LA-5343-MS

INFORMAL REPORT

LAMPF Data Handling Workshop

Introduction to PDP-11 DOS-8

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87544

UNITED STATES
ATOMIC ENERGY COMMISSION
CONTRACT W-7405-ENG. 36

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

In the interest of prompt distribution, this LAMS report was not edited by the Technical Information staff.

Printed in the United States of America. Available from
National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22151
Price: Printed Copy \$4.00; Microfiche \$0.95

LA-5343-MS

Informal Report

UC 32

ISSUED: July 1973

LAMPF Data Handling Workshop

Introduction to PDP-11 DOS-8

by

James Amann*

*Visiting Staff Member.

Permanent address: Carnegie-Mellon University, Pittsburgh, Pennsylvania.

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

LAMPF DATA HANDLING WORKSHOP
INTRODUCTION TO PDP-11 DOS-8

by
James Amann

ABSTRACT

The purpose of this document is to enable someone who is not familiar with the PDP-11 to enter, compile, link, and run a FORTRAN program by following the instructions contained herein. It should also provide the user with sufficient familiarity with the organization of the system that more advanced and detailed information may be readily obtained and understood from the appropriate systems manuals.

I. GENERAL SYSTEM ORGANIZATION AND INFORMATION

The primary avenue of communication between the user and the system is through the teletype (Dec-writer) keyboard. Through the keyboard the user may communicate with the Monitor (the resident system control program), DOS (the disk operating system) systems programs (e.g., PIP, FORTRN, EDIT, LINK, etc), or with his own programs written to run under the Monitor. Under normal circumstances all programs, both systems programs and user programs, are maintained in files on the disk.

A. User Identification Codes

In order for the system to be able to distinguish between various users for the purposes of keeping files, each user must have his own identification code. This code number is a sequence of two octal numbers separated by a comma. Each number must be in the range from 1 to 376 inclusive. Thus 15,172 and 45,11 are valid UIC's, whereas 208,16 and 57,400 are not. (208 is not a valid octal number and 400 is greater than 376.)

The only other restriction on the selection of a UIC is not to pick a set of numbers which is the same as any currently being used by the system or by other users. The system has UIC[1,1] on all machines. For each machine a list of UIC's already claimed by someone else should be available from

someone familiar with the particular computer. The procedure for entering one's UIC onto the disk's Master File Directory (MFD) or list of users will be described later.

B. User Files

A user filename consists of a string of up to six characters (legitimate characters are the letters A-Z and the numbers 0-9), followed by a . (period) and a three-character extension. The purpose of the extension is to allow the user to have several files with the same name but different extensions. For example, one may have the same filename but different extensions for the source program (extension .FTN), the compiled object module (extension .OBJ), and the load module or output of the LINK program (extension .LDA). The choice of the filename is quite arbitrary and may be anything you like, but for the purposes of later convenience all FORTRAN source programs should have the extension .FTN (e.g., a sample filename is JUNK.FTN). Additional information concerning extensions will be presented later during the discussion of the procedure for compiling and linking a FORTRAN program.

C. Communicating with the System

1. Interpreting Monitor Output

Under many circumstances the system will

request information from the user via the keyboard. In these cases a single character will be printed on the teleprinter. Depending on the type of information requested and the status of the system the following characters may be printed.

CHARACTER	MEANING
\$	The system is idle and is awaiting an operator command. This is the standard command issued when the Monitor has been entered either by bootstrapping the system (a procedure to be described shortly), or by terminating the program in progress through a KILL command, or when certain errors in the operation of a currently running program have been encountered.
.	(period) The Monitor has acknowledged a CTRL/C typed by the user and is in listening mode, ready to accept a command from the operator.
#	The system program or user's program requests an operator reply through a special routine called the Command String Interpreter (CSI). More will be said about the CSI in Section III.
*	A system program requests an input message directly (i.e., not through the CSI).

2. Special Function Keys

Certain keyboard keys have special functions as listed below.

KEY	FUNCTION
CTRL/C	A CTRL/C is typed by holding down the CTRL key while typing the C key. This alerts the Monitor to be ready to accept a command from the keyboard. Typing CTRL/C appears on the teleprinter as ^C with the Monitor acknowledgement . (period) on the next line.
CR (carriage return)	Pressing the key labelled CR terminates an operator command to the Monitor, or a line of input to a system or user program. The CR key produces a carriage return (CR) and line feed (LF) on the teleprinter.
RUBOUT	This key permits the correction of typing errors. Pressing RUBOUT once causes the last character typed to be deleted. RUBOUT does not delete characters other than on the current line of input (i.e., past the last CR). If the last remaining character has been deleted, a RUBOUT will be ignored. The Monitor will print out the deleted characters set off by backslashes. For example, if you wished to type EPICS and by mistake typed EPEX instead, the error could be corrected by typing two RUBOUTS and then the correct letters. The output on the keyboard would then look like

KEY	FUNCTION
	EPEX\XEN\CS. Note that the deleted characters are shown in reverse order, i.e., the order in which they are deleted.
CTRL/U	CTRL/U is typed by holding down the CTRL key while typing the letter U. It is used if the user has completely botched the current line and wishes to try it again. The typing of CTRL/U deletes the entire current line past the last CR. CTRL/U will echo on the teleprinter as ^U followed by a CR and a LF.

3. Commands to the Monitor

There are a number of commands one may give to the Monitor. Only the first two characters of the command word are significant although more characters (those given in parentheses) may be typed for the sake of intelligibility. Some commands are capable of having arguments after them, separated from the command by a space, others require that some arguments be given after the command (e.g., RUN). Some of the more commonly used commands are given below along with their function and arguments.

Hereafter the symbol CR will be used to indicate those places where the user is to type a carriage return.

COMMAND	FUNCTION
DA(TE)	This command allows the setting of an internal word in the Monitor to contain the current date. The format for input is DATE 29-JAN-73(CR) If no argument is given (i.e., just DATE(CR) is typed) the date previously entered will be printed. This command is OK any time the Monitor is listening (i.e., if a \$ of a . (period) has been typed by the Monitor).
TI(ME)	This command allows the setting of the time of day entry in the Monitor. The input format is TIME 17:23:40(CR) A 24-h clock is used, with the hour, minute, and second entries being separated by colons. One need not specify the entire string given above. For example, one may simply type time 17:23 without an entry for seconds. If no argument at all is given, the present contents of the time of day word is printed.
FI(NISH)	The FINISH command informs the Monitor that the current user is leaving the system. It should be used whenever you finish a session on the computer. The command is only valid when no program is currently loaded in core. If the last character printed on the teletype

<u>COMMAND</u>	<u>FUNCTION</u>
	is not a \$, the user should precede the FINISH by a CTRL/C and a KILL.
KI(LL)	The KILL command is used to terminate the execution of a program currently in core and enter the Monitor. It is necessary to give this command after running a program before any other program may be run. Some programs terminate themselves, leaving you in the Monitor. This is indicated by a \$ being typed with no error message.
LO(GIN)	This command allows the user to give his User Identification Code to the Monitor. It is necessary to give this command before the user attempts to enter a program or run one of his own programs. The format for input is LO 31,36(CR) where 31,36 is the UIC of the current user. This command is only valid if there is no currently logged in user. If a previous user is still logged in, a FINISH command must be given. The LOGIN command need only be given once for each time the user runs the machine, and it will remain in effect until a FINISH is issued or until the Monitor is reloaded from the disk.
RU(N)	This command is used to load your linked program into core and begin its execution. In order to run the program JUNK (extension .LDA) the following command sequence is typed RUN JUNK (CR) in response to the Monitor's \$. It is assumed that the file in question is contained in the Directory either of the currently logged in user or the system (UIC 1,1) and has extension .LDA. This command is only valid when there is no program already loaded.
CO(NTINUE)	This command is used after a recoverable error condition (an 'A' error message) to resume the program operation at the point where it was interrupted. It is most often used to continue operation after a PAUSE statement in a FORTRAN program. It is valid only if a program is already in core.

D. Getting on the Air for the First Time

1. Bootstrapping the System

In addition to the following discussion, an actual printout obtained by using the procedures given below is contained in the Appendix A.

The following is the procedure for "bootstrapping" the Monitor from disk. Although this procedure is not necessary each time one uses the computer, it is needed in the case of a system crash or if the power to the CPU (Central Processor Unit) has been turned off.

- Make sure that the power is on.
- Set the console switch labelled 'HALT' to the down position. Depress the switch labelled START. (On some machines it may be necessary to lift the START switch back to its 'up' position after depressing it.) Now return the 'HALT' switch to its up position.
- Set the octal number 173100 into the keys on the console switch register. (Note that for all PDP-11's up = 1 and down = 0.) Now depress the LOAD ADDRESS switch.
- The number one now sets in the switch register depends on the type of disk the particular computer possesses.
For a type 'RF' (small) disk set the octal number 177462
For a type 'RK' (big) disk set the octal number 177406
After setting the appropriate number depress the START switch (lifting it again if necessary). The Monitor will now be transferred from the disk to the memory and a message similar to the following will be typed:

TIME: - 08:35:27

18 - JUN - 73

BATCH V008-02

You are now ready to begin the procedure of logging in. Under normal circumstances it should not be necessary to use the procedure given above. This is the case if the Monitor has printed a \$ as the last character on the teleprinter, then the above steps may be skipped.

2. Logging In

After the \$ type LO and your UIC (e.g., if your UIC is 23,35 enter LO 23,35(CR)). The system will now type out the DATE and TIME. If these are incorrect or not to your satisfaction you may re-enter these quantities as described in Section 1.C.3.

3. Entering Yourself in the Master File Directory

This is a procedure you need only do once, that is the first time you use any given machine. First make sure that you have logged in as described above. Then after the \$, type RUN PIP(CR). After PIP has been loaded and types out a # simply type /EN(CR) and you are entered.

II. DOS SYSTEMS PROGRAMS

A. General Description

Contained below is a brief description of the function of some of the more commonly used systems programs involved in running FORTRAN programs. A fairly detailed set of instructions as to the use of each will follow.

<u>PROGRAM</u>	<u>FUNCTION</u>
EDIT	Is a program which allows the user to enter a new program or modify an already existing program from the Decwriter keyboard. It does not matter whether the text strings entered are a FORTRAN program, an assembly language program, or a letter to the editor, anything you wish to type is permitted.
FORTRN	Is the program which takes a FORTRAN source program (normally from the disk) and compiles it, producing an object module (the binary translation of the source) suitable for linking with other object modules by the LINK program.
LINK	Is the program which takes a collection of object modules (main program, sub-routines, functions, etc.) performs the necessary internal linkages and produces a load module suitable for running under the Monitor.
PIP	Is the acronym for Peripheral Inter-change Program. Its function is to allow the transfer of a file from one peripheral device to another. It may be used to get a listing of a source program on the keyboard or line printer, to transfer a file from disk to magtape or vice versa, etc.

B. A Guide to the Use of EDIT

A sample editing session, including commentary, is included in Appendix B. It may be helpful to refer to this as the reader progresses through this document. All things typed by the system have been underlined, anything typed by the user has not.

1. General Operating Procedures

a. Running Edit

In order to run EDIT type RUN EDIT(CR) after the Monitor types a \$.

b. Entering a New File

After the EDITOR has been loaded and types a # enter the name and extension desired for your file. For example, if it was desired to begin a file named JUNK.FTN one types

JUNK.FTN(CR)

The EDITOR will now echo a * on the keyboard to

indicate it is awaiting further instructions. Remember that it is very strongly advised that all FORTRAN source programs be given the extension .FTN as this will make life much easier later on.

The first command given to the EDITOR (after it types a *) when entering a new program is always an 'I' command. This command puts the EDITOR in the text input mode.

c. Modifying an Already Existing File

After the EDITOR has been loaded and types a # , enter on the keyboard the following instruction sequence.

FILE1.EXT<FILE1.EXT/B(CR)

where FILE1.EXT is the filename and extension of the file you wish to modify. NOTE: The symbol < is the "less than" sign.

In this case the first command given to the EDITOR after it types out an * , is the R command. This reads the specified file into core. This procedure will result in the following message being typed.

W311 000000 ('W' indicates a warning message)

This is no cause for concern and simply indicates that the end of the file has been reached.

Upon the proper termination of the Editing session, the procedure given above will result in the updating of the file FILE1.EXT, as modified during that session. If somehow you should manage to crash the system during the course of editing, your original file should still be on the disk.

d. Ending an Editing Session

The proper procedure for terminating an Editing session is to return to the edit command mode (indicated by the system's typing an *), and then to type the command EX(CR). This procedure will ensure that the file you have been editing is properly transferred to disk. Upon the successful completion of this transfer the system will type out an # indicating that it is ready to accept another filename or that you may return to the Monitor by typing a CTRL/C followed by a KILL. Attempting to KILL the EDITOR before it echoes the # , or without typing EX in the command mode, is likely to be disastrous.

2. Description of the Way EDIT Works

As described above, EDIT allows one to input and modify text strings in a given file from the teletype keyboard. Under normal circumstances the entire file being worked on is stored in memory at once (for extremely long files, of greater than 12000 symbols, this may not be possible). EDIT may operate in either of two modes: the command mode, and the text input mode. Once placed in a given mode, EDIT will remain in that mode until a specific operation is performed to switch to the other mode. The specific text strings to be operated upon are determined with reference to an internal pointer (hereafter referred to as 'DOT'). Some of the commands are used to move 'DOT' while others are used to modify the text given the current position of 'DOT', or to list certain text lines, or to change modes.

The EDITOR may operate in a manner oriented either towards entire lines of text or towards specific character strings, interchangeably. (A line is a text string between two successive (CR)'s and including the second.) The latter commands are more powerful, whereas the former are simpler to use. With a single exception, the commands discussed here will be the line-oriented commands. All editing may be performed using only these commands, although it may require more typing. If the user wishes further information on the character-oriented commands he is referred to the EDITOR manual.

To change from the command mode (indicated by a * at the beginning of the line) to the text input mode, the user types I(CR). There may not be any intervening characters or spaces between the 'I' and the (CR).

To change from text input mode to command mode one types an (LF) immediately following the last (CR) in the desired input text. Again no characters or spaces may be typed between the (CR) and the (LF).

3. EDITOR Commands

NOTE: All the commands listed below are to be immediately followed by a (CR) with no intervening characters or spaces.

COMMAND	FUNCTION AND OPERATION
I (insert)	Changes from the command mode to the text input mode. This is the first character typed when entering a program for the first time. When this command is issued to insert lines into already existing text, the text so inserted is all positioned before the line containing 'DOT'. To return from this mode to the command mode type an (LF) immediately following the (CR) of the last text line. Used as described above this command has no effect on the position of 'DOT'.
R (read in)	Reads the previously specified file into memory from disk. This is the first command typed when editing an already existing file.
B (go to beginning)	Positions 'DOT' at the beginning of the first line of the current file. (Beginning of the line means before the first character.)
K (kill)	This command deletes the entire line beginning at 'DOT' up to and including the next (CR). Note that deleting a line does not leave a line with nothing in it, but rather it 'closes up' the space where the line was. After this command 'DOT' is now at the beginning of the line immediately following the line just deleted.
V (verify)	Prints out the line containing DOT. It does not affect the position of 'DOT'.
L (list)	Prints out the current line beginning at 'DOT' up to and including the next (CR). The position of DOT is not affected.
NL	Where N is an unsigned integer specifying the number of lines to be printed out beginning at 'DOT'. The position of 'DOT' is not affected. Note that the L must immediately follow the integer without any intervening spaces.
A (advance)	Advances the position of 'DOT' to the beginning of the next line. It has no affect on the text.
-A	Positions 'DOT' at the beginning of the line immediately preceding the current line (it has no affect if you are at the beginning of the file).
NA	Where N is a signed integer specifying the number of lines 'DOT' is to be moved forward or back. Specifying a negative number of lines, which would take 'DOT' to before the beginning of the first line, simply positions 'DOT' at the beginning of the first line. Specifying a positive number of lines which would position 'DOT' further down the text than at the beginning of an imaginary line following the last line, merely positions DOT at the beginning of that imaginary 'last line'.
NG;text string;ØA (Ø=zero) (G is for get)	Where N is an unsigned integer, locates in the file beginning at 'DOT', the nth occurrence of the character sequence 'text string', and positions 'DOT' at

<u>COMMAND</u>	<u>FUNCTION AND OPERATION</u>
	the beginning of the line containing that occurrence. In order to locate just the next occurrence of the desired text string one may omit the integer and merely type G;text..., etc.

It should be pointed out that this command only searches for the desired text in the forward direction. If the end of the file is reached before encountering the specified number of occurrences of the desired text string, an error message (W307 000000) will be issued, and 'DOT' will be positioned at the beginning of the imaginary 'last line'.

By way of comment, this instruction is a combined command which makes use of both the line-oriented and character-oriented features of the EDITOR and should enable the user to select any desired line from the text.

EX (exit) This command takes care of transferring the edited file from memory back out onto disk. This should always be the last command for editing any file. Upon successful completion of this command a # will be typed on the teleprinter. Only then should the user enter a new file name or KILL the EDITOR and return to the Monitor.

4. Special Feature for FORTRAN Programs

The EDITOR has a special feature which allows the justification of FORTRAN statements so that they appear to the compiler to begin in column 7. (The fact that nine spaces are spaced on the teletype should be of no concern.) This justification is accomplished by typing the TAB key at the beginning of the line. For purposes of counting characters it should be noted that the TAB is treated by the EDITOR as a single character. In addition, if after typing a statement number, one strikes the TAB key, one may now begin typing the FORTRAN statement without worrying about skipping the appropriate number of spaces. Counting up the spaces yourself will also work but this procedure is much simpler.

This feature has one additional aspect, and that is concerning continuation lines. If after typing TAB the first character is a number, that line will be treated as a continuation line. It is not necessary to worry about placing a character in column 6.

C. A Guide to the Use of FORTRN

A sample use of the FORTRAN compiler 'FORTRN' is included in the Appendix C. All things

typed by the system have been underlined. All things typed by the user have not.

1. General Operating Procedures

a. Running FORTRN

In order to run FORTRN simply type RUN FORTRN(CR) after the Monitor types a \$. First be sure that you are appropriately logged in, otherwise your file is not likely to be found.

b. Compiling a FORTRAN Program

In order to compile a FORTRAN program which is contained in a file on the disk with the filename FILE.FTN, enter the following command sequence after FORTRN has been loaded and has typed a # .

FILE,KB:<FILE/ON(CR)

If no errors were detected in your program by the FORTRAN compiler, this procedure will result in the compiled object module being placed on the disk. It will have the filename FILE.OBJ. Any previous file with that name will be replaced by this new file automatically. The source version of your file will be listed on the teletype with any error lines being flagged. The above procedure should take care of locating most FORTRAN errors. Unfortunately, there is one class of errors for which the diagnostics are very poor, that is concerning undefined variables or statement numbers. Any errors of the above type are reported by the above procedure only as

U NNNNN 000000

where NNNNN is some integer seemingly unrelated to the missing variable or statement number. The only way to locate these errors is to go back to your program listing and look for them. Sometimes correcting other errors in the program will eliminate the undefined variable error. See the example in the Appendix for an illustration of this effect.

c. Exiting from FORTRN

The only precaution to be observed in KILLing FORTRN is to be sure that it has typed back a # following any command sequence you have given it. Failure to follow this procedure may corrupt your file on the disk.

2. General Information about FORTRAN for the PDP-11

Since the PDP-11 has a 16-bit word size, the range of integers it may contain is severely

restricted in comparison to some machines you may be accustomed to working with. The largest positive integer is 32767 and the largest negative integer is -32768. This implies that integer arithmetic rapidly overflows the bounds allowed. There is no double precision integer capability in the PDP-11 FORTRAN. Each integer on the PDP-11 requires one word of storage space (if /ON was specified when compiling the FORTRAN program). Each single precision floating point number requires two words for storage. Each double precision floating point number requires four words for storage. The only thing to be aware of here is that large floating point and double precision arrays eat up core very quickly (most machines at LAMPF have been 16- and 24-K of memory). The precision involved in single precision floating point arithmetic is approximately 7 decimal digits, for double precision it is about 17 decimal digits.

Since the PDP-11 is a 'hands on' machine, one need not worry about providing for an exit from your program. Stop commands are unnecessary. To terminate a program one merely types CTRL/C and KILL.

Regarding READ and WRITE statements, it should be noted that the standard device number for the teletype is 6. Thus to read a number IVALUE from the teletype, with FORMAT statement number 1000, the appropriate FORTRAN statement is

```
READ(6,1000)IVALUE
```

To write IVALUE on the teletype the appropriate statement is

```
WRITE(6,1000)IVALUE
```

D. A Guide to the Use of LINK

NOTE: A sample use of LINK is included in the Appendix D. All things typed by the system have been underlined, all things typed by the user have not.

1. General Operating Procedures and Information

a. Running LINK

In order to run LINK type RUN LINK(CR) after the Monitor types a \$.

b. Linking a FORTRAN Program

In order to LINK a FORTRAN program, first be sure that you are correctly logged in, then

RUN LINK. The object modules for the main program and all its associated user subroutines should be under your UIC. Suppose it was desired to LINK a main program whose filename was JUNK which required three user-supplied subroutines with filenames SUB1, SUB2, and SUB3. It is also desired that the final load module have the filename JUNK and the extension .LDA. The appropriate command sequence for this is given below.

```
JUNK<JUNK,SUB1,SUB2,SUB3,FTNLIB/L/U/E(CR)
```

Although a full explanation of the meaning of this command sequence will be delayed until the discussion of the CSI, a few comments will be made here. FTNLIB is a library of FORTRAN subroutines and functions (SIN, COS, EXP., etc.) along with a large number of subroutines which are called in the object code output by the FORTRAN compiler. For any FORTRAN program it is absolutely necessary to LINK this library. the /L command is used to indicate that this file is a library file (containing many separate object modules). The /U command will cause any undefined references to be listed. The /E command is needed to indicate the end of the list of files to be included in the linking procedure. The library file should always be the last file in the sequence of files listed to ensure that all necessary subroutines called by your program and subroutines are appropriately linked.

If your list of subroutines is so lengthy that it may not all fit on one line, simply type as many filenames as conveniently fit on the first line, completing the last filename fully (do not follow last name on first line by a comma). Then type a (CR). The LINK program will now search the disk for the files you have entered so far. Once it has located them all, it will type another # . At this point you may continue typing the desired filenames using as many lines as needed, proceeding for each line as you did for the first. After typing the last filename (which should be the FORTRAN library) type /E(CR) to indicate the end of the list.

IMPORTANT NOTE: The linking procedure with its subsequent writing on disk can take quite some time. With LINK, as with the other systems programs, be sure not to issue a KILL before the program has echoed a # indicating it is finished doing what it started out to do. Failure to observe this

precaution may not only corrupt your file, but also render large segments of the disk inaccessible to any user without special corrective measures.

c. Comment on Filenames and Subroutine Names

You will note that the names typed in the command string for the LINK program are names of files (each having the extension .OBJ) in the case where these files are object modules of user subroutines. There is not a necessary correlation between the name of the file and the name of the subroutine. For example, one might have a FORTRAN subroutine with the name

```
SUBROUTINE SCREW(IA,IB,IC)
```

and yet have the file in which the source version of that subroutine is stored have the name and extension SCREW1.FTN or SUBROT.FTN, etc.

Recall then, that you must enter the name of the file and not the name of the subroutine if the two are different (it is probably good procedure to make the filename correspond to the subroutine name).

d. What to do about a W322 or an F012

Two of the most common error messages encountered while linking programs are W322 and F012. The former indicates that a GLOBAL symbol has remained undefined after searching through all the files listed. In plain English, it means you forgot to list some file which contained a subroutine you called. An F012 error indicates that one of the files you listed could not be found, as typed, in your Directory of files.

If the /U command was used, the undefined references which caused the W322 error will be listed. To discover which file was the cause of the F012 you must use PIP to list out your Directory and make sure each file is present with the extension .OBJ.

E. A Guide to the Use of PIP

NOTE: A sample of the various uses of PIP discussed below is contained in the Appendix E. All things typed by the system have been underlined, all things typed by the user have not.

1. General Operating Procedures and Information

a. Running PIP

In order to run PIP type RUN PIP(CR) after the Monitor types a \$.

b. Uses of PIP

The primary use of PIP is that of transferring files from one peripheral device such as the disk to another peripheral device such as the teletype. Additional things that may be done with PIP are entering one's UIC in the Master File Directory, deleting files from the disk, and getting a directory of one's files. Again, a full understanding of the way to use PIP depends on understanding the syntax of the Command String Interpreter, to be discussed later.

However, some of the more commonly used commands will be given below without discussion of the detailed significance of what is typed in.

c. Using PIP

It is assumed for all of the following procedures that you have first logged in under your own UIC before running PIP. It is also assumed that you will wait for PIP to type out a # before entering any of the command strings listed and that you will not KILL PIP before it has completed its assigned task (as is indicated by it typing another #).

Entering your UIC in the Master File Directory (MFD), type the command sequence

```
/EN(CR)
```

Recall that this procedure is necessary only once for each computer that you make use of. It is included here only for the sake of completeness.

Getting a listing of the files in one's directory. To get what is known as a 'brief' listing enter the following

```
/BR(CR)
```

PIP will now print out on the teletype a listing of all one's files including the extension.

To get a complete listing enter the following command string

```
/DI(CR)
```

PIP will now print out on the teletype a listing of all one's files including the extension as before, but in addition, it will list for each file its creation date and something called the file protection code.

Deleting a source file from your Directory. To delete a file with the filename and extension JUNK.FTN from your Directory, enter the following command sequence

JUNK.FTN/DE(CR)

To list on the teletype a source (ASCII) file contained on disk, enter the following command sequence to list the file JUNK.FTN

KB:<JUNK.FTN(CR)

III. COMMAND STRING INTERPRETER

A. General Format

The Command String Interpreter (CSI) is a Monitor routine which allows information about a file such as its location (disk, magtape, etc.), filename, extension, UIC, and perhaps some additional information about the task to be performed, contained in something called "switches," to be entered when needed by a system program. Whenever a system program requests input through the CSI a # will be printed on the teleprinter and the program will wait for the operator's reply.

In its most general form a CSI command will consist of one or more output file specifications (output dataset specifications in the terminology of the manuals), followed by a < followed by one or more input file specifications. The command string is terminated when the user types a CR. Commands may not be continued from line to line. If more than one input file, or output file specification is used, then those file specifications must be separated from each other by commas. Thus a command string which involved two output files and three input files would have a form such as is given below

OUTPUT SPEC1,OUTPUT SPEC2<INPUT SPEC1,INPUT SPEC2,INPUT SPEC3(CR)

where OUTPUT SPEC is used to symbolize the desired output file specification and INPUT SPEC is used to symbolize the desired input file specification.

Most systems programs allow for the specification of multiple input and/or output files. For most of these programs the order in which the file specifications are given is important, since the program attaches a particular significance to the use of a given "positional field." For example, the system program FORTRN assumes that the first

"output positional field" will be used to specify information about the output object module, and the second "output positional field" will be used to specify information about the device for listing the program (if any), etc. The details of this will be discussed in the examples to be given later. The point of importance here is that if you wish to omit any of the positional fields (except those at the end) you must still indicate the order by typing a comma for the field omitted.

For example, if a particular system program allowed for three output file specifications and one input file specification, and it was desired only to make use of the first and third output files then the command string would look as follows in response to the # .

OUTPUT SPEC1,,OUTPUT SPEC3<INPUT SPEC(CR)

If it was desired only to make use of the third output file and neither of the first two, then the command string would be

,,OUTPUT SPEC3<INPUT SPEC(CR)

In the first example only the first and third output files would be generated. In the second example only the third output file would be generated.

A final comment regarding the specification of input and output fields is that the insertion of commas for missing fields, as mentioned above, is necessary only when another file specification is yet to be made on that side of the < . Hence, continuing with the examples in the format given above, if only the first of the three possible output files was desired, a perfectly satisfactory command sequence would be as follows

OUTPUT SPEC1<INPUT SPEC(CR)

Here, no commas are needed to indicate that the second and third files are not desired. Under certain circumstances trailing commas may be fatal.

B. File Specifications

As mentioned above, a file specification may consist of where to find the file, the filename and extension, the user identification code, and perhaps one or more switches which request various actions to be performed. Not all of this information need

appear in every file specification but it is all potentially enterable for every file specification. In those cases where certain of the information is omitted, various default assumptions are invoked by the system. Exactly what these assumptions are will be discussed shortly. For the moment we consider a complete file specification containing all of the elements mentioned above. In schematic form it would appear as follows.

DEV:FILENAME.EXT[UIC]/S1:V1:V2...:VN/S2:V1:V2....

where

DEV Is the device specification consisting of two letters and sometimes an octal digit followed by a colon. The letters identify the device and the digit identifies the device unit number on devices which can have more than one unit, such as magtape or dectape. This is a physical unit number as set by a thumb-wheel selector on the device and should not be confused with the logical unit number as specified in FORTRAN READ and WRITE statements. A list of allowable two-letter device names is given below along with the FORTRAN LOGICAL UNIT NUMBER.

DEVICE	CODE	LOGICAL UNIT NUMBER
RF disk	DF	1,2,3,7, and 8 (input and output)
RK disk	DK	None
dectape	DT	None (input and output)
paper reader	FR	4 (input only)
line printer	LP	5 (output only)
decwriter	KB	6 (input and output)
paper punch	PP	None
magtape	MT	None
card reader	CR	None

FILENAME The filename consists of up to six letters and digits with the first character being a letter.

.EXT The extension specification consists of a period followed by up to three letters or digits.

[UIC] The User Identification Code [UIC] specification consists of a left square bracket [followed by one or more octal digits, followed by a right square

bracket]. As usual, both numbers specified must be in octal with the largest valid number being 376 (0 is invalid).

/S

A switch specification consists of a slash (/) followed by one or two letters or digits followed by one or more value specifications. A value specification consists of a colon sometimes followed by one or more letters or digits. If more than one value specification is used for any given switch, it may simply be strung after the preceding one as in the example. If more than one switch specification is used it may be entered immediately after the input string for the previous switch. Thus in the above example of a complete file specification there are two switches entered. The first is S1 and has value specifications V1, V2, ... VN. While the second switch is S2 and has value specifications V1, V2, ..., VN. Note that each switch is preceded by a slash (/).

It is important to recognize that the order in which the various elements of the file specification are entered is critical. Items may be omitted from the specification, but all items actually entered must be in the order shown.

An example of a complete file specification such as might be given for the input file in FORTRAN would be

DF:JUNK.FTN[7,11]/ON

where the device is an RF disk, the filename and extension is JUNK.FTN, the UIC is 7,11 and the switch /ON has been specified indicating that integers are to be allocated one word of storage space.

C. Switches

As mentioned above, the various systems programs use something called "switches" to supply additional information about the task to be performed as given in the command string. The switch specification consists of a slash (/) followed by one or two letters followed by one or more value specifications. Each value specification consists of a colon (:) followed by one or more letters or digits, or by nothing. Multiple value specifications for a given switch may sometimes be appropriate and are simply strung out in sequence (each containing a colon). Some switches are applicable only to input files or to output files but not to both. Some switches refer to an individual file specification, while others may apply to the entire collection of file

specifications on that side of the < . Given below is a list of the more commonly used switches grouped by systems program. This is by no means an exhaustive list. If the user wishes more information concerning all the switches applicable to a given systems program, he is referred to the appropriate manual. NOTE: Some switches allow for no value specifications.

1. EDIT

/B This is the only switch applicable to 'EDIT'. If it is specified it must be on the input side of the < (right-hand side of the <) after the filename extension. Its function is to prevent the creation of a backup file on the disk (file has same name but extension .BAK).

Example: JUNK.FTN<JUNK.FTN/B(CR)

2. FORTRAN

/ON This switch is legal only on the input side of the < . Its function is to allocate one 16-bit PDP-11 storage word for each integer in the FORTRAN program instead of the two words (one unused) which would normally be allotted.

/SU This switch is legal only on the input side of the < . It does not admit of any values. Its function is to suppress the generation of the sequence numbers for each FORTRAN statement and the assembly language code associated with the sequencing. The function of the sequence numbers is to allow easy location of the statement which caused a run time error in the program. Specification of this switch is advisable only if core space is very valuable or time is at a premium. Specification of this switch shortens execution time for a FORTRAN program 5-10%.

/LI:N This switch is legal only in the second file specification on the output side of the < . N is an integer value from 0 - 3, inclusive. Its function is to specify the type of listing desired for the FORTRAN program by the FORTRAN compiler.

N = 0 List only lines containing errors and the size of the program and any common arrays and their sizes.

N = 1 List the entire source program including errors. Also lists size of program and common arrays. This is a default value if a listing device is specified but the /LI

switch is not specified. Default value if no value for N is given.

N = 2 List source program including errors. Then list the assembly language expansion of the source code. Also lists program size and common arrays.

N = 3 Does same as N = 2 but also lists symbol table for all symbols.

NOTE: Listing on the teleprinter for values of N greater than N = 1 can be very time consuming. It is also of little use unless you are familiar with assembly language programming.

EXAMPLE: To compile a FORTRAN program from disk file JUNK.FTN to create disk object module JUNK.OBJ, with a listing on the line printer, specifying one-word integers and suppressing the sequencing. The following is the appropriate command sequence.

oJUNK,LP:/LI:1<JUNK/ON/SU(CR)
JUNK,LP:<JUNK/ON/SU(CR)
would give the same output on the line printer due to the default assumption concerning the /LI switch.

3. LINK

/L This switch is only legal on the input side of the < . It does not admit of any values. Its function is to indicate that the file specified is a library file (contains many object modules in sequence). It must be specified for the FORTRAN library FTNL1B and for any user defined libraries which are referenced.

/U This switch is only legal on the input side of the < . It does not admit of any values. Its function is to cause a listing to be printed out of any symbols which are undefined as of the time it is encountered. It is helpful in diagnosing the cause of a W322 error, should such be encountered. It is normally the last switch to be specified (except for the /E switch to be discussed next) so that it will list only those symbols which have no possibility of yet being defined.

/E This switch is legal only on the input side of the < . It does not admit of any values. It must be the last switch specified in the command sequence and should be

immediately followed by a CR. Its function is to indicate to the LINK program that the end of the command sequence has been encountered.

Example: To link a FORTRAN object module on disk named JUNK.OBJ and produce a load module JUNK.LDA on disk with a listing of the load may on the line printer and specifying that all undefined references are to be listed. The following is the appropriate command sequence.

JUNK,LP:<JUNK,FTNLIB/L/U/E(CR)

4. PIP

/BR

This switch does not admit of any values. It causes to be listed all the filenames and extensions in the directory of the currently logged in user on the specified device (from the disk if no device specified).

Example: To get a brief listing of all files in your directory

/BR(CR)

To find out if a particular file JUNK.FTN is in your directory

JUNK.FTN/BR(CR)

/DE

This switch does not admit of any values. It causes all the previously specified files to be deleted from the designated (or default) device.

NOTE: It is only possible to delete files from a device which has a directory. There are only two such devices on the PDP-11 namely the disk and dectape. The magtape does not have a directory and hence individual files may not be deleted from it. The only way to delete a file from magtape is to zero the whole tape using the /ZE command.

Example: To delete the files JUNK.FTN and JUNK.OBJ from the disk after you are logged in

JUNK.FTN,JUNK.OBJ/DE(CR)

/DI

This switch does not admit of any values. It causes to be listed all filenames, extensions, creation dates, and protection codes in the directory of the currently logged in user from the specified (or default) device.

Example: To list your complete directory

/DI(CR)

To list the system directory while logged in under your own UIC

[1,1]/DI(CR)

To list the directory of user 7,11 on dectape unit number 3

DT3:[7,11]/DI(CR)

/EN

Legal only if no < has been specified. It does not admit of any values. Enters UIC of current user into Master File Directory (MFD).

Example: To enter your UIC in the Master File Directory on the disk type

/EN(CR)

To enter your UIC in the Master File Directory for dectape unit number 5 type

DT5:/EN(CR)

/FR

Legal only when no < has been specified and when no filename has been given. The only argument which may precede it is a device specification (disk or dectape only). It lists on the teleprinter the remaining number of free blocks.

Example: To find out the number of free blocks on the disk type

DF0:/FR(CR) or simply /FR(CR)

To find out the number of free blocks on dectape on unit number 3 type

DT3:/FR(CR)

/UN

Legal only when no < has been specified. It admits of no values. Its function is to unlock the specified file after a system crash so that it may be accessed by other systems programs.

Example: To unlock file JUNK.FTN on the disk type

DF0:JUNK.FTN/UN(CR) or simply JUNK.FTN/UN(CR)

/ZE

Legal only when no < has been specified. It admits of no values. It is valid only when simply a device has been specified before it. The only valid devices are the magtape and the dectape. Its function is to erase all files from the directory of the device (for dectape) or reinitialize the tape (for magtape) to indicate that no files exists.

EXTREMELY IMPORTANT NOTE:

Never issue this switch without specifying either the dectape or magtape as the device. Never give the disk as the device unless you are certain you wish to zero the disk.

Example: MT0:ZE(CR)

Will reinitialize the magtape and prevent any further access to files already on the magtape. It effectively erases the tape.

D. Default Assumptions

As mentioned above, it is permissible to enter only some of the elements which go into making up the complete file specification when using the CSI. Indeed, in all the examples shown in Sections I and II, this was done with regularity. When a particular element is omitted from the file specification the system invokes certain default assumptions as to the nature of the omitted information. These assumptions are of two types. First, those common to all systems programs, and second those peculiar to an individual systems program. We shall discuss each type in sequence. Examples of complete file specifications and the reductions allowable using the default assumptions will be given for each systems program discussed.

1. Universal Assumptions

The most commonly omitted specification is that for the device. If the device for a given file has not been specified, the device is taken to be the last device specified for any file on the same side of the <. If no device has yet been specified on that side of the <, then the disk is the assumed device (disk unit 0 if more than one exists). Note that the default assumption does not carry across the <, i.e., from the output file specifications to the input file specifications. If no digit appears, unit 0 is assumed. This is normally of no consequence except when dealing with magtapes or dectapes.

The other commonly omitted specification is the User Identification Code [UIC]. If this specification is omitted, it is assumed to be that of the currently logged in user. If a UIC is entered in a particular file specification it does not carry over into other file specifications which may follow it, as does the device specification.

2. Assumptions Specific to a Particular System Program

These assumptions all have to do with the extension for a given filename. The assumptions will be listed separately for each program which makes them. If the program is not listed, it does not make any assumptions about the extension. Hence for these system programs the extension must always be specified along with the Filename.

a. EDIT

EDIT makes no assumptions as to the extensions of the files involved. One must specify the input filename including the extension. However, the assumption as to device (i.e., the disk) and UIC (current user) are valid.

Example: To get an output file of the same name as the input file JUNK.FTN both being on disk (UIC 7,11) with no backup file being generated. The command sequence

DF0:JUNK.FTN[7,11]<DF0:JUNK.FTN[7,11]/B(CR)
might be shortened to
JUNK.FTN<JUNK.FTN/B(CR)

b. FORTRAN

The default assumption for the input file extension is .FTN. The default assumption for the first output file extension (that of the object module) is .OBJ. The default assumption for the second output file extension (that of the listing) is .LST. Note that the filename and extension are only relevant when the list file is being output to a file structured device such as disk, magtape, or dectape. It is not necessary to specify a filename and extension for a nonfile structured device such as the keyboard.

Example: To compile a FORTRAN program found under UIC[7,11] whose name is JUNK.FTN and yielding an output file (object module) with name JUNK.OBJ and using the line printer as a listing device. The following is a complete command sequence in response to the #.

DF0:JUNK.OBJ[7,11],LP:/L1:1<DF0:JUNK.FTN[7,11]/ON(CR)
Using the default assumptions this may be reduced to
JUNK,LP:<JUNK/ON(CR)

c. LINK

The default assumption for all input file extensions is .OBJ. The default assumption

for the first output file extension (that of the linked load module) is .LDA. The default assumption for the second output file extension (that of the load map) is .MAP. The default assumption for the third output file extension (that of the GLOBL symbol table) is .STB. This third output file is only specified when using the overlay facility for FORTRAN programs. It is a binary file and hence is not suitable for being output on the line printer or keyboard. It is normally kept on the disk to be referenced later on in linking other overlays.

To link a FORTRAN program while logged in under UIC[7,11] with object module name JUNK.OBJ and produce a load module on disk named JUNK.LDA with a listing on the line printer the following is the complete command sequence.

```
DF0:JUNK.LDA[7,11],LP:<DF0:JUNK.OBJ[7,11],DF0:FTNLIB.OBJ[1,1]/L/U/E(CR)
```

Using the default assumptions this may be reduced to

```
JUNK,LP:<JUNK,FTNLIB/L/U/E(CR)
```

which is a good deal less trouble to type.

d. PIP

PIP makes no assumptions, directly, concerning the extensions involved in its actions. It does, however, assume that in a transfer operation from one device to another, if no output filenames are specified, that the output filenames and extensions are the same as those for the given input files.

Example: To transfer the three files JUNK.FTN, JUNK.OBJ, and JUNK.LDA under UIC[7,11] from the disk to magtape unit 1, a nearly complete command sequence might be

```
MT1:JUNK.FTN,JUNK.OBJ,JUNK.LDA<DF0:JUNK.FTN,JUNK.OBJ,JUNK.LDA(CR)
```

where use has already been made of the default assumption concerning the UIC (currently logged in user).

This might be shortened by using the above default assumption to

```
MT1:<JUNK.FTN,JUNK.OBJ,JUNK.LDA(CR)
```

Using the * convention to be described next, this sequence could be shortened even further.

3. The * Convention

The asterisk (*) convention when used in a command string provides the user of PIP (and only PIP) with a convenient means of transferring large numbers of related files. It is used to symbolize all of a particular class of files as specified by the remainder of the command string.

Example:

JUNK.*	Refers to all files with filename JUNK whatever the extension. WARNING: DO NOT USE FOR AN OUTPUT SPECIFICATION.
*.FTN	Refers to all files with the extension .FTN regardless of the filename.
,	Refers to all files regardless of filename or extension.

[*,11]	Refers to all users with second number of UIC being 11, no matter what the first number.
--------	--

[*,*]	Refers to all users.
-------	----------------------

The asterisk can only be used in place of filenames, extensions, and UIC's and not for device names or switches. And recall that it is only valid in PIP and not in any other systems program.

Examples of Usage:

To transfer all your files with extension .FTN to magtape unit 3 type

```
MT3:*.FTN(CR)
```

To list a complete directory of all your files with extension .OBJ which are on decatape unit 5 type

```
DT5:*.OBJ/DI (CR)
```

To obtain a complete directory of all users and all files on a magtape mounted on drive number 2 you type

```
MT2:[*,*]/DI (CR)
```

A note of caution to the user.

This is a very powerful convention but it has its dangers. Be aware of the drastically different results of typing

```
*,*/DI (CR) which produces a directory
```

and

*./DE(CR) which deletes all files!!!!!!
verbum sapientis satis.

IV. FORTRAN I/O

Thus far all references to input and output in FORTRAN have been to the keyboard, assigned logical unit number 6. It is of course possible to do input and output to other devices using FORTRAN. Suppose one issued a WRITE command for logical unit 3 (the RF disk) what would happen?

First an output file would be generated on the disk containing the data in the format specified by the user. The filename and extension would be

FUR003.DAT

and it would be under the directory of the currently logged in user. This file could later be read under a FORTRAN READ command with unit 3 specified as the logical device. All this would happen with no intervention from the user. If another of the logical device numbers assigned to the disk (2, 7, or 8) were used then the name and extension of the output file generated would be

FUR00N.DAT (where N is the logical unit number).

If one specified unit 5 for output for example, that output would go to the line printer (without a filename of course).

All this is quite easy, but it is somewhat restrictive in that it allows only one filename to be used for a given logical unit number. Suppose one wished to enter one's data through the EDITOR into a series of files on the disk. These sets of data are all meant to run in the same program but you may wish to have a different set of data read in depending on exactly what you wish the program to do. Since FORTRAN READ from the disk looks only for a single filename for a particular logical unit, you would have some problem distinguishing between the sets of data using the method described above. Fortunately, you may easily specify which set of data you desire by using the ASSIGN command in the Monitor. While in the Monitor, before running your program, you enter the following command sequence in response to the \$.

AS(SIGN) DEV: FILENM.EXT[UIC],N

where DEV:FILENAM.EXT[UIC] is the standard file specification as given before. The UIC is optional

if the data file is under the UIC of the currently logged in user. In this case, however, there is no default assumption regarding the device, it must be specified. The same is true of the extension. If the device is not file structured (i.e., is not disk, dectape, or magtape) the filename, if one is specified, is ignored. The file specification is followed by a comma, which is in turn followed by the integer (N) which is the logical unit number which will be assigned to this file for all READ and WRITE statements. This assignment will apply until changed by the user, even if you run another program, until the Monitor has been re-bootstrapped into core. You may return to the default assignments as listed earlier by simply typing

AS(CR)

(that is ASSIGN with no arguments) while in the Monitor.

Example: To read a file named JNKDAT.DAT from the disk under your UIC with a logical unit number of 5 you would enter the following.

AS DF:JNKDAT.DAT,5(CR)

You will note that this use of the ASSIGN statement overrides any of the assumptions as to the logical unit number-device correspondence given earlier.

This procedure may also be accomplished in your program directly using the "SETFIL" routine. For more information concerning this routine the user is referred to the FORTRAN manuals.

APPENDIX A

Printout obtained in bootstrapping the system, entering date and time and using PIP to enter your UIC into the Master File Directory.

JUNE 28, 1973
BATCH V08-02

\$LO 35.37
DATE:-01-JAN-72
TIME:-00:00:25
\$DA 5-JUL-73
\$TIME 20:45
\$DA
05-JUL-73
\$TIME
20:45:10
\$RUN PIP
PIP-11 V7A01
\$/EN

\$^C
..KI

\$

APPENDIX B

Sample use of EDIT to enter a program for the first time.

\$RUN EDIT
EDIT-11 V006A

#DEMONS. FTM

*I

C THIS IS A DEMONSTRATION PROGRAM
C WHICH WILL BE USED TO ILLUSTRATE
C VARIOUS ASPECTS OF THE EDITOR, THE
C FORTRAN COMPOILER, THE LINK
C PROGRAM AND ALSO OF PIP.

C THIS FIRST VERSION CONTAINS
C SOME INTENTIONAL ERRORS.

C
1 IA=10
IB=7
READ(6,1000)IC
1000 FORNTH(I7)
ID=((IA*IB)+C
WRITE(6,1000)ID
GOT TO 1
END

entering program name for first time
go to input mode
user enters text beginning here

*B
*V
C THIS IS A DEMONSTRATION PROGRAM
*AV
C WHICH WILL BE USED TO ILLUSTRATE
*G: *C: 0AV
ID=((IA*IB)+C
*I
ID=((IA*IB)+IC

(L) typed to return to command mode
move DOT to beginning of file
list line containing DOT

<u>*V</u>	<u>ID=((IA*IB)+C</u>	locate character string "+C"
<u>*K</u>		insert new line
<u>*2A</u>		
<u>*V</u>	<u>END</u>	
<u>*-A</u>		
<u>*V</u>	<u>GOT TO 1</u>	
<u>*I</u>	<u>GO TO 1</u>	
<u>*V</u>	<u>GOT TO 1</u>	verify old line
<u>*K</u>		kill old line
<u>*-4A</u>		
<u>*4L</u>		
<u>1000</u>	<u>FORMAT(17)</u>	
	<u>ID=((IA*IB)+IC</u>	
	<u>WRITE(6,1000)ID</u>	
	<u>GO TO 1</u>	
<u>*EX</u>		exit
<u>*^C</u>		
<u>*KI</u>		kill the EDITOR
<u>\$</u>		

Sample use of EDIT to modify an already existing program to correct errors.

\$RUN EDIT
EDIT-11 V006A

<u>\$DEMONS.FIN<DEMONS.FIN/B</u>	enter filename, inhibit backup file
<u>*R</u>	read in file
<u>M311 000000</u>	message indicating end of file reached
<u>*G;MTA;0AV</u>	locate line containing "MTA"
<u>1000</u>	<u>FORMAT(17)</u>
<u>*I</u>	
<u>1000</u>	<u>FORMAT(17)</u>
<u>*V</u>	
<u>1000</u>	<u>FORMAT(17)</u>
<u>*K</u>	kill incorrect line
<u>*G;((;0AV</u>	
	<u>ID=((IA*IB)+IC</u>
<u>*I</u>	
	<u>ID=(IA*IB)+IC</u>
<u>*V</u>	
	<u>ID=((IA*IB)+IC</u>
<u>*K</u>	
<u>*EX</u>	
<u>*^C</u>	
<u>*KI</u>	
<u>\$</u>	

APPENDIX C

Sample use of FORTRAN to compile a program while giving a listing on the teletype. The error in the FORMAT statement causes statement 1000 to be undefined which results in the two U errors.

```
$RUN FORTRN
FORTRAN Y004A
#DEMONS,KB:<DEMONS/ON
```

```
FORTRAN Y004A                20:54:32    05-JUL-73    PAGE    1
```

```
      C      THIS IS A DEMONSTRATION PROGRAM
      C      WHICH WILL BE USED TO ILLUSTRATE
      C      VARIOUS ASPECTS OF THE EDITOR. THE
      C      FORTRAN COMPILER, THE LINK
      C      PROGRAM AND ALSO OF PIP.
```

```
      C      THIS FIRST VERSION CONTAINS
      C      SOME INTENTIONAL ERRORS.
```

```
0001      I      IA=10
0002      IB=7
0003      READ(6,1000)IC
0004      1000    FORMTA(17)
```

```
[MTA(17)]
ERROR    21
W UNRECOGNIZED STATEMENT.
```

```
0005      ID=((IA*IB)+IC
```

```
[A*IB)+IC]
ERROR    25
F MISMATCHED PARENTHESIS.
```

```
0006      WRITE(6,1000)ID
0007      GO TO 1
0008      END
```

```
SWITCHES = /ON
```

	BLOCK	LENGTH
	MAIN.	82 (000244)*
U 000052	000000G	\$PSH,\$1000
U 000132	000000G	\$PSH,\$1000

```
**COMPILER ----- CORE**
```

PHASE	USED	FREE
DECLARATIVES	00366	04719
EXECUTABLES	00446	04639
ASSEMBLY	00934	07068

```
0004 ERRORS
```

```
  ^C
  KI
```

```
  $
```

Sample use of FORTRN to recompile same program as above after errors have been corrected. No listing is requested.

```

$RUN FORTRN
FURTRN V004A
#DEMONS<DEMONS/ON
#^C
_KI
$

```

APPENDIX D

Sample use of LINK to link demonstration program.

```

$RUN LINK
LINK V11A01
#DEMONS<DEMONS.FTN LIB/L/USE
TRANSFER ADDRESS: 124632
LOW LIMIT: 124632
HIGH LIMIT: 137460
LINK V11A01
#^C
_KI
$

```

APPENDIX E

Sample use of PIP to list a user directory, delete a file, and list the directory again.

```

$RUN PIP
PIF-11 V7A01
#^C
_KI
$

```

DIRECTORY OK0: [35,37]

05-JUL-73

DEMONS.OBJ	2	05-JUL-73	<233>
DEMONS.LDA	14	05-JUL-73	<233>
DEMONS.FTN	1	05-JUL-73	<000>

TOTL BLKS: 17

TOTL FILES: 3

```

#DEMONS.LDA/DE
#^C
_KI
$

```

DIRECTORY OK0: [35,37]

05-JUL-73

DEMONS.OBJ	2	05-JUL-73	<233>
DEMONS.FTN	1	05-JUL-73	<000>

TOTL BLKS: 3

TOTL FILES: 2

```

#^C
_KI
$

```