DECUS

DIGITAL EQUIPMENT COMPUTER USERS SOCIETY

MAYNARD, MASSACHUSETTS / TEL. 897-8821 / TWX 710 347-0212

November 18, 1965

Mr. G. A. Michael
Lawrence Radiation Laboratory
L-63
Livermore, California

Dear Mr. Michael:

Enclosed are the programs which you requested from the DECUS Program Library.

In order to increase the usefulness of the DECUS Program Library, we request that you review each of these programs for accuracy, ease of use, clarity of documentation, etc., and return your comments to DECUS for the benefit of future users.

If you have any major problems with the program, or programs, please let us know and we will try to assist you in any way possible.

Please send all comments to: DECUS Executive Secretary, Digital Equipment Computer Users Society, Maynard, Massachusetts 01754.

Thank you for your interest.

Sincerely,

Angela J. Cossette

Angela J. Cossette (Mrs.)
DECUS Executive Secretary/Editor

Enclosures:

DECUS No. 82 - tapes and write-up

DIGITAL EQUIPMENT COMPUTER USERS SOCIETY

Maynard, Massachusetts

DECUS Library - Program Writeup | DECUS No. 82

TITLE:	FORTRAN FOR THE PDP-1
AUTHOR:	Developed by the Air Force Technical Applications Center (AFTAC) and The Geotechnical Corporation.*
DATE:	Received by DECUS: July 1965
HARDWARE REQUIRED:	Standard PDP-1, typewriter, paper tape reader and punch.

*Programs by The Geotechnical Corporation were developed under contract to AFTAC. Any questions regarding specific details of the Program should be directed to: HQ., USAF (AFTAC) TD-1A, Attn: SSgt. John Davidson, Washington, D. C. 20333.

INDEX

DECUS NO. 82

TITLE: FORTRAN FOR THE PDP-1

Notes concerning compiling and loading the FORTRAN compiler and associated routines.

The DS tapes for the FLIP 8/28 floating point package should all be compiled and the DL tapes labeled so that the necessary ones may be linked with segments of the FORTRAN system.

The FLIP 8/28 floating point package may also be used for non-FORTRAN programs. A complete writeup of the package is being prepared by the author of the FLIP routines.

All of the included routines are for machines with mul div hardware, however, they may be converted for use with mus dis by making ig's for mul, mpy, div, and dvd which simulate the mul div instructions by subroutines.

Compiling the FORTRAN compiler:

The tapes labeled A1, A2, and A3 should be compiled into a single DL tape. This is the lower segment of the compiler. The tape labeled B should be compiled into DL tape. The tape labeled C should be compiled into DL tape. The DL tapes produced by tapes B and C are the upper memory segment of the system.

To load and link the upper and lower memory segments, follow the instructions given on page 62 of the DECAL Manual (DECUS 39, September 26, 1963).

If additions or changes are made to the compiler, be sure that the main program does not extend over address 6777 as in the current versions (versions 1 and 2). The area extending from 7000 through 7777 is used for various tables and for the input buffer area.

Compiling and linking the subroutine package:

The tapes labeled S1 and S2 should be compiled into a single DL tape. The tape labeled S3 should be compiled into a DL tape. Using the low linking loader, load the DL tape produced from tapes S1 and S2 then load the tape produced by tape S3. After these are loaded, put sense switch 6 up to see which of the floating point routines are required and load the necessary ones. The subroutine package may actually be located anywhere in memory, however, addresses 7775, 7776, and 7777 are used no matter where the package is located. After the subroutine package is loaded, punch off a symbol tape which is to be used for linking the subroutine package and FORTRAN programs. After punching the symbol tape, punch off the subroutine package.

FORTRAN FOR THE PDP-1 - VERSION 1 and 2

The FORTRAN compiler for the PDP-1 is not intended to be a replacement language for the other compiler and assembly languages already in use on the PDP-1, however, it is useful for short programs which may easily be coded in FORTRAN.

The first version of the FORTRAN system for the PDP-1 uses mus and dis instructions; and mpy and dvd subroutines.

Version 2 is for machines with mul div hardware. Be sure that the mul/mus switch is set to mul, and the div/dis switch is set to div.

A statement number is required only if the statement is referred to elsewhere in the program.

Tabs, spaces, and redundant carriage returns may be used freely.

Backspace causes the entire line to be ignored.

I. CONSTANTS

Integer constants magnitude $\leq$ 131071.

Floating point constants may be up to 10 digits to the left of the decimal point and/or up to 10 digits to the right of the decimal point. Floating point constants may be followed by the letter e and an exponent.

The exponent of a floating point value must never exceed ± 38.

Floating point values are in a 2-word format with an 8-bit exponent and a 28-bit mantissa.

Statement numbers may be up to 6 decimal digits.

Variable names may be up to 6 characters.

All alphabetic characters are lower case.

A fortran comment is a line beginning with a letter c followed by a tab.

Maximum statement length varies, however, up to 156 characters are allowed for any type of statement.

If sense switch 5 is up, the input to the compiler is from the typewriter; if sense switch 5 is down, the input is a FIO-DEC paper tape.

If sense switch 6 is down, the output is a DECAL symbolic tape; and if sense switch 6 is up, the output is typed. This is sometimes useful for seeing how a particular statement compiles.

Parentheses or brackets may be used interchangeably.

II. ARITHMETIC STATEMENTS follow the general rules of the FORTRAN language.

Examples of arithmetic statements:

```
abc = sinf(x(3)-(abc x any(n,5)-(cosf(a/b))) + 45.6
n(4,j,3) = i x [ k/45 + (match-4)] / iabsf(k-4)
a= -b + c + (-d)
```

III. FUNCTIONS AVAILABLE:

sinf - floating point sine (argument in radians)
cosf - floating point cosine (argument in radians)
acosf - floating point arc cosine (argument is a cosine, returns with radians)
asinf - floating point arc sine (argument is a sine, returns with radians)
atanf - floating point arc tangent (argument is a tangent, returns with radians)
expf - floating point exponential
sqrtf - floating point square root
logf - floating point log
log10f- floating point log (base 10)
absf - floating point absolute value
iabsf - integer absolute value

Names of integer variables begin with letters i, j, k, l, m, or n.

Names of floating point variables begin with any letter except i, j, k, l, m, or n.

Variable names may not end with the letter f.

Mixed mode is not allowed and is not checked by the compiler.

a= a-b/j	is illegal as j is an integer variable.
i= j+k-b/3	is illegal as b is a floating point variable.
a= i-5+k	is allowed; i-5+k is calculated in integer mode, the result converted to floating point, and stored in variable a.
i= a-b/7.3	is allowed; a-b/7.3 is calculated in floating point mode, the result converted to integer mode (with any fractional part dropped), and the result stored in integer variable i.

The DECAL subscript interpreter is used, so up to 32 subscripts on a single variable should be okay.

Subscripts begin with 1, not 0 as in DECAL.

Recursive subscripts are not allowed.

Subscript arithmetic is allowed in integer calculations, however, it is <u>not</u> allowed in floating point.

Examples:

k= i(3/k+n, j)	okay
b= a(3/k+n, j)	is illegal as the subscript arithmetic will be performed in floating point mode resulting in an incorrect subscript.

IV. GO TO STATEMENTS

go to 100	program transfers to statement 100.
go to (45,600,3,888),k	causes transfer to the 1st, 2nd, 3rd, etc., statement on the list depending upon whether k is 1, 2, 3, etc.
assign 400 to i	saves location of statement 400 in i.
go to i, (34,500,400)	transfers to statement specified by i which has been set by a previous "assign" statement.

The list of statement numbers in the parentheses is optional.

go to i	is a legal assigned go to.

V. SENSE LIGHTS AND SENSE SWITCHES

if (sense switch 3) 12, 34	transfers to statement 12 if the console sense switch 3 is on (up), transfers to statement 34 if the switch is off (down). Switches 1 thru 6 may be used. If sense switch 7 is tested it will cause a transfer if any switch is on.
sense light 0	turns off sense lights 1 thru 6.
sense light 5	turns on sense light 5. Sense lights 1 thru 6 are normally used, however, sense lights 7 thru 99 may also be used but the sense light 0 statement will not turn off lights 7 thru 99.
if (sense light 4) 34,2	transfers to statement 34 if sense light 4 is on, transfers to statement 2 if sense light 4 is off. If (sense light) statements always turn off the sense light that is being tested.

Note: The FORTRAN sense lights are not the same as the program flags on the console; however, at a "pause" or "stop" statement sense lights 1 through 6 are displayed in the program flags.

VI. IF STATEMENTS

if (a) 12,34,56	"a" may be any arithmetic expression. Mixed mode is not allowed.
if (a(3)-(b/x+56.7)) 300,4,5032	the arithmetic calculations are performed and the program transfers to statement 300 if the result is negative, statement 4 if zero, or statement 5032 if positive.

VII. PAUSE AND STOP STATEMENTS

pause 45	The line "pause45" is typed out and the program halts. Pushing continue, on the console, causes the program to resume.
stop 6667	The line "stop6667" is typed out and the program halts. This is a terminal halt and pushing continue has no effect.

After the "pause" or "stop" any decimal number is leagal or you may type a special message by using a series of words separated by - [dash].

Examples:

```
pause 987654399
stop--this-is-the-end-of-the-program
```

The entire "pause" or "stop" line is typed out. Don't use commas or periods as part of the line.

VIII. DO STATEMENTS

Examples:

do 500 j=1,300	The following statements through statement 500 are executed. The first time through the loop j will be 1; j is incremented by 1 each time until the final time through the loop it is 300, then the statement following statement 500 is executed.

The increment may also be specified

do 67 k=1,45,3	the increment is 3

The values used for the "do" may also be unsubscripted integer variables.

The values may have a minus sign.

do 689 kr = 100,-n,-2	the increment is a negative 2 so kr will be 100 the first time through the loop and will be 98 the second time, etc.

"do" loops may end on the same statement number.

```
do 400 k3= 1,300
do 400 k4= 5,m,7
```

After a "do" loop is completed and the program transfers out of the loop, the integer variable used for the index will contain the same value as the last time through the loop.

The second and/or third parameters of a "do" statement may not be the same variable name as the index variable.

Examples:	do 50 k=1,k	is not allowed.
	do 50 k=1,4,k	is not allowed.

The above rule is not checked by the compiler.

The "continue" statement may be used for terminating a "do" loop where the final statement would otherwise be an "if", "go to", or another "do".

Example:

```
        do 500 j=1,34
        if (a(j)-value) 600,600,500
500 continue
```

The "continue" statement may also be used freely in the program; however, if it is not the final statement of a "do" loop, a "nop" instruction is executed wasting one word.

IX. END STATEMENT

All programs must end with an "end" statement. If the "end" statement is in the flow of statements being executed, it must either have a statement number or have characters following the "end" in the same manner as a "pause" or "stop". If the "end" statement is not to be executed, it may be by itself on a line.

X. DIMENSION STATEMENT

The "dimension" statement is the same as in other FORTRAN systems except in this system it must be in the executed flow of the program. It is executed once and then bypassed if the program flow repeats through the same path of the program.

A subscripted variable must be dimensioned before it occurs in any other statement.

Examples of the "dimension" statement:

```
dimension i (45)
dimension j (4,5), abc (55), k(5), x(2,2,2,2)
```

XI. INPUT/OUTPUT

The standard PDP-1 consists of a typewriter and paper tape input/output, thus only those devices will be provided for in version 1 of the FORTRAN system.

type 45,a,i,x(4),y(i)	the variables on the list will be typed according to format 45.
punch flex 45,a,i,x(4)	the variables on the list will be punched in FIO-DEC code according to format 45.
accept 74,i,a(34),k(j)	the variables on the list will be accepted from the typewriter according to format 74.
read flex 85,i,j,xx(j)	the variables on the list will be read from FIO-DEC coded paper tape according to format 85. If a stop code (oct 13) is read, the program enters a one-word loop in the subroutine package. The program may provide for stop codes being read by using a DECAL insert (see SPECIAL FEATURES).

"Do" type indexing within input/output statements of the following form is not allowed. Examples:

type 66, (a(k), k=1, 34)	is not allowed
feed flex, n	This is a non-standard statement and causes n blank lines of paper tape to be punched. The argument, n, may be an unsubscripted integer variable or integer constant.
end flex	This is a non-standard statement and causes a stop code to be punched.

XII. FORMAT

This statement specifies how data is to be transmitted between input/output devices and the computer.

Example of a format:

35 format (a5, i6, f13.6, 3x, i2)

"i" specification

The format specification i10 may be used to type a number which exists in the computer as an integer quantity. Ten type positions are reserved for the number. It is typed in this 10-digit field right-justified (that is, the units position is at the extreme right). Positions in the field to the left of the most significant digit are blank.

If the format specification i10 is used for input, 10 characters may be typed in or read from paper tape and the resulting integer value will be stored in the variable specified by the input statement. The largest integer value that should be used as input is 131071. There is no check for exceeding 131071 magnitude. Non-digit characters (except minus sign) will cause an error halt.

"f" specification

If the format specification f10.3 is used to type a number which exists in the computer, 10 type positions are reserved for the number and there will be 3 positions to the right of the decimal point.

If the format specification f12.5 is used for input, 12 characters will be read from the input device. If there is no decimal point in the string of input characters, one will be assumed to be in the value and the rightmost 5 digits are the decimal fraction. If a decimal point is one of the input characters its position in the field over-rides the specification.

"x" specification

Blank characters may be provided in an output record, or characters of an input record may be skipped by means of the "x" specification.
3x in a format would cause 3 characters to be skipped.

"h" specification

The "h" specification is used for input/output of alphanumerical information that will not be manipulated by the program.

5habcde in a format used for output would cause 5h characters "abcde" to be output.

3habc in an input format would cause 3 characters from the input device to replace the 3 characters in the "h" field.

All PDP-1 typewriter characters are usable in "h" fields, however, the specification must include upper and lower shift as characters.

"a" specification

If the format specification a3 is used for output, the word from the output list will be outputted as alphanumeric characters (FIO-DEC code). If the format specification a4 is used for input, the characters are stored in FIO-DEC code. Only 3 "a" characters will fit in a word, so for input, the last 3 of the field, if it exceeds 3, will be stored.

"a" fields may only be stored in integer variables.

A slash (/) in a format indicates end of a line.

NOTE: There is some problem with input/output of large "i" and "f" format specifications. For this version try to limit the field width to about 20 characters.

For "i" or "f" output, try to allow extra character positions in the field width so there will be room for a minus sign.

For "f" specifications where the output field is only to be decimal fractions with no integers, allow an extra position so a leading zero may be output. Example: Consider the floating point value 0.0345 which is to be typed. If the field is f5.4, there will be no room for the leading zero so a field overflow occurs. If the field is f7.4, it will type correctly.

If there is a field overflow (too many digits to fit in the field) during output, the field will be output as dashes (-).

Example: Consider the value 500.678 which is to be output under specification f6.3. There are too many digits so it will be output as 6 dashes (------).

The input/output routine scans the input/output statement list, and for each variable gets the corresponding format specification. If the format has no more specifications, it starts again with the first specification.

Example: Consider the variables a,b,c which are to be output as f10.3 fields type 30, a,b,c.
30 format (f10.3)

The variables will all be typed as f10.3 fields, however, whenever the format routine has to restart scanning the format, it types a carriage return so each variable will be on a separate line.

When the format has more specifications than the input/output list requires, the extra specifications are ignored.

When the input/output list requires the format to be restarted and the input/output list finishes before the format is used up and there are "h" fields in the format, the routine requires the entire line to be input or output. However, the "i", "a", and "f" fields are treated as "x" fields.

Another feature of the format statements is the ability to repeat a specification by preceeding the specification with a number.

Example:

```
10      format (a4,5f7.3,3x,f10.4)      will work in the same manner as
10      format (a4,f7.3,f7.3,f7.3,f7.3,f7.3,3x,f10.4)
```

Only one pair of parentheses is allowed in a format.

There is no fixed line length for input/output so you are not limited to 120 characters per line as in most FORTRAN systems.

There are two special escape characters usable with the input routines.
During input under an a, i, x, or f specification if either a tab or carriage return is input, the input variable being input is stored and the routine scans ahead for the next input variable on the list.
The tab and carriage return escape characters may be replaced by using DECAL inserts. (See SPECIAL FEATURES.)

XIII. OPERATION OF THE FORTRAN SYSTEM

The FORTRAN system consists of 7 paper tapes for compiling, loading, and executing FORTRAN programs.

Tape 1 - FORTRAN compiler

Tape 2 - DECAL compiler (modified)

Tape 3 - HI LINKING LOADER

Tape 4 - Symbols for linking the subroutine package to your FORTRAN program.

Tape 5 - Libetape of subroutines required by some programs.

Tape 6 - Subroutine package - input/output routines, subscript interpreter, and other routines required by most FORTRAN programs.

Tape 7 - Punch off routine

To compile, load, and execute a FORTRAN program:

1. Make a FORTRAN symbolic FIO-DEC paper tape of your FORTRAN program. You may skip this step by typing the program directly into the FORTRAN compiler, however, if you have program errors you will then usually have to retype the entire program.

2. Load the FORTRAN compiler. The compiler is in a self-loading, condensed tape form. Place in reader, turn on, push READ-IN on the console.

3. Ready your program tape in the reader (unless you are going to type your program in, then have sense switch 5 up). START at 0000. If sense switch 3 is up during compilation, the input statement is not included in the output as a comment, resulting in a slightly shorter output tape.

4. If your program compiled without any errors load the DECAL compiler.

5. Ready the FORTRAN compiler output tape, which is a DECAL symbolic tape, in the reader and depress the space bar on the typewriter.

6. If your program compiled without any errors load the HI LINKING LOADER.

7. Ready the DECAL output tape (DL tape) in the reader. If you don't want to type out system symbols while loading set sw.1 up. If you want to origin your program at 0000 to use all available memory, set sw.2 up and clear the Test Word Switches (all down) START at 6000. After starting put sw.2 down.

8. Ready the symbol tape in the reader for linking the subroutine package. Push CONTINUE on the console.

9. Put sw.6 up and push CONTINUE to store integer constants. If the letters rq are typed in red then your program probably requires some of the libetape subroutines.

   9a. To read the libetape - Ready tape in reader, put sw.5 up, and push CONTINUE. After the libetape is read in then go to step 9 above. If the letters rq are again typed out, your program must require some subroutine that is not provided with the system.

10. After your program has been loaded (and the highest address does not overlap the subroutine package) then the subroutine package must be loaded. Ready the tape in the reader and push READ-IN on the console.

Your program is now ready to execute or be punched off. The first time your program is started the location of the array storage area is typed out. Be sure that the lowest address of the array area does not overlap the highest address of your program.

The punch off routine "pofmem" punches off a loader and all non-zero words from 4000 through 7777 and 0000 through 3537. The resulting tape, when loaded, is not self starting.

The punch-off routine occupies 3540 through 3777.

When your program is loaded using the output tape of "pofmem", if the Memory Buffer is a normal hlt (760400), you may start your program. However, if the Memory Buffer is octal 141414, then there was a check-sum error indicating either a punch or reader error.

XIV. ERRORS

The FORTRAN compiler checks most of the common errors made in FORTRAN programs.

If an error is found the following is typed out:

erN lineJ
where N is the error type and J is the line number in your program.

After an error has been found the rest of the program is scanned for further errors, but the normal output tape is not punched.

A. FORTRAN errors detected by the compiler:

Number	Error
1	Statement too long.
2	Illegal character.
3	Unmatched parentheses or brackets.
4	Statement number too long (over 6 digits).
5	Unrecognized statement.
6	Comma missing after variable name in assigned go to.
7	"do" loop terminated by a "do", "if", "stop", or "go to".
8	go to N statement number N is over 6 digits.
9	Not fixed point variable is assigned go to.
10	No comma after right parentheses of computed go to.
11	Can't find letters "to" of assign statement.
12	Too many floating point constants.
13	"if" statement has illegal path.
14	Illegal sense switch number > 7.
15	Bad "if (sense light)" statement.
16	Some unterminated "do" loops.
17	Nothing to right of equals sign in arithmetic statement.
18	"do" terminated previously.
19	Too many "do" loops.
20	Index variables name in "do" statement is not an integer variable.
21	Variable name over 6 characters.
22	Only one parameter for a "do" loop (do 123 k=2).
23	Comma after third parameter of a "do" loop (do 123 k=1,2,3,4).
24	Variable name used for "do" loop parameter begins with a digit.
25	Variable name used for "do" loop parameter is not an integer name.
26	"do" has been terminated previously.
27	"format" has no statement number.
28	Error in "format" statement
29	No carriage return after) in "format" [12 format (a4, (f5.6)) not allowed].
30	No decimal point in "f" field specification of "format".
31	Field specification too large (Max. = 63).

32	No format number in input/output statement.
33	Variable name over 6 characters.
34	Too many undimensioned variables.
35	"dimension" has a statement number.
36	Variable name ends with letter "f".
37	Undimensioned variable has a subscript.
38	Dimensioned variable has no subscript.
39	Subscripted variable has not been dimensioned.
40	Too many dimensioned variables.
41	Parity error in FIO-DEC input paper tape.
42	Over 10 digits to right or left of decimal point in floating point constant.
43	Exponent overflow in floating constant routine.

The only time a variable name is checked to see if it is too long, (ends with letter "f", etc.), is when it is being defined. Fortran variables are defined by being on the left side of an equals symbol, in a "dimension", in an input list, or in an "assign" statement.

Error type 26 and error type 18 above appear to be the same, however, they indicate different errors.

Error 18 Example:

```
        do 200 k=1,300
        do 200 j=1,34
200     continue
c       do loop 200 has been satisfied.

        do 200 k=1,50
```

Error 18 - a do loop that is being defined by a "do" statement has already been terminated.

Error 26 Example:

```
        do 100 k=1,100
100     continue
c       do loop 100 has been terminated.

100     a=b-c
```

Error 26 - Terminating statement number of a "dc" loop has been found previously.

Errors 37 and 39 appear to be the same, however, they indicate different errors.

Example of errors 37, 38, and 39:

```
        dimension a(100)
        b= 57.29578
        b(2) =
```

Error 37 - "b" has a subscript, has not been dimensioned, but has been defined as an unsubscripted variable.

```
        a=
```

Error 38 - "a" has no subscript, has been dimensioned.

```
        c(3)=
```

Error 39 - "c" has a subscript, has not been dimensioned, and has not been defined as an unsubscripted variable.

Duplicate statement numbers are detected during the pass through DECAL.

Other errors or program halts:

hlt instruction (760400) = divide halt - probably will never occur.

Unused instruction 140000 is used to indicate various other troubles. The rightmost octal digits indicate the type of trouble.

140001 = stop code (13) - no "end" statement.

140002 = overflow set at entry to floating constant routine.

140003 = overflow set during processing of previous statement.

140004 = floating divide by zero. Probably will never occur.

Not all errors are detected during the pass through the FORTRAN compiler, however, they will usually be detected during the pass through the DECAL compiler. A common error detected by the DECAL compiler is a name typed out on the DECAL MAP as an "aps". This indicates an undefined or improperly defined FORTRAN variable name. The only exception to this is the name may have been used in a DECAL insert

Consult the DECAL manual for other errors.

Note: The names of dimensioned variables are typed out on the DECAL MAP as being only one word. That word is not the location of the array, but is a word where the address of the array will be stored.

B. Errors Detected During Execution Of Your Program.

hlt instruction (760400) indicates a divide halt.

Unused instruction 140000 is used for other error halts. The rightmost digits indicate the type of error. To continue after a 140000 instruction the CONTINUE button on the console must be pushed twice.

140002	FIO-DEC paper tape parity error, the character is in the IO. Put the correct character in the Test Word switches and push CONTINUE twice.
140003	Illegal format. No recovery.
140004	No carriage return at end of line or format error (paper tape input).
140005	Non-digit character in "i" or "f" field. Resume and char. is used anyway.
140006	Divide halt during "f" format processing. Should never happen.
140011	Index of a "do" loop $\geq$131071. Resume and invalid index is used anyway.
140012	Floating number too large to convert to integer variable. Resume and trash is used for the integer value.
140013	Not enough room in array storage area. Indicates compiler error.
140014	Too many subscripts or out of range. Resume and invalid subsc. used anyway.
140015	Out of bounds of array. Resume and invalid subsc. used anyway.

140016	Recursive subscript. Not allowed.
140017	Exponent underflow - no recovery.
140020	Exponent overflow - no recovery.
140021	"sqrtf" has a negative argument. Resume and argument is made positive.
140022	"logf" or "log10f" argument $\leq$ zero. No recovery.

The overflow indicator is not checked during execution of FORTRAN programs and is cleared and used by the floating point interpreter and "do" loop subroutine.

XV. SPECIAL FEATURES

During execution of a FORTRAN program address 7777 is used for various special switches.

If bit 0 is on, an overbar is typed at the start of every format field.
If bit 1 is on, output field overflow is allowed.
If bit 17 is on, "type" statements are punched instead of typed.

Address 7776 and address 7775 are used for escape characters during input and if the input character matches either word, the format routine stores the input variable and goes to the next field.

7776 normally contains octal 77 (carriage return).

7775 normally contains octal 36 (tab).

The output routines for the typewriter and punch use the status bits to try and keep up a reasonable speed; so if you are going to use DECAL inserts to type or punch, use the following subroutines.

Subroutine typ' will type the character in the right 6 bits of the IO.
Entrance to typ is a jsp typ.
Subroutine pcf' will punch the right 6 bits of the AC with correct parity.
Entrance to pcf is a jda pcf.

The paper tape input, using "read flex", is buffered. Up to 30 characters are read at a time unless a carriage return or stop code is read.

If stop code is read, the subroutine goes into a one-word loop at system symbol eof'. If you wish to provide for stop codes in you program, deposit an address in eof.

Example:

```
→          law st100; dap eof
```

If a stop code is read, the program transfers to statement 100. If you are going to use system symbol eof; at the start of your program →dss eof.

XVI. DECAL INSERTS

Some of the features of DECAL have been expunged, however, they may be put back in for use with DECAL inserts. They were expunged merely to give more room on the symbol table.

A DECAL insert begins with a right arrow → .
A line beginning with → is copied directly into the output.

The program is always in fixed-point mode at the start of DECAL insert and if the insert uses "efm 2", be sure and give an "lfm" before resuming with FORTRAN statements.

The FORTRAN compiler uses block symbols for some "do" and "format" symbols and they are expunged whenever possible so DECAL inserts should not use block symbols.

FORTRAN or DECAL symbols (variable names) should not be any of the following:

1. May not be the same as any symbol on the DECAL symbol table.
2. May not be the same as any of the FORTRAN statement types; such as "if", "end", etc.
3. May not be of the form "stN' where N is a decimal number.
4. May not be of the form "cN" where N is a decimal number.

The following is a listing of the tape used to modify DECAL for the FORTRAN system.

```
... fortranize decal
... 18 May 65

xsy ~
~ dig                   beg lv7 op1 rs1
                        fde lst; cma en
xsy abs
absf dig                beg lv7 op1 rs1
                        fde; spa
                        fde lst; cma        end
iabsf esy absf

xsy / ×
× dig                   beg lv6 op2 rs1 cmt
                        jda fsy; bci.imp.
                        lac 2 lst           end
/ dig                   beg lv6 op2 rs1
                        jda fsy; bci.idv.
                        lac 2
                        fde lst; hlt        end

xsy mpy dvd
mpy dig                 beg lv7 op1 rs1 nlc
                        jda ths
                        lac 1 lst           end
dvd esy mpy


sinf dig                beg lv7 op1 rs1
                        jsp ths lst         end
cosf                    esy sinf
sqrtf                   esy sinf
atanf                   esy sinf
acosf                   esy sinf
asinf                   esy sinf
logf                    esy sinf
log10f                  esy sinf
expf                    esy sinf


xsy nop
nop                     ewd 760000
```

```
xsy op1 op2 op3 rs1 cmt ctr ths fsy lst mns nac
xsy nlc lv0 lv1 lv2 lv3 lv4 lv5 lv6 lv7
xsy dip iot opr skp usk cal lap mus dis
xsy fde fct dpy rrb srb rcb cnv esm lsm
xsy cbs msm mwc mrc mcb rck cac cks eem lem
xsy mcs chn mec
xsy isd iso asd aso asc dsc isb bac bpc bio bjm dal
xsy lss <= =>|
xsy goto ≡ ⊃ ⊇ < ≤ > ≥ ≠
xsy sin cos atn sqrt ln log exp exp10 ↑
xsy if then else clear set for stepu stepd until while
xsy do lar ina arrase array realarray
xsy ∨ ∧ xor
xsy procedure tmp ppa rpa tyi tyo dig •

fix fin.
```

DECAL permanent symbol table (modified version)

```
add  400000; and  020000; dac  240000; dap  260000; dio  320000
idx  440000; ior  040000; isp  460000; jmp  600000; jsp  620000
lac  200000; law  700000; lio  220000; sad  500000; sas  520000
sub  420000; sma  640400; spa  640200; spi  642000; sza  640100
szf  640000; szo  641000; cla  760200; clf  760000; cli  764000
cma  761000; hlt  760400; lat  762200; xct  100000; dzm  340000
jda  170000; clo  651600; mul  540000; div  560000; nop  760000
..   000000; loc  000000; stf  760010; ppb  720006; rpb  720002
```

action operators:

```
fin     stp     dao     fix     ...     :.      :     ewd     esy
'       dss     ral     rar     rcl     rcr     ril     rir     sal
sar     scl     scr     sil     sir     szs     oct     dec     poi
noi     ndi     pdi     AC      (       bci     str     org     efm
lfm     xsy     blk     lve     xss     odv     oda     opt     [
```

instruction generators:

```
;       beg     end     =>      →       =       +       -       ×
/       adr     |       )       tpo     ,       ]       ~       absf    iabsf
×       /       mpy     dvd     sinf    cosf    sqrtf   atanf   acosf   asinf
logf    log10f  expf
```

None of the above symbols may be used for variable names.
The instruction generators and action operators that have been expunged may be put back in if needed; however some of them take quite a bit of storage in DECAL so there will not be room for many statement numbers or variable names in the DECAL symbol table during compilation of FORTRAN programs.

```
c                       sample program
c                       type in up to 100 numbers
c                       put numbers in ascending or descending order
                        (according to sw.2)
c                       type out ordered numbers
                        dimension a(100)
```

```
c                       clear array a
1                       do 2 j=1,100
2                       a(j)= 0.0


                        type 3
3                       format (//5hinput)

                        do 20 j=1,100
                        accept 10,a(j)
10                      format (f10.5)
c                       parentheses and brackets may be used interchangeably
                        if [a(j)] 20,50,20
20                      continue

50                      do 150 m= 1,j
100                     do 150 k= m,j
                        if (sense switch 3) 115,110
110                     if (a[m]-a[k]) 150,150,140
115                     if (a(k)-a(m)) 150,150,140
140                     temp = a(m)
                        a(m)= a(k)
                        a(k)= temp
                        go to 100
150                     continue

                        type 160,j
160                     format (1hj,i4/8h ordered)
                        do 200 k=1,j
200                     type 10,a(k)
c                       spaces and tabs may be used freely
                        g o  t o            1
                        end
```

Listing of the output tape (DS tape which is read by DECAL)

```
dss rdf wrf tif tof xf ff dff cff dof f1f f2f f3f f4f f5f f6f
 fdf eff arf i1f i2f
mainprog'               lac laf
                        jda arf

... c                   sample program
... c                   type in up to 100 numbers
... c                   put numbers in ascending or descending order
                        (according to sw.2)
... c                   type out ordered numbers
... dimensiona(100)
jsp i2f
a:.                     lac a[100]


... c                   clear array a
... 1do2j=1,100
st1:.                   pdi 1
st2n1s:                 dac j
```

```
                        jmp *+3
                        dec 100
st2n1:                  ..1
... 2a(j)=0.0
st2:.                   efm 2
                        c1=>a[j]
                        law st2n1
                        jda dof
                        lac st2n1s
blk


... type3
                        law st3+1
                        jda tof
                        nop
... 3format (//5hinput)
st3:.                   jmp st3a
                        ..210000
                        ..210000
                        ..700005
                        ..714547
                        ..242376
st3a:                   nop
blk

... do20j=1,100
                        pdi 1
st20n1s:                dac j
                        jmp *+3
                        dec 100
st20n1:                 ..1
... accept10,a(j)
                        law st10+1
                        jda tif
                        dac a[j]
                        nop
... 10format (f10.5)
st10:.                  jmp st10a
                        ..661205
st10a:                  nop
... c                   parentheses and brackets may be used interchangeably
... if(a(j))20,50,20
                        efm 2
                        a[j]=>strf
                        lfm
                        lac strf
                        spa
                        jmp st20
                        sza'
                        jmp st50
```

```
... 20continue
st20:.                  law st20n1
                        jda dof
                        lac st20n1s
blk

... 50do150m=1,j
st50:.                  pdi 1
st150n1s:               dac m
                        lac j
                        dac st150n1-1
                        jmp .+3
                        ..
st150n1:                ..1
... 100do150k=m,j
st100:.                 lac m
st150n2s:               dac k
                        lac j
                        dac st150n2-1
                        jmp .+3
                        ..
st150n2:                ..1
... if(senseswitch3)115,110
                        szs 3
                        jmp st115

... 110if(a(m)-a(k))150,150,140
st110:.                 efm 2
                        a[m]-a[k]=>strf
                        lfm
                        lac strf
                        spa
                        jmp st150
                        sza'
                        jmp st150
... 115if(a(k)-a(m))150,150,140
                        jmp st140
st115:.                 efm 2
                        a[k]-a[m]=>strf
                        lfm
                        lac strf
                        spa
                        jmp st150
                        sza'
                        jmp st150
... 140temp=a(m)
st140:.                 efm 2
                        a[m]=>temp
... a(m)=a(k)
                        a[k]=>a[m]
... a(k)=temp
                        temp=>a[k]
... goto100
                        lfm
                        jmp st100
```

```
... 150continue
st150:.                 law st150n2
                        jda dof
                        lac st150n2s
                        law st150n1
                        jda dof
                        lac st150n1s
blk

... type160,j
                        law st160+1
                        jda tof
                        lac j
                        nop
... 160format (1hj,i4/8h ordered)
st160:.                 jmp st160a
                        ..700001
                        ..417676
                        ..710400
                        ..210000
                        ..700010
                        ..004651
                        ..646551
                        ..656476
st160a:                 nop
blk
... do200k=1,j
                        pdi 1
st200n1s:               dac k
                        lac j
                        dac st200n1-1
                        jmp .+3
                        ..
st200n1:                ..1
... 200type10,a(k)
st200:.                 law st10+1
                        jda tof
                        lac a[k]
                        nop
                        law st200n1
                        jda dof
                        lac st200n1s
blk
... c                   spaces and tabs may be used freely
... goto1
                        jmp st1
... end
strf:.                  .. ; ..
laf:.                   dec 201
blk
j:.                     ..
m:.                     ..
k:.                     ..
temp:.                  .. ; ..


c1:.                    oct 0; oct 0
fin.
```

Normally the DS tape need not be listed; however if you have errors and can't find them by examining the FORTRAN coding it may be useful to have a listing of the DS tape.

Following is the Decal Map produced by DECAL after compilation

```
Decal Map

ssd

mainprog            0000

ps

a                   0003      (a pointer to the array "a")
st1                 0004      (location of statement number 1)
st2                 0011
st3                 0022
st10                0042
st20                0056
st50                0061
st100               0070
st110               0101
st115               0114
st140               0126
st150               0137
st160               0151
st200               0172
strf                0202      (temporary storage used by "if" statements)
laf                 0204      (contains size of array area)
j                   0205
m                   0206
k                   0207
temp                0210
c1                  0212      (floating point constant)

fin                 0300
```