516-64
HL
1/27/72

FSNAP Designer's Guide

Introduction

The FSNAP language (Document No. 516-51) has been implemented on the 516-TSS virtual memory time-sharing system. The source code is compiled in a compact format so as to facilitate execution in an interpretive mode. The source code and object code are both treated as linear strings of 8-bit bytes.

The purpose of this document will be to describe the implementation mechanics in enough detail to allow comprehension of the structure of the compiled code generated and the manner in which the compiled code is run interpretively. A few flow charts will be provided to show how all of the various FSNAP program segments are interrelated. Allocation of the user relocatable pointers and the user temporary storage area will be shown to enable a user to write his own machine language application programs callable from FSNAP. The actual structure of the compiled code will then be discussed so that one could conceivably add FUNCTIØN's and MACRØ's other than those that exist at present.

FSNAP Code

The machine language code written for FSNAP can be broken up into five sections, as follows:

(1) FSNAP control routines - these routines interpret the commands (along with arguments) entered by a user and transfer control to the appropriate routine.

(2) FSNAP compile routines - these routines compile the code from a given source file into a compact string of interpretive code.

(3) FSNAP execution routines - these routines execute the compiled code interpretively

(4) FSNAP math routines - these routines perform the actual mathematical calculations on the data and variables.

(5) FSNAP user routines - these routines are callable from an FSNAP program and are special application machine language programs.

A complete list of all of the routines in each section along with their size in octal words follows.

(1) FSNAP Control Routines

Name	Length	Function
CALCUL	57	compile and execute FSNAP program
FSNAP	314	main control routine
GETCOM	172	get one letter command and parameters
FSUNLK	135	unlink all files and data segments
FSLINK	117	link up to all data segments
FSEXEC	174	save state and execute system program
FSNPRT	174	print out values of variables
FSPDAT	162	print out contents of data files
FSDUMP	170	dump octal compiled code
FSNPMN	55	print various FSNAP menu's
	$2060_8 = 1072_{10}$	

(2) FSNAP Compile Routines

Name	Length	Function
FSCMPL	567	main compilation routine
FSCMPX	75	data segment initialization routine
LNCMSU	332	set up ID, char. pointer pairs for statement numbers
FSERSU	473	type out compilation error
GETNAM	257	get code for variable name
FSNXPD	140	expand variable's segment size
DMVSU	136	generate code for dimensioned variable
CALLSU	251	set up code for CALL macro
SETSUP	575	set up reverse polish string for expression
FSUSFN	33	copy user function into object code
FØRSU	174	set up FØR macro
NEXTSU	151	set up NEXT macro
IFSTUP	123	set up IF statement
TYASSU	323	set up ASK, TYPE, READ, WRITE macro's
FUNCSU	165	set up reverse polish string for user function
DATASU	276	set up data in a file
DIMSU	322	set aside space for dimensioned variables
FGTNUM	163	get floating point representation of number
	$5772_8 = 3066_{10}$	

(3) FSNAP Execution Routines

Name	Length	Function
FSNPXC	532	main execution routine
FSNPBR	63	detects DEL hit by user
FSEVAL	373	evaluates a reverse polish string
FSNPFN	166	evaluates a function
FSNPER	363	types out an execution error
DIMVAR	276	obtains address and value of dimension var.
FSNPFR	165	sets up a FØR/NEXT loop
FSNPIF	163	determines IF condition
FSCALL	250	performs CALL to user machine language program
FSNPTP	261	executes ASK, TYPE, READ, WRITE macros
FSREAD	174	reads data from a file
FSINFL	373	asks user for name of input file
FSOTFL	76	asks user for name of output file

$4245_8 = 2213_{10}$

(4) FSNAP Math Routines

Name	Length	Function
FLPIN	73	input a floating point number
FLTDCD	375	decode the number read in
FLPOUT	47	output a floating point number
FLTFMT	544	format the number being typed
FADD	173	add two F.P. numbers
FSUB	176	subtract two F.P. numbers
FMULT	167	multiply two F.P. numbers
FDIVD	174	divide two F.P. numbers
FSQRT	115	take square root of number
DINTGR	106	convert F.P. number to double precision integer
FSNPIP	76	take integer part of F.P. number
FLØAT	47	convert integer to F.P. number
ALØG	46	take $\log_e$ or $\log_{10}$ of F.P. number
LØGCØM	167	common log. calculation routine
CØSINE	262	cosine of angle in radians
SINE	264	sine of angle in radians
TANGNT	56	tangent, cotangent of angle in radians
ATAN	337	Arctangent function
FPØW	120	raise F.P. number to a power
EXP	361	take exponential of a F.P. number
RANDØM	123	generate a random number
	$5113_8 = 2635_{10}$	

(5) FSNAP User Routines

Name	Length	Function
FSRAND	131	generate a random number
FSCHAR	15	input one character
FSTIME	21	output date and time of day
FSFEØF	164	check for end of data file
FSRØND	137	round fraction to N bits
FSTRUN	127	truncate fraction to N bits
	$641_8 = 417_{10}$	

Total Code

(1)	Control	2060	1072
(2)	Compile	5772	3066
(3)	Execution	4245	2213
(4)	Math	5113	2635
(5)	User	641	417
		22273_8 =	9403_{10}

The following three (3) flow charts are attached, showing how all the various routines are interrelated,

(1) Flow Chart of FSNAP Control Routines.

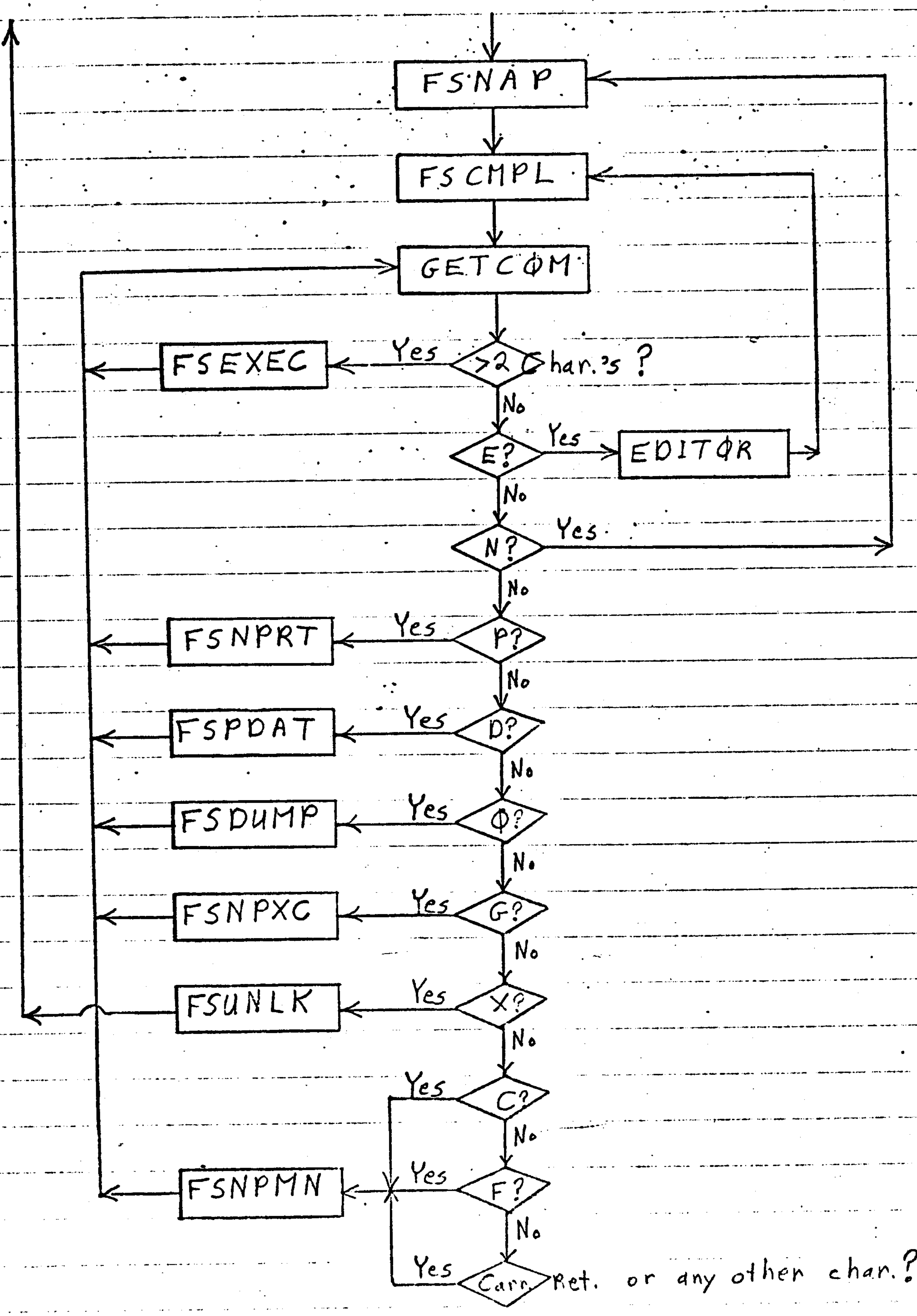
(2) Flow Chart of FSNAP Compile Routines

(3) Flow Chart of FSNAP Execution Routines

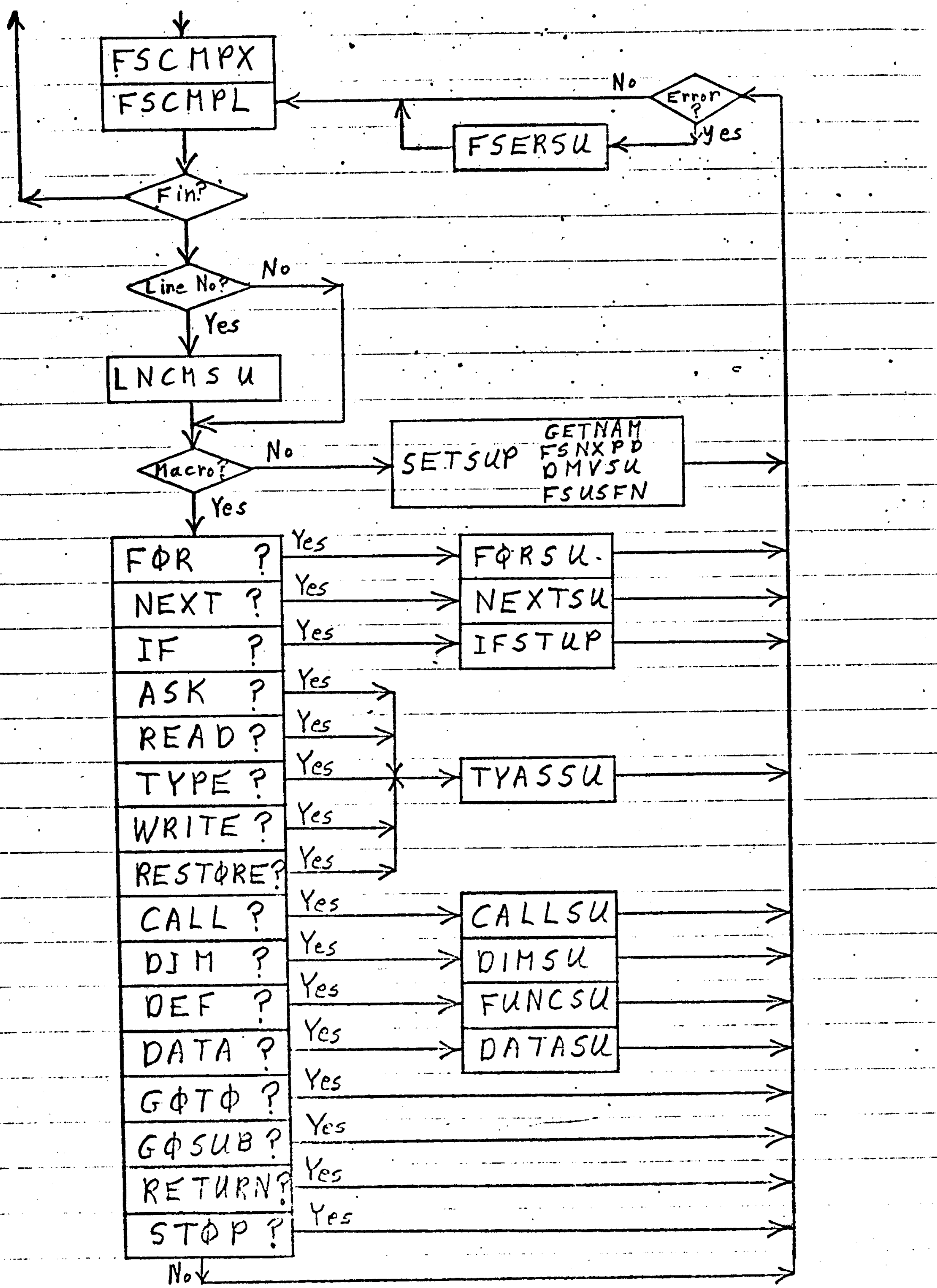
The math routines are discussed in a previous 516 document. They are in fact system callable routines in that they do not use any of the user temporary storage area (.T's) or the user relocatable pointers (.RP's). The routines are completely reentrant and have an error exit besides the normal exit. The user routines are discussed in various 516 documents. These routines may use the user temporary area (.T7 → .T17). The implementation of the CALL macro will be deferred until the allocation of the user .T's and .RP's is discussed.

The user .T's are allocated separately during the compilation and the execution phases. Some are used globally, whereas others are used only in certain program segments and so are available to other program segments. The following two tables outline the allocation of the user .T's during the compilation and execution phases.

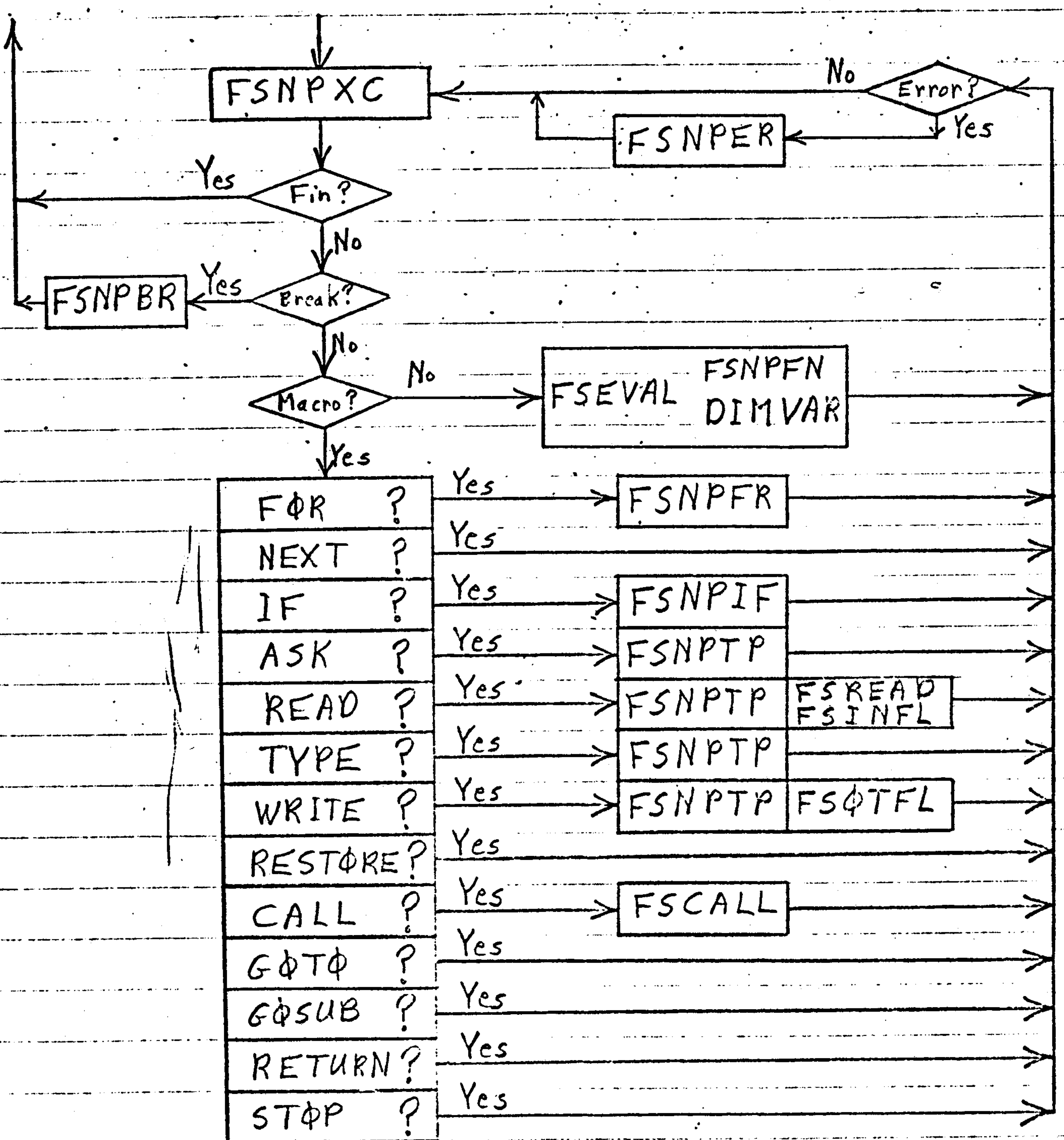
FSNAP Control Routines



FSNAP Compile Routines



FSNAP Execution Routines



Allocation of .T's (Compilation)

.T0	CALLSU,SETSUP,FØRSU,NEXTSU,DIMSU
.T1	SETSUP,FSCMPL,FSCMPX
.T2	program file character pointer
.T3	pointer into dimensioned data block
.T4	last character
.T5	CALLSU,SETSUP,IFSTUP
.T6	SETSUP
.T7	object file character pointer
.T10	LNCMSU,CALLSU,SETSUP,FUNCSU,DATASU
.T11	LNCMSU,CALLSU
.T12	LNCMSU,CALLSU
.T13	LNCMSU,CALLSU
.T14	LNCMSU,CALLSU
.T15	LNCMSU,CALLSU
.T16	CALLSU,SETSUP
.T17	CALLSU,SETSUP

Allocation of User .T's (Execution)

.T0	variable to be stored into
.T1	FSCALL,FSNPTP,FSREAD
.T2	object code character pointer
.T3	highest possible variable code
.T4	last character
.T5	ID ofcurrent dimension data block
.T6	ID of dimension data block required
.T7	operator code
.T10	FSNPTP,FSREAD,FSOTFL
.T11	FSEVAL
.T12	FSEVAL
.T13	format specification word
.T14	FSINFL
.T15	function opcode
.T16	operand storage (high order word)
.T17	operand storage (low order word)

The next several pages cover the allocation of the user .RP's during both compilation and execution. The layout of all of the data blocks is shown in detail to enable one to interface special application user programs to the FSNAP language interpreter.

Allocation of

	Compilation Phase		Execution Phase	
.RP0	Source File	PRGID	Object Code	ΦBJID
.RP1	Data File Set Up *		Read Data File	* **
.RP2	Pointers * Temp. Data		Pointers * Temp. Data	
.RP3	DEF * FΦR Temp. Data		Unused	
.RP4	Program Variables	CΦNID	Program Variables	CΦNID
.RP5	Ptrs. to Dim. Var	* DIMID	Ptrs. to Dim. Var.	* DIMID
.RP6	Object Code	ΦBJID	Dimensioned Variables	*
.RP7	Prog. Var. Names	NAMID	Unused	

* Used only if required

** May be used in between Read Statements

RP2 - Pointers - Compilation Phase

4

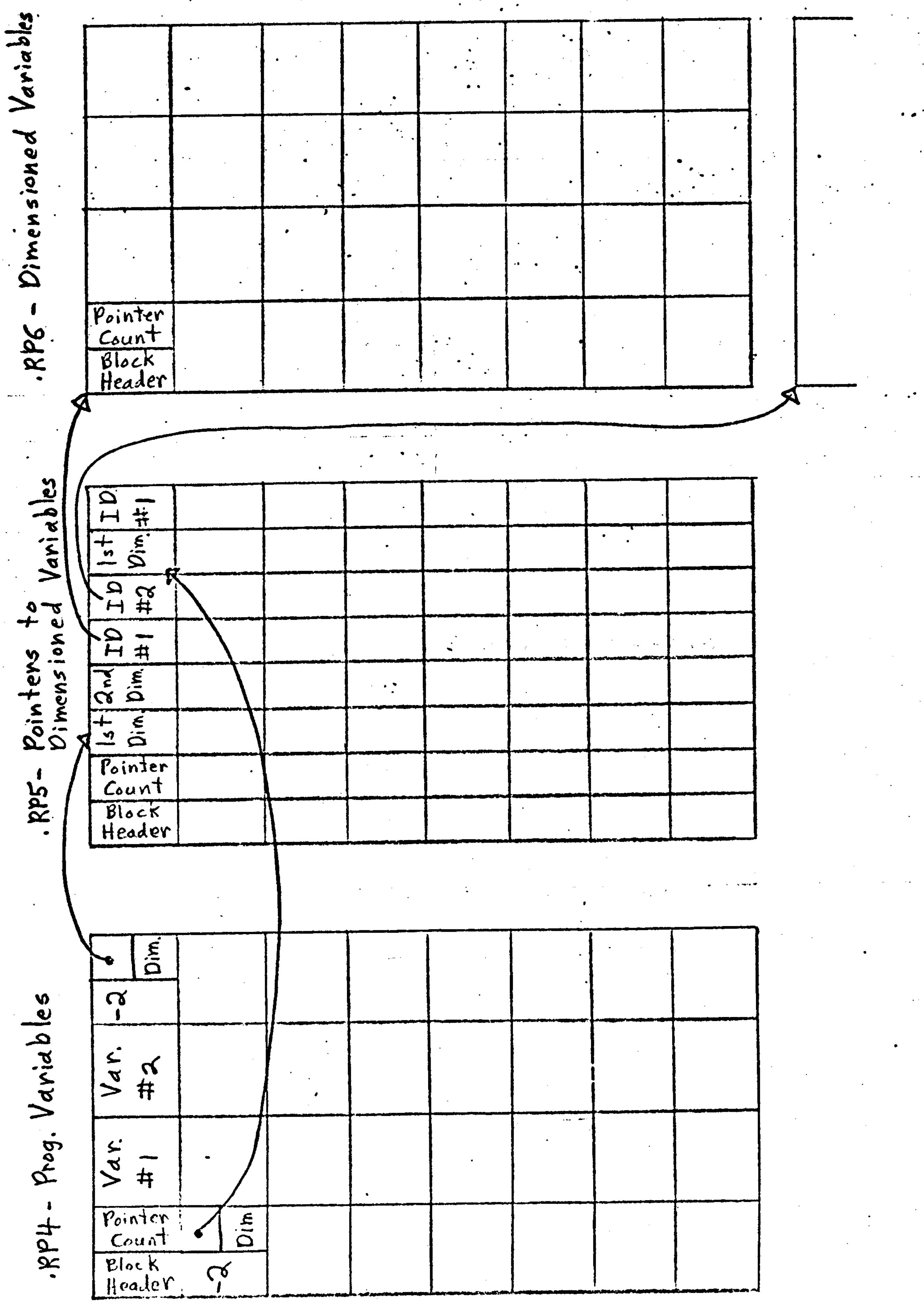
0	Data Block Header	Ptr. Count	Ø177777	Ø177777	STACK		DATA	
8		STACK			DATA			
16		STACK			DATA			
24		STACK			DATA			
32		STACK			DATA		DEF Temp.	DEF Temp.
40	File #1 ID	Current Position ID	R/W Bit File # File Ptr.	File #2 ID			File #3 ID	
48		File #4 ID			File #5 ID			File #6 ID
56			LINID	DIMID	NAMID	PRGID	ØBJID	CØNID
			58	59	60	61	62	63

RP2 - Pointers - Execution Phase

4

0	Data Block Header	Ptr. Count	Ø177777	Ø177777	ST	ACK	DAT	A
8		ST	A CK	DA	TA			
16		ST	ACK	DA	TA			
24	GLØ	BAL	SAV	E	ARE	A	RETID 1	RETCHR 1
32	RETID 2	RETCHR 2	RETID 3	RETCHR 3	FØR, IF, CALL TEMP. DATA	FØR, IF, CALL TEMP. DATA	IF TEMP. DATA	GØSUB STACK PTR.
40	File #1 ID	Current Position ID	R/W Bit File # File Ptr.	File #2 ID			File #3 ID	
48		File #4 ID			File #5 ID			File #6 ID
56			ØUTID in FSNPTP	DIMID	NAMID	PRGID	ØBJID	CØNID
			58	59	60	61	62	63

Scalar and Dimensioned Variables



.RP1 - Used in Compilation and Execution

.RP7 - Used in Compilation

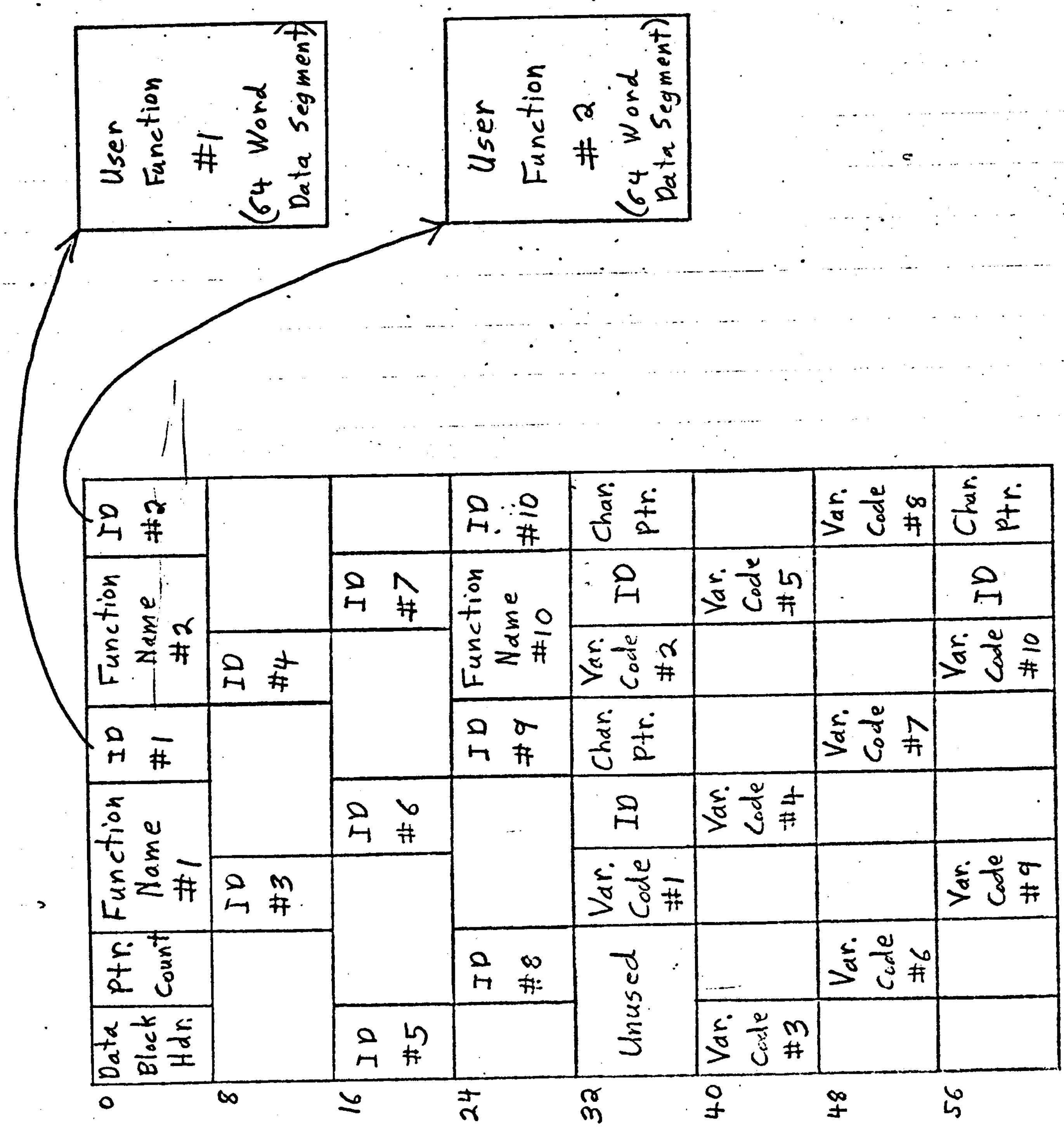
.RP1 - Data Segment

0	Data Block Hdr.	Ptr. Count	Back Ptg. ID	Forward Ptg. ID	No. of Numbers in Block / Unused	Floating point No. #1
8						
16						
24						
32						
40						
48						
56						Floating point No. #29

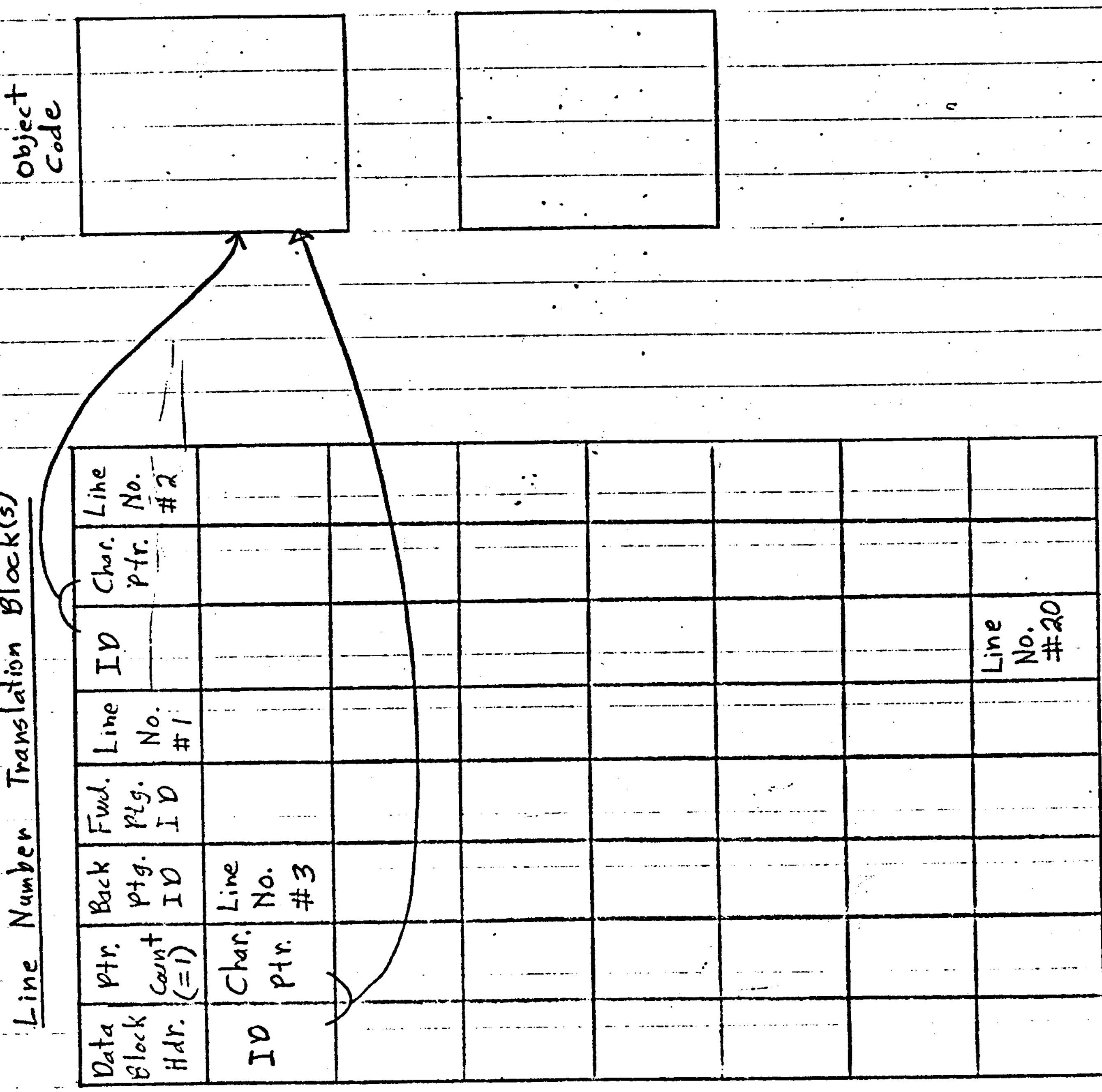
.RP7 - Variable Names

0	Data Block Hdr. / Ptr. Count.	Variable Name #1	Variable Name #2	
8				
16				
24				
32				
40				
48				
56				Variable Name #31

.RP3 - Temporary Data for up to 10 User Functions and 10 For/Next Loops (Compilation Phase Only)



Statement Number Translation Block(s) - LINID



FSNAP CALL Mechanism

- up to three arguments may be passed via the user .T's (scalars, dimensioned variables or floating-point numbers).

.T7	Arg. #1 Name	Arg. #1
.T10	Value	
.T11	Value	
.T12	Arg. #2 Name	Arg. #2
.T13	Value	
.T14	Value	
.T15	Arg. #3 Name	Arg. #3
.T16	Value	
.T17	Value	

- argument values may be passed back to the user program through scalar variables only by leaving the corresponding argument name non-zero.
- all other user .T'S must be left untouched in the called subroutine.
- a global save area (words $24_{10} \rightarrow 29_{10}$ in the block pointed to by .RP2) may be used by the called subroutines throughout the entire execution phase of an FSNAP program.
- The rel. ptr.'s .RP3 and .RP7 may be used globally, as well as .RP1 between READ statements, by any called subroutine.

FSNAP Compiled Code Structure

Variables - variables are given an eight-bit octal code. $101_8 \rightarrow 277_8$ which is used as a direct pointer into the data block which contains the values of the variables.

Dim. Variables - parenthesis, comma's and plus and minus signs are included in the compiled code for interpretation.

Numbers - numbers are defined by a zero byte followed by four bytes which determine the two word floating-point number.

Operators - code used for the operators is as follows:

+	1
-	2
*	3
/	4
↑	5
- unary	6

Expressions - expressions are compiled in reverse polish notation for ease of decoding during execution.

Functions - functions are given an octal code from 16 to 33 (presently) and are treated as "unary" operators during execution.

Function	Octal Code
ABS	16
NEG	17
SQR	20
SIN	21
CØS	22
TAN	23
CØT	24
ATN	25
EXP	26
LGT	27
LØG	30
SGN	31
INT	32
RND	33

Examples

```
 A = B
101 102
 A = 5 * B
101  000  101  320  000  000  102  003
 A              5              B    *
```

A = B(I+J-3,K)

101	102	050	103	053	104	055	<u>300 003</u>	054	105	051
A	B	(	I	+	J	-	3	,	K	)

A = B*C+SIN(D)

101	102	103	003	104	021	001
A	B	C	*	D	SIN	+

A = - (B+C)

101	102	103	001	006
A	B	C	+	-

<u>User Defined Functions</u> - the reverse polish string for a user defined function is copied directly into the compiled object code using the octal code fourteen (14) to designate a user function and the octal code fifteen (15) to designate and argument. During execution code 14 indicates that an argument is popped from the stack and saved, while code 15 indicates that the saved argument is pushed onto the stack.

e.g. DEF CØSH(X) = (EXP(X) + EXP(-X))/2

A = B * CØSH(C)

101	102	103	014	015	026	015	006	026	001
A	B	C	fn.	arg	EXP	arg.	-	EXP	+
000	101	100	000	000	004	003			
2					/	*			

Macros - macros are given an octal code from 16 to 32 (presently) and occur as the first byte in a line of compiled code. The macros are as follows:

Macro	Octal Code
STØP	16
FØR	17
NEXT	20
READ	21
ASK	22
TYPE	23
WRITE	24
RETURN	25
IF	26
GØTØ	27
GØSUB	30
CALL	31
RESTØRE	32
DIM	not compiled
DATA	not compiled
DEF	not compiled

STØP 016

<u>FØR</u> I = 1,N,M

```
017  106  000  100  300  000 000  077  107  077
FØR  I              1             ,    N    ,
110  073  234  320  201
M    ;    ID        (CHPTR+100)
```

- the ID,CHPTR pair points to the place in the object code where the final value and the increment value of the loop will be stored, i.e. immediately following the correxponding compiled NEXT statement.

<u>NEXT</u> I

```
020   106   377   377   377   377   377   377   377   377
NEXT  I     final value             increment
234   320   172
      ID    (CHPTR+100)
```

- the ID,CHPTR pair points to the next statement following the corresponding FOR statement to which control will be transferred if the loop conditions are satisfied.

<u>READ</u> A,B

```
021   050   104  077  105  077
READ  FBLK  A    ,    B
```

- FBLK is a pointer to the file control block stored in the data block pointed to by the user relocatable pointer .RP2.

```
ASK   XIN
022   101   077
ASK   XIN
```

- code 77 is used as separator between variable names.

```
TYPE   #   !   $3   %8.6   "TEXT"   A*B
023   043   041   044   003   045   010   006
TYPE   #     !     $     3     %     8     .6
042   124   105   130   124   042   101   102   003   077
 "     T     E     X     T     "     A     B     *
```

```
WRITE   (15)   %2,A
024   050   045   002   000   101   077
WRITE FBLK %     2     .0     A
```

- FBLK is a pointer to the file control block stored in the data block pointed to by the user relocatable pointer .RP2.

```
RETURN   025
```

```
IF   (A > B) STØP
026   116   076   117   077   016
IF     A     >     B     )     STØP
```

- code 077 marks end of expression in IF statement
- the codes used for the comparison operators are as follows:

<	74
=	75
>	76
=<	71
<>	72
=>	73

IF (A) 10,20,30

```
026  101  077  160  050  205  160  050  223  160  050  235
IF    A    )   ID,CHPTR(10)    ID,CHPTR(20)    ID,CHPTR(30)
```

- the ID,CHPTR pairs are pointers to statements with statement number labels 10, 20 and 30 respectively.
- the CHPTR has 100_8 added so that the code will not interfere with a line feed code (012_8).

GØTØ 10

```
027   215  020  122
GØTØ  ID,CHPTR pair
```

GØSUB 20

```
030    215  020  156
GØSUB  ID,CHPTR pair
```

CALL FSRAND (A)

```
031   055 001  101
CALL    ID      A
```

- the ID of the program segment, FSRAND is compiled directly into the code.

RESTØRE

032 050

RESTØRE FBLK

- FBLK is a pointer to the file control block stored in the data block pointed to by the user relocatable pointer .RP2.

Note: All compiled statements are ended with a semicolon (073_8) or a line-feed code (012_8) depending on whether the source code ends with a semicolon or a line-feed.

- if an error is detected during compilation phase, the object code is padded with ten (10) error codes (377_8) and then two STØP codes (16_8).

- normally only two STOP codes are appended to the compiled code.

MH-1352-HL-JER

H. LYCKLAMA