT1,CON0,LAB13);
145         BREAKPOINT;
146
147   LAB13:
148         WAITI(CNT1,CNT1,CON0);
149         WAITI(CON1,CNT1,15,LAB14);
150         BREAKPOINT;
151
152   LAB14:
153         BREAKPOINT;
154
155   FINI:
156         LINEAR(0);
157         PMOVE(PT(650,0,0,0));
158         BREAKPOINT;
159
160   END;
```

APPENDIX G. TEST PROGRAM NUMBER 2

```
--    TEST PROGRAM NO. 2

--    THIS PROGRAM TESTS ALL THE AML/E COMMANDS, WITH PARAMETERS
--    BEING PASSED INTO SUBROUTINES. THE "SYSTEM CONTROL"
--    PROGRAM CAN HANDLE UP TO FIVE FORMAL PARAMETERS. THIS
--    PROGRAM ALSO TESTS CASES WHEN THE NAMES OF LOCAL VARIABLES
--    ARE IDENTICAL IN DIFFERENT SUBROUTINES.

--    PROGRAM WRITTEN BY RADHAKRISHNAN JAYARAMAN  (MARCH 1986)

CNT1 : STATIC COUNTER;
CNT2 : STATIC COUNTER;
CNT3 : STATIC COUNTER;
CNT4 : STATIC COUNTER;

CON0 : NEW 0;
CON1 : NEW 1;
CON2 : NEW 2;
CON3 : NEW 3;
CON5 : NEW -10;
CON6 : NEW -10;

LR1  : NEW PT(-396.07,5,-100.,180);
UR1  : NEW PT(-441.07,82.94,-50,180);


MAIN : SUBR;

 SUB1  : SUBR;

      PMOVE(LR1);
      PMOVE(PT(650,0,0,0));
 END;

 SUB2  : SUBR(PNT);

      LINEAR(1);
      LINEAR(0);
      PMOVE(PNT);
      PMOVE(PT(650,0,0,0)) ;
 END;

 SUB3  : SUBR(PNT,CNT,CNTT,CONS1,CONS2);

 CONS3 : NEW 0;
 CONS4 : NEW 1;
 LL1   : NEW PT(-500,-55.00,-100.,180.0);
 CNT8  : STATIC COUNTER;
 CNT9  : STATIC COUNTER;
 DP1   : NEW <-10,-10,-10,-10>;


      SETC(CNT8,CONS3);
      SETC(CNT9,CONS4);
      SETC(CNT4,CONS2);

      ZONE(CONS2);
      PAYLOAD(CONS1);
      PMOVE(PNT);
      LINEAR(CONS1);
      LINEAR(0);
      PAYLOAD(0);
      ZONE(0);
      PMOVE(LL1);
      LINEAR(CNT);
      DPMOVE(DP1);
```

```
      ZMOVE(CNTT);
      GRASP;
      DELAY(CONS1);
      PAYLOAD(CNT);
      ZMOVE(-30);
      SETC(CNT8,76);
      SETC(CNT9,490);
      SETC(CNT3,0);
      SETC(CNT4,0);
      PMOVE(PT(CNT8,CNT9,CNT3,CNT));
      RELEASE;
      DELAY(2.0);
      SETC(CNT2,-10);
      SETC(CNT3,10);
      ZMOVE(CNT2);
      DELAY(CONS2);
END;

SUB4  : SUBR(PNT,CNT,CONS1,CONS2,CNTT);

CONS3 : NEW 1;
CONS4 : NEW 0;
LL1   : NEW PT(-441.07,82.94,-50,180);
CNT8  : STATIC COUNTER;
CNT9  : STATIC COUNTER;
DP1   : NEW <-30,-30,-30,-30>;


      SETC(CNT8,CONS1);
      SETC(CNT9,CONS2);
      SETC(CNT3,5);

      LINEAR(0);
      PAYLOAD(CONS1);
      ZONE(CNT);
      PMOVE(LR1);
      PMOVE(PNT);
      PMOVE(LL1);
      ZMOVE(0);
      DPMOVE(DP1);

      TESTC(CNT3,5,LAB1);
      BREAKPOINT;

LAB1: TESTC(CNT3,CNT8,FINI);
      TESTC(CNT9,CONS1,FINI);
      TESTC(CNT8,CONS1,LAB2);
      BREAKPOINT;

LAB2: DECR(CNT3);
      DECR(CNT3);
      TESTC(CNT3,1,FINI);
      TESTC(CNT3,CONS2,LAB3);
      BREAKPOINT;

LAB3: COMPC(CNT3 = CNTT,FINI);
      COMPC(CNT3 <= CNTT,FINI);
      COMPC(CNT3 <> 3,FINI);
      COMPC(CNT3 >= 9,FINI);
      COMPC(CNT3 < 0,FINI);
      COMPC(CNT3 > 3,FINI);
      COMPC(CNT3 = CONS2,LAB4);
      BREAKPOINT;

LAB4: COMPC(CNT3 > CON0,LAB5);
      BREAKPOINT;

LAB5: COMPC(CNT3 <> 0,LAB6);
      BREAKPOINT;
```

```
LAB6: COMPC(CNT3 <= 6,LAB7);
      BREAKPOINT;

LAB7: COMPC(CNT3 < 900,LAB8);
      BREAKPOINT;

LAB8: DECR(CNT2);
      DECR(CNT2);
      COMPC(CNT3 = CONS2,LAB9);
      BREAKPOINT;

LAB9:
      WRITEO(CONS1,CON1);
      WRITEO(15,CNTT);
      BREAKPOINT;
      SETC(CNT3,15);
      WRITEO(CNT3,0);
      TESTI(CNT1,1,LAB10);
      BREAKPOINT;

LAB10:
      TESTI(CNT1,CON0,LAB11);
      BREAKPOINT;

LAB11:
      WAITI(CNT1,CNT1,CON0);
      WAITI(CON1,CNT1,15,LAB12);
      BREAKPOINT;

LAB12:
      BREAKPOINT;

FINI:
      LINEAR(0);
      PMOVE(PT(650,0,0,0));
      BREAKPOINT;

 END;

      SUB1;
      SUB2(LR1);
      SETC(CNT1,1);
      SETC(CNT2,2);
      SUB3(LR1,CNT1,0,CON1,CNT2);
      SETC(CNT1,1);
      SETC(CNT4,0);
      SUB4(UR1,CNT1,CON2,3,CNT4);
      LINEAR(0);
      PMOVE(PT(650,0,0,0));
      BREAKPOINT;

END;
```

The vita has been removed from the scanned document