



# DECUS

## PROGRAM LIBRARY

DECUS NO.	82
TITLE	FORTRAN FOR THE PDP-1 VERSION 3
AUTHOR	
COMPANY	
DATE	November 1965
FORMAT	FORTRAN

## FORTTRAN for the PDP-1 - VERSION 3

Version 003 is for machines with mul div hardware so be sure that the mul/mus switch is set to mul and the div/dis switch is set to div.

This version is to replace version 002. One of the more important changes is that there is a check for mixed mode arithmetic or "if" statements (mixed mode means that fixed point and floating point variables have been mixed in one statement). Also, a couple of bugs that were found in versions 001 and 002 have been fixed. One of the bugs that was found in the earlier versions was caused by the "log10f" function appearing as the first symbol (other than special characters) in an arithmetic or "if" statement. In the previous versions the statement was compiled as though it was in fixed point mode which was incorrect.

Another bug was in the "format" statement. In the previous versions if there were blanks following a format specification they were incorrectly used as zeros. For example the format statement "23 format (f10.3,a3 ,f4.1)" has a blank after the "a3" specification so the blank caused the specification to be "a30".

This version includes the FORTRAN "print" statement and is for use with a DEC type 64 line printer. For systems that do not have a printer the following change must be made... The tape labeled "subroutine package" has a clear print buffer instruction (730445) in location 4527. This must be changed to a nop instruction (760000).

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.

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

Tabs and spaces may be used freely.

Redundant carriage returns may be used freely.

Backspace causes the entire line to be ignored.

### 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. There may be a minus sign between the e and the exponent.

A floating point number may never exceed  $1 \times 10$  to the  $\pm 38$ th power.

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.

Names of variables 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 should be okay 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.

Parentheses or brackets may be used interchangeably.

ARITHMETIC STATEMENTS follow the general rules of the FORTRAN language.

Examples of arithmetic statements...

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

The following functions are provided for use in arithmetic expressions.

```
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

Names of variables may not end with the letter "f".

Mixed mode is not allowed.

```
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
```



## 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 used 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.

## "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.



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.

### "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 in the flow of statev being executed, and is only a program terminator, it may appear by itself on a line.

## "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 appear in a dimension statement before it appears 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)
```

## INPUT/OUTPUT

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.

print 683,a,i,x(3,j)

The variables on the list will be printed according to format 683.

NOTE ... The printer provided for in this version is a DEC type 64 with FIO-DEC character set.

accept 74,1,a(34),k(j)

The variables on the list will be accepted from the typewriter according to format 74.

read flex 85,1,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.

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.

### FORMAT

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

Example of a format:

35 format (a5,16,f13.6,3x,12)

### "i" specification

The format specification i10 may be used to output a number which exists in the computer as an integer quantity. 10 positions are used 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

### "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 output 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.

## "c" specification

The "c" specification is used for control of the printer paper advance. The "c" is followed by a number which indicates which track of the carriage control tape is to be used to stop the paper advance. If the standard carriage control tape is used paper advance will be as follows

c0 = track 1	1 line spacing
c1 = " 2	2 line spacing
c2 = " 3	3 line spacing
c3 = " 4	6 line spacing
c4 = " 5	11 line spacing
c5 = " 6	22 line spacing
c6 = " 7	33 line spacing (half page)
c7 = " 8	66 line spacing (full page)
c8 or greater =	no paper spacing (overprint)

The paper advance control is reset to 1 line spacing after every print operation. If no "c" specification is given the printer will give 1 line spacing.

The "c" specification controls the paper advance after the current line is printed so if the format "34 format (c7,f10.3,i17)" was used the "f" field and the "i" field would be printed on one line and the "c" specification would cause the paper to eject to the top of the next page.

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 re-scanned 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 preceding 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. The exception to this is the print statement. Characters after the 120th are lost and will not be printed.

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.

#### 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 - "lss" tape of symbols for linking the subroutine package to a 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 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

Page missing from original document



13 "if" statement has illegal path.  
14 Illegal sense switch number > 7  
15 Non-specific error in 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 Error in "sense light" statement (light > 99)  
42 Over 10 digits to right or left of decimal point in floating point  
constant.  
43 Exponent overflow in floating constant routine.  
44 arithmetic or "if" statement is mixed mode.  
45 illegal subscript in arithmetic or "if" statement.  
46 variable name begins with digit.

The only time when 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:

```
200      do 200 k=1,300
c         do 200 j=1,34
          continue
          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:

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

```
100      a=b-c      Error 26 - Terminating statement number of a "do" loop
                    has been found previously.
```

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

Examples 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.
```

"dda" type errors (found by DECAL) may indicate one of the following

1. Duplicate statement number.
2. Variables name begins "st" followed by digits the same as a statement number.
3. Variables name begins with "c" followed by digits. The floating point constants generated by the compiler are named "c1" through "c50"

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.

140005 = mul div switches not on (up)

140006 = negative number being floated (floating constants routine ) = compiler error

If a parity error in the input FIO-DEC tape is found the letters "fpe" are typed out and the compiler stops. Pushing continue will have no effect.

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 exceptions to this are:

1. The name may have been used legally in a DECAL insert.
2. A sense light number > 6 may have been used. Sense light numbered up to 99 may be used. These will appear as fNf on the DECAL map where N is the light number.

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.

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.

140001	mul div switches not on [up]
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.
140007	Overflow - "i" input
140010	printer error status bit is on - push continue twice and status is rechecked
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 some of the routines in the FORTRAN subroutine package.

## SPECIAL FEATURES

A clear print buffer instruction is executed at the start of the FORTRAN program execution so if the start button on the printer is not on the program will hang up until the button is on.

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.

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 input/output routines 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 no parity change.  
Entrance to pcf is a jda pcf

Subroutine wrp' will punch the right 6 bits of the IO with correct parity.

If a stop code is read there is a one word loop at system symbol eof'. If you wish to provide for stop codes, in your program, deposit an address in eof.

Example:

→ law st100; dap eof

If a stop code is read the program transfers to statement 100.

## 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 a 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.

The program flags are all used by the various subroutines so if a program flag is used in a decal insert its setting will probably not be the same after any fortran statement.

To reference a fortran statement in a decal insert precede the number by the letters "st".

```
→                jmp st555                ... will cause a transfer to statement  
                                     555
```

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 end

xsf abs
absf dig      beg lv7 op1 rs1
               fde; spa
               fde lst; cma          end

iabsf esy absf

xsf / x
x dig         beg lv6 op2 rs1 cmt
               jda fsy; bcl.imp.
               lac 2 lst          end
/ dig         beg lv6 op2 rs1
               jda fsy; bcl.idv.
               lac 2
               fde lst; hlt      end

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

xsf 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 ide 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 ty1 tyo dig .

```

fix fin.

DECAL symbol table (modified version for FORTRAN system)

```

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
mpy 540000; dvd 560000

```

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
x	/	iabsf	sinf	cosf	sqrtf	atanf	acosf	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
c          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
  
```

```

150          go to 100
           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
           go to 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)

a:..           jsp i2f
                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

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

st3a:
blk

... do20j=1,100
st20n1s:
pd1 1
dac j
jmp →+3
dec 100
..1

st20n1:
... accept10,a(j)
law st10+1
jda tif
dac a[j]
nop

... 10format (f10.5)
st10:..
jmp st10a
..661205

st10a:
nop
... c
... if(a(j))20,50,20
parentheses and brackets may be used interchangeably

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

```

```

... 140temp=a(m)
st140:.
... a(m)=a(k)
... a(k)=temp
... goto100

... 150continue
st150:.

blk

... type160,j

... 160format (1hj,i4/8h ordered)
st160:.

st160a:
blk
... do200k=1,j

spa
jmp st150
sza'
jmp st150

efm 2
a[m]=>temp
a[k]=>a[m]
temp=>a[k]

lfm
jmp st100

law st150n2
jda dof
lac st150n2s
law st150n1
jda dof
lac st150n1s

law st160+1
jda tof
lac j
nop

..700001
..417676
..710400
..210000
..700010
..004651
..646551
..656476

nop

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              0202

(contains size of array area)

j                0204

m                0205

k                0206

temp             0207

c1               0211

(floating point constant)

fin              0277

## PROGRAM ERRORS NOT DETECTED BY THE COMPILER

The following errors have been found in programs being debugged by users of the FORTRAN system.

(1) a fortran statement

```
relg(8)= expf(y(2) × 2.308)
```

The variable "y(2)" was intended to be "y2" however there was a variable "y" in the program so "y" was not typed out on the list of aps during the pass through DECAL. Undimensioned variables that appear with a subscript are only detected if they are being defined as fortran variables. Variables are defined only by being to the left of an equals symbol, in an input list, in an "assign" statement, or appear in a "dimension" statement.

## COMPILING and LOADING the FORTRAN SYSTEM

The compiler is in 3 segments. Segment "A" consists of 3 DS (DECAL symbolic) tapes which are compiled by the standard DECAL compiler into 1 DL (DECAL Load) tape which should be labeled "segment A, DL". Segments "B" and "C" are each 1 DS tape and they should be compiled by the standard DECAL compiler and the DL tapes labeled "segment B, DL" and "segment C, DL".

To load the DL tapes

1. Read in high linking loader
2. Load "segment A,DL" which has an org 0000
3. Load the constants (sw.4 up, continue)
4. Punch off an "lss" tape of the symbols (see page 62 of the DECAL manual for a writeup on linking programs that overlap the linking loader)
5. Read in low linking loader.
6. Load the "lss" tape punched off at step 4 above.
7. Load "segment B,DL" with the org following the constants of segment "A"
8. Load "segment C,DL"
9. Load the constants (sw.6 up, continue)
10. Punch off an "lss" tape of the symbols
11. Punch off segments "B" and "C" using any punch off routine that does not extend below 7000

12. Read in high linking loader
13. Load the "lss" tape produced by step 10 above
14. Load segment "A". The system symbols "bad", "jps", and "start here" are typed out as "dda" type errors, this may be ignored as long as the address typeouts match each other.
15. Load the constants (sw.6 up, continue)
16. Load the punch off tape produced by step 11 above
17. Punch off the entire linked program using any punch off routine that does not extend below 7000

To compile and load the subroutine package:

The subroutine package consists of several tapes. Compile the tapes labeled "s-1" and "s-2" (DS tapes) into 1 DL tape which should be labeled "s". Compile the tape labeled "s-3" and label it "imp, idv, tpo for subr. package".

1. Read in low linking loader.
2. Set the org in the Test Word switches to 4437 and set sw.2 up. (the org may be changed if there are changes made to the subr. package, however the final address type out must be "nxt 7775" when the subroutine package is loaded.)
3. Load the tape labeled "s"
4. Put sw.2 down and load the tape labeled "imp, idv, tpo for subr. package"
5. Load the "flip" libetape (sw.5 up while loading the libetape)
6. Store the constants (sw.6 up, continue)
7. Punch a "lss" tape of the symbols and label it "lss tape of symbols for linking subroutine package"
8. Punch off the subroutine package (4437-7777)

NOTE...For systems that do not have a type 64 printer the following change must be made to the subroutine package before compiling tapes "s-1" and "s-2". On the DS tape "s-1" there is a clear print buffer instruction (730445) located 2 words before the system symbol "arf". This must be changed to a nop instruction (760000).

To create a new FORTRAN system libetape:

1. Read in "libetape maker" (standard DECAL system tape)

2. Load the following DL tapes (see page 53-54 of the DECAL manual for instructions for operating the "libetape maker").

```
atanf
logf log10f (1 DL tape)
sinf cosf (1 DL tape)
asini' acosf (1 DL tape)
expf
sqrtf
errsqf
errlgf
eff
fdf
cff
prf
wrf
rdf
paf
```