516-51
HL
9/29/71

FSNAP User's Guide

Introduction

FSNAP is a file oriented simple numerical arithmetic processing language designed to run in a time-sharing environment on a virtual memory multi-programmed small computer. It has been designed with the following concepts in mind:

(1) easy to use.

(2) quick response.

(3) good debugging aids.

(4) flexible and powerful input/output capabilities.

(5) possibility of running long jobs in "batch" mode.

The language has been implemented using the existing file structure, supported by the TSS-516 multi-programming operating system, for the storing of programs, input data and output data.

The FSNAP language structure is very similar to BASIC but does not require a statement number for each statement in the program and is not limited to two-character variable names. Statement numbers are used in the same manner as in FORTRAN. Input statements allow for free-formatted input of numbers from either the user's terminal or a file specified at "run" time. Output statements are flexible, allowing the user to specify output format in detail. Output may be directed either to the user's terminal or to a file which is specified at "run" time and created automatically if it did not exist previously.

FSNAP programs to be run are compiled into executable code in one pass, giving error diagnostics if any errors are encountered. The user may then call the system EDITØR to correct the error and recompile the program. The program is then run in an interpretive mode again giving error diagnostics if any errors are detected.

Writing a Program

Programs may be written using the system EDITØR (see 516 document #30). A program consists of a series of specification statements and executable statements. Any number of statements may appear on one line provided the statements are separated by semi-colons. Comments may appear in a program provided they are preceded by a backslash. Executable statements consist of an optional statement number, a macro command word and a subsequent series of parameters which are interpreted according to the type of command. The syntax of all statements is "free-format", i.e., all extraneous spaces are ignored. An example of a program follows: (user response underlined)

```
PRØGRAM? F↵
PRØGRAM FILE? ↵
FSNAP-E↵
*EDIT
+PROG↵
I↵
\SAMPLE PRØGRAM↵
SUM=0                    \INITIALIZE↵
FØR I=1,100              \SET UP LØØP↵
SUM=SUM+I/(I+1)↵
NEXT I↵                  \END ØF LØØP↵
TYPE "SUM=" %8.2 SUM↵
STØP↵
↵
↵
X↵
FSNAP-G↵
SUM = 95.80
FSNAP-X↵
PRØGRAM?
```

Variables

Variable names may consist of any number of alphanumeric characters with the restriction that the first character must be an alphabetic character. Only the first four characters are used to identify the variable; all succeeding characters in the name are ignored during compilation. No distinction is made between upper and lower case letters. Up to 192 different variables may be used in any FSNAP program.

e.g., allowable names AB32, AB3XF, A, ABCDEFG

illegal names 13, 2A, A!3

Dimensioned Variables

Any number of dimensioned variables may be used in the program with any number of dimensions on each array. A one-dimensional array of size N has N elements from 1 to N. Dimensioned variables can have any name allowed for variables, as discussed above, and may be referenced by an index expression consisting of integer constants, positive integer variables or any combination thereof separated by the addition or subtraction operators.

e.g., ABC(I), AB(3), XY(I+3,J-3), A(I+J-K+2)
are allowable dimensioned variables.

ABC(2*I), XY(0)
are illegal dimensioned variables.

Numbers

Numbers may be expressed as:

	integer	5, 1256, -2
	fixed	12.32, -0.325
or	E-format	12E4, 3.6E-2

Internally all numbers are represented in the floating point format, i.e., 2 words per number - 1-bit sign, 8-bit exponent and a 23-bit normalized characteristic. A number is accurate to one part in eight million, i.e., to about 7 decimal places. The range of numbers which can be represented is

$$-1.7E38 < \text{number} < 1.7E38.$$

Statement Numbers

Statement numbers are only required when a transfer may be made to the statement in question. They are to be distinguished from line numbers used by the system TEXT EDITOR. Any non-zero, positive integer from 1 to 32767 may be used as a statement number.

Operators

The following operators are used in the FSNAP language:

+	addition
-	subtraction
*	multiplication
/	division
↑	exponentiation.

System Defined Functions

The following functions are recognized by the FSNAP compiler. These names may not be used as variable names.

ABS	absolute value	
NEG	negative value	
SQR	square root	
SIN	sine	angle expressed in radians
CØS	cosine	
TAN	tangent	
CØT	cotangent	
ATN	arctangent	
EXP	exponential (base e)	
LGT	log (base 10)	
LØG	log (base e)	
SGN	sign (-1, 0, +1)	
INT	integer part	
RND	random number	

User Defined Functions

The user may define up to 10 functions. The name of the user function may be any legal variable name not used previously as a variable or as a system defined function.

e.g., COSH(X) = (EXP(X)+EXP(-X))/2

Expressions

During the compilation phase, the reverse polish string for the expression is generated. The string is evaluated during execution time. Any arbitrary level of nested parenthesis are allowed in an expression. The priority of the operators from low to high is as follows:

+, -
*, /
↑
negative unary operator, system function, user function.

Note that: (1) expressions inside parenthesis are computed before being used in further computation.

(2) expressions involving operators of equal priority are evaluated from left to right.

Some examples of allowable expressions are:

```
DX(2) * ((A+1)/B+3.2) + X↑2) + 31

AX3↑2 + LØG(SIN(B(I)+A)÷3)
```

Specification Statements

Specification statements are not executed during program "run" time.

DEF defines a user function. Up to ten user functions may be defined. The name of the function may be any legal variable name not conflicting with previously defined functions or system functions or a variable name already used. DEF statements must occur before the functions are actually used in an expression. The variable used as the parameter in the user function must be a one-letter alphabetic character.

```
e.g., DEF COSH(X) = (EXP(X)+EXP(-X))/2
      DEF DEL(D) = 1 + SQR(1+D*V)
```

The functions may be used in an expression as any system function is used.

```
e.g., 5*COSH(DEL(2*STP)+1) - 4*DEL(STP)
```

DIM sets aside virtual memory space for dimensioned variables. The largest linear array which can be specified consists of 1891 elements. However, any number of array variables may be specified provided the total number of elements does not exceed 1891. Each array may have any number of dimensions. DIM statements must occur before the dimensioned variables are actually referenced in the program.

```
e.g., DIM XYZ(3,3,12), ABC(20), D3(3,200)
```

DATA enters data into a data file in the two-word floating-point format so as to be directly readable by the corresponding READ statement. As many DATA statements may be used as required; however, for a large data file it is usually more convenient to put the data directly into another file. The DATA statement may occur anywhere in the program. For further discussion, see the section on I/Ø statements.

```
e.g., DATA 35, 36.2, 0.0, 52, 41E-3
      DATA 3, 4, 5, 6, 23
```

Executable Statements

Executable statements are compiled during the compilation phase and the object code produced is executed at run time. If errors are detected during compilation, the object code is stuffed with an error code to ensure that execution is never attempted beyond the error.

Arithmetic Statement

Arithmetic statements consist of setting a variable equal to an expression. No macro word is required for this purpose.

```
e.g.,    ABC = DX1*(A(I+1)*(B+3.2)+X)+30
         A(I+2) = A(I)*CX(I+2)+2
     300 IX = 30+I
```

Logic Statements

GØTØ - used to make an unconditional transfer
GØ TØ to a statement other than the one following directly in the program. A space between the two words GØ and TØ is optional.

```
e.g.,    GØ TØ  56
           -
           -
         56 I = I+1
```

STØP - used to stop the execution of a program. There may be more than one occurrence of this statement within a program. The use of STØP as the last statement in a program is optional.

IF - used to provide a two-or three-way conditional transfer and/or execution of a statement.

(A). Three-way IF statement

IF (expression) 10, 20, 30 -

- if expression is less than zero transfer to statement 10
- if expression is equal to zero transfer to statement 20
- if expression is greater than zero transfer to statement 30

(B). Conditional IF statement

If the condition of the IF statement is met, the statement immediately after the IF statement is executed. Otherwise the subsequent statement is executed. Six different relations between two different expressions may be tested for:

=	equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
<>	not equal to

e.g.,
```
IF (A(I)*3=B*2)GØTØ 35
IF (A <> I)I=A
IF (B↑2 > 25)RETURN
```

Loop Statements

FØR - used to execute statements within a loop a specified number of times.
- FØR I = A,B,C will cause the program to loop on the variable I where:

  I - is any non-dimensioned variable.

  A - is an expression to which I is initially set.

  B - is an expression giving the final value of I in the loop.

  C - is an expression giving the increment which is to be added to I at each step through the loop.

- If C is not specified, a default value of one is taken. The expression for C must be a non-zero positive or negative value.

NEXT - The NEXT statement is the last one in the loop and causes processing to repeat the loop or continue sequential execution if the value of the loop parameter plus the increment is outside the range specified by the initial and final values calculated in the corresponding FØR statement, i.e., the loop is entered again only if:

$I = <B$ and $C > 0$

or, $I = >B$ and $C < 0$.

- Nested loops up to 9 levels deep may be used in a program.

```
e.g.,  FØR I = 1, 5*SQR(B),0.5
       FØR J = 1, 4
         -
         -
       NEXT J
       NEXT I
```

- The following is an example of an illegal use of FØR/NEXT.

```
       FØR I = 1,10
       FØR J = 2,-10,-1
         -
         -
       NEXT I
       NEXT J
```

Subroutine Statements

GØSUB - transfers execution of the program to an internal subroutine starting at the specified statement number.

-- e.g., GØSUB 100

RETURN - returns execution of the program to the next statement following the GØSUB statement used to call the subroutine.

- Subroutines may be written up to three levels deep.

```
e.g., X = 35
      GØSUB 100
      -

 100  Y = INT(X*3.14)
      IF (Y > 1000) Y=1000
      GØSUB 200
      -
      RETURN
 200  -
      -
      RETURN
```

CALL - used to call a machine language subroutine from an FSNAP program. Up to three arguments may be passed between the calling program and the subroutine. An argument may be any one of:

(1) a number
(2) a variable
(3) a dimensioned variable

If more than three arguments are required for a subroutine, they may be passed through a dimensioned array.

```
e.g., A(1)=ARG1; A(2)=ARG2;---; A(N)=ARGN
      CALL SUBRX (A(1),X)

      CALL FSRAND(X)
```

Any number of machine language programs may be called from an FSNAP program. Machine language routines called from FSNAP have two exits, RET1 for an error return and RET2 for the normal return.

A number of routines have been written which are directly callable from FSNAP. These include FSTIME, FSRAND and FSCHAR.

CALL FSTIME - will result in the date and the time of day being printed out at the user's terminal. An FSNAP user may wish to use this to time certain computation sequences.

CALL FSRAND(RAN) - will generate a random number between 0.0 and 1.0. The user must pass an argument between 0.0 and 1.0 through RAN. The random number generated is passed back to the user in RAN.

CALL FSCHAR(X) - permits the user to input any one character from his terminal. The octal code for the character is converted to a floating point number.

```
e.g., 1 => 49.0
      A =>. 65.0
      a => 65.0
```

This allows the user to test the character against any specified character for a match. String characters in FSNAP are specified by using the dollar sign. For instance, $A is compiled as the floating point number 65.0. The following set of FSNAP statements can be used to test for the answer "YES" or "NØ" to a question:

```
20   TYPE! "DØ YØU WISH TØ CØNTINUE?"
     CALL FSCHAR(X)
     IF(X=$Y)GØTØ 30
     IF(X=$N)STØP
     GØTØ 20
30   SUM=0
     -
```

Other system subroutines may also be written in machine language code in order to provide specialized user routines. Refer to a subsequent TSS-516 document for details on how to interface these routines to FSNAP.

Input/Output Statements

ASK	- input data from user terminal.
READ	- input data from file (or user terminal).
TYPE	- output at terminal.
WRITE	- output into a file (or user terminal).
RESTØRE	- restore data file input pointer.
DATA	- put data into a file.

An FSNAP program may use up to 6 different input and/or output files during execution of the program. The names of the files are specified at "run" time. In the program the files are specified by a file number from 1 to 126 enclosed in parenthesis. The file number is optional, however, if only one input or output file is required.

The input format of numbers may be integer, fixed or E-format. The output format can be specified in detail by the use of certain parameters listed below. These parameters may be used with the ASK, READ, TYPE and WRITE statements.

"TEXT STRING" output text string
! output a carriage return, line feed
# output a carriage return only
$ output one space
$N output N spaces where N may be any positive integer (e.g., $25).
%N.M set format of numbers to N digits total (including leading spaces but excluding the sign and decimal point) and including M digits after the decimal point, (e.g., %8.2).

e.g., TYPE "RESULT=" %5.2, (A*2-3)*100 "PERCENT"
would produce
RESULT = 11.00 PERCENT

ASK - will always request input from the user's terminal. Input of a number may be terminated by either a carriage return or a space, e.g.,

```
ASK !! "ANGLE(DEG.) =" ANG
```

TYPE - will always output the specified parameters at the user's terminal. Note that carriage return, line feed combinations may also be included in text strings to be output, e.g.

```
TYPE !!"TITLE
SUB-TITLE
HEADING"
```

READ - generates a request for an input file at "run" time. For example, READ(35) will type out: INPUT FILE 35? to the user. In response the user may give the name of the file which contains the data. The response:

```
DFL,3,30
```

will convert the numbers in the file DFL starting at line number 3 and ending at line number 30 to a two-word format in a temporary data string. If the first and last line numbers to be used in the file are not specified, line number 1 will be assumed for the starting line number and line number 9999 will be taken as the default value for the ending line number. The format of the input data file is very flexible, i.e., numbers may be separated by spaces, commas, semi-colons or carriage returns and line feeds. A carriage return in response to

```
INPUT FILE 35?
```

will result in input being taken from the user terminal in the same manner as the ASK statement. However, upon a subsequent execution of the program, a carriage return will result in data being taken from the previously specified file again, if a data file had been specified previously.

DATA - will cause data to be put in the specified file during the compilation phase.

- e.g., DATA (35) 3,4,5,25.3

will result in the four numbers being put in file number 35. During program execution the proper response to

INPUT FILE 35?

would be a carriage return unless one wished to take input from another data file.

RESTØRE - this command will restore the data pointer to the beginning of the data file.

```
e.g.,  READ (35) A,B,C,
       RESTØRE (35)
       READ (35) D,E,F
       DATA (35) 1,2,3
```

would produce the results:

```
A=D=1
B=E=2
C=F=3
```

WRITE - generates a request for

ØUTPUT FILE?

at "run" time. The user may specify the name of a file in which he wishes the output to be stored. If the file does not exist, it will be created automatically. If the file already exists, output will be put at the end of the file. No information will be overwritten or destroyed. A carriage return in response to the ØUTPUT FILE? question will result in all output being directed to the user's terminal for the file number specified

FSNAP Executive

The FSNAP subsystem consists of an executive itself, for the purpose of executing FSNAP commands as well as calling any program which can be called from the PRØGRAM? level. One may enter the FSNAP executive in the following ways:

(1) PRØGRAM? FSNAP,PFL↵
FSNAP-

(2) PRØGRAM? F,PFL↵
FSNAP-

(3) PRØGRAM? F↵
PRØGRAM FILE? PFL↵
FSNAP-

(4) PRØGRAM? F↵
PROGRAM FILE?↵
FSNAP-

where all user responses are underlined. The file PFL contains the program to be run. Giving only a carriage return at the FSNAP-level will produce a menu of the one letter commands which can be given at the FSNAP-level as shown on the next two pages, where a typical user interaction is portrayed. Here a program is written, corrected for an error and then executed. The value of the variables used in the program are printed out. At any time output may be stopped by hitting the break key. During computation, execution of the program may be stopped by hitting CNTRL C. This will bring the user back to the PRØGRAM? level. A back-up deck is produced on the GE as well as a listing by invoking the GEBKUP and GELIST programs. Examining the STATUS of the TSS-516 system periodically will indicate when the jobs on the queue have been transmitted to the GE-635.

```
PASSWØRD? HL↵

PRØGRAM? F↵

PRØGRAM FILE?↵
FSNAP-↵


GENERAL FØRM ØF CØMMAND - X,8%8.2,VNAM
8%8.2  FØRMAT SPECIFICATIØN (ØPTIØNAL)
VNAM   VARIABLE NAME (ØPTIØNAL)

C LIST CØMMANDS
D PRINT DATA FILE
E GØ TØ EDITØR
F LIST FUNCTIØNS
G GØ EXECUTE PRØGRAM
N GET NEW PRØGRAM FILE
Ø DUMP CØMPILED CØDE
P PRINT VARIABLES
X RETURN TØ EXECUTIVE

FSNAP- E↵

*EDIT
+DEMØ↵ <—— CREATE PRØGRAM FILE
I↵  <—— INSERT MØDE
\ SUM ØF TWØ SINE WAVES↵
ASK ! "AMPLITUDE ØF SINE WAVE ØNE =" AMP1↵
ASK ! "AMPLITUDE ØF SINE WAVE TWØ  Ø=" AMP2↵
TYPE !!! " ANGLE AMPLITUDE"↵
FØR ANG = 0,360,30↵
RAD=ANG*3.1416*Ø/180↵
TYPE ! %4.1  ANG  %7.4  AMP1*SIN(RAD)+AMP2*SIN(RAD/2)↵
NEXT I↵
↵ <—— CARRIAGE RETURN TØ GET BACK TØ EDITØR CØMMAND LEVEL
X↵
UNMATCHED FØR/NEXT, LINE NUMBER     8

SNAP- E↵

*EDIT
P8↵
008 NEXT I
D8↵
I↵
NEXT ANG↵

P1,9↵
001 \ SUM ØF TWØ SINE WAVES
002 ASK ! "AMPLITUDE ØF SINE WAVE ØNE =" AMP1
003 ASK ! "AMPLITUDE ØF SINE WAVE TWØ =" AMP2
004 TYPE !!! " ANGLE AMPLITUDE"
005 FØR ANG = 0,360,30
006 RAD=ANG*3.1416/180
007 TYPE ! %4.1  ANG  %7.4  AMP1*SIN(RAD)+AMP2*SIN(RAD/2)
008 NEXT ANG
009 <EØF>
X↵
```

Ø DELETES ØNE CHAR.

CØRRECTIØN ØF ERRØR

```
FSNAP- G↵

AMPLITUDE ØF SINE WAVE ØNE =4.5↵
AMPLITUDE ØF SINE WAVE TWØ =2.25↵


 ANGLE AMPLITUDE
   0.0    0.0000
  30.0    2.8323
  60.0    5.0221
  90.0    6.0910
 120.0    5.8457
 150.0    4.4233
 180.0    2.2500
 210.0   -0.0767
 240.0   -1.9486
 270.0   -2.9090
 300.0   -2.7721
 330.0   -1.6676
 360.0    0.0000
FSNAP- P,%8.4↵

AMP1 =     4.5000
AMP2 =     2.2500
ANG  =   360.0000
RAD  =     6.2832
FSNAP- GEBKUP,DEMØ↵ <---- BACK-UP DECK ØN GE-635

GE JØB ACCEPTED  7B157    09/28/71AM 08:53:49
FSNAP- GELIST,DEMØ↵ <---- LISTING ØN GE-635

GE JØB ACCEPTED  7B158    09/28/71AM 08:54:05
FSNAP- STATUS↵

516 TSS STATUS 09/28/71AM 08:54:15
     2 158/02    1467        0 09/20/71AM     698    2612     404 101434 110745

 USERS GEDATA AVLDSK LSTERR SRTINGDATE ACCESS CNTIME DSKERR RNGERR LSTERR

FSNAP- X↵                    (TWØ JØBS ØN
                              GE-QUEUE

PRØGRAM? U↵

516 TSS USERS  09/28/71AM 08:54:57
          GE GREM  .GESWT    200    116
          CØN TTY  SYSNAP    600    316
TØTAL CØRE IN USE  2034  /  7661

PRØGRAM? S↵

516 TSS STATUS 09/28/71AM 08:55:20
     1 158/00    1482        0 09/20/71AM     698    2612     404 101434 110745

 USERS GEDATA AVLDSK LSTERR SRTINGDATE ACCESS CNTIME DSKERR RNGERR LSTERR

                              JØBS HAVE BEEN
PRØGRAM? U↵                   SENT TØ GE-635

516 TSS USERS  09/28/71AM 08:55:47
          CØN TTY  SYSNAP    600    316
TØTAL CØRE IN USE  1116  /  7661
```

FSNAP One Letter Commands

In all cases use of the two arguments following the one letter command (separated by commas) is optional. However use of the arguments gives the user better control of the results.

C - lists the commands available to the FSNAP programmer. No arguments need be specified.

D - prints out contents of specified data file.

e.g., D,8%6.2,23

will print out the complete contents of data file number 23 in the format %6.2 at 8 words per line.

E - control is transferred to the editor, with the editor linked to the program file attached to FSNAP if one has been specified. No arguments need be specified. The current compiled code is lost. Upon return from the editor, the program file which is currently attached will be compiled and ready to run.

F - lists the system functions which are available to the FSNAP programmer. No arguments need be specified.

G - execution of the FSNAP program is begun. Output format may be specified in the first argument. However, if a format specification is used within the FSNAP program, it will over-ride the argument used. At any time output and hence execution may be stopped by hitting the break key. Control will return to the FSNAP-level. Execution of the program during straight computation (no output) may be stopped by hitting CNTRL C. This will bring the user back to the PRØGRAM? level.

N - allows a user to run a new (i.e., another) program file. No arguments are specified. The user will then be required to answer the PRØGRAM FILE? query.

Ø - dumps the code compiled for each line of program. One may selectively dump certain lines by specifying the first line to be dumped in argument number one and the last line to be dumped in argument number two.

P - prints out the values of the variables used in the program. The first argument may be used to specify the output format of the numbers. The second argument may be used to give the name of the variable at which the start of printing is desired. The names of dimensioned variables are printed but the elements of the arrays are not.

X - return to PRØGRAM? level.

User Programs called from FSNAP-Level

The following provides a list of programs which are frequently called from the FSNAP-level. A brief description of how to invoke the programs is included. The compiled object code remains intact throughout these CALL's, saving all pointers to the compiled code in the user file, SNP*F.

CØPYFL/PWD/FN - copies a file by the name of FN from the directory of the user with password, PWD, to the directory of the user calling the program. Any previous file in the user's directory by the name, FN, will be overwritten.

DESK - user may enter the DESK Calculator to carry out some immediate computations and then return to the FSNAP program. (See subsequent 516 document for further information.)

EDITØR - allows user to use the TEXT editor without recompilation of the program file. The program file cannot be altered by using the editor in this manner. This gives the user a "back door" into the EDITØR to modify or prepare some input data files or to examine some output files, without modifying the compiled object code.

FLIST - will list the names of the user's files along with the size of each file (in number of disk blocks).

GEBKUP,FN1,FN2,... - allows user to back up as many as 10 files at a time on cards punched on the GE-635 in a compressed format. Here FN1 is the name of one file, FN2 is the name of another file, etc.

FLØAD - allows user to load files from cards produced by GEBKUP into his directory. (See 516 document #43 for further information.)

GELIST,FN1,FN2,... - allows user to obtain a listing of up to 10 files at a time of the GE-635 line printer.

GEPLØT,FN - will generate a STARE plot of the data contained in the file, FN. The parameters for the call to TPLØT are contained in the file. See 516 document #47 for the parameters required and the general layout of the plot file.

PRINT,FN - will print out the contents of the file, FN. Printing may be stopped at any time by hitting the interrupt key.

STATUS - will print out the status of TSS-516 indicating the number of users logged in, the number of jobs on the GE queue as well as the SNUMB of the last job entered, the available disk space left for files and other pertinent information.

SYSNAP - will print a list of the users (by terminal name) on TSS-516 indicating how many words of core (in octal) each is holding down and the total amount of core being held down.

Modes of Running FSNAP Programs

FSNAP programs written in a file may be run in either the time-sharing mode or in the "batch" mode by invoking different control programs. Basically FSNAP programs can be run in one of the four following modes, each subsequent mode invoking a higher level of control. The DEMØ program which was discussed previously will be used as an example here.

(1) F,DEMØ - will compile the program contained in DEMØ and transfer control to the FSNAP executive to wait for the user to issue the G command.

(2) C,DEMØ - will compile and run the program without going through the FSNAP executive. The program will wait for the user to input the data only. Upon completion of the program, control will be returned to the PRØGRAM? level. This mode of running an FSNAP program is useful for completely debugged programs.

(3) I,SRC - will take input from the file, SRC and perform the commands exactly as if received from the user's terminal.

e.g., SRC would contain:

```
F,DEMØ
G
4.5
2.25
X
```

to run the DEMØ program discussed previously.

(4) G,SRC - will generate an "ELF" to run the job contained in the file, SRC and free up the user's terminal for further TSS activities. Input commands are taken from SRC and all output is directed into a file SRC*Ø which is created at run time. An "ELF" may be stopped at any time by means of the program, STØP, ELF13 if ELF13 is the one which has been spawned. All "*Ø" files may be deleted from the user's directory by issuing the "*" command at the program level.

One also has the option of passing arguments to the source file used by the INPUT or SPAWN commands. For instance, one may wish to run the DEMØ program for two different sets of values of AMP1 and AMP2. This can be done by means of:

```
I,SRC,4.5,2.25
I,SRC,4.5,3.0
```

or

```
G,SRC,4.5,2.25
G,SRC,4.5,3.0
```

if one does not wish to wait for the output to be completed before proceeding to some other TSS activity. In this case the SRC file would contain:

```
F,DEMØ
G
#1
#2
X
```

Arguments 1 and 2 will be filled in upon calling the source file mechanism.

Some General FSNAP Usage Conventions

To enable FSNAP users to share programs that have been written to solve general problems certain conventions have been established.

(1) All programs which are self-contained will be stored in files under the password MATHLIB and may be copied into a user's directory by means of the CØPYFL/MATHLIB/FN program and then executed immediately without any editing being necessary.

(2) Subroutines will also be stored in MATHLIB and may be copied into a user's directory and then merged into a user's program either as a linear piece of code or as a subroutine. All variables used in the MATHLIB subroutines will start with the letter Z so as not to cause a conflict of names with the user's variables.

(3) All statement numbers used in the MATHLIB routines will be between the numbers 30000 and 32767 inclusive so as not to cause conflicts with the user's statement numbers.

(4) Some general usage programs which have been written are:

PØLRTS - solve for complex roots of a polynomial using BAIRSTØW's method.

PØLRT - solve for complex roots of a polynomial using the NEWTØN-RAPHSØN method.

ZSIMPSØN - subroutine to perform Simpson's integration on any given function.