J-4FT

PDP-4 FORTRAN

USERS' MANUAL

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

PDP-4

FORTRAN

USERS' MANUAL

INTRODUCTION

This manual details standard operating procedures for the several available PDP-4 FORTRAN systems. The procedures are standard in the sense that they are directed to the person who wants to write and execute FORTRAN programs with little or no attention paid to the details of the operation beyond the source language level. PDP-4 FORTRAN is written for two significantly different hardware configurations, the one an exclusively paper-tape configuration and the other a configuration which includes minimally two logical tape units (either normal magnetic tape or a dual Micro Tape unit) and an 8K memory.

The principal subsections of the FORTRAN system for paper tape are:

Compiler
Assembler
Operating System
Library.

The compiler accepts input in the FORTRAN language and produces an output in an intermediate language acceptable to the assembler. The assembler accepts the compiler output and produces a binary relocatable version of the program and (in 8K systems) a binary version of the linking loader. In 4K systems this loader is too large to be kept in memory with the assembler, and it constitutes an additional primary element of the system. When the user is ready, he loads his main program and any sub-programs, followed by any built-in functions called from the library. Once the total program is in memory, he loads the operating system and executes his program. The operating system contains an interpreter for floating point arithmetic, an interpreter for format statements, red tape routines such as fix a floating number and vice versa, and the I/O routines. The operating system must be in memory when a FORTRAN program is executed.

PROCEDURE FOR USING FORTRAN WITH A PDP-4 PAPER TAPE SYSTEM

Since the available memory is small (74,000 bits), the FORTRAN system for 4K paper tape systems is slightly less flexible than normal systems. "A" format has been omitted as well as Hollerith input. Floating point numbers may be output in "E" format only. About 1400 decimal locations are available for program and data.

The following procedure, with differences noted, applies equally well to 4K and 8K systems. In an 8K system about 4600 decimal registers are available for program and data. The bootstrap loader with starting address 7770 (for 4K machines) or 17770 (for 8K machines) is called the readin mode loader or RIM loader. Pressing the START switch on the console with the appropriate address in the ADDRESS switches is called RIM "start."

Step 1:	Prepare the programs to be compiled in accordance with the conventions established in the "Summary Description of PDP-4 FORTRAN."
Step 2:	Place the paper tape labelled FORTRAN in the reader and RIM "start."
Step 3:	Place the program to be compiled in the reader and press CONTINUE. The punch must be on. FORTRAN will punch out the intermediate object tape.
Step 4:	If other programs are to be compiled, repeat step 3. If an accidental error occurs at any time, such as the punch running out of paper tape before compilation is completed, the compilation procedure may be restarted by replacing the source tape in the reader, putting octal 22 in the ADDRESS switches, and depressing START.
Step 5:	If an error occurs in the source language, the compiler will type a three-letter plus two-digit code on the teleprinter followed by the current (last encountered) statement number. See Appendix I for the associated error conditions. As a rule, a source language error will make any further action useless. The error must be corrected and the program compiled again. However, compilation should be completed to uncover all errors in the same program. In 8K systems, the compiler will also print the offending line with the errant character flagged (by line feed).
Step 6:	When all necessary compilations have been successfully completed, remove the output tape(s) from the punch.
Step 7:	Load the FORTRAN Assembler through RIM "start."
Step 8:	Place the first program to be assembled in the reader. If several programs were compiled together they will be separated from each other by a short length of blank tape. The punch must be on, and the accumulator switches must be all in the down position. Depress CONTINUE. The assembler will punch a partial binary output.

Step 9: Depress CONTINUE to finish punching of the binary output. Symbols which are used in the source program, but never appear on the left hand side of an arithmetic statement or in an input statement or as the argument of a subroutine call, will be printed with a relative location automatically assigned by the assembler. Any statement number which is referred to but never used as a statement label will be printed also.

Step 10: If a printout of the relative locations of program symbols is desired, put the least significant (right most) switch of the ACCUMULATOR switches (test switches) to the up position and press CONTINUE. If the printout is not desired, leave the switch in the down position and press CONTINUE to restore the assembler for the next assembly. For details about the assembly capabilities of the PDP-4 Assembler see Digital-4-15-S. Many of the operations possible with the FORTRAN Assembler have not been delineated here. Should an error occur during the assembly procedure, the Assembler will print a message on the teleprinter. For a summary see Appendix II. An error printed by the assembler usually is the result of an original program error which was not detected by FORTRAN.

Step 11: If more programs are to be assembled, place the next tape in the reader and complete step 8. If several programs were compiled together, be sure that the blank tape area is under the reader light before continuing.

Step 12: Remove the assembled programs from the punch. Each program will have its title punched in readable format at the beginning. Since the FORTRAN Assembler is a one-pass assembler, this will be at the physical end of the punched tape.

Step 13: Load the FORTRAN linking loader through RIM "start," for 4K systems. Omit this step in an 8K system.

Step 14: Load the main program through RIM "start." It is important that the main program be loaded first. In an 8K system, the linking loader is punched on the main program tape only. The loader is a lengthy strip of tape with the eighth level punched at every frame and is unmistakable once recognized.

Step 15: Place any sub-programs in the reader (readable title is always the leader), and load through RIM "start." The loader will handle the problems of linking between programs.

Step 16: To obtain a printout of the absolute locations of sub-program labels put 5 in the ADDRESS switches and depress START. Should a subroutine call or library function call not yet be satisfied (i.e. not yet loaded), the symbol will be preceded on the line by a minus sign followed by an address designating the last referenced link to this symbol.

Step 17: If no library functions are needed go to step 18. Place the library tape in the reader, put 6 in the ADDRESS switches and depress START. The linking loader will search the library for functions called by the program. When all the called functions have been loaded, the loader will halt, perhaps part way through the library tape, displaying in the accumulator lights the last program address. In general, overlays of program and common storage will not be detected by the loader; this responsibility is left to the programmer.

Step 18: Load the operating system through RIM "start." Put octal 22 in the ADDRESS switches and depress START to execute the program. If paper tape input to the FORTRAN program is used, this should be ready in the reader.

THE FORTRAN SYSTEM

When Micro Tape or normal magnetic tape is available, a FORTRAN oriented job-processing executive system, called DECSYS, may be used. A minimum of two physical tape units is required. An integral part of DECSYS is the FORTRAN debugging system which is described later. The basic jobs which DECSYS can do are detailed in the following paragraph. The system is organizationally open-ended in the sense that its executive list (job list) may readily be expanded or changed to suit the purposes of a given installation.

The DECSYS Job-List

The convention of the context in which jobs are defined is that two tapes are available, a systems tape and a scratch tape. In general, the systems tape may not be written on, the scratch may be read. Paragraph headings are control commands and are typed in on the teleprinter.

Get Name - DECSYS will retrieve the named program from the system tape and relinquish control to that program.

Compile N Name, Name, Name; M Name, Name.! - N and M are both two valued; they indicate the physical source for the list of programs which follow; "1" means paper tape, "2" magnetic tape. If "copy" is typed before "compile," DECSYS will copy the systems tape onto the scratch tape and append any programs now on paper tape before proceeding.

Either list of names may have one element only, and either may be omitted. If a source designation is not typed, the source is understood to be paper tape. Unless errors occur, the job will proceed to the point of execution (load and almost go). Depress CONTINUE to execute the program.

All intermediate output is stored on the scratch tape and may be retrieved at any time by again designating the scratch tape as a systems tape and restarting DECSYS. If errors occur during compilation, the compilation will be allowed to proceed although the output will be lost and DECSYS will automatically restart. If errors occur during assembly, however, normal assembly is allowed to continue.

DECSYS will return control to the programmer in the assembly error mode with the following eventualities possible:

Cancel.!	DECSYS will restart
Symbols.!	DECSYS will print output symbol definitions.
Change.!	DECSYS will prefix the lines typed following "change" to the intermediate language and reassemble.
Print.!	DECSYS will print a listing of the assembly input.
Ignore.!	DECSYS will ignore the error and continue the job.

Assemble N Name, Name, Name; M Name, Name.! - The remarks about N, M, and the lists made in the previous section apply here. If "copy" precedes "assemble," DECSYS will copy the systems tape onto the scratch tape and append all paper tape programs before proceeding. Assembly procedure will follow the path outlined in the preceding section.

Execute Name - DECSYS will search the systems tape for the named program including subprograms called by it and proceed to the point of execution. Depress CONTINUE to execute the program.

Edit - DECSYS will retrieve the editing program and relinquish control to it. English language sources either in FORTRAN or assembly (machine) language serve as input.

Debugging in the FORTRAN System

An integral part of DECSYS is the FORTRAN debug mode. In place of "compile," the control word "debug" is typed. FORTRAN is then switched into the debug mode and the compiled program includes entries to the Debug routine incorporated in the (debug) opsys. All executable statement numbers are available as labels to the debugging program. When the point of execution is reached and DECSYS is in the debug mode, the programmer may type:

XXX	LIST
TRACE	XXX
MUST	XXX

Where XXX is a statement number, "list" is a list of variables to be printed out when the associated statement number is encountered, "trace" will cause the printout of the last five statement numbers encountered when the designated statement number occurs, and "must" designates a statement number which must be encountered in a reasonable time or else an error halt should occur. Whenever the debug program is in control, i.e. just after a printout, its "live" list of debug commands may be altered.

APPENDIX I

DIAGNOSTICS

The following diagnostics may be printed during compilations. Each diagnostic is identified by a three-letter name, and a two-digit number. In addition, when using an 8K compiler, the offending statement will be printed, with a line feed after the last character processed. For all errors, except those which indicate storage capacity exceeded, processing will continue. For both systems, the diagnostic error print (below) will be followed by the current statement number.

Name	Number	Why
CON		CONTROL STATEMENT
	1	Illegal Control Statement. The statement will be ignored.
	2	Upper case character in Control Statement. The statement will be ignored.
COM		COMMON STATEMENT
	1	Illegal entry in list. The entry will be ignored.
	2	Symbol appears twice in COMMON. The second occurrence will be ignored.
ASG		ASSIGN
	1	N not a fixed point number. Attempt to use it anyway. This will probably result in an undefined symbol at assembly.
	2	Number not followed by to. Ignore the statement.
	3	No fixed point variable. Use the variable as fixed point.
	4	Illegal format - variable. Ignore the rest of the statement.
SUB		SUBROUTINE AND FUNCTION
	1	Name not a variable. Compile anyway.

Name	Number	Why
	2	Dummy symbol not a variable. Treat as a symbol.
	3	Dummy symbol used twice. Use anyway. Will result in a multiple symbol definition at assembly.
DIM		DIMENSION
	1	Array name not a variable. Treat as a variable.
	2	Array dimensioned twice. Process anyway.
	3	Dimension not a fixed point number. Process anyway.
DO		DO STATEMENT
	1	First two letters not "do." Ignore statement.
	2	No statement number. Process anyway.
	3	No end test value specified. Process anyway.
	4	Too many characters. Ignore excessive characters.
ILF		ILLEGAL FORMAT
	1	Non-statement number at left margin. Process as statement number.
	2	Missing left parenthesis. Assume present.
	3	Missing right parenthesis. Assume present.
	4	Missing left parenthesis. Assume present.
	5	Missing right parenthesis. Assume present.
	6	Comma missing in go to. Assume present.
	7	Variable missing in arithmetic statements, ignore second operator.
	11	Illegal device number in input or output statement. Process anyway.
	12	Illegal format in accept statement. Ignore statement.
	13	Illegal format statement number in an I/O statement. Process anyway.

Name	Number	Why
	17	Extra right parenthesis.
	20	Extra characters in statement. Ignore excessive characters.
	22	Comma missing in repetitive element in I/O list. Assume present.
	24	Illegal format in I/O list element. Ignore all characters to next comma.
ICH		ILLEGAL CHARACTER
	1	Illegal character. The character will be ignored.
	2	Illegal upper case character. Treat as lower case.
	4	No more characters after an illegal one.
DIT		CALL DIT MORSE. Cannot proceed.
	1	Immediately
	2	Wrong place in table
	3	Dispatch number too big
	10	Too many cal's
	11	Illegal cal
	12	Too many exits.

If any of the errors labeled DIT occurs, please note any pertinent data and send to DEC-Programming Group.

Name	Number	Why
UFX		UNSEEN FIXED POINT
	1	Empty or punctuation. Statement done, or a punctuation character. Process as if it were fixed point.
	2	Floating point quantity, treat as fixed point.
	3	Not a fixed point constant. Treat as if it were.
FOR		FORMAT STATEMENT
	1	Character missing.
	2	Illegal format. Ignore remainder of statement.

Name	Number	Why
	3	Characters missing. Ignore remainder of statement.
	4	Illegal control character. Process anyway.
	5	Illegal punctuation.
	6	Letter other than, I, F, E, X, H. Ignore remainder of statement.
	7	N too large in H.
IFU		ILLEGAL FUNCTION USAGE
	1	Function name on left side outside function definition.
SCE		STORAGE CAPACITY EXCEEDED
		Processing may not proceed.
	1	Polish stack exhausted.
	2	Table exceeded.
	3	Table exceeded.
	4	Symbol generator exhausted.
	5	Table exceeded.
	6	Statement too long.
	7	Push down stack exceeded. (too many nested do's).

APPENDIX II

ERROR MESSAGES (FORTRAN ASSEMBLER)

With the exception of sce (storage capacity exceeded) and ilp (illegal parity), assembly continues after the error message has been printed, unless assembling a library tape. An error message may occur in one of three formats.

Format A

ERR VAL1 SYM VAL2

Format A is used to indicate errors in the redefinition of symbols. ERR is a three letter mnemonic for the particular error. VAL1 is the old octal value of the symbol SYM. VAL2 is the value of the new definition attempted. Whether the symbol was redefined depends upon the particular error.

ERR	Meaning
mdt	The symbol was redefined with a comma.
rsp	A permanent symbol was redefined.
rda	An attempt to redefine a symbol was made. The symbol was not redefined.

Format B

ERR OCT SYM

The general error message is printed in Format B. ERR indicates the particular error. OCT is the octal address at which the error occurred. SYM is the symbolic address at which the error occurred.

ERR	Meaning
ifp	Illegal format in parameter assignment
ifc	Illegal format in a comma assignment
ifq	Illegal format in library list
ify	Illegal format in internal declaration

tms	Toomany symbols in internal declaration
liq	Illegal term punctuation in library list
mdt	The value and address disagree in an address assignment.
tua	Too many undefined symbols in an address assignment.
ift	Illegal format in an absolute address assignment
lit	Illegal terminator in a <u>pundef</u> or <u>external</u> list
ifl	Illegal format in a <u>pundef</u> or <u>external</u> list
ifs	Illegal format in a <u>start</u>
ifi	Illegal format in an input pseudo instruction
sce	Storage capacity exceeded
lns	non-symbol in a <u>pundef</u> list

Format C

ERR OCT SYM CAUSE

Format C is an expanded version of Format B. CAUSE is additional information to help the programmer ascertain the cause of the error. For example, in the case of an error caused by an undefined symbol, the symbol will be printed.

ERR	CAUSE	Meaning
ilp	character	Illegal parity (place correct character in ACS and "continue")
ust	symbol	Undefined symbol in a <u>start</u> or <u>pause</u>
uaa	symbol	Undefined symbol in an absolute address assignment
upa	symbol	Undefined symbol in a parameter assignment
ich	character	Illegal character
sys	symbol	Already defined symbol in internal declaration

<u>Undefined Symbol Assignment</u>

At the end of assembly, before the loader is punched, the undefined symbols and their definitions will be printed. Each undefined symbol which was used in a storage word will be defined as the address of a register at the end of the program, and the definition printed. If the symbol was not used in a storage word, then just the symbol will be printed and the symbol will not be defined. An example of the latter is a symbol which appears to the right in a parameter assignment, but nowhere else.