FORTRAN AUTOMATIC CODE EVALUATION SYSTEM (FACES)

PART I

by

C. V. Ramamoorthy and S. F. Ho

Memorandum No. ERL-M466

25 July 1974

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

FORTRAN AUTOMATIC CODE EVALUATION SYSTEM

(FACES)

PART I

by

C. V. Ramamoorthy and S. F. Ho

FORTRAN AUTOMATIC CODE EVALUATION SYSTEM

(FACES)

PART I

C. V. Ramamoorthy and S. F. Ho

University of California, Berkeley
Department of Electrical Engineering and Computer Sciences
Computer Science Division
Electronics Research Laboratory

Introduction

This report is the documentation of the FORTRAN Automatic Code Evaluation System (FACES) which is designed to serve as an automatic aid in analysing, debugging and partially validating large FORTRAN programs. The documentation includes the user manual, source listing and a sample run.

Under a contract from the U. S. Army Safeguard Evaluation Agency, a system called ACES was developed at the University of Texas at Austin to analyse and partially validate large real time software systems written in CENTRAN. With the ACES experience, an original version of FACES was designed and developed for NASA. Under the auspices of ONR, an improved version of FACES was developed and installed on CDC 6400 at the Computer Center of the University of California, Berkeley and is available to all users upon request.

ACES was developed and implemented with outstanding support from Mr. James Turner and Mr. Robert Meeker, who also took part in the original version of FACES along with Mrs. Ann Haller. The new version incorporates many suggestions of Dr. R. L. Kleir and Mr. Irving Wendell.

The research was sponsored by the Office of Naval Research Contract N00014-69-A-0200-1064.

FORTRAN AUTOMATIC CODE EVALUATION SYSTEM

USER MANUAL

August 10 , 1974

TABLE OF CONTENTS

I. GENERAL DESCRIPTION

The FORTRAN Automatic Code Evaluation System (FACES) is designed to serve as an automatic aid in analyzing and debugging FORTRAN programs. FACES is a software package that takes as input a FORTRAN program which may contain many modules (subroutines and functions). The system is composed of two main parts, the FORTRAN Front End which gathers information about the input program, and the Diagnostic Package which evaluates the information and prints warning messages where possible errors could occur. At present, all information for the analysis is obtained statically from the source code of the FORTRAN program. However, extensions to the system can be made to include dynamic traces and other run-time analyses.

The FORTRAN Front End scans and parses the FORTRAN input program, gathering information about the source code as the parsing is done. For example, all variables and a record of their usage is obtained. A graph structure of each routine analyzed is also formed, and information concerning interface between routines is gathered (common block structure and argument lists in routine calls). This information is stored in sets of tables as described in Section III.

The second part of FACES consists of diagnostic routines which search the information stored in the tables, checking for possible danger signs. Each routine checks for a specific type of problem in the program and may be optionally chosen for execution by the user. The diagnostic routines that are currently available check the source program for correct parameter alignment, common block alignment, proof of variable initialization and trace the future and history of specified variables. These routines are described in more detail in Section II.

II. Instructions for Using FACES

In order to implement FACES, one should be familiar with the basic structure of the system. The FORTRAN Front End is executed first with the source program to be analyzed as data. (See diagram of deck structure.) Before the source program, one data card must be inserted which contains the table print and store options. The options are as follows:

01-Print tables
10-Store tables on file.
11-Print and store tables.

} in column one and two.

If any of the diagnostic routines are going to be executed, the tables must be stored on file. (Tape or common file.)

After the Front End is executed, calls to the diagnostic routines may be made in a main program provided by user. These routines are independent of the Front End and depend only on access to the tables stored on file. Therefore, the diagnostic routines may be executed any time after the tables have been stored.

The user may choose any or all of the diagnostic routines to be executed by inserting calls to the routines chosen. Descriptions of the currently available routines follow:

PARAL

The diagnostic routines PARAL may be used to check alignment of all parameters in subroutines and function calls. The routine is called by the statement

CALL PARAL (IATOPT)

where IATOPT is an option chosen by the user indicating whether array parameters are to be checked for equal dimensions. If IATOPT=0, array dimensions will not be checked. Otherwise, array dimensions will be checked.

PARAL will check the alignment of all argument lists in calls to subroutines and functions that have been analyzed by the FORTRAN Front End. Each argument list will be checked against the defined parameter list to insure that the following conditions exist:

(1) Corresponding parameter lists have the same number of parameters.

(2) Corresponding parameters within each list are of the same type.

(3) Corresponding array parameters within each list have the same dimensions. (if IATOPT≠0)

Warning messages will be printed when the above conditions are not met.

TRACY

The diagnostic routine TRACY may be used to determine if all variables are initialized before being used in a manner that might presume prior initialization (called an input usage). Those variables which are in common, in a DATA statement, or always used as entry parameters are assumed to be handled correctly. The routine is referenced by the statement

CALL TRACY (MDNAM1, MDNAM2)

where the parameters contain the name (four characters per word, left-justified and blank-filled) of the routine to be analyzed. The name of the programmer's main routine is always "$MAIN$".

Input usages are those in which the variable affects: the value of any other variable, the flow of control of the routine, or the output. From each input usage a backward trace is performed along all possible entry paths. Each such path must pass through an initialization of the variable or a diagnostic is produced.

COMBAL

The diagnostic routine COMBAL may be used to verify the alignment

of all common blocks in the routines that have been analyzed by the FORTRAN Front End. It is referenced by the statement

CALL COMBAL (NMCK).

The parameter NMCK allows the user the option of checking names in the common blocks. The alignment conditions that are checked for are the following:

(1) Corresponding common blocks must have the same number of entries.

(2) Corresponding elements within each common block must have identical dimensions.

(3) Corresponding elements within each common block must be of the same type.

(4) Corresponding elements within each common block must have identical names (only if NMCK=1).

COMBAL produces output for each common block. The output consists of the common block name, the routines it appears in, and, when the above alignment conditions are not met, diagnostic messages associated with those routines.

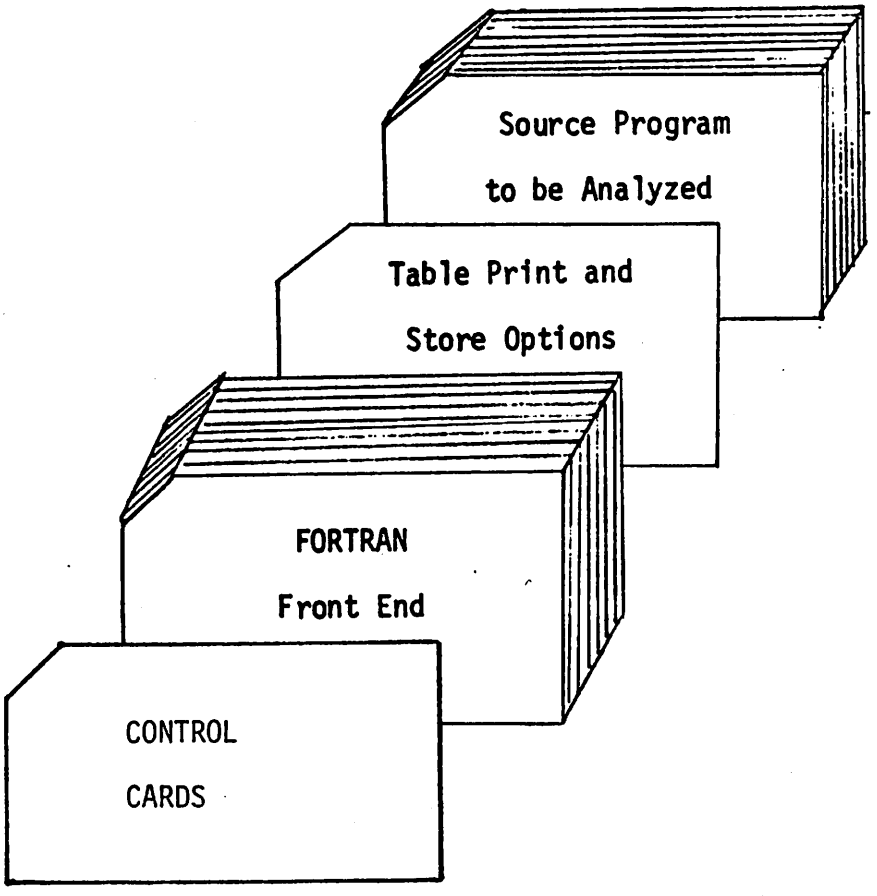
PATHS

This routine may be used to either trace the history or chart the future of the value of a particular variable at a particular statement in a program. The call statement for PATHS is:

CALL PATHS (I,MD1,MD2,IV1,IV2,NODE).

MD1 and MD2 refer to the name of the module to be considered, and IV1 and IV2 contain the name of the variable to be traced. In both cases the name is stored four characters per word, left-justified and blank-filled, (i.e. in H format). NODE is the statement number from which the trace is initiated. I is a parameter specifying either the backward or forward trace. If I=0,

PATHS will produce a list of all variables which might have affected the subject variables value at NODE and, associated with each variable, the node at which the revaluation is noted. If I = 1, PATHS produces a list of all variables which may be affected by the subject variable's value at NODE and the associated node for each revaluation. Variables may appear more than once in the output if there is more than one associated node to be listed. It should be noted that the value traced is the value of the subject variable immediately prior to the execution of statement number NODE.



Deck Structure for Front End Execution

Control cards for the execution of the FORTRAN Front End on CDC 6400 at the Computer Center, U.C. Berkeley will look like the following :

A. Store tables in Common File

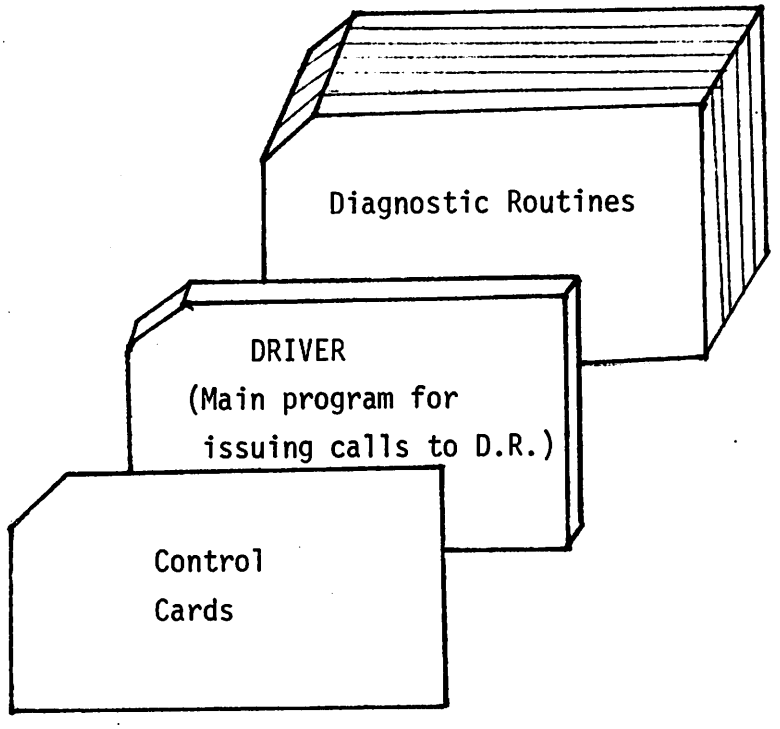
```
JOB CARD
RUNW
COMMON, RANDOM, FC
CLDR
7 8 9
```

B. Store tables in tape

```
JOB CARD
RUNW
CLDR
REQUEST, TAPE1, HY, I. TAPE NUMBER, WRITE, name
REWIND, RANDOM
COPYBF, RANDOM, TAPE1
REWIND, TAPE1
7 8 9
```

RANDOM is a file used by TSDISK which is a CAL system routine for handling random access files (Appendix A).

FORTRAN Front End currently requires $130{,}000_8$ words of core memory for execution.



Deck Structure For The Execution of Diagnostic Routines

Control cards for the execution of the Diagnostic Routines on CDC 6400 at the Computer Center, U.C. Berkeley will look like the following :

A. Tables stored in Common File

```
JOB CARD
RUNW
COMMON, RANDOM, SC
CLDR
78_9
```

B. Tables stored in tape

```
JOB CARD
RUNW
REQUEST, TAPE1, HY, I. TAPE NUMBER
COPYBF, TAPE1, RANDOM
CLDR
78_9
```

Diagnostic Routines currently require $120{,}000_8$ words of core memory for execution.

Sample main program :

```
PROGRAM DRIVER ( OUTPUT )
IMPLICIT INTEGER (A-Z)
DIMENSION INDEX (1000)
CALL OPDISK (1000, INDEX)
CALL COMBAL (1)
CALL PARAL (1)
   :
calls to other diagnostic routines
   :
STOP
END
```

Note: OPDISK is called to prepare filset RANDOM for accessing.

III.

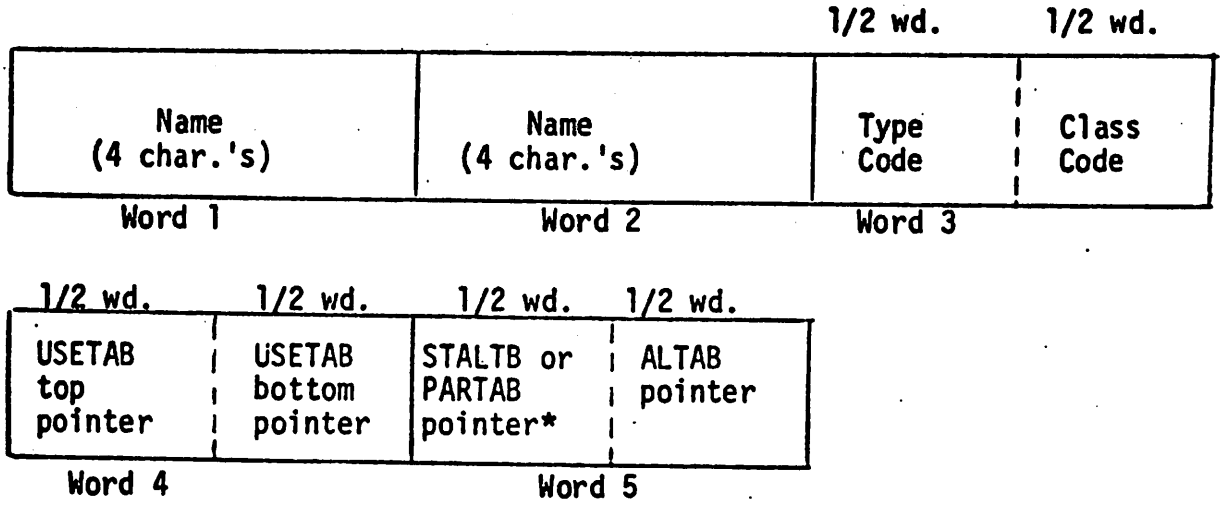
Tables Produced by the FORTRAN Analysis Package

The analysis of a Fortran task (program, subroutines, and user defined functions) results in the generation of two distinct table types. First there are the global tables that describe the interactions and interfaces of the various program modules ofthe task. Included in this group are the task directory (DIREC), the common block name table (CNTAB), and the parameter table (PARTAB).

In the second table type are the local tables that describe a single routine or function. Unlike the global tables, for which one and only one of each table is generated for the task analysis, each routine or function analyzed generates a complete set of local tables. For each module there is generated a symbol table (SYMTAB), name usage table (USETAB), array length table (ALTAB),the predecessor and successor tables (NODTAB, SUCTAB, PRETAB), DO loop limits table (DOTAB), and the storage allocation table (STALTB). Explanations of the local and global table structures follow.

LOCAL TABLES:

SYMTAB (5,1800) - (Symbol Table)



SYMTAB contains the names of all variables, labels, modules, and common blocks that appear in the module. The names are hash coded and stored in the table one time only. The first two words of the table are used for storing the name, four characters per word, left-justified, blank-filled. The TYPE and CLASS codes, as listed below, are stored in word 3 of the table for each name. Word 4 contains top and bottom pointers to USETAB, the usage table, where each name is stored once for each time it appears in the module. The USETAB top pointer points to the entry for the first occurrence of the name, and the bottom pointer points to its last occurrence. The lower half of word 5 contains a pointer to ALTAB, the array length table, when the name in SYMTAB is an array. ALTAB contains the dimensions of the array. If the name is not an array, the lower half of word 5 is zero. The upper half of word 5 can contain one of two types of pointers. If the name is a common variable, a pointer to STALTAB, the

storage allocation table, will be found in this place. If the name is a function or subroutine name, a pointer to PARTAB, the parameter table, is stored here. If the name is neither a function name, subroutine name nor a common variable, this field is 0.

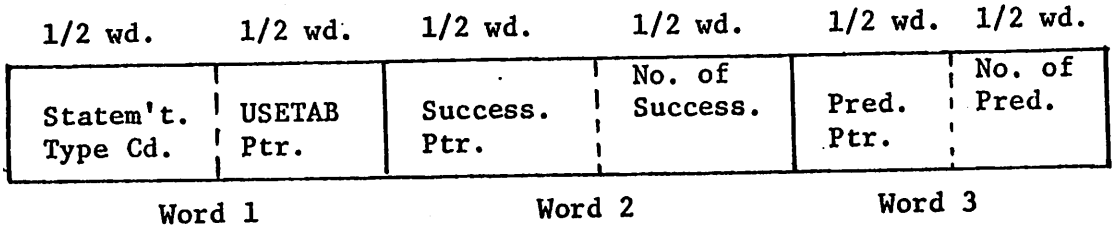
Class Code

Program name	0
Subroutine name	1
Statement function name	2
Array name	3
Function name	4
Undefined Label	5
Variable	6
Common Block name	7
Label	10000 + statement number of label definition
Integer Constant	8
Entry Point Name	9
PARAMETER variable	10
NAMELIST Name	11

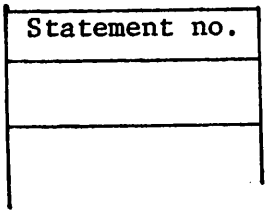
Type Codes

Integer	0
Floating Point	1
Double Precision	2
Complex	3
Logical	4
Neutral	5

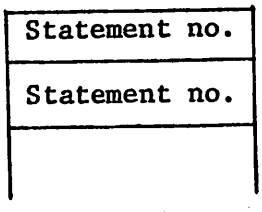
NODTAB (3,1000) - (Node Table)



SUCTAB (1500)



PRETAB (1500)



NODTAB, SUCTAB, and PRETAB together serve to link each node (each statement is treated as a node) to its immediate successors and predecessors. The index of NODTAB identifies the statement number of a node. The statement type code for that statement is stored in word 1 along with a pointer to the beginning of the USETAB list for this statement. Word 2 of NODTAB contains a pointer to the list of immediate successors in SUCTAB and the number of successors the node has. Entries in SUCTAB (SUCCESSOR PTR) to SUCTAB (SUCCESSOR PTR + NUM of SUCCESSORS) points to statements that are successors of the node specified. Word 3 contains the same information for immediate predecessors of the node which are stored in PRETAB.

Special codes are stored in SUCTAB and PRETAB for certain transitions where the successor or predecessor does not exist.

SUCTAB

10,000	SUBROUTINE reference
20,000	RETURN or STOP statement
30,000	END statement
60,000	transfer to an undefined label
70,000	transfer to an undefined DO-LOOP label

PRETAB

40,000	ENTRY statement
50,000	PROGRAM, SUBROUTINE or FUNCTION entry point
70,000	Predecessor is an undefined DO-LOOP label
90,000	Predecessor not found

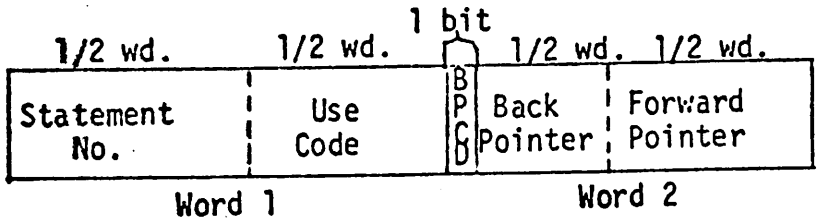
STATEMENT TYPES

INDEX	MATCH ENTRY	FORTRAN St. Type
1		
2		
3	FORM	* FORMAT
4	PRIN	PRINT
5	IMPL	IMPLICIT
6	NAME	* NAMELIST
7	ENCO	* ENCODE
8	DECO	* DECODE
9	PUNC	PUNCH
10	IF	IF
11	COMP	COMPLEX
12	EXTE	EXTERNAL
13	BLOC	BLOCK DATA
14	END	END
15	READ	READ
16		
17		
18		
19		
20	ENDF	* ENDFILE
21		
22		
23	REAL	REAL
24	BACK	* BACKSPACE
25		
26	LOGI	LOGICAL
27	FUNC	FUNCTION
28	DIME	DIMENSION
29		
30	SUBR	SUBROUTINE
31	DATA	DATA
32	PROG	PROGRAM
33	DOUB	DOUBLE PRECISION
34	CALL	CALL
35	INTE	INTEGER
36	COMM	COMMON
37		
38	ASSI	ASSIGN
39	WRIT	WRITE
40	EQUI	* EQUIVALENCE
41	STOP	STOP
42	REWI	* REWIND
43		
44	CONT	CONTINUE
45	GOTO	GOTO
46		
47		
48	ENTR	ENTRY
49		
50	PAUS	* PAUSE
51	RETU	RETURN
52		ASSIGNMENT STATEMENT
53		DO STATEMENT

Note: Index also indicates statement type code.

* - currently ignored by PARSER

USETAB (2,2500) - (Usage Table)



Each occurrence of a name in a module creates one entry in USETAB. The statement number and use code (as listed below) for each entry are stored in word one. A back pointer to the last occurrence of the name is stored in the upper half of word 2. For the first entry of a name, the back pointer points to the address of the name in SYMTAB and the B-P code is 1. Otherwise, the back pointer points to the previous entry in USETAB and the B-P code is 0. The lower half of word 2 contains a forward pointer to the next entry in USETAB. For the final occurrence of the name, the forward pointer is 0.

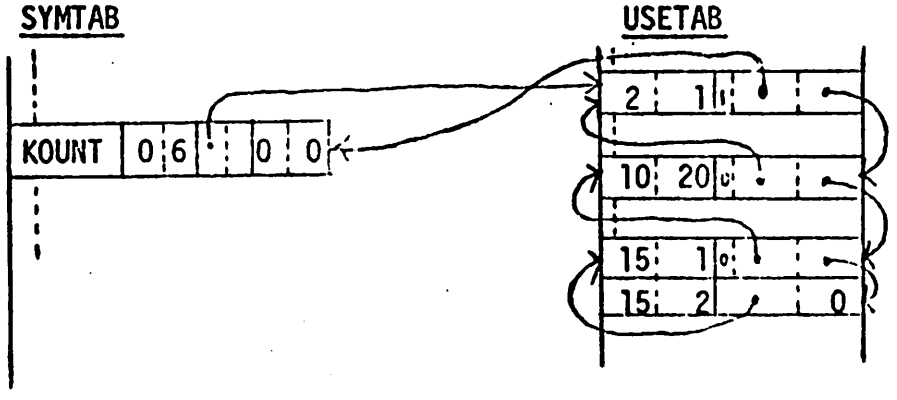
Use Codes

Declaration	0
Output variable (in assignment statement)	1
Input variable (in assignment statement)	2
External output variable	3
External input variable	4
Do loop index variable	5
Do loop starting index	6
Do loop ending index	7
Do loop increment value	8
Label definition	9
Label reference	10
Common Block entry	11
Data statement entry	12
Dimension statement entry	13
Type statement entry	14
Subscript	10000 + address of array name in SYMTAB
Function dummy parameter	16
Subroutine dummy parameter	17
Function actual parameter	18
Subroutine actual parameter	19
Conditional branch decision variable	20
External procedure reference	21
Variable referenced in ASSIGN statement	22
Variable referenced in ASSIGNED GO TO statement	23
Variable referenced in COMPUTED GO TO statement	24
Label referenced in DO statement	25
I/O unit reference	26
FORMAT reference	27
Multiple return parameter	28
Switch variable in IF statement	29

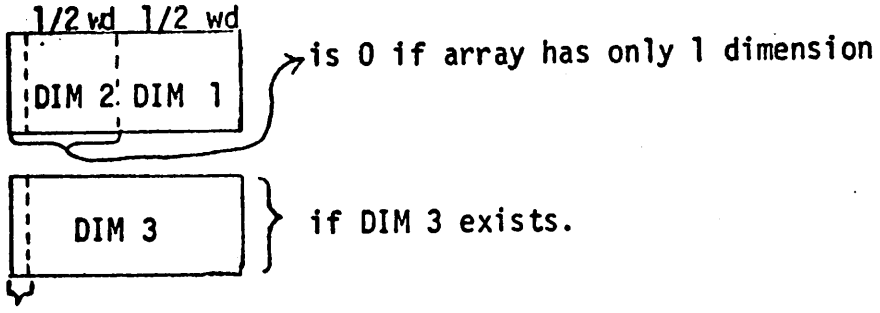
Illustration of Interaction Between SYMTAB and USETAB

Example program

```
 1        SUBROUTINE DUMMY
 2        KOUNT = 1
 :          :
10        IF (KOUNT.EQ.10) GO TO 50
 :          :
15        KOUNT = KOUNT + 1
```

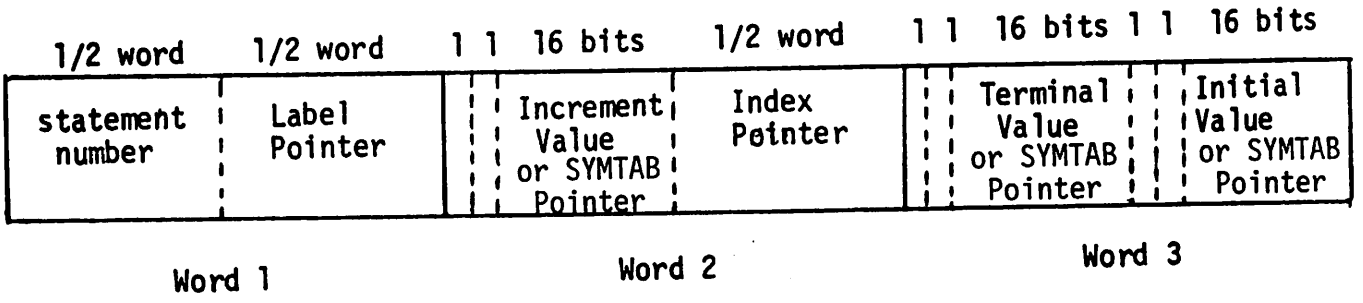


ALTAB (1,56) - (Array Length Table)



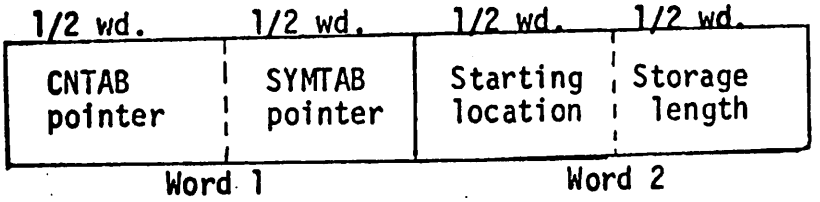
ALTAB contains the dimensions of each array in the module. SYMTAB contains a pointer to the appropriate word of this table for each array stored in SYMTAB. If the array has less than three dimensions, only one word of ALTAB is used. If a third dimension exists, it is stored in the next word of ALTAB with a code 1 in the high order 3 bits. All dimensions are stored in integer format, right-justified in the appropriate field.

DOTAB (3,50) - (DO Loop Limits Table)



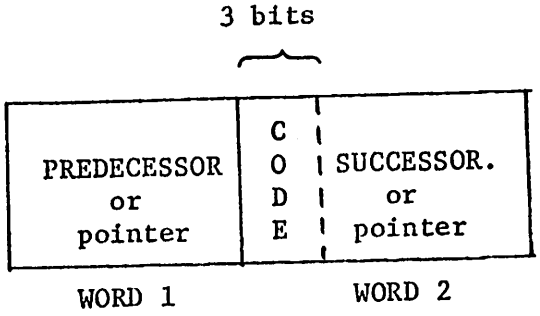
Each DO statement generates one entry in DOTAB. The first word of the entry contains the statement number of the DO statement and a pointer to the end of the DO label in SYMTAB. The second word contains the index and increment values, and the third word contains the terminal and initial values of the DO index. Associated with each value is a one bit code to indicate whether the value is an integer constant (code = 0) or a pointer to a variable in SYMTAB (code = 1), and a one bit code to indicate whether the entry has positive or negative value.

STALTB (2,100) - (Storage Allocation Table)



STALTB contains one entry for each declaration of a common variable in a common block. For each common block, the variables in that block are entered serially according to the order in which they appear in the COMMOM BLOCK declaration. A pointer to the name of the variable in SYMTAB is stored in the lower half of word 1 and a pointer to the name of the common block in CNTAB is stored in the upper half of word 1. The storage length (total number of words) of the variable is stored in the lower half of word 2. (For example, if the variable is a 2 x 10 array, the storage length is 20.) The starting location of the variable, relative to its common block, is stored in the upper half of word 2.

TRIP (2,1000) - (Transition Pair Table).



TRIP is a tempórary local table generated by the Front End to record all program transfers.

Since the predecessor and successor may be undefined when a program transfer is parsed, the predecessor and successor field may contain pointers instead of statement numbers. The contents are identified by the code field. Post-processing will convert pointers into statement numbers for the generation of NODTAB, SUCTAB and PRETAB. The above configuration is a layout of the TRIP table before post-processing.

Code

0 predecessor field contains predecessor statement number. successor field contains a pointer to a label name in the symbol table.

1 predecessor field contains a pointer to a label name in the symbol table. Successor field contains successor statement number.

2 predecessor and successor field are empty (zero). Reserve space for DO-loop exit transition.

3 both predecessor and successor field contain statement numbers.

4 external subroutine reference

predecessor field is the predecessor statement number.

successor field is empty (zero).

Special codes are stored in the successor and predecessor field for transitions where the successor or predecessor does not exist

Predecessor

40,000 entry statement

Successor

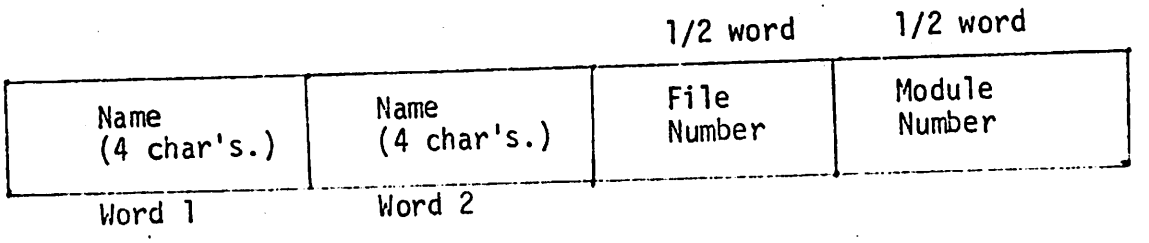
20,000 RETURN or STOP statement.

30,000 END statement.

After post processing by PPTRIP, the code field is eliminated and all pointers are converted into statement numbers. The TRIP table printed by Front End is the TRIP table after post-processing.

GLOBAL TABLES

DIREC (3,100) - (Directory)



The Directory is a hash coded table which contains the names and associated module numbers of each routine or function referenced in a task. The following convention is used in assigning module numbers:

1## - Program
2## - Subroutine
3## - Function
4## - Block Data

The assignment of a module number to a routine takes place when the module is defined. System supplied functions and undefined externals will not be assigned module numbers.

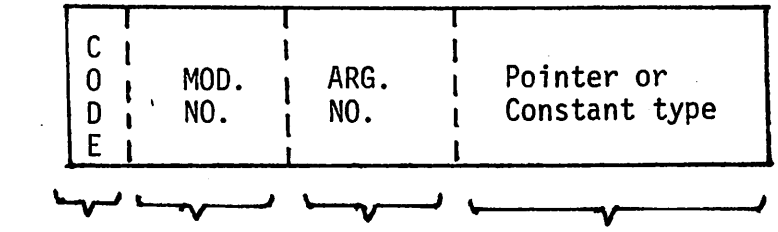
The file number indicates what file contains the local tables for a module. Only one set of local tables will be held in main memory at a time. The main routine that is analyzed by the FORTRAN Front End is always given the name "$MAIN$" so that a name will appear in the directory. (Sometimes, the main program may not have a program name.)

CNTAB (4,300) - (Common Name Table)

Common Blk. Name (4 char's.)	Common Blk.Name (4 char's.)	STALTB top pointer	STALTB bottom pointer	Module No.
Word 1	Word 2	Word 3		Word 4

The Common Name Table contains the names of all common blocks for all modules. Each block name is storedin a separate entry every time it is declared. For each entry, pointers to the top and bottom of the list of common variables for that block in STALTAB are stored in word 3 of CNTAB. Word 4 contains the module number in which the common block is declared.

PARTAB (2000)- (Parameter Table)



PARTAB contains one entry for every occurrence of an actual parameter or dummy parameter in all modules. Pointers to the parameters in SYMTAB are stored in serial lists, the first and last words of the list being marked by special codes. The module number in which this procedure call or definition appears is also stored with each entry.

Codes

0--Ptr. to dummy parameter in SYMTAB

1--Ptr. to actual parameter in SYMTAB

2--Ptr. to parameter list for next call of same routine

3--Ptr. to Sub. or function name in SYMTAB

4--This word marks end of current parameter list

5--This is a constant parameter. Pointer field contains constant type.

*Dummy parameters shall be defined as a parameter which occurs in a subroutine or function declaration. For example, in the statement

SUBROUTINE HELP (A, B, C)

A, B, and C are dummy parameters.

*Actual parameter shall be defined as a parameter which occurs in a call to a subroutine or function, for example, in the statement

CALL HELP (X, Y, Z)

X, Y, and Z are actual parameters.

IV. Notes on Table Sizes

In order to make FACES as flexible as possible, the system has been designed in such a way that the table sizes can easily be changed by the user. The table sizes as they exist now should be adequate for most large, multi-module programs unless they exceed any of the limits listed below. If a program does exceed one or more of these limits, or if a table overflow message is printed, the table sizes can be expanded according to the following instructions given in Section V.

Limits on Program Size with Current Table Sizes

Maximum number of modules: 100

Statements per module: 1000

Arrays per module: 56

DO statements per module: 50

Common variables per module: 100

Instructions for Changing Table Sizes

In order to change the size of a table, the user must do the following three things:

(1) Change the initialization of the appropriate table length variable in the BLOCK DATA subroutines of the FORTRAN FRONT END and of the Diagnostic Package.* (The name of the Table length variable is L followed by the first two or three letters of the table name.)

(2) Change the dimension declaration of the table in all occurrences of the common block in which that table appears in the FORTRAN Front End and the Diagnostic Package

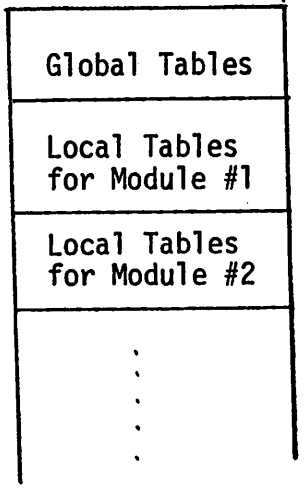
* If a hash-coded table length (SYMTAB or DIREC) is changed, the prime number associated with it and used in hash coding must also be changed to equal the largest prime number less than the table length.

V. File Handling Routines

In order to execute the diagnostic routines, tables produced by the FORTRAN Front End must be stored in a file. However, different machines have different system library routines for handling files. Even though FACES is designed for easy transferrability, subroutines using these system dependent file handling routines have to be modified when moved from one installation to another. With well defined functions, these subroutines are well-insulated from the rest of the FACES system. Changing these routines to adapt to a new working environment shouldn't cause any major problems.

General File Structure

The FORTRAN Front End produces a set of local tables for each module that is analyzed. Each set is written to the file immediately after a module has been analyzed. Each table has a standard length (i.e. the entire table is written to the file, even if it is not full). At the beginning of execution, enough space is reserved at the beginning of the file to store the three global tables. Then, at the end of execution, the global tables are stored in the pre-reserved space. The structure of the tables on file will finally appear as shown in the following diagram:



File Routines

FORTRAN Front End:

SUBROUTINE FILSET*

FILSET prepares the file for writing, e.g. rewind and position the file past the space reserved for global tables.

SUBROUTINE OUTLT (MODNO).

The FORTRAN Front End produces a set of eight tables for each module that is analyzed. Each set is written to the file immediately by OUTLT after a module has been analyzed. (Pointers to the last used entry of local tables are stored in the first row of the corresponding tables.) MODNO specifies the module number. Local tables are stored in the order indicated:

```
          symbol table
SYMTAB :  size = LSYM *5

          node table
NODTAB :  size = LNOD *3

          successor table
SUCTAB :  size = LPS
          SUCTAB(1) ←  ISUC

          predecessor table
PRETAB :  size = LPS
          PRETAB (1)  ←  IPRE

          use table
USETAB :  size = LAL
          USETAB (1,1)  ←  INDUT-1

          array length table
ALTAB  :  size = LAL
          ALTAB (1)  ←  INDAL-1
```

```
                DO table
DOTAB :         size = LDO *3
                DOTAB (1,1) ← INDDO

                storage allocation table
STALTB :        size = LSTAL *2
                STALTB (1,1) ← INDST-1
```

SUBROUTINE OUTGT

OUTGT stores the global tables in the file after all the modules have been analyzed. Pointer to the least used entry of some global tables are stored in the first row of the corresponding tables. Global tables are stored in the following order:

```
                directory
DIREC :         size = LDIR *3

                common name table
CNTAB :         size = LCN *4
                CNTAB (1,1) = INDCN

                parameter table
PARTAB :        size = LPAR
                PARTAB (1) = IPAR-1
```

SUBROUTINE EOFRIT*

EOFRIT closes the file and prepares it for subsequent job or job step. This routine is called by the Front End before it terminates.

Diagnostic Routines

SUBROUTINE OPDISK

OPDISK prepares the file for reading. This is called by the main program before any diagnostic routines are called.

File opening may not be required for some system file handling routines.

* Opening and closing of a file may not be required for some system file handling routines.

SUBROUTINE READLTS (MODNO, NTABLE)

READLTS is called by the diagnostic routines whenever a local table of a certain module is needed. The corresponding local table is brought in by READLTS. MODNO and NTABLE specify the module and table number respectively. Pointers to the last used entry in some tables are retrieved from the first row of the corresponding tables.

Table Number

1 : SYMTAB (Symbol Table).
size = LSYM *5

2 : NODTAB (Node Table).
size = LNOD *3

3 : SUCTAB (Successor Table).
size = LPS
ISUC ← SUCTAB (1)

4 : PRETAB (Predecessor Table).
size = LPS
IPRE ← PRETAB (1)

5 : USETAB (Use Table).
size = LUSE *2
INDUT ← USETAB (1,1)

6 : ALTAB (Array Length Table).
size = LAL
INDAL ← INDAL-1

7 : DOTAB (DO Table).
size = LDO *3
INDDO ← DOTAB (1,1)

8 : STALTB (Storage Allocation Table)
size = LSTAL *2
INDST ← STALTB (1,1)

SUBROUTINE READGTS (NTABLE)

READGTS is called by diagnostic routines to bring in global tables. NTABLE specify the global table number. Pointers to the last used entry in some tables are retrieved from the first row of the corresponding tables.

Global Table Number

1 : DIREC (Directory).
size = LDIR *3

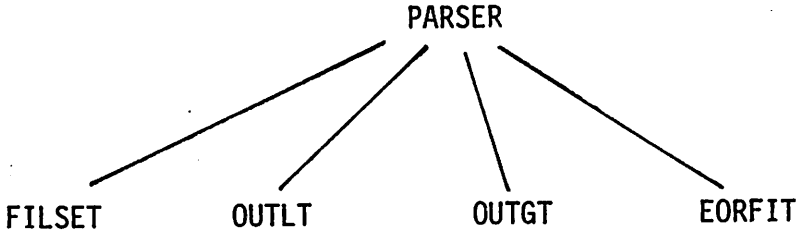
2 : CNTAB (Common Name Table).
size = LCN *4
INDCN←CNTAB (1,1)

3 : PARTAB (Parameter Table).
size = LPAR
IPAR←PARTAB (1)

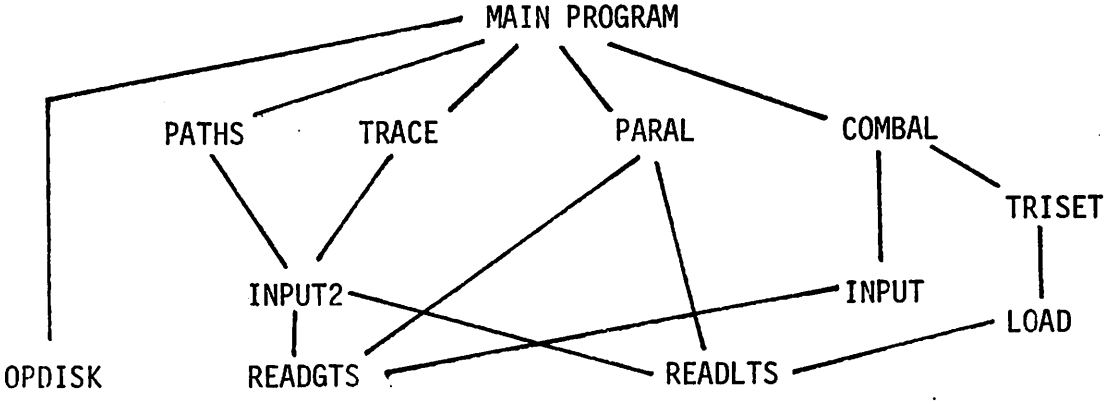
Appendix B contains a set of file storage and retrieval routines using CAL TSDISK. (A CAL system library routine which performs random access disk input and output).

Calling Hierarchy

FORTRAN Front End



Diagnostic Routines



VI. FORTRAN FRONT END DOCUMENTATION

General Description

There are three main tasks for the FORTRAN Front End; parsing, table generation and program graph generation. The main routines for each task are PARSER, TABS and TRANS respectively. The main routine for the entire Front End is PARSER which also drives TABS and TRANS.

PARSING

Parsing is done on a statement by statement basis. Since program modules analysed by FACES are assumed to be compilable without any syntax error, syntax checking is limited to aid information extraction. Statement types* are identified by matching keywords (e.g. PRIN for PRINT statement). Once the statement type is identified, special purpose parsing routine designed to accept the syntax for that particular statement type is called to extract and pass information to TRANS and TABS for making table entries.

Generation of System Tables

Each occurence of a logical symbol (name or label) is recorded in system tables by TABS (and related routines call by TABS).

TABS is called with the symbol name, the class, type and use code passed as parameters. Class code identifies the symbol type (label, variable, subroutine name etc.). Type code is assigned to variable or function name to designate variable type (integer, logical etc.). Use code determines how the symbol is referenced in the statement (subscript, DO-loop index etc.). Based on this information, TABS will make the appropriate entries in the system tables. Detailed descriptions of table

*Statement types are listed in Appendix D.

structures can be found in section III of User Mannual.

Program Graph Generation

The FACES system represents program structure as a directed graph with nodes representing program statements and edges representing lines of program flow. Each statement is treated as a node and is given a (pseudo) statement number as it is processed.

"Normal transitions" (i.e. the only successor of statement i is statement i+1) are not recorded in the transition table (TRIP). Therefore, if statement is not found in the TRIP table as a predecessor, normal transition is assumed. Program transfers recognized by the parsing routines are recorded in the TRIP table as "non-normal transition pairs" by subroutine TRANS. "Non-normal transition pair" (i,j) represent permissable transition from statement i to statement j.

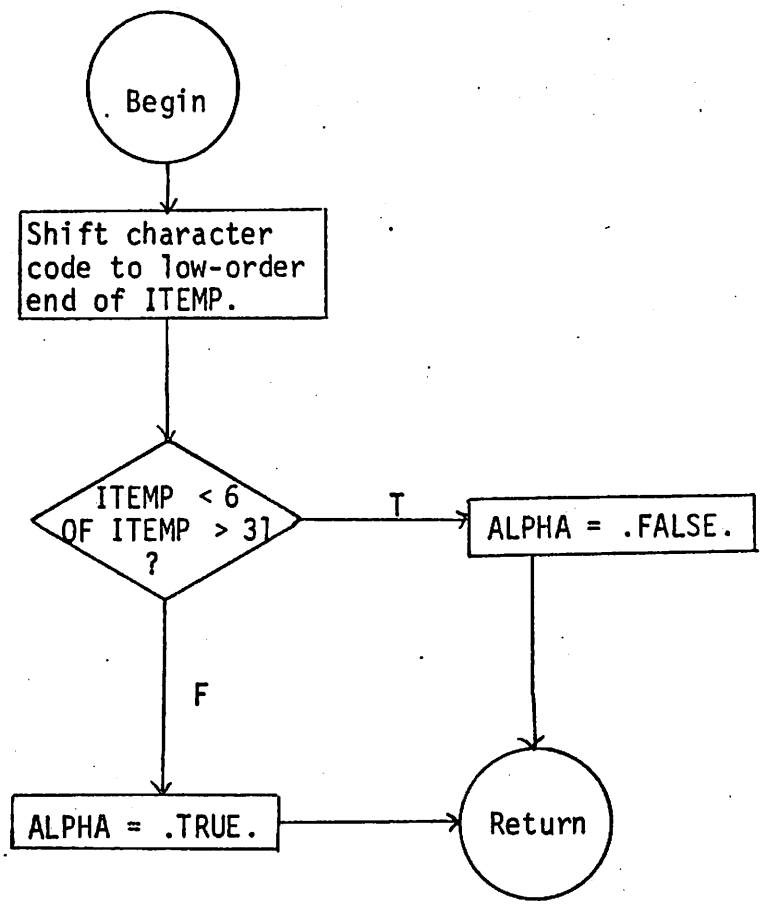
While program transfers are parsed, their destination (statement label) may be undefined at that point and i, j is recorded as undefined labels. Post-processing of TRIP table by PPTRIP will convert these undefined labels into statement numbers for the generation of NODTAB, SUCTAB, and PRETAB.

Documentation for subroutine PARSER contains a detailed description of the parsing algorithm used.

Logical Function Alpha (CHAR)

This function determines whether the input parameter CHAR is a letter A-Z. If it is a letter, the value .TRUE. is returned. Otherwise, .FALSE. is returned.

Flowchart for ALPHA



FUNCTION BITSET(I,KB1,KB2) AND SUBROUTINE BITRET(IWD,ICD,IPM)

BITSET and BITRET are used to manipulate the left-most 2 bits of the right half of a given word. They are used specifically in setting and retrieving the one-bit indicators in the DO loop table (DOTAB).

BITSET is used in subroutine DORANG to set the proper bits depending on parameters KB1 and KB2. The contents of BITSET on return is the contents of input parameter I with one, two, or none of the following modifications: (1) If KB1 = 1, the left-most bit of the right half word is set to 1, (2) If KB2 = 1, the second bit of the right half is set to 1. The contents of I remains unchanged.

BITRET is called by subroutine LOCPRT to extract the one-bit indicators from DOTAB entries for printing purposes. From the right half word of parameter IWD, BITRET extracts the left-most bit and second from left-most bit, storing them in ICD and IB2, respectively. The value of IB2 (0 or 1) is then used to set parameter IPM to the proper print character (+ or -).

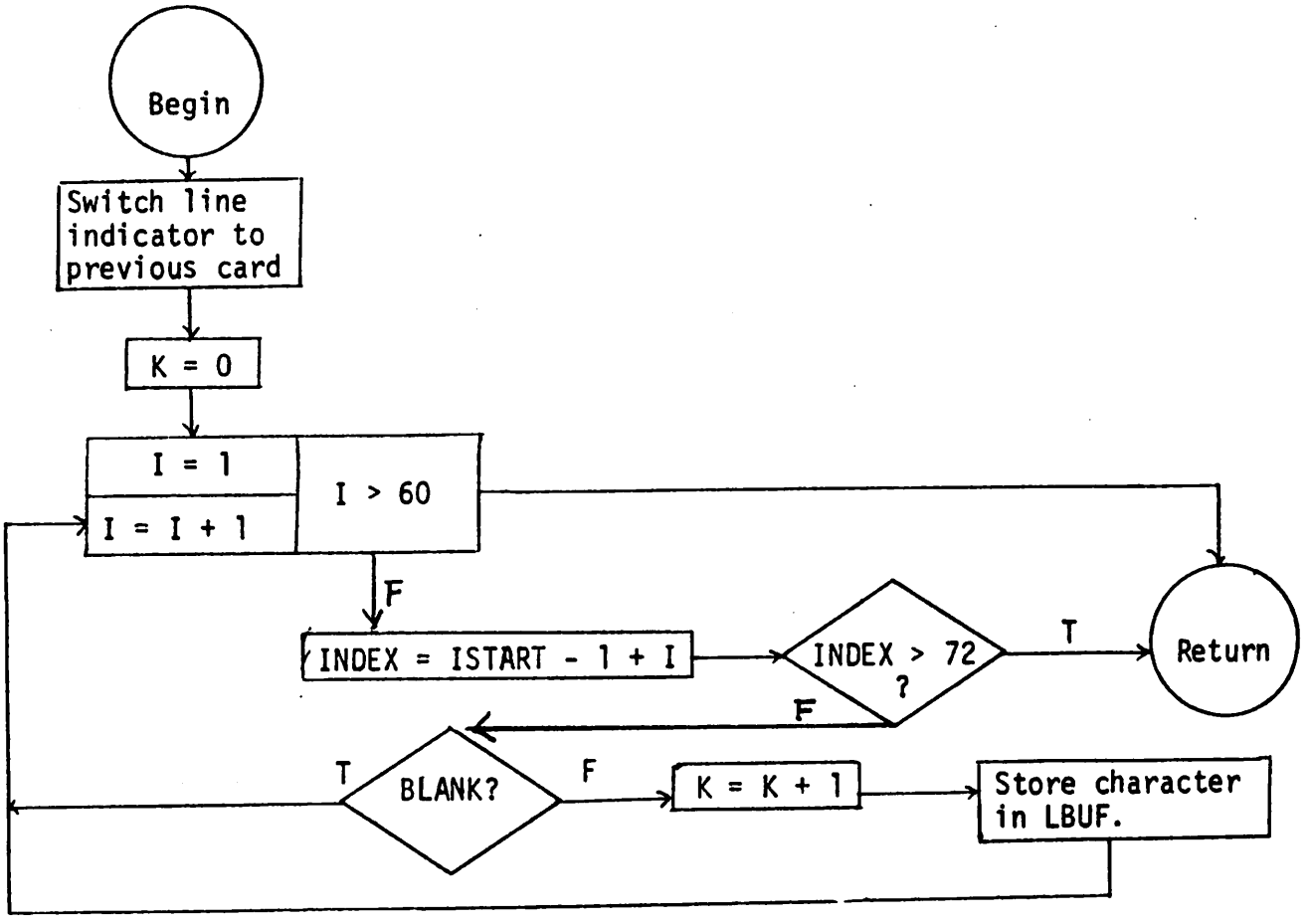
Bit manipulation in BITSET and BITRET is accomplished by the FLD function.

Both bits retrieved by BITRET are reset to 0 in parameter IWD on return.

Subroutine BUFF(ISTART)

This routine stores an alphanumeric string from the end of a card into the temporary buffer LBUF so that the next card can be scanned without losing the string. The string is stored one character per word and at most ten characters can be stored. Blanks are not stored. The input parameter ISTART marks the starting point where the letters are to be picked up and stored.

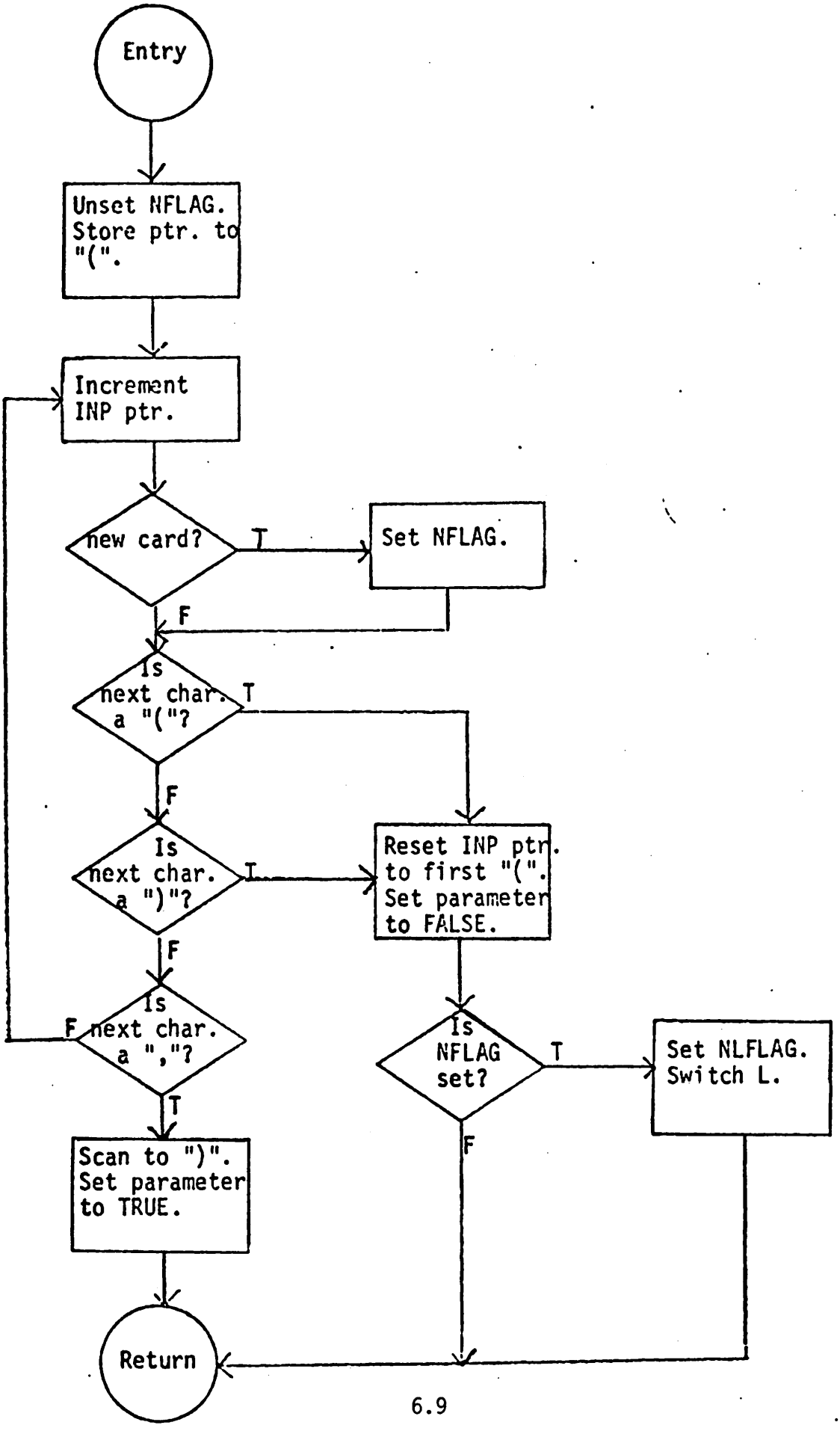
Flowchart for BUFF



Subroutine COMPLX(B)

This routine is called by SCAN to check for a complex constant when a left parenthesis is found. The characters following the left parenthesis are scanned, and if a comma is found before a right parenthesis (indicating a complex constant) the output parameter B is set to .TRUE. . Otherwise, B is set to .FALSE. .

Flowchart for COMPLX

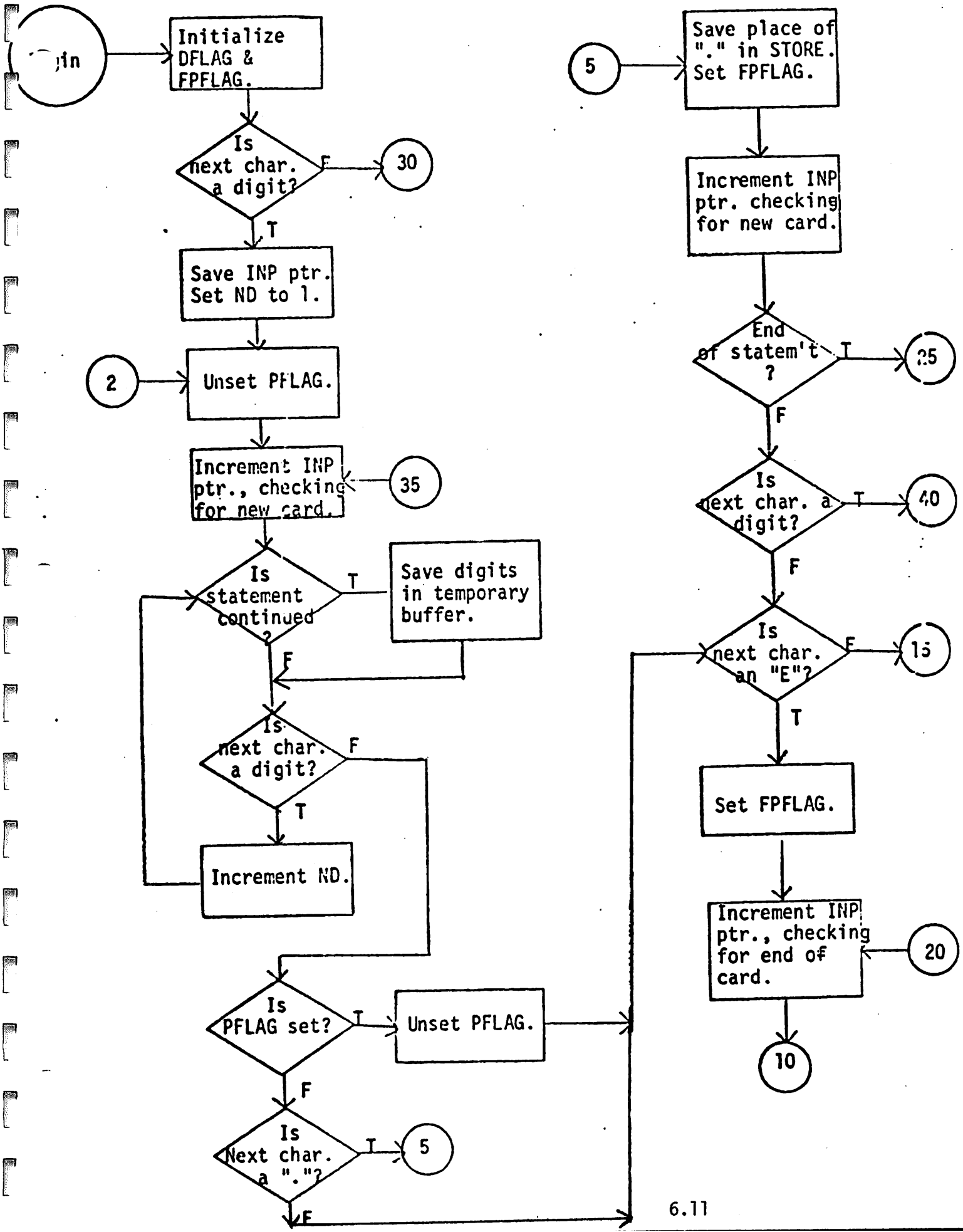


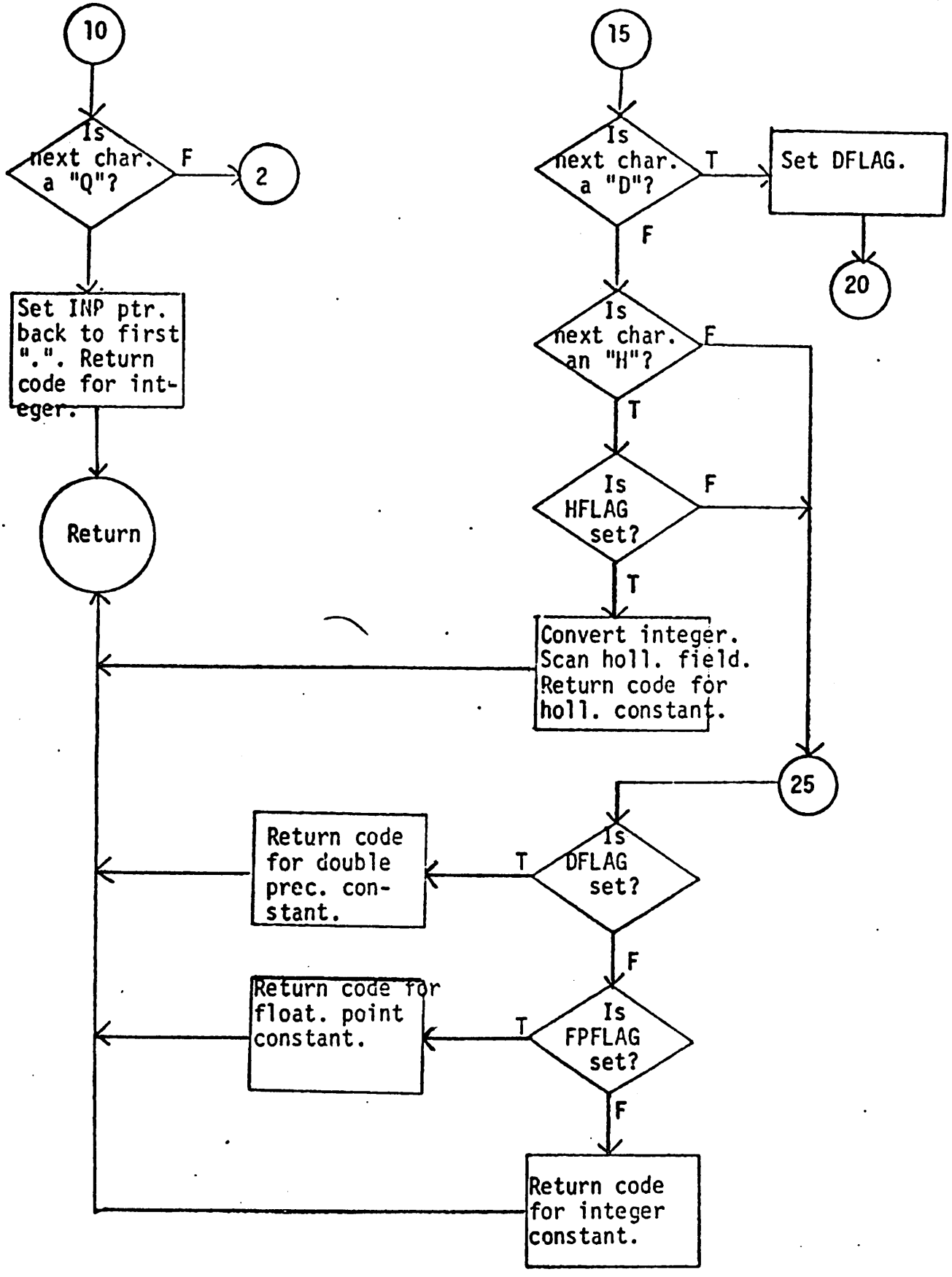
Subroutine CONS(IP)

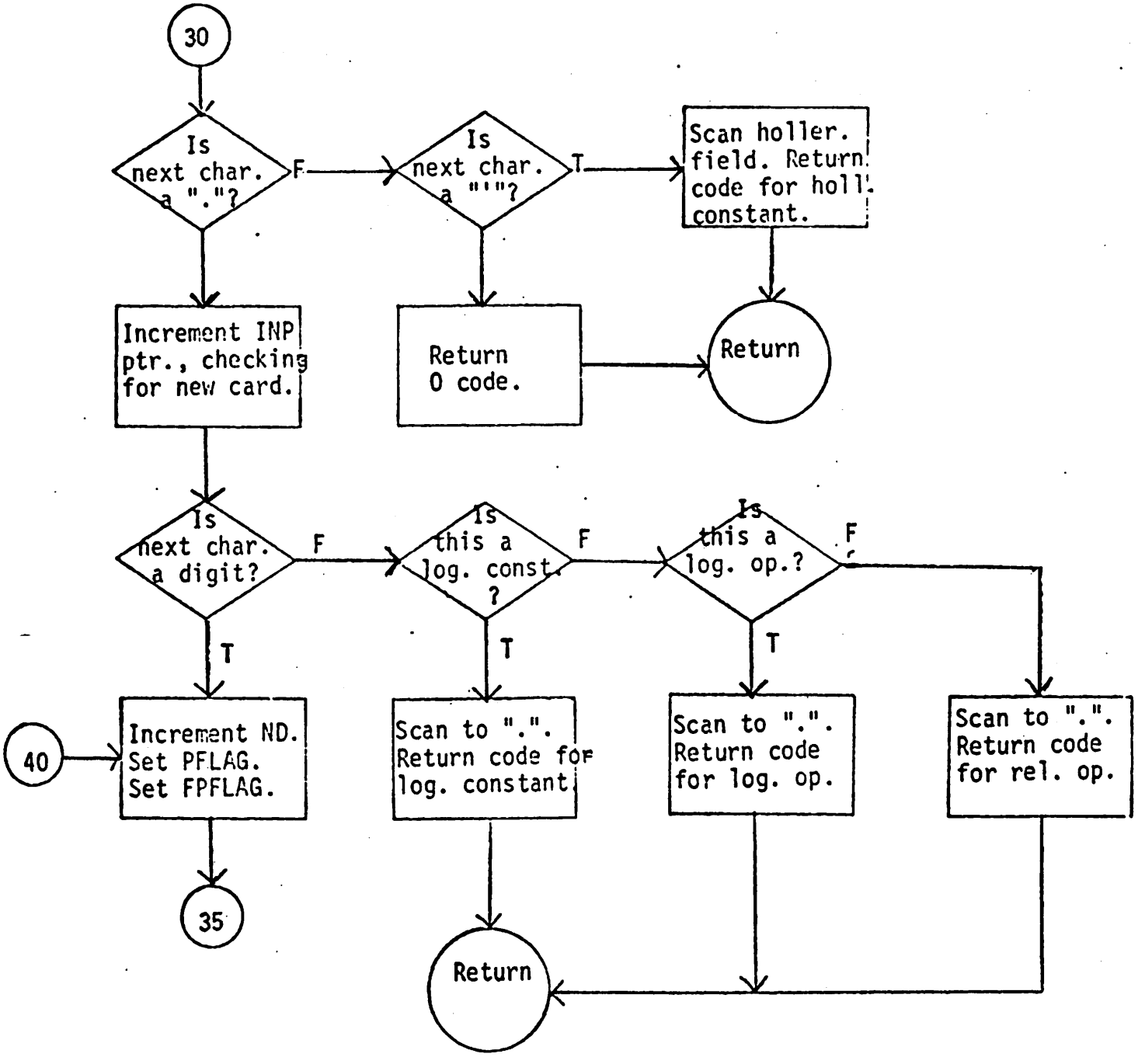
This routine is called by SCAN to check for integer, floating point, double precision, hollerith, and logical constants and relational and logical operators. If none of these is found, the output parameter IP is set to 0. Otherwise it will be set to one of the values listed below:

Integer	1
Floating Point	2
Double Precision	3
Hollerith Constant	4
Logical Operator	5
Lggical Constant	6
Relational Operator	9

Flowchart for CONS




10
Is next char. a "Q"?
F
2
Set INP ptr. back to first ".". Return code for integer.
Return
15
Is next char. a "D"?
T
Set DFLAG.
20
F
Is next char. an "H"?
F
T
Is HFLAG set?
F
T
Convert integer. Scan holl. field. Return code for holl. constant.
25
Is DFLAG set?
T
Return code for double prec. constant.
F
Is FPFLAG set?
T
Return code for float. point constant.
F
Return code for integer constant.


30
Is next char. a "."?
F
Is next char. a "'"?
T
Scan holler. field. Return code for holl. constant.
Increment INP ptr., checking for new card.
Return 0 code.
Return
Is next char. a digit?
F
Is this a log. const. ?
F
Is this a log. op.?
F
T
T
T
40
Increment ND. Set PFLAG. Set FPFLAG.
Scan to ".". Return code for log. constant
Scan to ".". Return code for log. op.
Scan to ".". Return code for rel. op.
35
Return

FUNCTION CONST(N1,N2)

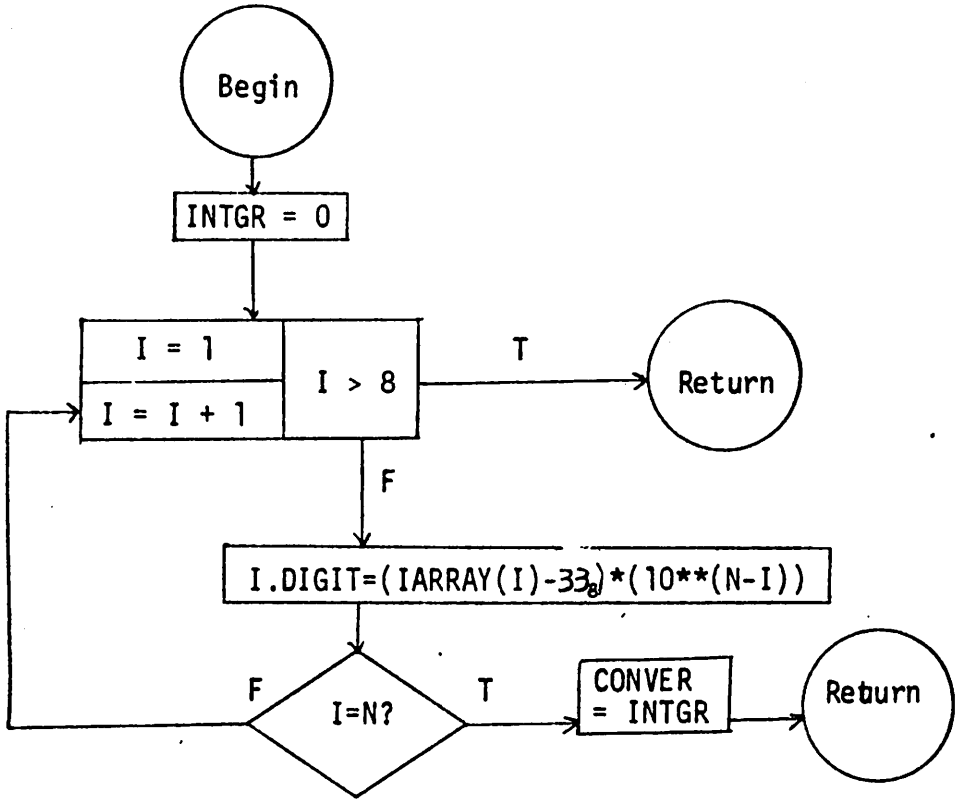
Function CONST performs transformation from the character code of an integer to the value of that integer. Input parameters N1,N2 contain the character codes as an 8 character string - 4 characters per word, left-justified, blank filled. The integer value is returned as the value of the function.

CONST is machine dependent, since it converts character codes to value. In particular, the CDC character codes assumed are 55_8 for blank and 33_8-44_8 for digits 0-9. Each character code for characters making up the integer is extracted by the FLD function.

Integer Function CONVER (IARRAY,N)

This function converts a string of digits (in character code) from input array IARRAY to their integer value. Input parameter N indicates the number of digits to be converted (maximum 8). The integer value is returned as the value of the function.

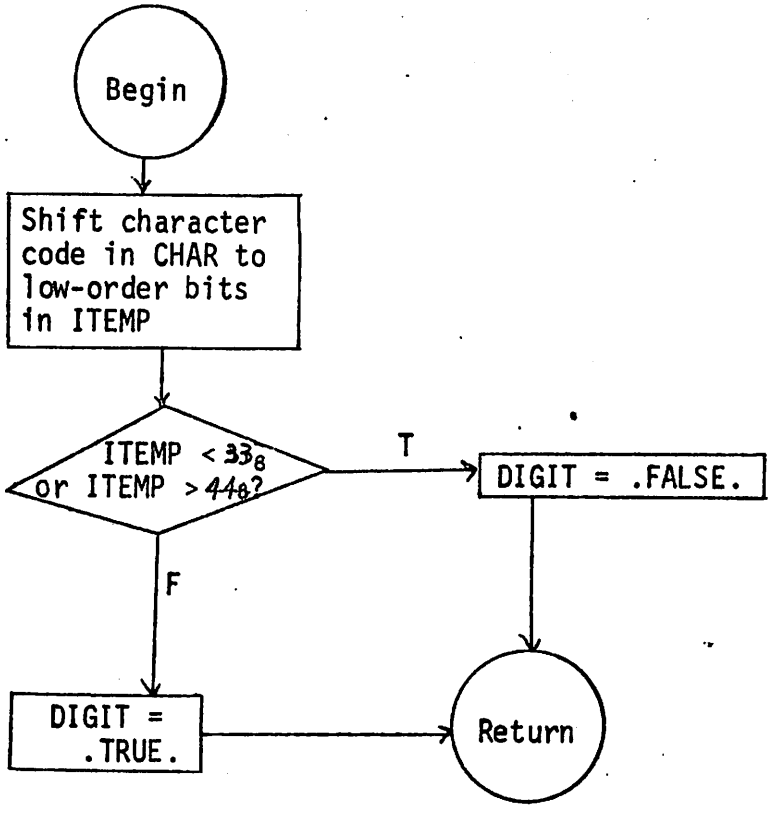
Flowchart for CONVER



Logical Function DIGIT (CHAR)

This function determines whether the input parameter CHAR is a digit 0-9. If it is, the value of the function returns as .TRUE. . Otherwise, the value is .FALSE. .

Flowchart for DIGIT



SUBROUTINE ERROR(L)

Subroutine ERROR is called by one of the parsing routines when an error condition is detected. These errors are generally syntax errors (in the source statement being processed) or conditions which cannot be resolved by the parsing routines. If such a condition occurs, the general procedure is to call ERROR to store an error message indicator (L) for the particular statement - identified by statement number (STNO) - and to ignore the remainder of the statement.

Parameter L is an index to a table of error messages (see documentation for PARSER). Array NERROR consists of two-word entries containing a message number (L) and statement number (STNO).

When called with L=999, ERROR prints the messages accumulated for the FORTRAN module being processed.

SUBROUTINE FARITH

Subroutine FARITH is called from PARSER to process arithmetic and logical statements (also referred to as assignment statements), which have the form

v=e

where v is a subscripted or unsubscripted variable and e is an arithmetic or logical expression. FARITH also processes statement function definitions, which have the form

$f(a_1,a_2,\ldots,a_n)=e$

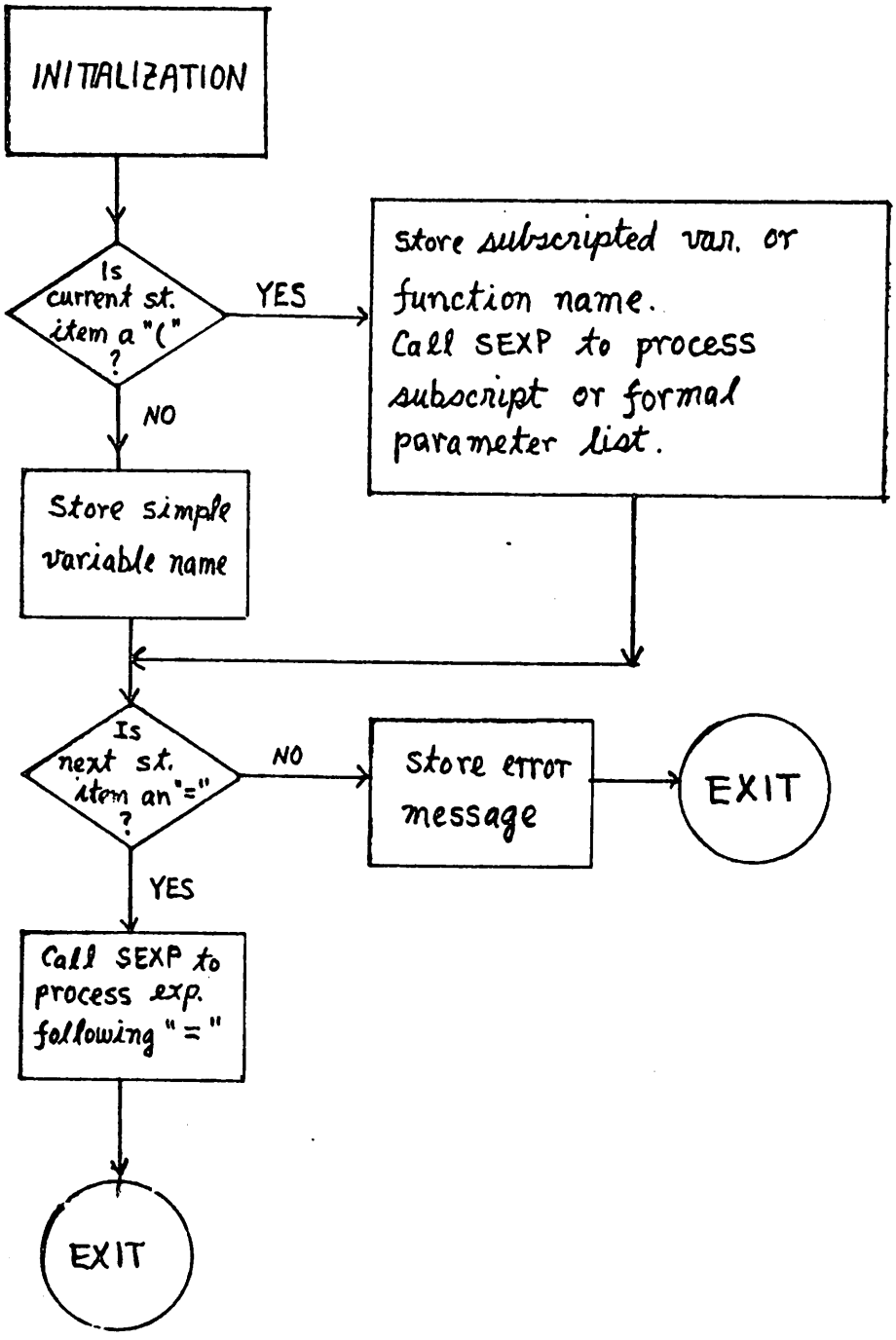
where f is the function name, $a_1,a_2,\ldots,a_n$ are dummy parameters, and e is an expression.

If v is a simple (unsubscripted) variable, it is passed to subroutine TABS with class code 6 (variable), type code 5 (neutral), and use code 1 (output variable in assignment statement). Otherwise, the entity v (or f) in the above form is passed with class code 4 (function name), type code 5 (neutral), and use code 1 (output variable in assignment statement). In this latter case, the ambiguity between variable name and function name is resolved by TABS, i.e. if the name has been declared as an array name, the class code (KC) is changed to 3 (array name). The value of KC after the call to TABS then determines whether the following parenthesized list contains subscripts (use code 15) or dummy parameters (use code 16); the value of parameter MODE is set to 15 or 16 accordingly, and the list is processed by subroutine SEXP. If the list is a parameter list, a special

call to TABS (with MPOP=1) is made to indicate the end of the list.

The expression e is also processed by subroutine SEXP. The parameter MODE to SEXP is set to 2 if the statement is an assignment statement, indicating that names found in the expression are used as input to an assignment statement. If the statement is a function definition, MODE is set to 16, indicating that names found in the expression are dummy parameters.

FARITH


INITIALIZATION
Is current st. item a "(" ?
YES
NO
Store subscripted var. or function name. Call SEXP to process subscript or formal parameter list.
Store simple variable name
Is next st. item an "=" ?
NO
Store error message
EXIT
YES
Call SEXP to process exp. following "="
EXIT

SUBROUTINE FASS

Subroutine FASS is called from PARSER to process ASSIGN statement, which have the following form:

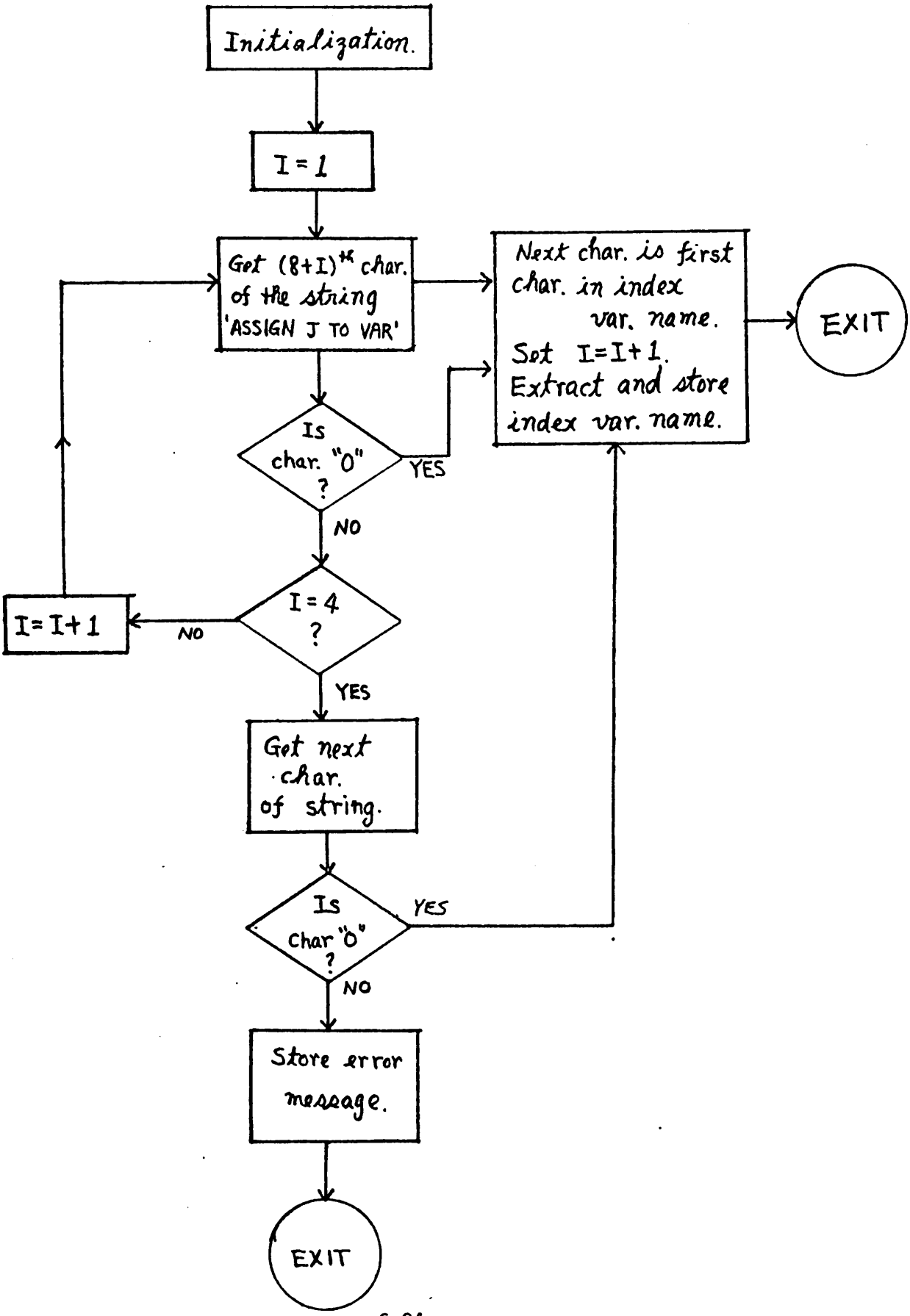
ASSIGN n TO m

where n is a statement number and m is a switch variable.

FASS currently ignores n and stores m (passes m to TABS) with class code 6 (variable), type code 5 (neutral), and use code 22 (variable reference in ASSIGN statement).

Since the ASSIGN statement appears as one continuous alphanumeric string (in the TSTAB array), subroutine SHIFTY is used to determine the first character of m (first character following the "O" in "TO") and to extract the characters of m.

FASS


Initialization.
I = 1
Get (8+I)th char. of the string 'ASSIGN J TO VAR'
Is char. "0" ?
YES
NO
Next char. is first char. in index var. name.
Set I=I+1.
Extract and store index var. name.
EXIT
I = 4 ?
NO
I= I+1
YES
Get next char. of string.
Is Char "0" ?
YES
NO
Store error message.
EXIT

SUBROUTINE FCALL

Subroutine FCALL is called from PARSER to process FORTRAN CALL statements, which are assumed to have the form:

CALL S(a_1,a_2,...,a_n)

or CALL S

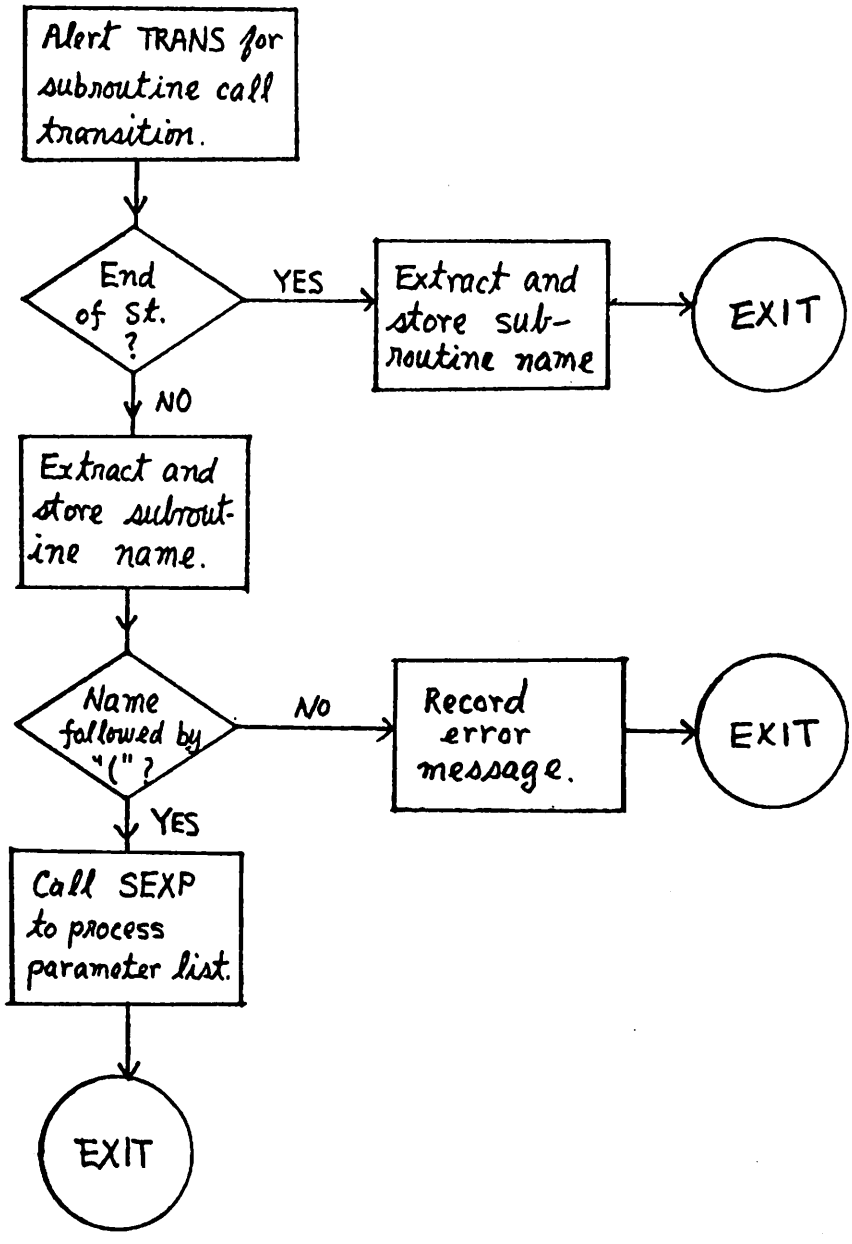
where S is the subroutine name or entry point and a_1,a_2,...,a_n are the actual arguments. Additionally, a check is made for the form

CALL EXIT.

The subroutine name or entry point name is passed to subroutine TABS with class code 1 (subroutine name), type code 5 (neutral), and use code 21 (external procedure reference in CALL statement). The argument list is processed by subroutine SEXP with parameter MODE=19, indicating list items are subroutine actual parameters.

A transition from the CALL statement to an external point (subroutine or entry point) is created in the program graph through a call to subroutine TRANS (with MOD=5). If a CALL EXIT is recognized, a call to TRANS is made with MOD=3.

FCALL


Alert TRANS for subroutine call transition.
End of St. ?
YES
Extract and store sub-routine name
EXIT
NO
Extract and store subrout-ine name.
Name followed by "(" ?
NO
Record error message.
EXIT
YES
Call SEXP to process parameter list.
EXIT

SUBROUTINE FCOMON

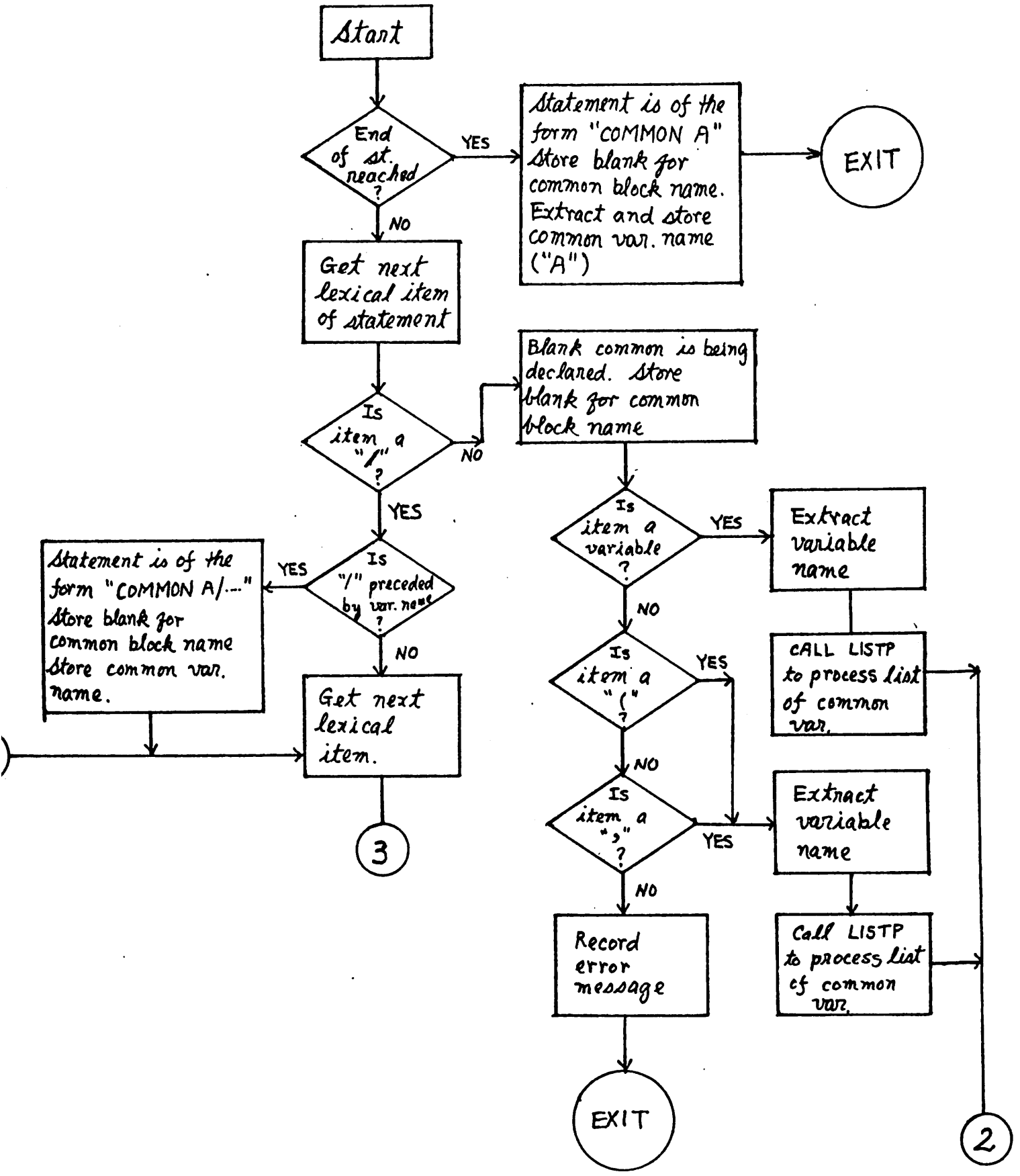
$$\text{COMMON}/X_1/a_1, a_2, \ldots, a_n/X_2/a_n+1, \ldots, a_m$$

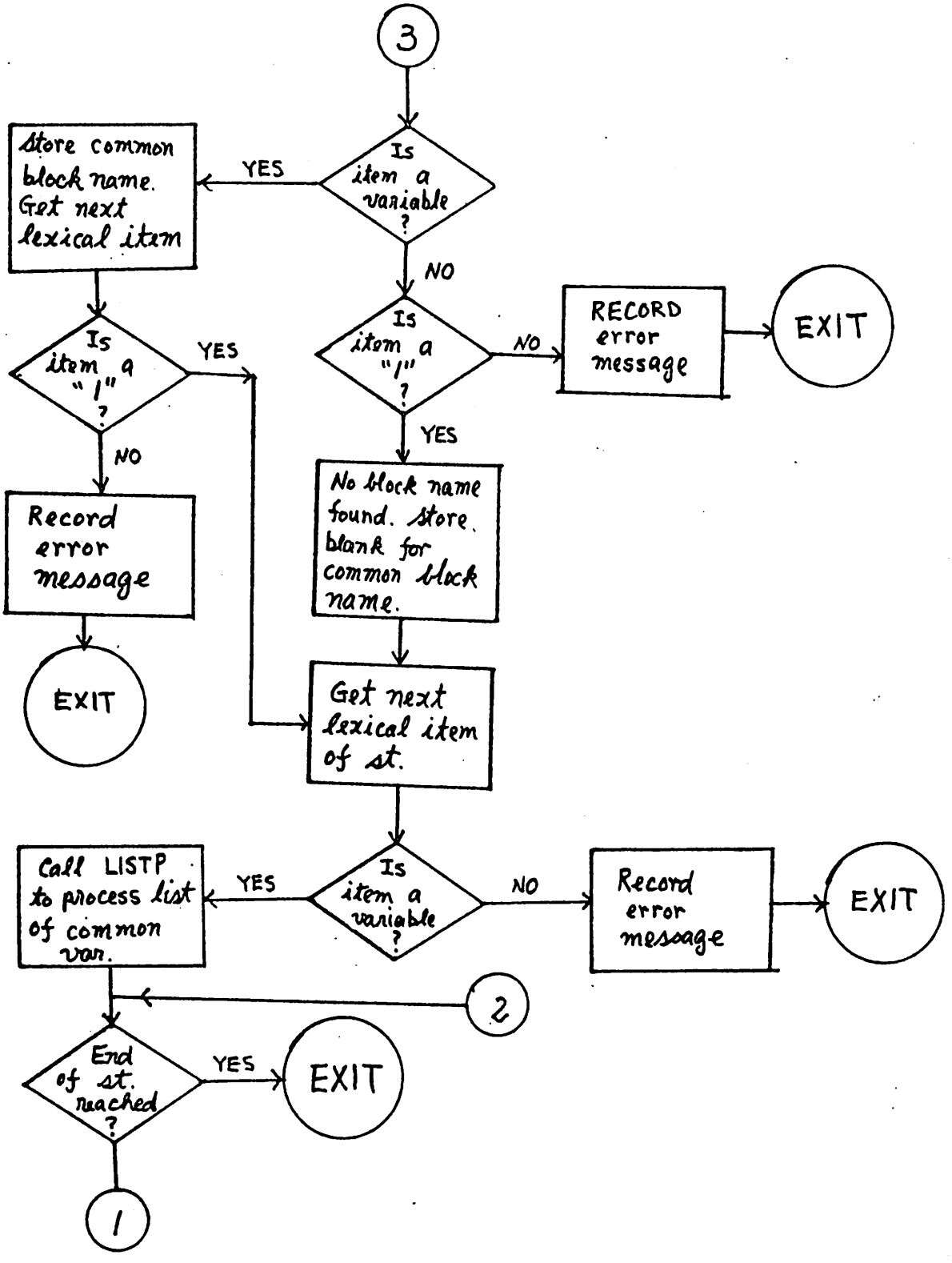
where X_i represents a block name and a_i represents a variable or array name or array declaration. This form may appear somewhat different when variables are assigned to blank common; first, the string "/X_1/" may be omitted so that the list $a_1, a_2, \ldots, a_n$ is assigned to blank common, and secondly the appearance of two slashes with no block name between them assigns the variables which follow to blank common. Recognition of the various combinations of these forms is the major function of FCOMON. If blank common is being assigned, a block name consisting of all blanks is passed to subroutine TABS, indicating that the succeeding variables are assigned to blank common.

Each variable or array list ($a_1, a_2, \ldots, a_n$) is processed by subroutine LISTP. FCOMON extracts the first variable of a list, sets the use code (11 for common block entry) and passes control to LISTP, which processes the list and returns to FCOMON when a slash or end of statement is reached.

Common block names are recognized by FCOMON and passed to subroutine TABS with class code 7 (common block name) and use code 0 (declaration).

FCOMON


Start
End of st. reached ?
YES
NO
Statement is of the form "COMMON A" Store blank for common block name. Extract and store common var. name ("A")
EXIT
Get next lexical item of statement
Is item a "/" ?
NO
Blank common is being declared. Store blank for common block name
YES
Is "/" preceded by var. name ?
YES
Statement is of the form "COMMON A/..." Store blank for common block name Store common var. name.
NO
Get next lexical item.
3
Is item a variable ?
YES
Extract variable name
CALL LISTP to process list of common var.
NO
Is item a "(" ?
YES
NO
Is item a "," ?
YES
Extract variable name
Call LISTP to process list of common var.
NO
Record error message
EXIT
2


3
Is item a variable ?
YES
Store common block name. Get next lexical item
NO
Is item a "/" ?
NO
RECORD error message
EXIT
YES
No block name found. Store blank for common block name.
Is item a "/" ?
YES
NO
Record error message
EXIT
Get next lexical item of st.
Is item a variable ?
YES
Call LISTP to process list of common var.
NO
Record error message
EXIT
2
End of st. reached ?
YES
EXIT
1

SUBROUTINE FDATA

Subroutine FDATA is called from PARSER to process FORTRAN DATA statements, which are assumed to have the general form

$$\text{DATA } v_1/\ell_1/v_2/\ell_2/, \ldots, /v_n/\ell_n/$$

$$\text{or DATA } v_1/\ell_1, v_2/\ell_2/, \ldots, v_n/\ell_n/ .$$

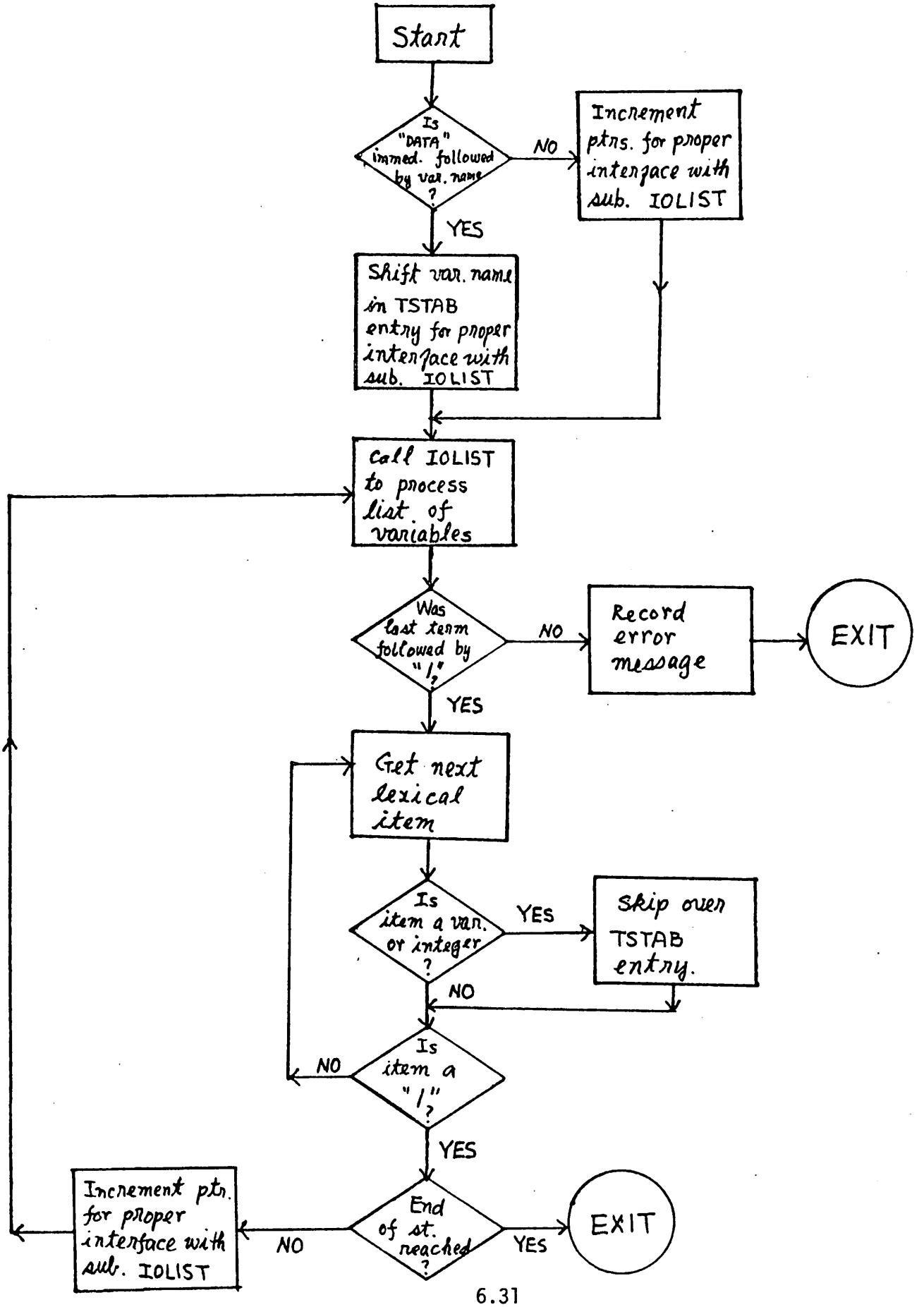
where v_i is a variable list and ℓ_i is a literal list. The variable lists are processed by subroutine IOLIST (IOLIST is called once for each variable list). Literal lists are simply ignored.

In order to interface with IOLIST, the string DATA stored in the first row of TSTAB is eliminated. The first data variable is shifted left to fill the empty space. When IOLIST is called, JTST and JISS (pointers associated with TSTAB and ISS resp.) point to the first data variable in the list.

For example: DATA NUMBER, ALPHA /1,2.0/

UPON ENTRY		BEFORE IOLIST IS CALLED	
TSTAB	ISS	TSTAB	ISS
DATA NUMB	V	NUMB ER	V
ER	V		V
ALPHA A	,	ALPHA A	,
	V		V
	/		/
	I		I
	,		,
	F		F
	/		/

FDATA


Start
Is "DATA" immed. followed by var. name ?
NO
Increment ptrs. for proper interface with sub. IOLIST
YES
Shift var. name in TSTAB entry for proper interface with sub. IOLIST
Call IOLIST to process list of variables
Was last term followed by "/" ?
NO
Record error message
EXIT
YES
Get next lexical item
Is item a var. or integer ?
YES
Skip over TSTAB entry.
NO
Is item a "/" ?
NO
YES
End of st. reached ?
NO
YES
EXIT
Increment ptr. for proper interface with sub. IOLIST

SUBROUTINES FDIME

Subroutines FDIME is called from PARSER to process FORTRAN DIMENSION statements:

DIMENSION $v_1/\ell_1/v_2/\ell_2/ \ldots, v_n/\ell_n/$

where v_i represents a list of array declarations, and ℓ_i represents a literal list. Since the v_i and ℓ_i lists appear the same as those in type statements, they are processed by subroutine LISTP, the list processor for type statements. Thus the only function of FDIME is the extraction of the first variable in the list and setting of the use code (13 for use in DIMENSION statement). This information is passed to LISTP, which processes the remainder of the statement.

SUBROUTINES FDO AND DORANG

Subroutine FDO is called from PARSER to process FORTRAN DO statements, which are assumed to have the form

DO n i=m_1,m_2,m_3

where n is the label of the terminal statement of the DO loop, i is the index variable, and m_1,m_2,m_3 are the initial, terminal, and incrementation parameters, respectively. Index variable i is assumed to be an unsigned, unsubscripted, integer variable, and each index parameter, m_i, may be a signed or unsigned integer constant, a signed or unsigned unsubscripted integer variable, or a signed or unsigned PARAMETER variable.

Subroutine FDO extracts the label and index variable from the DO statement and passes them to subroutine TABS with the proper codes for storage in the FACES tables. Since the portion "DO n i" is stored (by SCAN) in the temporary storage table (TSTAB) as one continuous character string, subroutine SHIFTY is called to extract the character strings for "n" and "i". Subroutine DORANG is then called to process the range parameters m_1,m_2,m_3.

DO statement entities with symbolic names are stored in the FACES tables by subroutine TABS just as those found in other FORTRAN statements. However, an additional table (DOTAB) is maintained for easy access to information regarding DO loops found in a FORTRAN module. DOTAB contains a 3-word record for each DO loop and has the following form:

DOTAB (1, I) :	Statement number of DO statement	Pointer to SYMTAB for label reference
DOTAB (2, I) :	* + Incrementation parameter	Pointer to SYMTAB for index variable
DOTAB (3, I) :	* + Terminal parameter	* + Initial parameter
	½ word	½ word

* 1-bit indicator -

  0 indicates contents is an integer constant

  1 indicates contents is a pointer to a symbolic name in SYMTAB

+ 1-bit indicator -

  0 indicates parameter is unsigned or is preceded by "+" in the DO statement

  1 indicates parameter is preceded by "-" in the DO statement

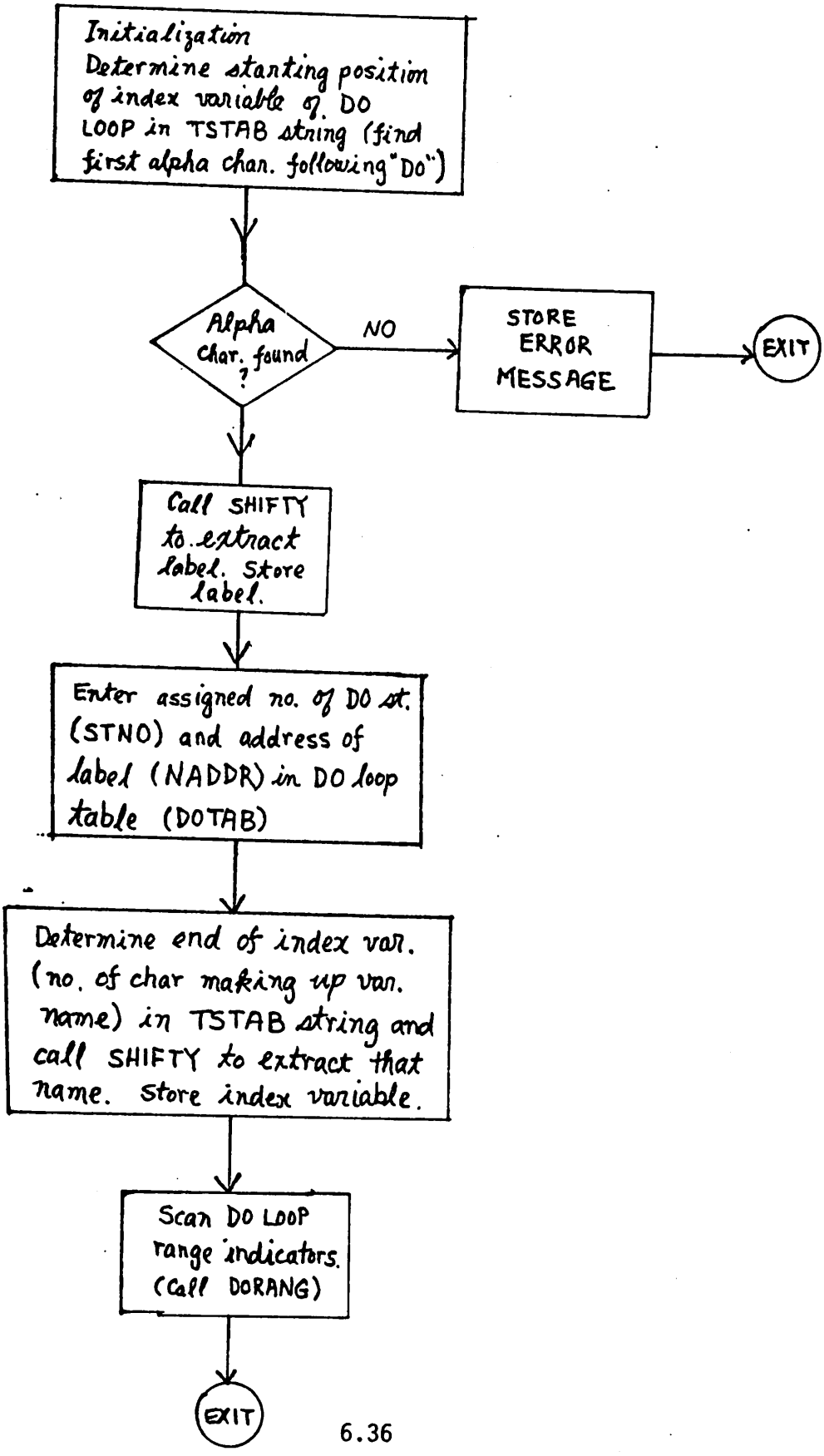
The statement number of the DO statement is the sequential number assigned by FACES. The label reference and index variable refer to "n" and "i" in the above statement form - their symbolic names are found in SYMTAB at the locations specified by the DOTAB entries. The initial, terminal, and incrementation parameter refer to "m_1,m_2,m_3" in the above statement form. For each parameter, if a constant is found, the value of that constant is stored in DOTAB; if a variable is found, the address to SYMTAB for the symbolic name is stored and the indicator bit is set. An additional indicator bit is set if a "-" is found preceding the parameter.

Entries in the first word of a DOTAB record are made by subroutine FDO while entries in the second and third words are made by

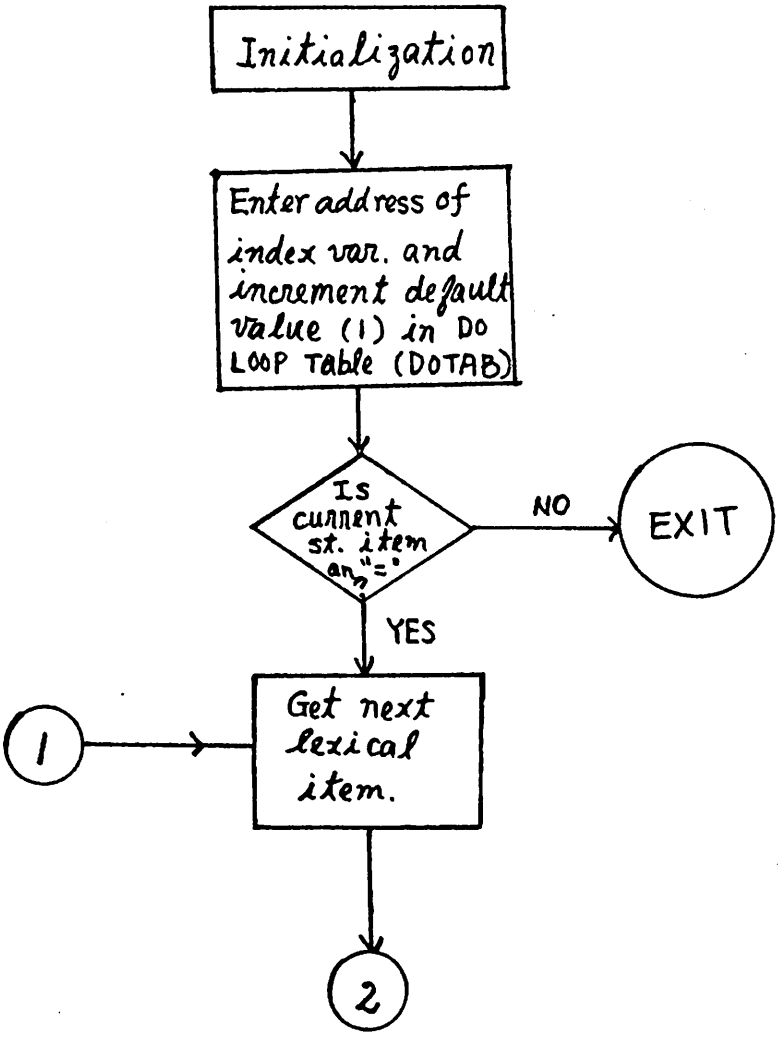
subroutine DORANG. The necessary bit manipulation and shifting are performed by functions BITSET and HISTOR, respectively.

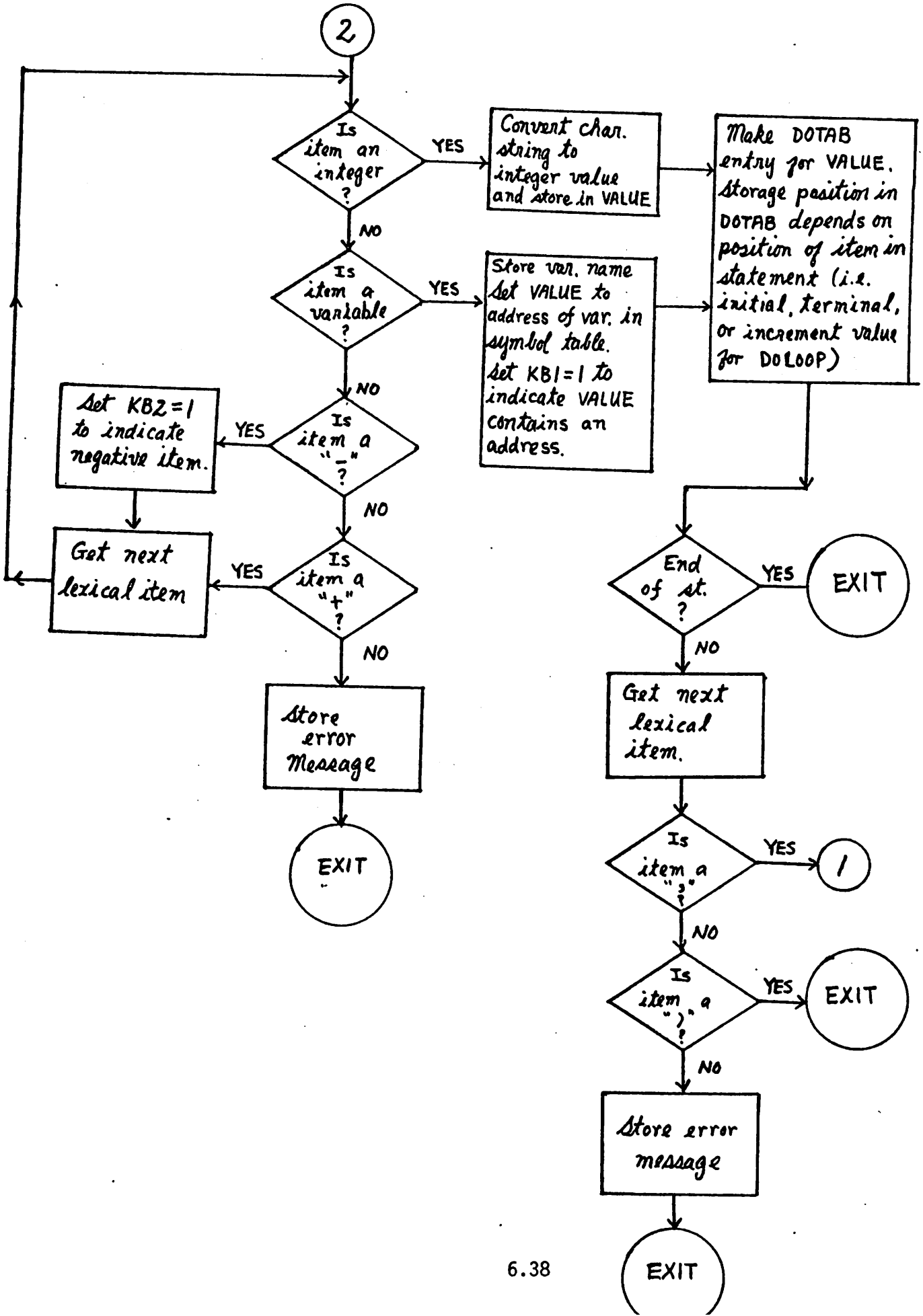
Subroutine DORANG may also be called from subroutine IOLIST when an implied DO loop is encountered in an I/O list. Entries in DOTAB are made in the same manner as in DO statements except that no label reference will be present.

FDO


Initialization
Determine starting position of index variable of DO LOOP in TSTAB string (find first alpha char. following "DO")
Alpha Char. found ?
NO
STORE ERROR MESSAGE
EXIT
Call SHIFTY to extract label. Store label.
Enter assigned no. of DO st. (STNO) and address of label (NADDR) in DO loop table (DOTAB)
Determine end of index var. (no. of char making up var. name) in TSTAB string and call SHIFTY to extract that name. Store index variable.
Scan DO LOOP range indicators. (Call DORANG)
EXIT

DORANG


Initialization
Enter address of index var. and increment default value (1) in DO LOOP Table (DOTAB)
Is current st. item an "="?
NO
EXIT
YES
Get next lexical item.
1
2


2
Is item an integer ?
YES
Convert char. string to integer value and store in VALUE
Make DOTAB entry for VALUE. Storage position in DOTAB depends on position of item in statement (i.e. initial, terminal, or increment value for DOLOOP)
NO
Is item a variable ?
YES
Store var. name Set VALUE to address of var. in symbol table. Set KB1=1 to indicate VALUE contains an address.
NO
Is item a "-" ?
YES
Set KB2=1 to indicate negative item.
NO
Is item a "+" ?
YES
Get next lexical item
NO
Store error Message
EXIT
End of st. ?
YES
EXIT
NO
Get next lexical item.
Is item a "," ?
YES
1
NO
Is item a ")" ?
YES
EXIT
NO
Store error message
EXIT

SUBROUTINE FENTRY

Subroutine FENTRY is called from PARSER to process ENTRY statements, which have the form:

ENTRY ff(b_1,b_2,...,b_n)

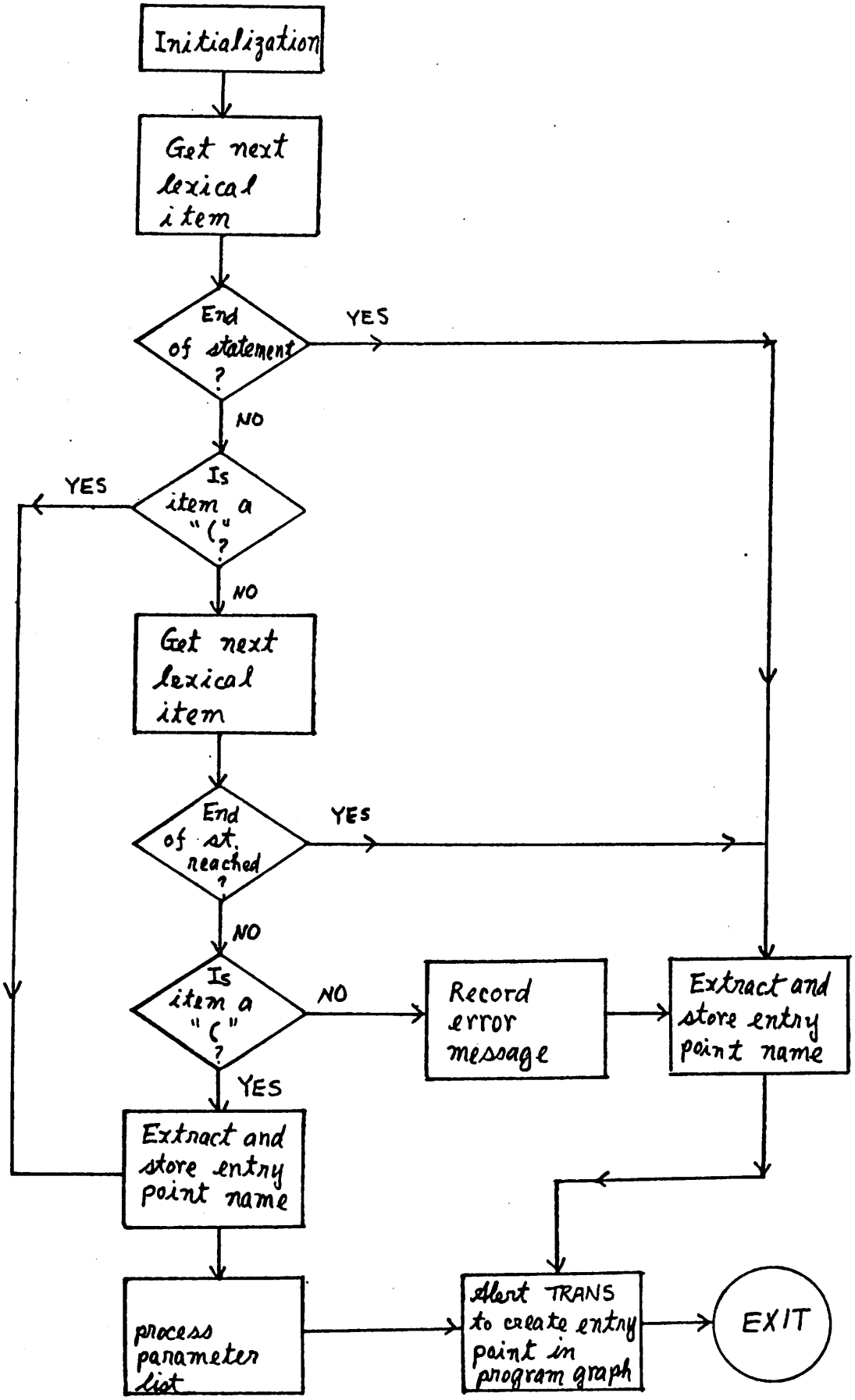
where ff is the symbolic name of the entry point and b_1,b_2,...,b_n is a list of formal (dummy) parameters.

The entry point name is extracted from the symbol string in TSTAB by subroutine SHIFTY and passed to subroutine TABS with class code 9 (entry point name), type code 5 (neutral), and use code 0 (declaration).

Formal parameters are recognized by FENTRY and passed to TABS with class code 6 (variable), type code 5 (neutral) and use code 16, 17 (function or subroutine dummy parameters). A special call to TABS is made after the last name is processed with MPOP = 1 to indicate the end of the parameter list.

Finally a call is made from FENTRY to subroutine TRANS (with LOC=0 and MOD=7) to record a transition from an external point to the ENTRY statement.

FENTRY


Initialization
Get next lexical item
End of statement ?
YES
NO
Is item a "(" ?
YES
NO
Get next lexical item
End of st. reached ?
YES
NO
Is item a "(" ?
NO
YES
Record error message
Extract and store entry point name
Extract and store entry point name
process parameter list
Alert TRANS to create entry point in program graph
EXIT

SUBROUTINE FGOTO

Subroutine FGOTO is called from PARSER to process FORTRAN GOTO statements, which are assumed to have the following forms:

Unconditional:		GO TO n
Assigned	:	GO TO m
		GO TO m, $(n_1,n_2,\ldots,n_k)$
Computed	:	GO TO $(K_1,k_2,\ldots,k_j)$,i

where n is a statement number, m is a switch variable, $n_1,n_2,\ldots,n_k$ is a list of statement numbers, $k_1,k_2,\ldots,k_j$ are statement numbers or switch variables, and i is a nonsubscripted integer variable.

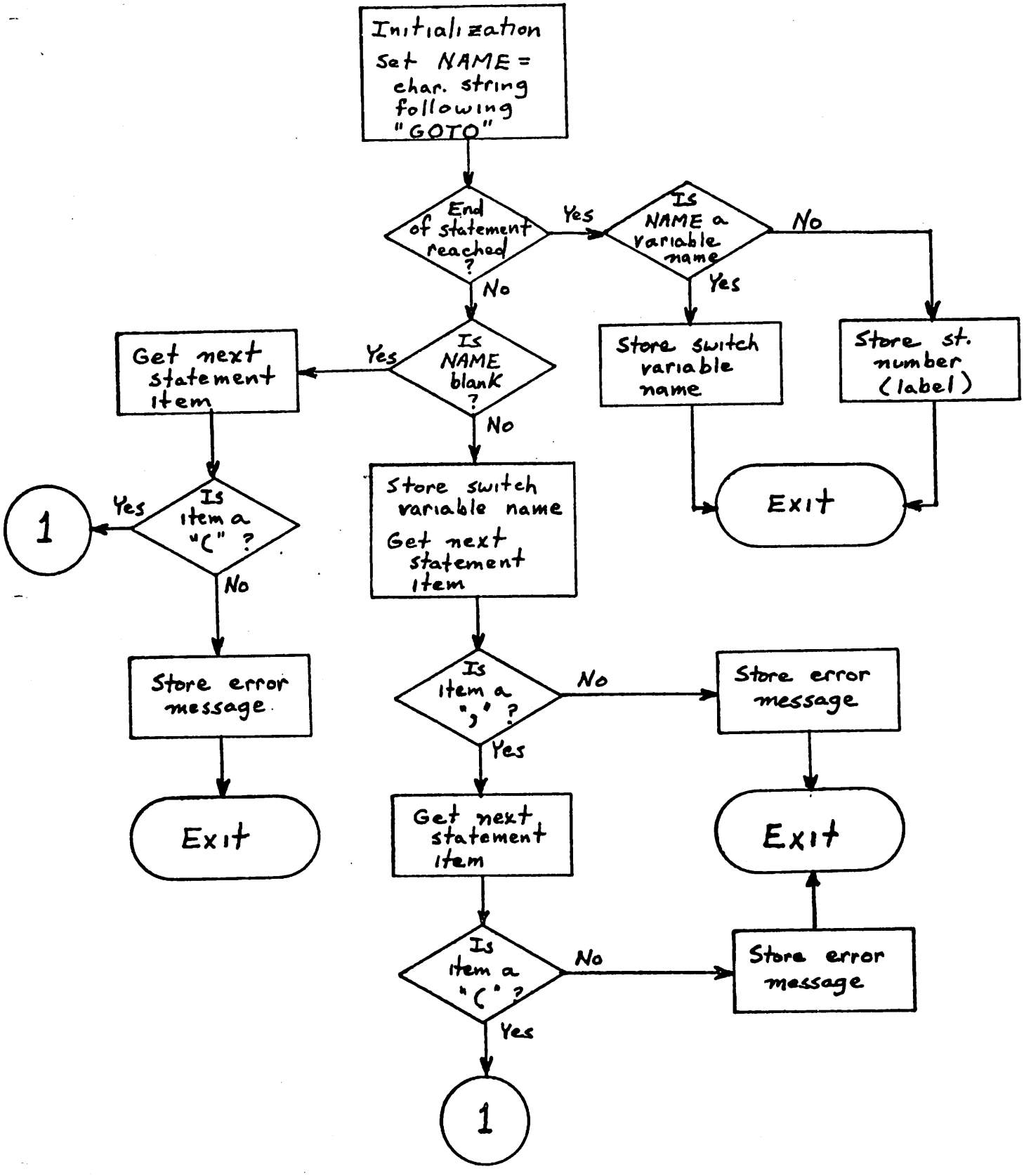
The above forms are recognized by FGOTO and the various entities found are passed to subroutine TABS for proper storage in the FACES tables; TABS also initiates procedures for recording the proper transitions in the program graph.

The following summarizes the information passed to subroutine TABS for the entities in the above forms.

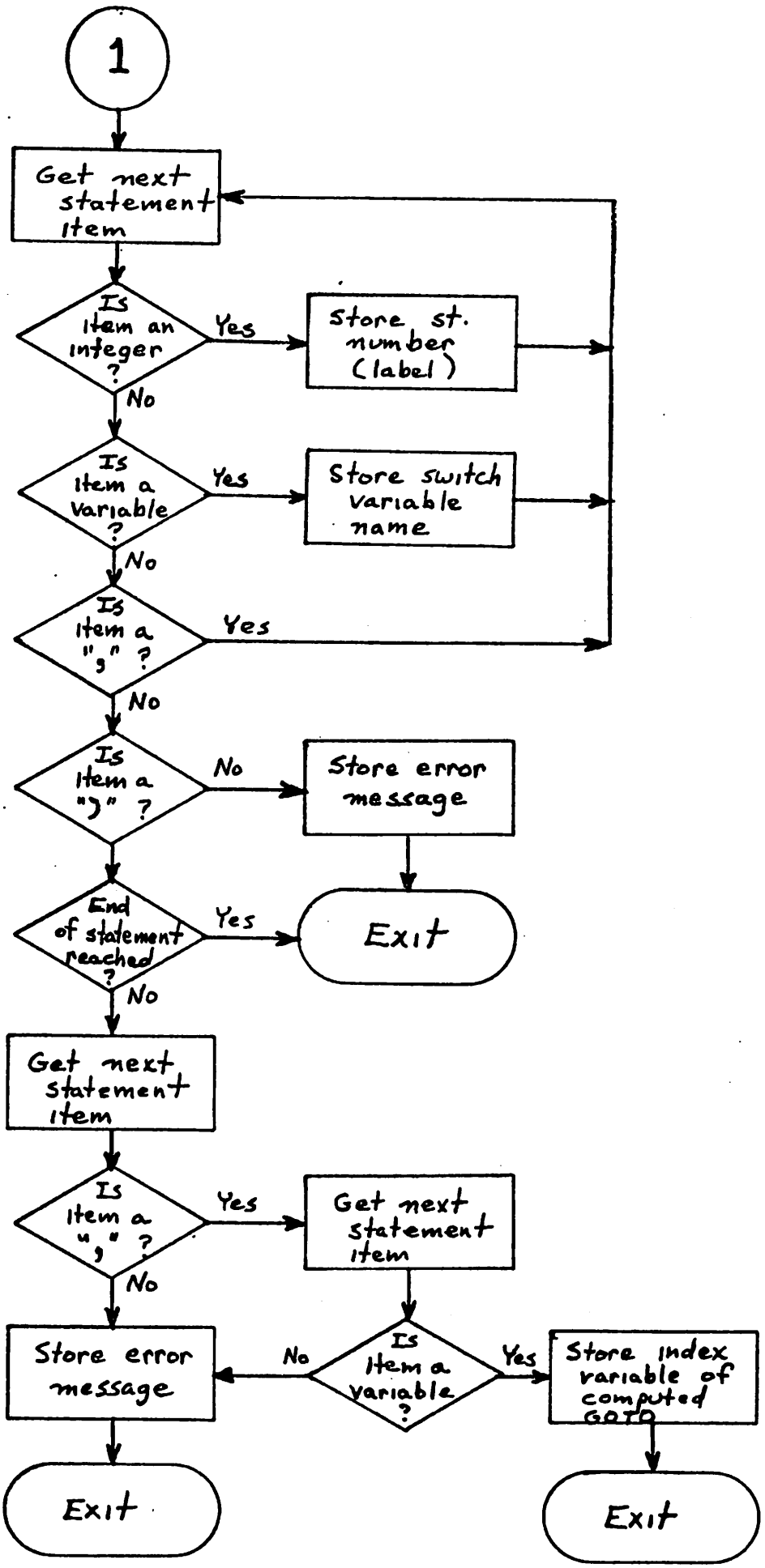
n	class code 5 (label)
	type code 5 (neutral)
	use code 10 (label reference)
m	class code 6 (variable)
	type code 5 (neutral)
	use code 23 (variable reference in assigned GO TO)

$n_1, n_2, \ldots, n_k$	class code 5 (label)
	type code 5 (neutral)
	use code 10 (label reference)
$k_1, k_2, \ldots, k_j$	
if st. number:	class code 5 (label)
	type code 5 (neutral)
	use code 10 (label reference)
if switch variable:	class code 6 (variable)
	type code 5 (neutral)
	use code 24 (variable reference in computed GO TO)
i :	class code 6 (variable)
	type code 5 (neutral)
	use code 24 (variable reference in computed GO TO)

FGOTO


Initialization
Set NAME = char. string following "GOTO"
End of statement reached ?
Yes
Is NAME a variable name
No
Yes
No
Is NAME blank ?
Yes
Get next statement item
Store switch variable name
Store st. number (label)
Exit
1
Is item a "(" ?
Yes
No
Store switch variable name
Get next statement item
Store error message
Exit
Is item a "," ?
No
Yes
Store error message
Exit
Get next statement item
Is item a ")" ?
No
Yes
Store error message
1

FGOTO (cont.)


1
Get next statement item
Is item an integer ?
Yes
Store st. number (label)
No
Is item a variable ?
Yes
Store switch variable name
No
Is item a "," ?
Yes
No
Is item a ")" ?
No
Store error message
End of statement reached ?
Yes
Exit
No
Get next statement item
Is item a "," ?
Yes
Get next statement item
No
Store error message
No
Is item a variable ?
Yes
Store index variable of computed GOTO
Exit
Exit

SUBROUTINE FIF(PARAM)

Subroutine FIF is called from PARSER to process FORTRAN IF statements, which have one of the following general forms:

Arithmetic IF: IF (⟨arithmetic expression⟩) n_1,n_2,n_3

One Way Logical IF: IF (⟨ logical expression ⟩) ⟨statement⟩

Two Way Logical IF: IF (⟨ logical expression ⟩) n_1,n_2

where n_1,n_2,n_3 are statement numbers or switch variables, or are omitted.

The arithmetic or logical expression is processed by subroutine SEXP; the parameter (MODE) to SEXP is set to 20, indicating names in the expression are stored with use code 20 (conditional branch decision variable).

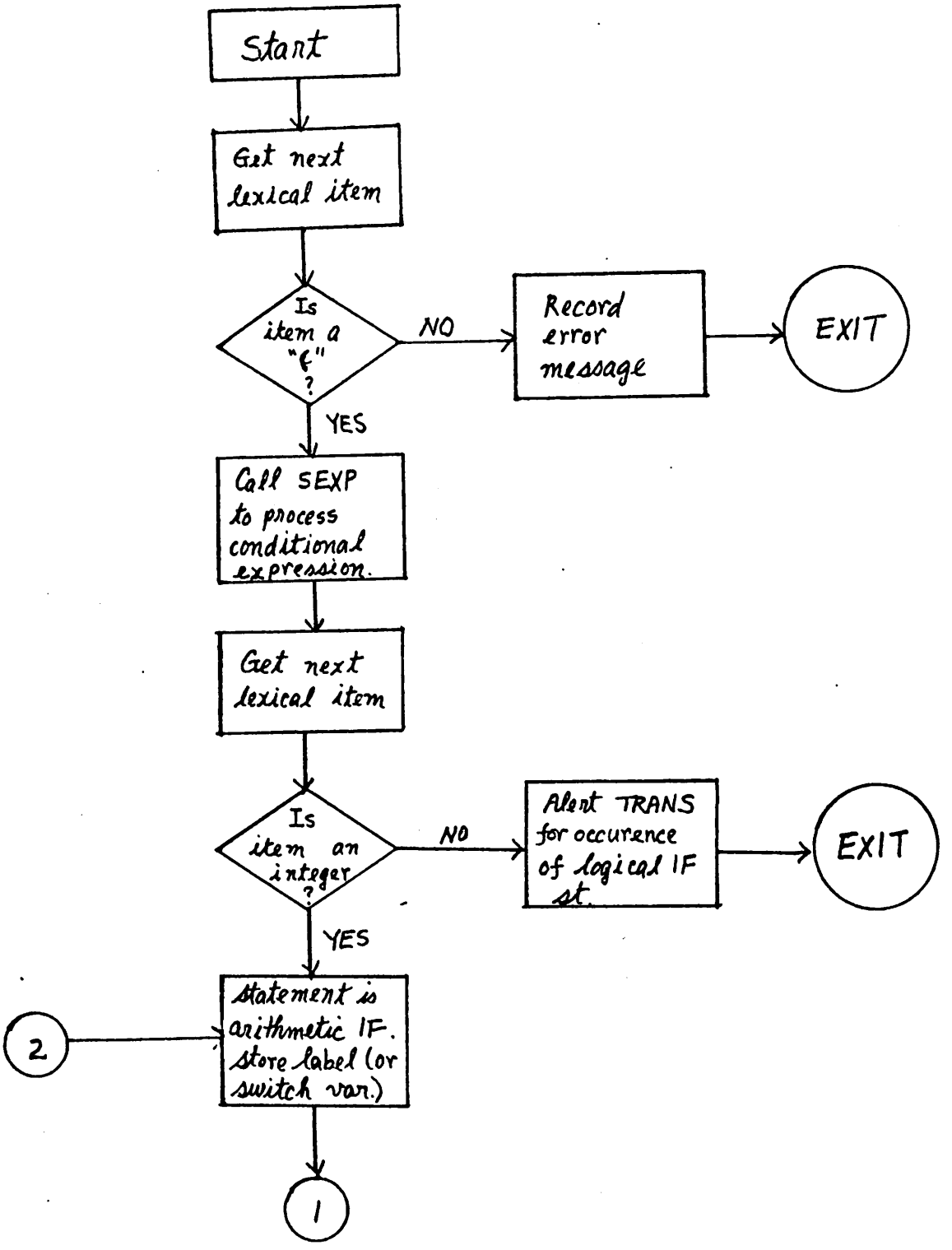
The distinction between an executable statement and statement labels following an IF expression is made by the presence or absence of an integer following the ")" in the above forms, i.e. n_1 is an integer. If n_1 is a switch variable, FIF will currently mistakenly recognize the statement as a logical IF. Since the occurrence was judged to be rare, and since a more involved recognition process is required to resolve the situation, the current version of FACES restricts n_1 to be an integer. However, the necessary modifications could be made if required by more frequent use of switch variables.

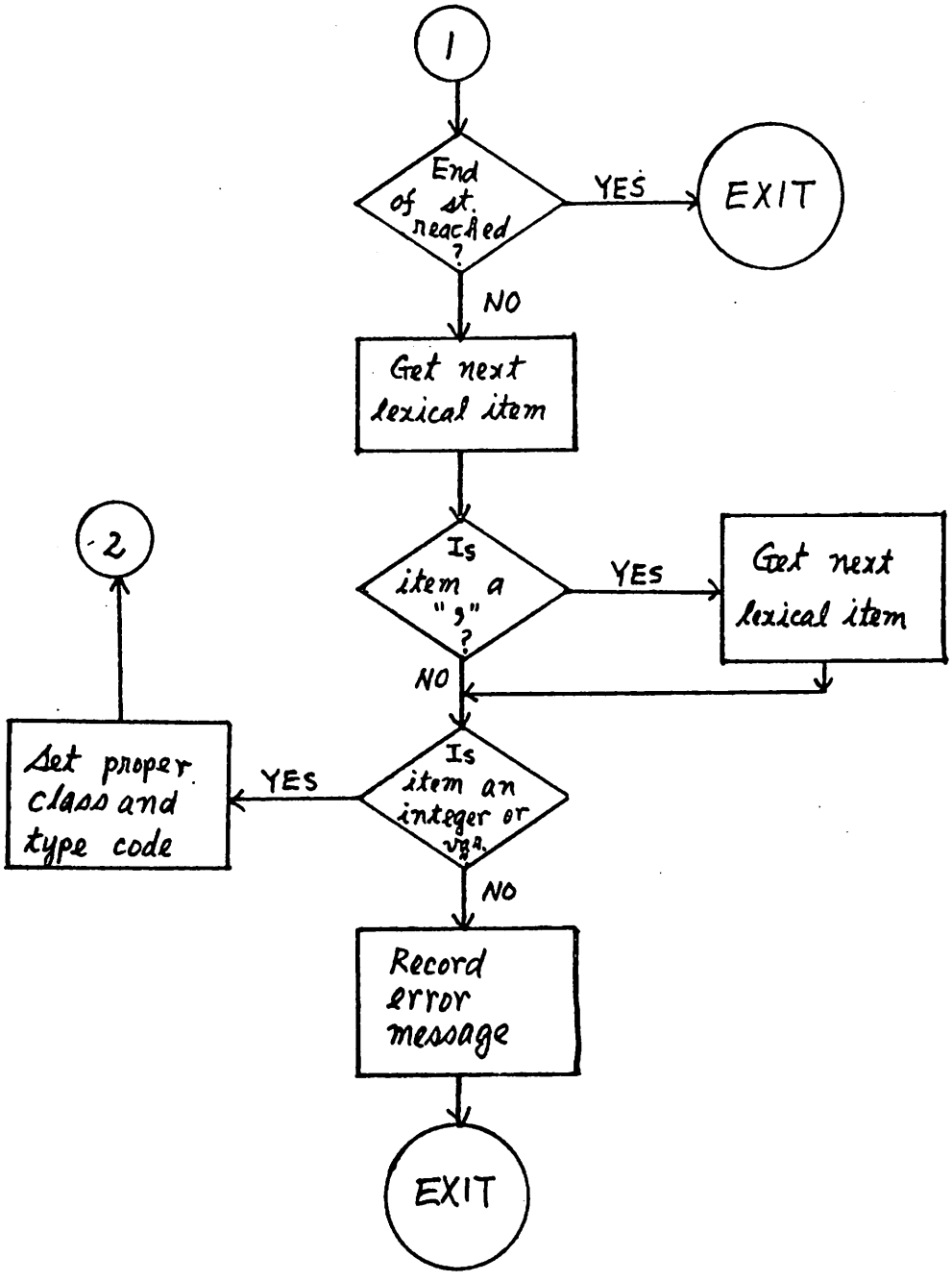
The transfer labels n_2 and n_3 may be statement numbers or switch variables, or may be omitted. If a transfer label is a statement number, it is passed to TABS with class code 5 (label), type code 5 (neutral), use code 10 (label reference). If a transfer label is a switch variable, it is passed to TABS with class code 6 (variable), type code 5 (neutral), and use code 29 (switch variable in IF statement).

If a logical IF statement is recognized, a call (with MOD=4) is made to subroutine TRANS to allow transitions in the program graph from the IF statement to the following two statements. Additionally, the value of parameter PARAM is set to 1 to indicate to PARSER that a logical IF was detected; PARSER then assigns a new number to executable statement , effectively splitting the logical IF into two nodes in the program graph.

Output parameter PARAM remains 0 if an arithmetic IF is detected by FIF.

F I F


Start
Get next lexical item
Is item a "(" ?
NO
Record error message
EXIT
YES
Call SEXP to process conditional expression.
Get next lexical item
Is item an integer ?
NO
Alert TRANS for occurence of logical IF st.
EXIT
YES
2
statement is arithmetic IF. store label (or switch var.)
1


1
End of st. reached ?
YES
EXIT
NO
Get next lexical item
2
Is item a "," ?
YES
Get next lexical item
NO
Set proper class and type code
YES
Is item an integer or var.
NO
Record error message
EXIT

SUBROUTINE FIMP

This routine processes IMPLICIT type statements. A table (array LTRS) of twenty six entries is used to store the information in the IMPLICIT statement. Each entry in the table corresponds to one letter of the alphabet and each contains five one-bit fields:

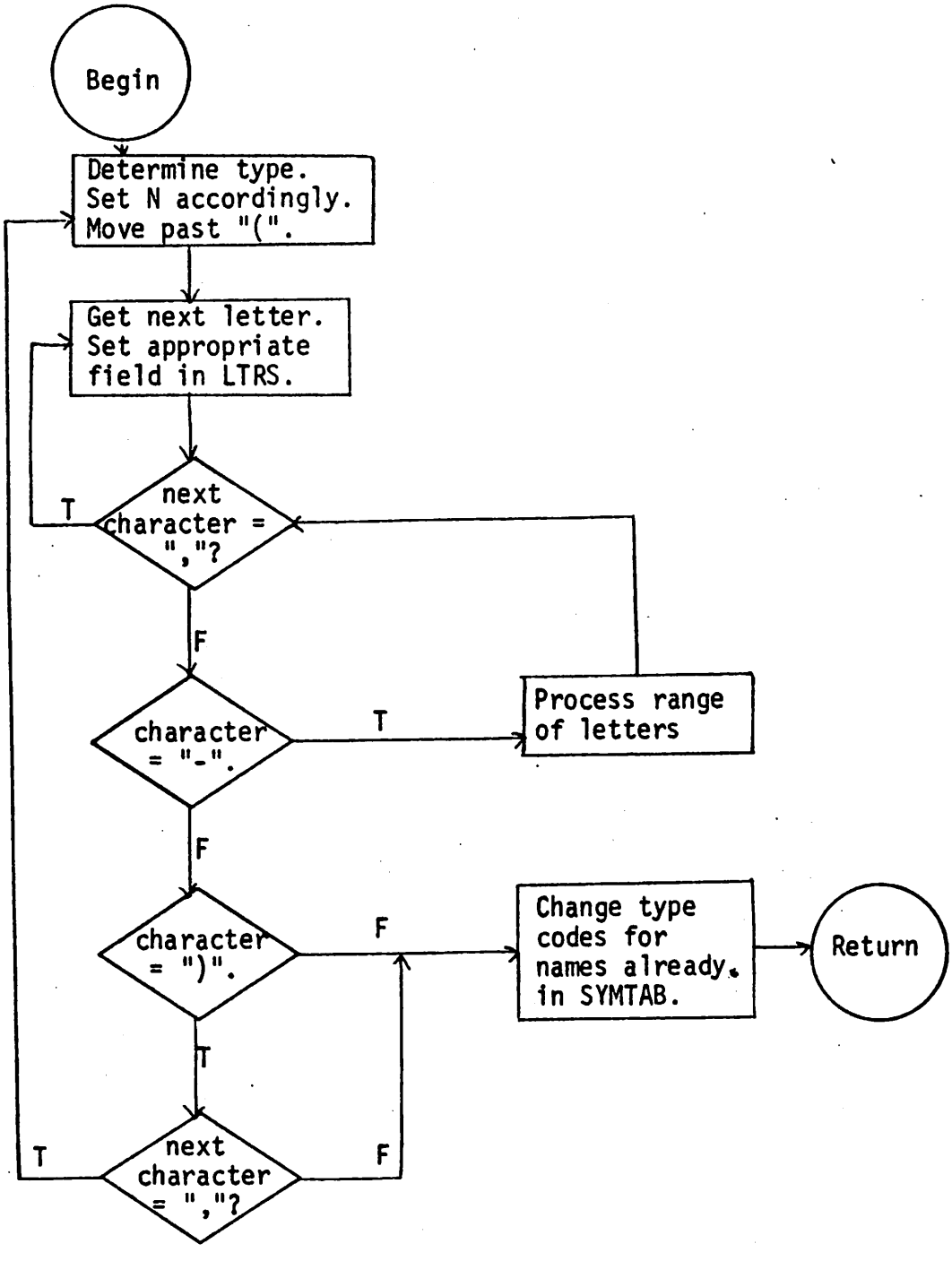
bit set	
1	integer
2	real
3	double precision
4	complex
5	logical

When an IMPLICIT statement is encountered, FIMP scans the statement, determining the type specification and the letters to be implicitly typed. It then marks the appropriate fields in the LTRS table. For example, for the statement

IMPLICIT INTEGER (A-C)

the integer fields of the entries for A, B, and C in the LTRS table would be marked to indicate that the letters are implicitly typed integer.

Flowchart for FIMP



SUBROUTINE FIRST

Subroutine FIRST is called from PARSER to perform a pre-scan of the first statement of a FORTRAN program module. Its purpose is to prepare the parser to parse the next module. Cards that are not module declaration statements (program, subroutine, function or block data statements) are ignored by FIRST.

Module identification is accomplished in the following manner. Subroutines, functions, and block data are recognized by the key words SUBROUTINE, FUNCTION, and BLOCK DATA in the first statement. Since a FUNCTION statement may also contain a type specification, a check is made for the key words INTEGER, LOGICAL, COMPLEX, DOUBLE PRECISION, REAL preceding FUNCTION.

If a subroutine, function, or block data module is recognized, no action is taken, since module identification information is assigned in FSUB, FUNCI, or PARSER for these module types. If none of these modules is recognized, the subsequent statements are ignored (skipped) until a module declaration statement is reached.

INTEGER FUNCTION FLD (IPOS, N, IWORD)

Integer Function FLD is used for bit string manipulation. The position and width of the bit string to be accessed are specified by IPOS ($0 \leq IPOS \leq 59$) and N ($I \leq N \leq 60$). IWORD is an integer variable. The bit string to be accessed is N bits wide and starts with the IPOSth bit of IWORD and returned as the value of FLD. If IPOS or N is out of the range specified, FLD is set to 0 upon return. The bits of IWORD are counted from left to right (leftmost bit is 0).

FLD depends on the word length of the machine used.

SUBROUTINE FPRPU

Subroutine FPRPU is called by PARS ER to process FORTRAN PRINT and PUNCH statements.

```
PRINT  f, list
PUNCH  f, list
```

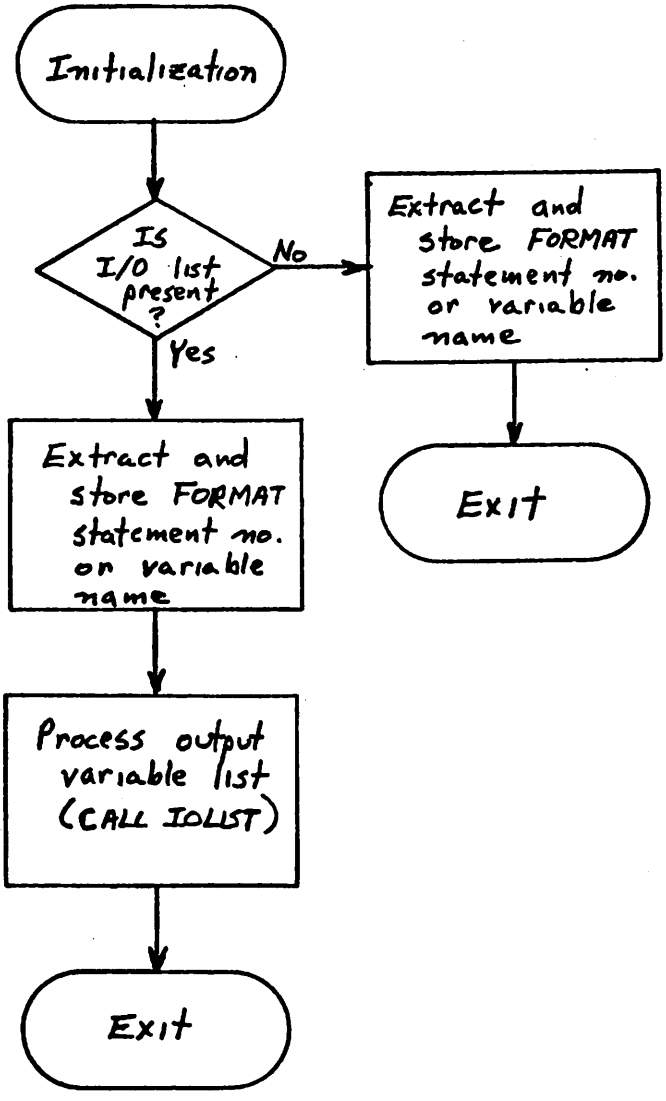
where f = format label.

list = a list of variables or arrays separated by commas.

The format label is extracted by FPRPU. The format label is assumed to be the label of a FORMAT statement or the name of an array that contains FORMAT specifications. If an integer is found, it is assumed to be a label and is passed to TABS with class code 5 (label), type code 5 (neutral), and use code 27 (format reference); if a variable is found, it is passed to TABS with class code 6 (variable), type code 5 (neutral), and use code 27 (format reference).

The list of variables to be printed or punched are processed by a call to subroutine IOLIST with IOF=3 to indicate this is a call from FPRPU.

FPRPU


Initialization
IS I/O list present ?
No
Yes
Extract and store FORMAT statement no. or variable name
Exit
Extract and store FORMAT statement no. or variable name
Process output variable list (CALL IOLIST)
Exit

SUBROUTINES FREAD

Subroutines FREAD is called by PARSER to process FORTRAN I/O statements. The main function is to recognize the various forms for READ statements and to call subroutines IOARGS and IOLIST when argument lists and I/O lists are encountered. The following forms are recognized:

READ (argument list) list

READ INPUT TAPE unit, f, list

READ f, list

READ TAPE unit, list

where argument list has the form

unit, f, ERR=ℓ_1, END=ℓ_2

where ℓ_1 is the next statement for execution if error is detected in READ/WRITE

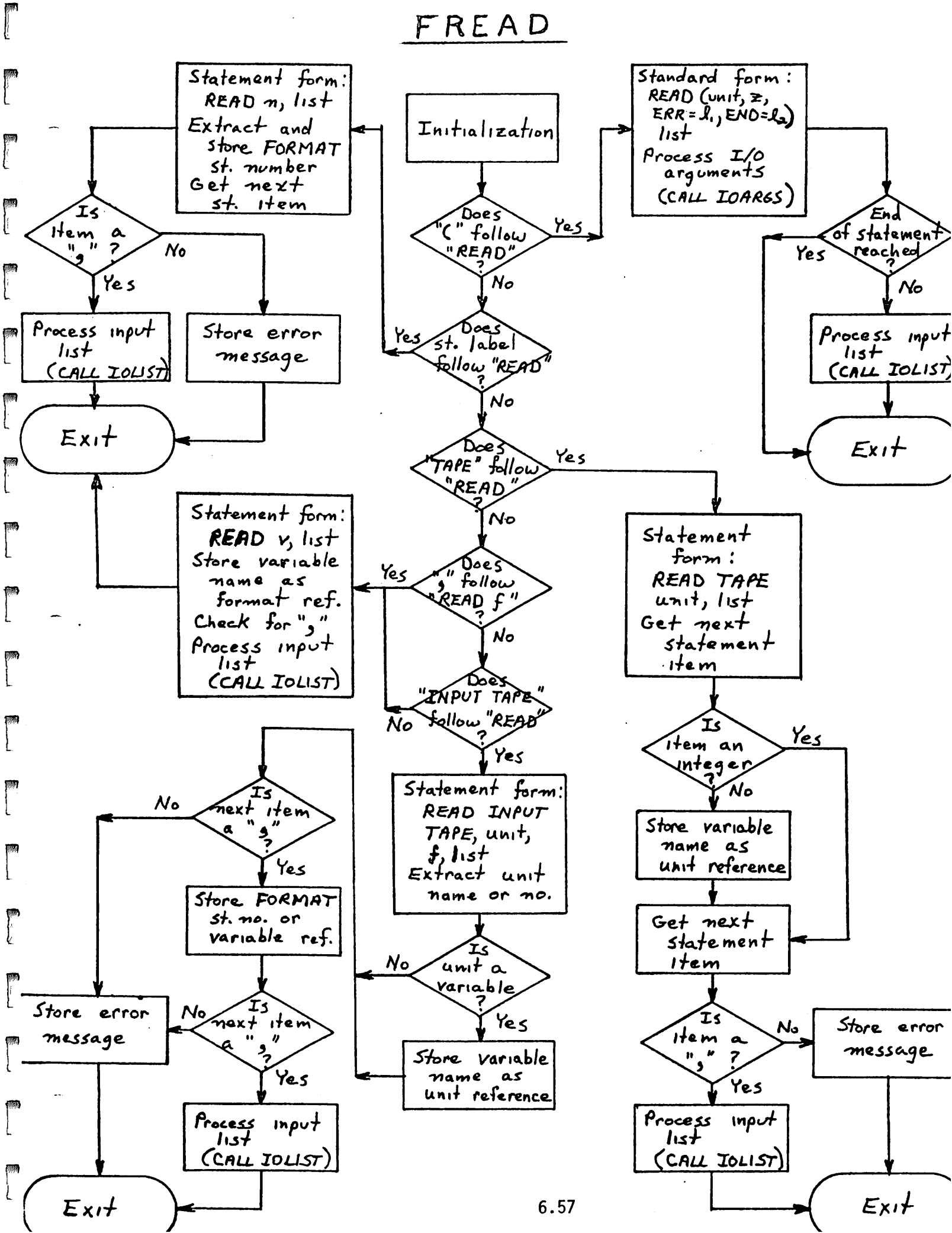
ℓ_2 is the next statement for execution if EOF

Entities extracted by FREAD, are the unit reference (unit) and the format reference (f). The unit reference is assumed to be either an integer constant or a PARAMETER variable. If a constant is found, it is simply ignored; if a variable is found, it is passed to TABS with class code 10 (PARAMETER variable), type code 0 (integer) and use code 26 (unit reference). The format reference is assumed to be the label of a FORMAT statement or the name of an array that contains FORMAT specifications. If an integer is

found, it is assumed to be a label and is passed to TABS with class code 5 (label), type code 5 (neutral), and use code 27 (format reference); if a variable is found, it is passed to TABS with class code 6 (variable), type code 5 (neutral), and use code 27 (format reference).

The argument list and I/O variable list are processed by subroutines IOARGS and IOLIST, respectively (see documentation for these routines).

FREAD


Initialization
Does "(" follow "READ" ?
Yes
No
Does st. label follow "READ" ?
Yes
No
Does "TAPE" follow "READ" ?
Yes
No
Does "," follow "READ f" ?
Yes
No
Does "INPUT TAPE" follow "READ" ?
No
Yes
Statement form: READ n, list
Extract and store FORMAT st. number
Get next st. item
Is item a "," ?
No
Yes
Process input list (CALL IOLIST)
Store error message
Exit
Statement form: READ v, list
Store variable name as format ref.
Check for ","
Process input list (CALL IOLIST)
Standard form: READ (unit, z, ERR=l1, END=l2) list
Process I/O arguments (CALL IOARGS)
End of statement reached ?
Yes
No
Process input list (CALL IOLIST)
Exit
Statement form: READ TAPE unit, list
Get next statement item
Is item an integer ?
Yes
No
Store variable name as unit reference
Get next statement item
Is item a "," ?
No
Yes
Store error message
Process input list (CALL IOLIST)
Exit
Statement form: READ INPUT TAPE, unit, f, list
Extract unit name or no.
Is unit a variable ?
No
Yes
Store variable name as unit reference
Is next item a "," ?
No
Yes
Store FORMAT st. no. or variable ref.
Is next item a "," ?
No
Yes
Store error message
Process input list (CALL IOLIST)
Exit

SUBROUTINES FREAL, FTYPE(KT)

Subroutines FREAL and FTYPE are called from PARSER to process FORTRAN type statements. The general form for type statements is

$$t\ v_1/\ell_1/v_2/\ell_2/\ \ldots\ /v_n/\ell_n/$$

where t stands for INTEGER, REAL, DOUBLE PRECISION, COMPLEX, or LOGICAL: v_i represents a list of variables, arrays (with or without dimensions), function names, and ENTRY names; ℓ_i represents a literal list, i.e., a list of constants which are to be assigned to the last variable or array in the variable list.

An additional function of FTYPE and FREAL is the recognition of statements of the form

type FUNCTION f $(a_1, a_2, \ldots, a_n)$.

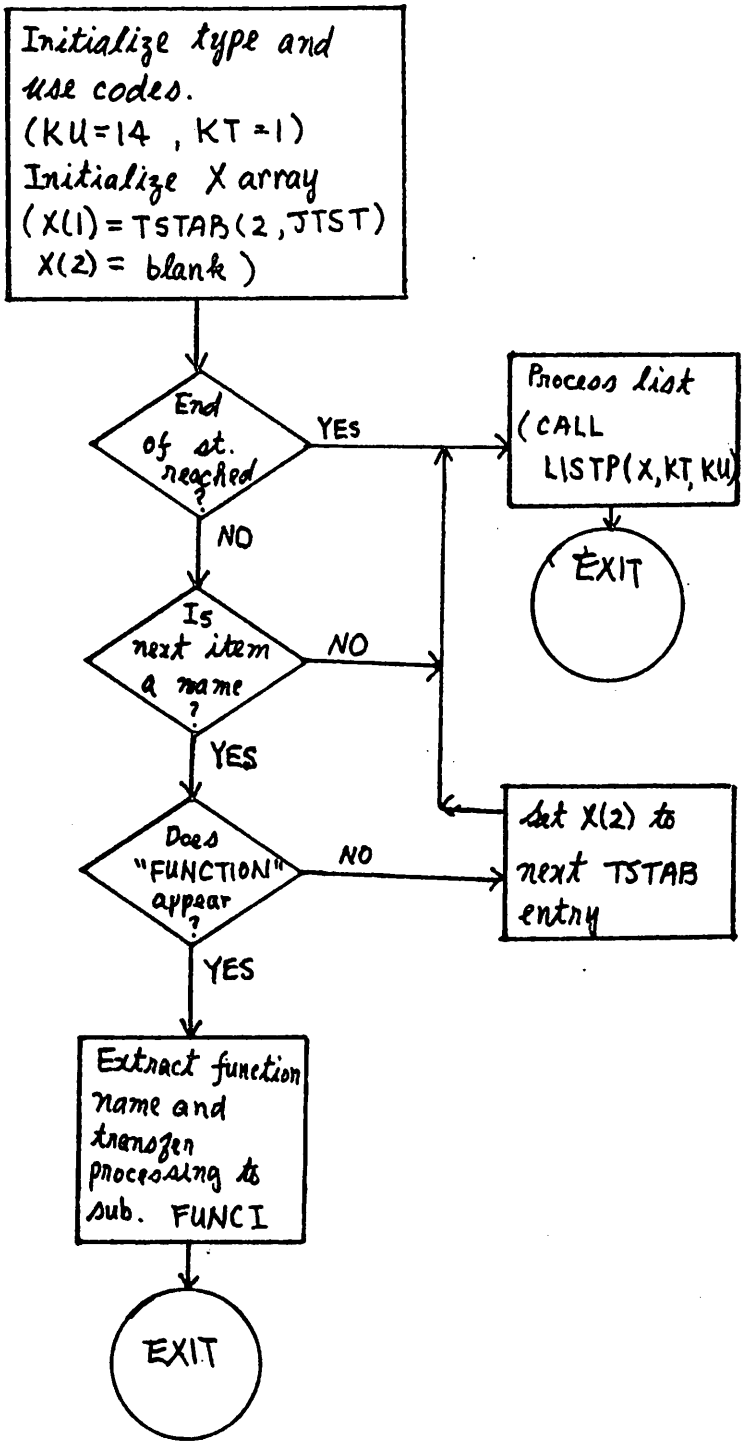
Thus a check is made for the presence of the character string "FUNCTION" and, if found, the function name (f) is extracted and passed to subroutine FUNCI for processing of the remainder of the statement.

Subroutine LISTP is called by FREAL and FTYPE to process variable and literal lists, i.e. that portion of the statement following the type (t) in the first form.

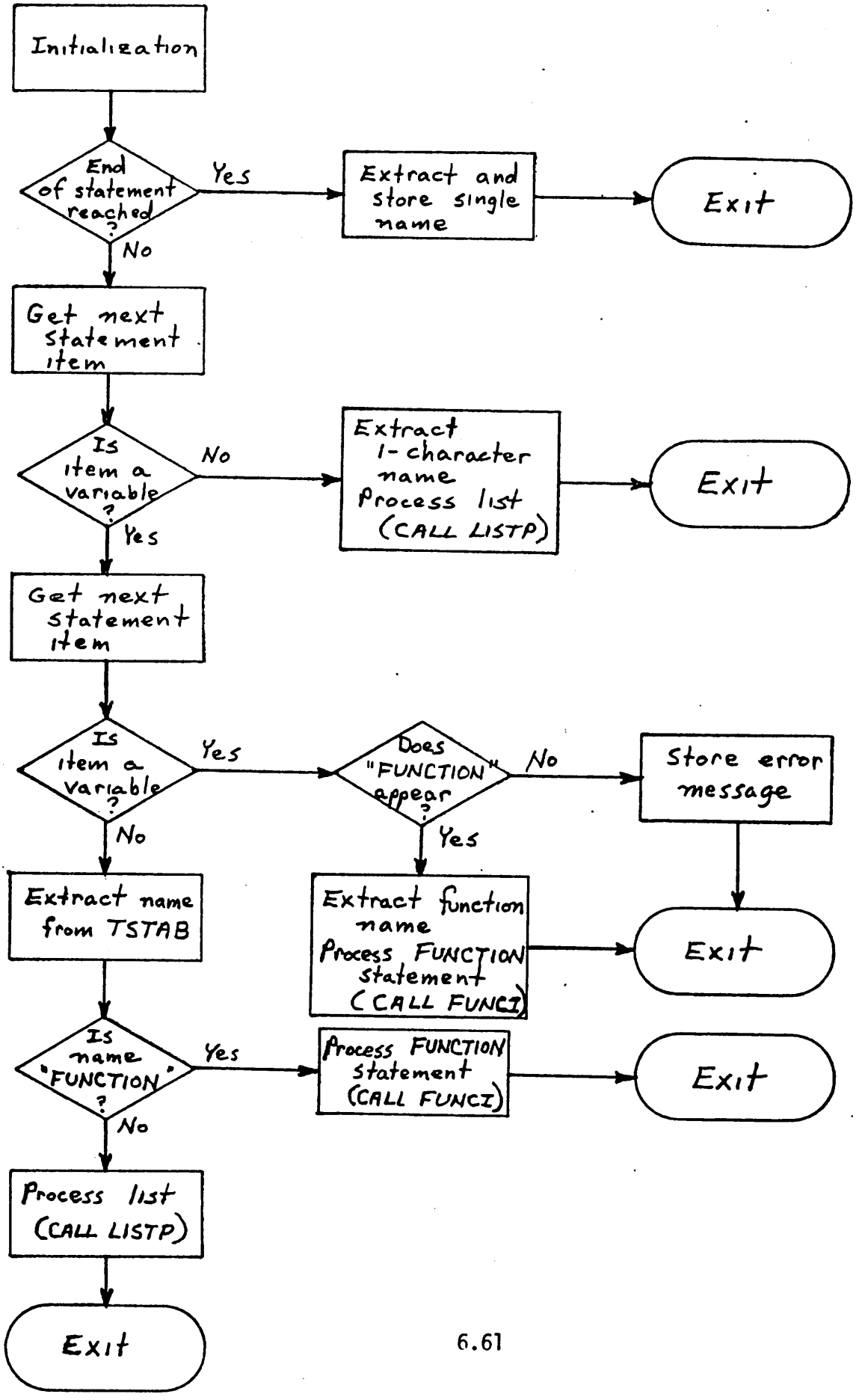
The various type statements are handled in the following manner. PARSER recognizes the keyword of particular type statement (INTE, REAL, DOUB, COMP), sets the type code (KT) accordingly, and passes control to subroutine FTYPE. Since there is no break character between the type specification and the first name in the list (e.g. INTEGER X appears as one continuous character string in array TSTAB), FTYPE extracts the first name (using subroutine

SHIFTY) and passes it, along with the type code and use code (14 for type statement entry), to LISTP for processing of the remainder of the statement. REAL statements are an exception, since the 4 characters of "REAL" are stored in exactly one word of TSTAB: consequently, no shifting is required to extract the first list item, and REAL statements are processed be FREAL and LISTP rather than FTYPE.

FREAL


Initialize type and use codes.
(KU=14 , KT=1)
Initialize X array
(X(1) = TSTAB(2,JTST)
X(2) = blank)
End of st. reached ?
YES
Process list (CALL LISTP(X,KT,KU)
EXIT
NO
Is next item a name ?
NO
YES
Does "FUNCTION" appear ?
NO
Set X(2) to next TSTAB entry
YES
Extract function name and transfer processing to sub. FUNCI
EXIT

FTYPE


Initialization
End of statement reached ?
Yes
Extract and store single name
Exit
No
Get next statement item
Is item a variable ?
No
Extract 1-character name Process list (CALL LISTP)
Exit
Yes
Get next statement item
Is item a variable ?
Yes
Does "FUNCTION" appear ?
No
Store error message
No
Yes
Extract name from TSTAB
Extract function name Process FUNCTION statement (CALL FUNCI)
Exit
Is name "FUNCTION" ?
Yes
Process FUNCTION statement (CALL FUNCI)
Exit
No
Process list (CALL LISTP)
Exit

SUBROUTINES FWRITE

Subroutines FWRITE is called by PARSER to process FORTRAN I/O statements. The main function is to recognize the various forms for WRITE statements and to call subroutines IOARGS and IOLIST when argument lists and I/O lists are encountered. The following forms are recognized:

WRITE (argument list) list

WRITE OUTPUT TAPE unit, f, list

WRITE TAPE unit, list

where argument list has the form

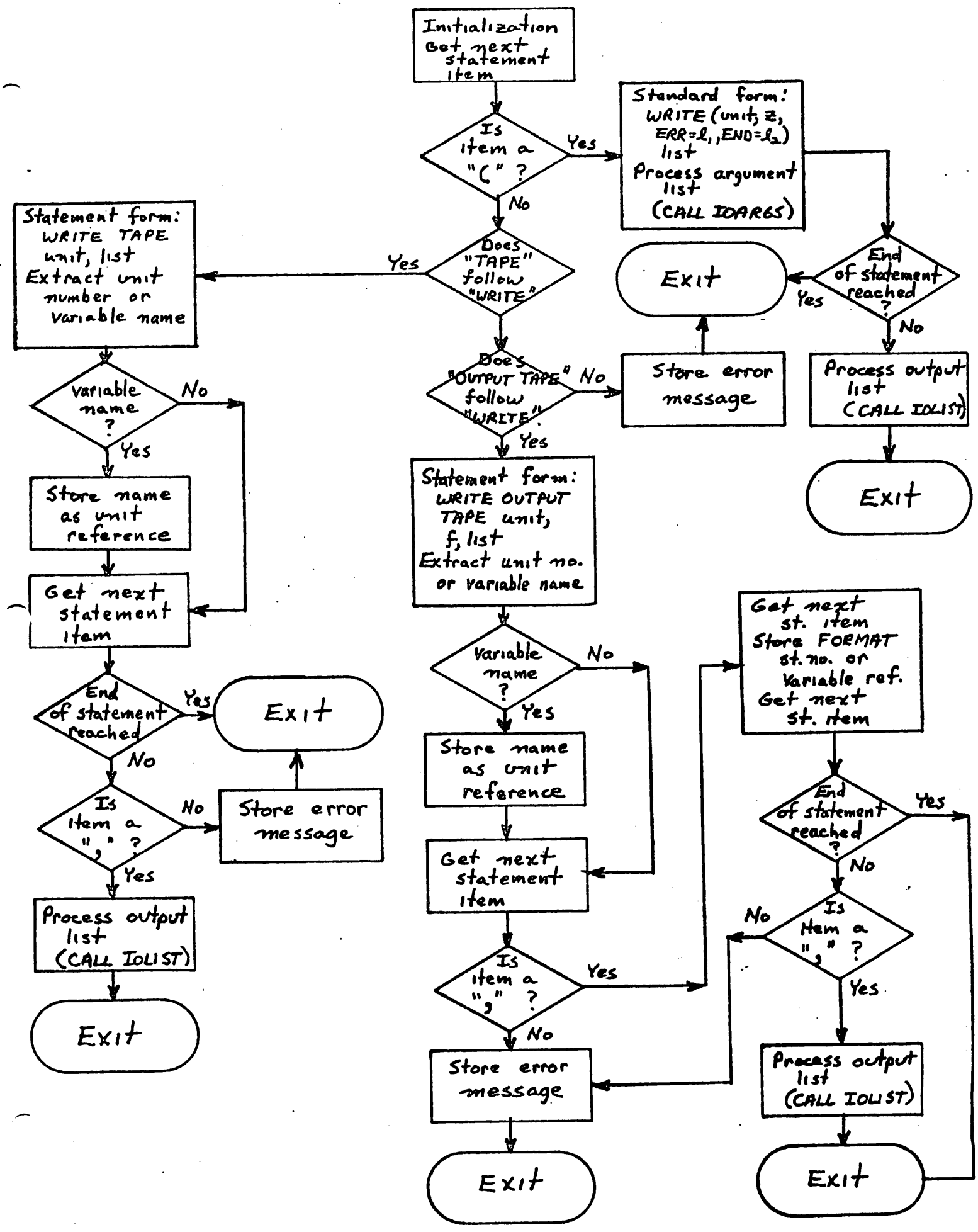
unit, f, ERR=ℓ_1, END=ℓ_2

where ℓ_1 is the next statement for execution if error is detected in READ/WRITE

ℓ_2 is the next statement for execution if EOF

Entities extracted by FWRITE are the unit reference (unit) and the format reference (f). The unit reference is assumed to be either an integer constant or a PARAMETER variable. If a constant is found, it is simply ignored; if a variable is found, it is passed to TABS with class code 10 (PARAMETER variable), type code 0 (integer) and use code 26 (unit reference). The format reference is assumed to be the label of a FORMAT statement or the name of an array that contains FORMAT specifications. If an integer is

FWRITE


Initialization
Get next statement item
Is item a "C" ?
Yes
Standard form: WRITE (unit, z, ERR=l1, END=l2) list
Process argument list (CALL IOARGS)
No
Does "TAPE" follow "WRITE"
Yes
Statement form: WRITE TAPE unit, list
Extract unit number or variable name
Variable name ?
No
Yes
Store name as unit reference
Get next statement item
End of statement reached
Yes
Exit
No
Is item a "," ?
No
Store error message
Yes
Process output list (CALL IOLIST)
Exit
Exit
End of statement reached ?
Yes
No
Process output list (CALL IOLIST)
Exit
Store error message
Does "OUTPUT TAPE" follow "WRITE"
No
Yes
Statement form: WRITE OUTPUT TAPE unit, f, list
Extract unit no. or variable name
Variable name ?
No
Yes
Store name as unit reference
Get next statement item
Is item a "," ?
Yes
No
Store error message
Exit
Get next st. item
Store FORMAT st. no. or Variable ref.
Get next st. item
End of statement reached ?
Yes
No
Is Item a "," ?
No
Yes
Process output list (CALL IOLIST)
Exit

SUBROUTINE FSUB

Subroutine FSUB is called from PARSER to process FORTRAN SUBROUTINE statements, which have the following form:

SUBROUTINE $S(a_1,a_2, \ldots,a_n)$

where S is the subroutine name and $a_1,a_2,\ldots,a_n$ is a list of formal (dummy) parameters.

Since the SUBROUTINE statement is the first statement of a subroutine module, the FACES module identification is updated by FSUB, i.e. the sequence number (NSUB) and file number (NFILE) are incremented and the module number is set to NSUB+200 (subroutine modules are identified by module number in the range 201-299).

Subroutine SHIFTY is used to extract the subroutine name, which is passed to subroutine TABS with class code 1 (subroutine name), type code 5 (neutral), and use code 0 (declaration).

Names of formal parameters are recognized by FSUB and passed to TABS with class code 6 (variable), type code 5 (neutral), and use code 17 (subroutine dummy parameter). A special call to TABS is made after the last name with MPOP=1 to indicate the end of the parameter list.

Formal parameters are assumed to be symbolic names; multiple returns ($ used as argument in connection with RETURN K) are not currently recognized by FACES.

SUBROUTINE FUNCI(X,KT,KU)

Subroutine FUNCI is called from PARSER or a type statement processor to process FUNCTION statements, which are assumed to be of the following form:

t FUNTION $f(b_1,b_2, \ldots,b_n)$

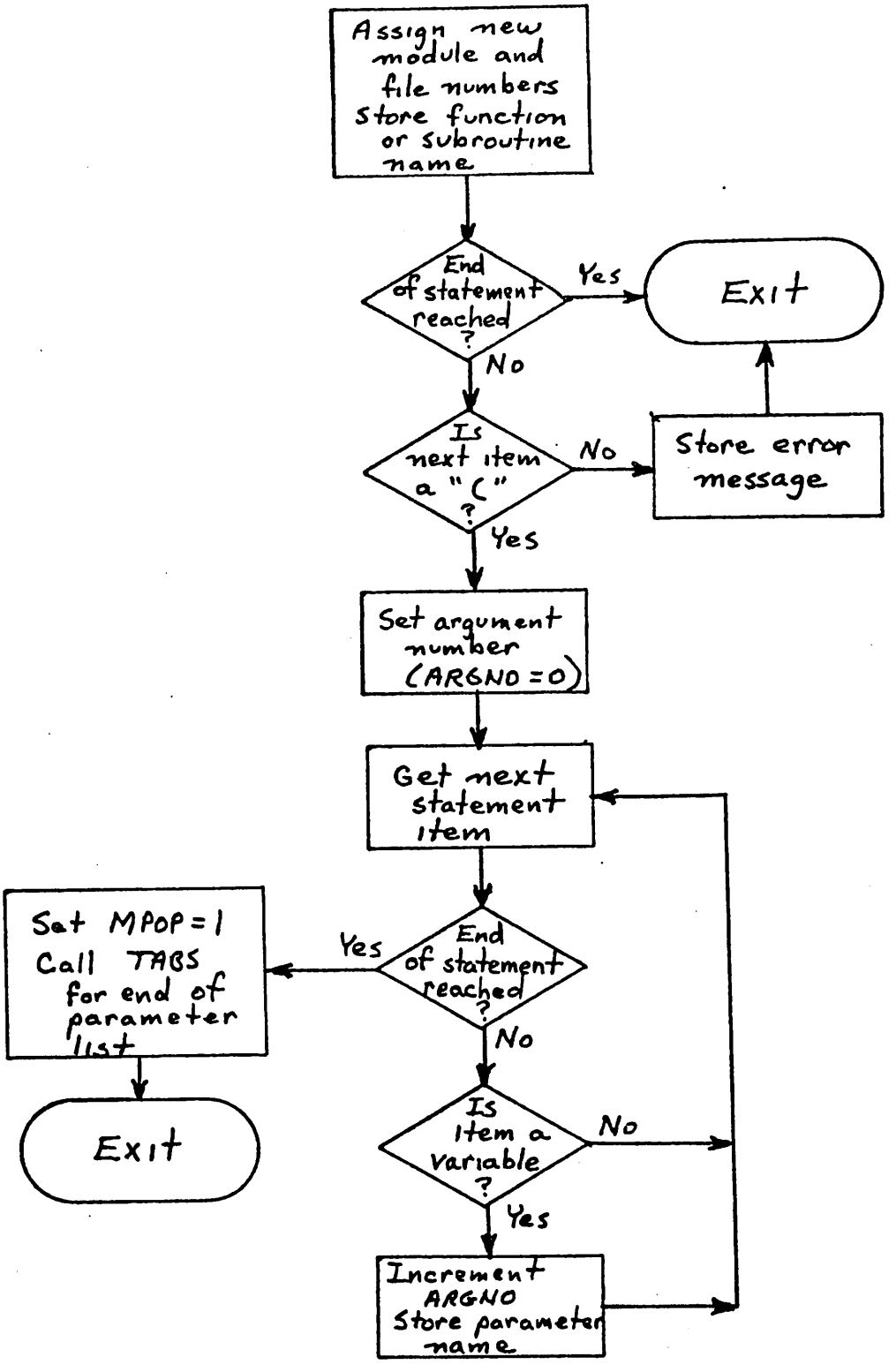
where t represents the type (INTEGER, REAL, DOUBLE PRECISION, COMPLEX, or LOGICAL) and may be omitted, f is the function name, and $b_1,b_2, \ldots,B_n$ are formal (dummy) arguments. The formal arguments are assumed to be symbolic names; multiple returns ($ used as argument in connection with RETURN K) are not currently recognized by FACES.

FUNCTION statements are recognized by PARSER or by a type statement processor (FTYPE, FREAL); these routines extract the function name from the symbol string in array TSTAB and pass this name to FUNCI through parameter X, a three word array. In addition, the type and use codes are passed to FUNCI through parameters KT, KU, respectively. If FUNCI is called from FTYPE or FREAL, the type code (KT) will be set according to the particular type (t) recognized and the use code (KU) will be set to 14 (entry in type statement). In this case the function name will be passed to TABS once as a declaration (use code 0) and once as a type statement entry (use code 14). If FUNCI is called from PARSER, the type code will be 5 (neutral) and use code will be 0 (declaration). In this case only one call is made to subroutine TABS for the function name.

Since the FUNCTION statement is the initial statement of a module, subroutine FUNCI updates the module identification information; i.e., the sequence number (NSUB) and the file number (NFILE) ARE INCREMENTED AND THE MODULE NUMBER (MN) is set to NSUB+300 (FUNCTION modules are identified by module number in the range 301-399).

Each formal parameter (b_i in the above form) is recognized by FUNCI and passed to subroutine TABS with class code 6 (variable), type code 5 (neutral), and use code 16 (function dummy parameter). A special call to TABS is made after the last parameter with MPOP=1 to indicate the end of the parameter list.

FUNCI and FSUB

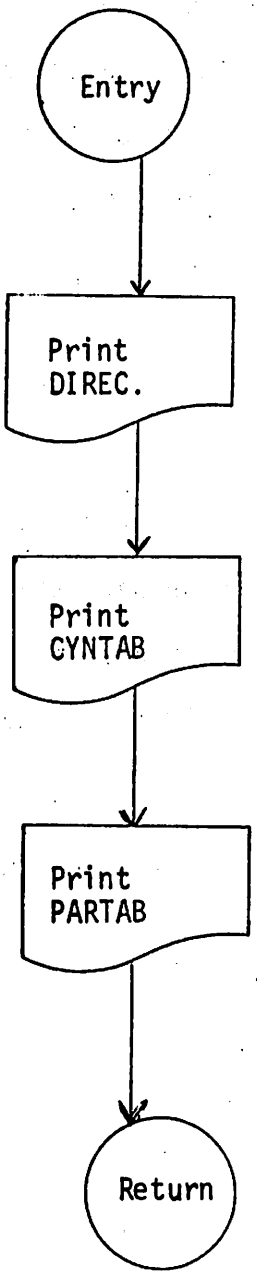


SUBROUTINE GLOPRT

This routine is called to print the three global tables: DIREC - directory, CNTAB - common name table, and PARTAB - parameter table.

DIREC is a hash coded table. Only non-empty entries are printed.

Flowchart for GLOPRT



Subroutine HASH (INFLAG, NAM1, NAM2)

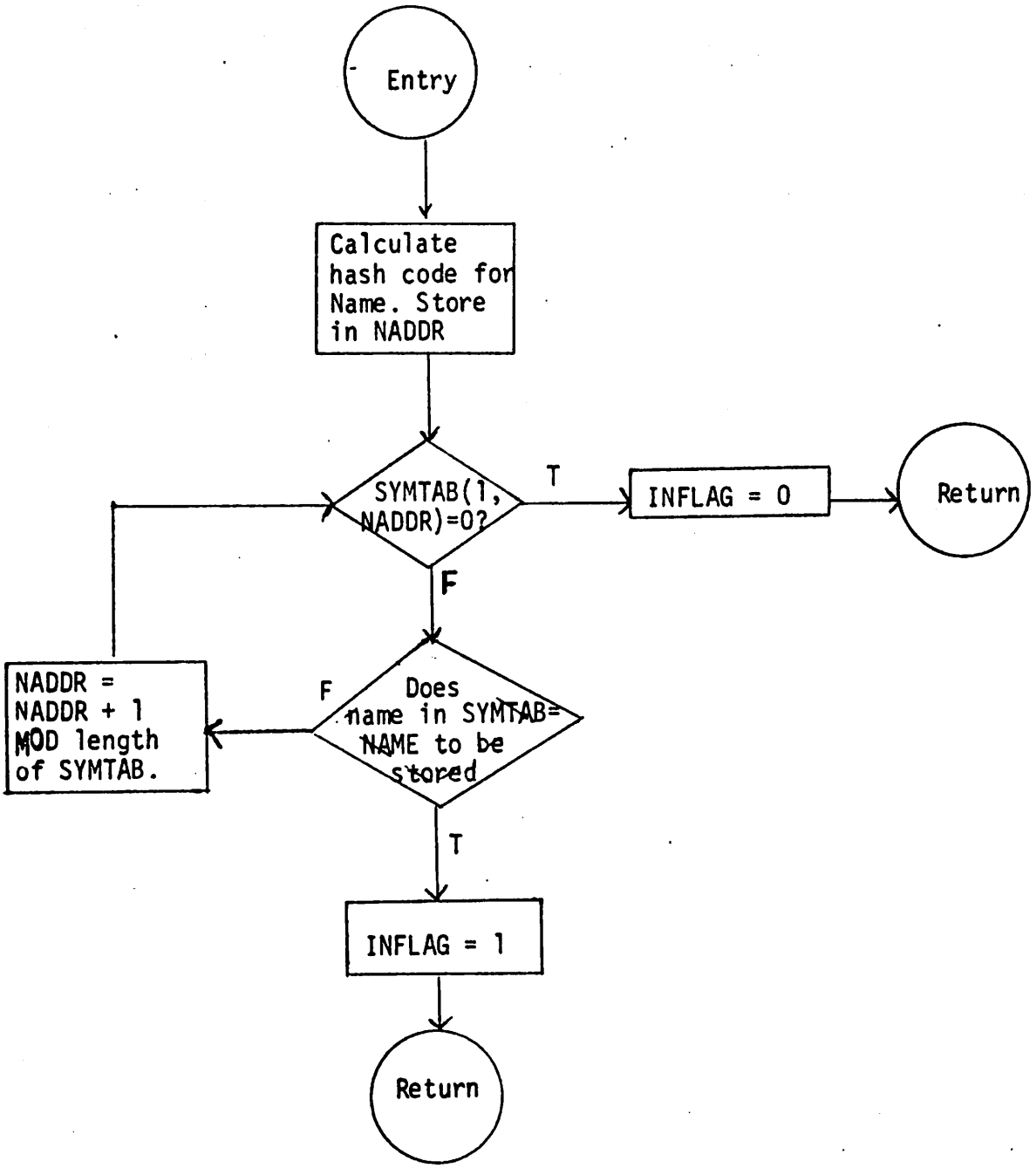
This routine computes the hash code for entering a name in SYMTAB. The variables NAM1 and NAM2 contain the character code (left-justified, blank-filled, four characters per word for the name to be stored. The hash address is calculated accordingtto the following formula:

$$\text{Hash address} = \text{MOD}\left(\frac{\text{NAM1} + \text{NAM2}}{2}, \text{IPRIME}\right) + 1$$

where NAM1 and NAM2 have been right-justified, zero-filled and IPRIME is the largest prime number less than the length of SYMTAB. If a collision occurs (another name already stored at the hash address), the name is stored in the next blank table entry. The parameter INFALG is used to indicate whether the name has already been stored in SYMTAB. If it has, INFLAG is set to 1; otherwise it is 0. In either case, common variable NADDR contains the index to SYMTAB at which the name is stored.

The hash address returned depends on the internal character representation of the machine used.

Flowchart for HASH



FUNCTION HASHIN(N)

Function HASHIN computes a hash index based on the sum of the third and fourth characters of a given character string. The character string is contained in input parameter N, 4 characters per word, left-justified.

The formula for the hash index is

HASHIN = 3rd char of N + 4th char of N + 10

The value returned is used by PARSER to index the table of key words (MATCH, Appendix C).

The value returned depends on the internal character representation of the machine.

SUBROUTINE IOARGS

Subroutine IOARGS is called from FREAD and FWRITE to process the argument list for READ and WRITE statements. The form for the argument list (see documentation for FREAD, FWRITE, FPRPU) is the following:

(unit, F, ERR=ℓ_1, END=ℓ_2) where unit = I/O unit number

F = format label

ℓ_1 = next statement for execution if error detected in the I/O operation

ℓ_2 = next statement for execution if EOF

The unit reference (unit) is assumed to always present while any combination of the remaining arguments may be omitted in the I/O statement.

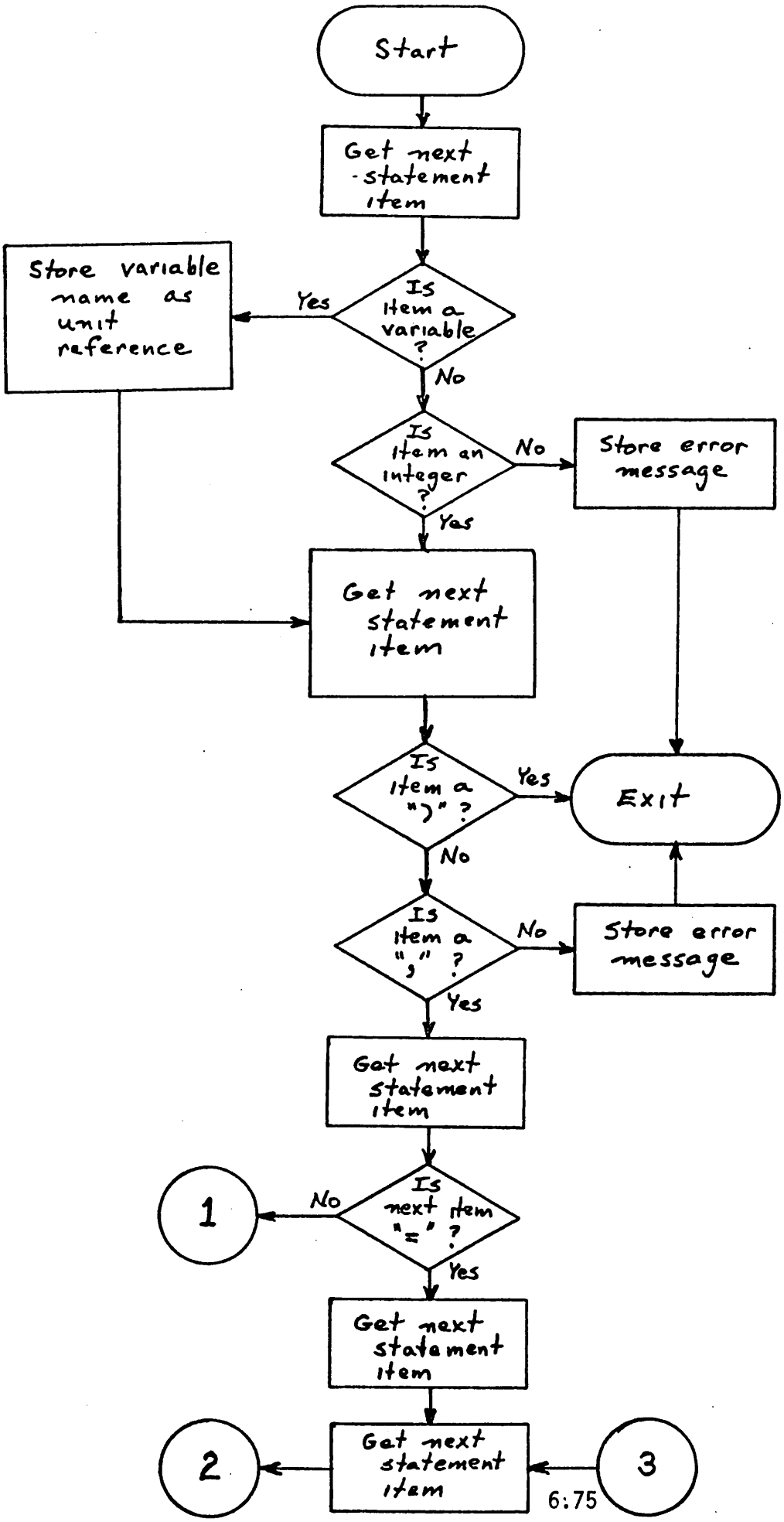
The unit reference is assumed to be either an integer constant or a PARAMETER variable. If a constant is found, it is simply ignored; if a variable is found, it is passed to TABS with class code 10 (PARAMETER variable), type code 0 (integer), and use code 26 (unit reference).

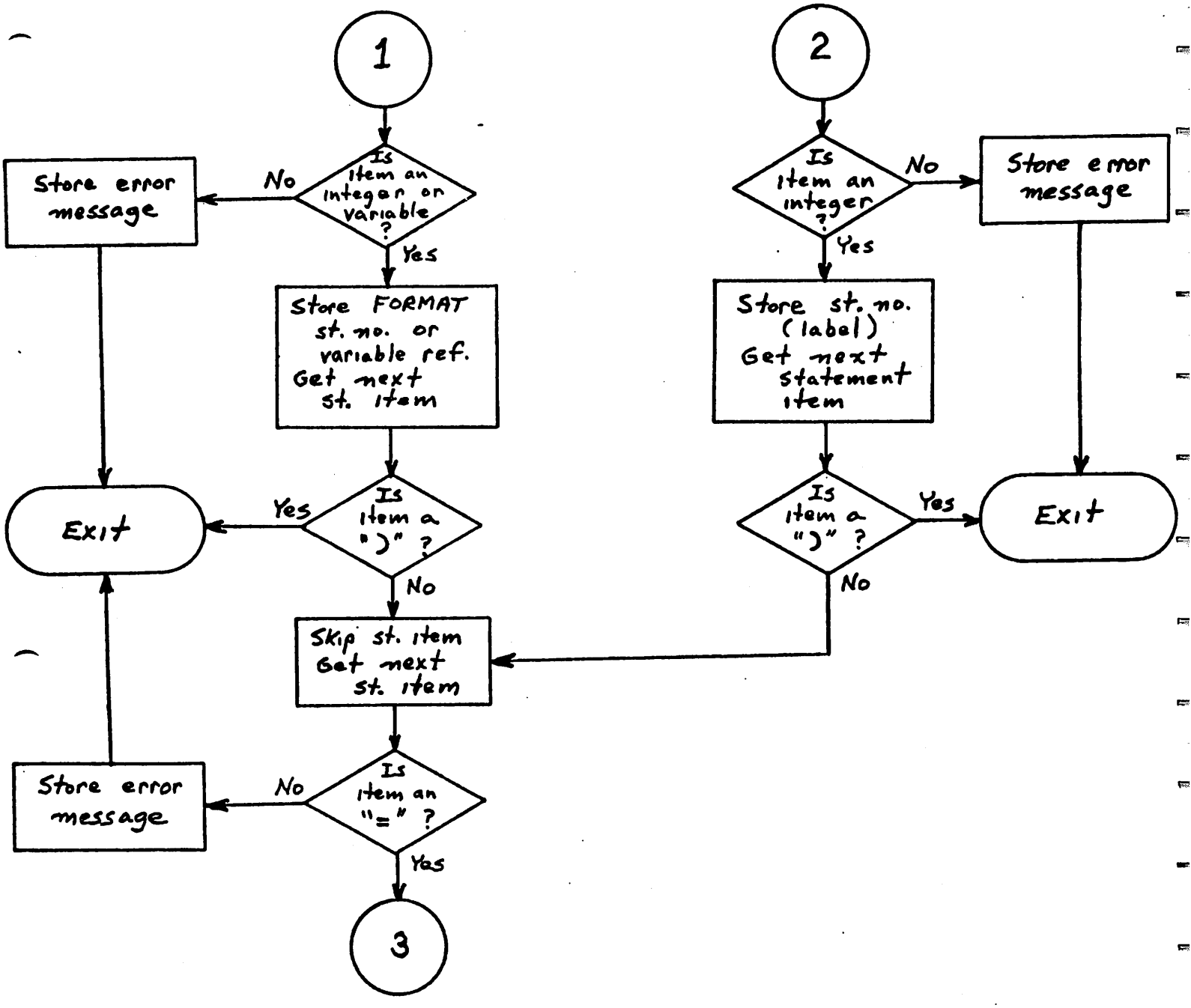
The format reference is assumed to be the label of a FORMAT statement or the name of an array that contains FORMAT specifications. If an integer is found, it is assumed to be a label and is passed to TABS with class code 5 (label), type code 5 (neutral), and use code 27 (format reference); if a variable is found, it is passed to TABS with class code 6 (variable), type code 5, and use code 27. Reference to a namelist name is not currently recognized, since FACES does not currently recognize NAMELIST statements.

For the arguments ERR=ℓ_1 and ERR=ℓ_2, ℓ_1 and ℓ_2 represent labels of statements to which control can be transferred. Thus the label names for ℓ_1 and ℓ_2 are passed to TABS with use code 10 (label reference), type code 5 (neutral), and class code 5 (label).

Control is returned from IOARGS to FREAD or FWRITE upon recognition of the ")" representing the end of the argument list.

IOARGS


Start
Get next statement item
Is item a variable ?
Yes
Store variable name as unit reference
No
Is item an integer ?
No
Store error message
Yes
Get next statement item
Is item a ")" ?
Yes
Exit
No
Is item a "," ?
No
Store error message
Yes
Get next statement item
Is next item "=" ?
No
1
Yes
Get next statement item
Get next statement item
2
3


1
2
Is item an integer or variable ?
No
Yes
Store error message
Store FORMAT st. no. or variable ref. Get next st. item
Is item an integer ?
No
Yes
Store error message
Store st. no. (label) Get next statement item
Exit
Is item a ")" ?
Yes
No
Is item a ")" ?
Yes
No
Exit
Skip st. item Get next st. item
Store error message
Is item an "=" ?
No
Yes
3

SUBROUTINE IOLIST(IOF)

Subroutine IOLIST is called from an I/O statement processor or the DATA statement processor to process I/O variable lists. Input parameter IOF is a use code indicator which may have the following values:

IOF = 3 If called from FWRITE or FPRPU
Indicates names in list are external output variables

IOF = 4 if called from FREAD
Indicates names in list are external input variables

IOF = 12 if called from FDATA
Indicates names in list are assigned data in a DATA statement

List elements are assumed to be subscripted or unsubscripted variable, array names, or implied DO-loops.

Since nesting is allowed in the I/O list (e.g. implied DO Loops), a use code stack (STACK) is maintained to record the use code for names found at the various levels of nesting. The initial entry in the stack is the value of IOF (namely, 3, 4, or 12 as described above), since names at the highest level are list elements. When a "(" is found after a variable name, the STACK is pushed and STACK (TOP) = 15 (use code for subscripts). If a "(" is found and is not after a variable name, then the STACK is pushed and STACK (TOP) = STACK (TOP-1). The variable PC is used as the index to STACK as well as representing the parentheses count. Each variable name found in the I/O list is passed to subroutine TABS for storage in the FACES table.

The following example illustrate the use of the use code stack (STACK) and PC. Consider the following PRINT statement.

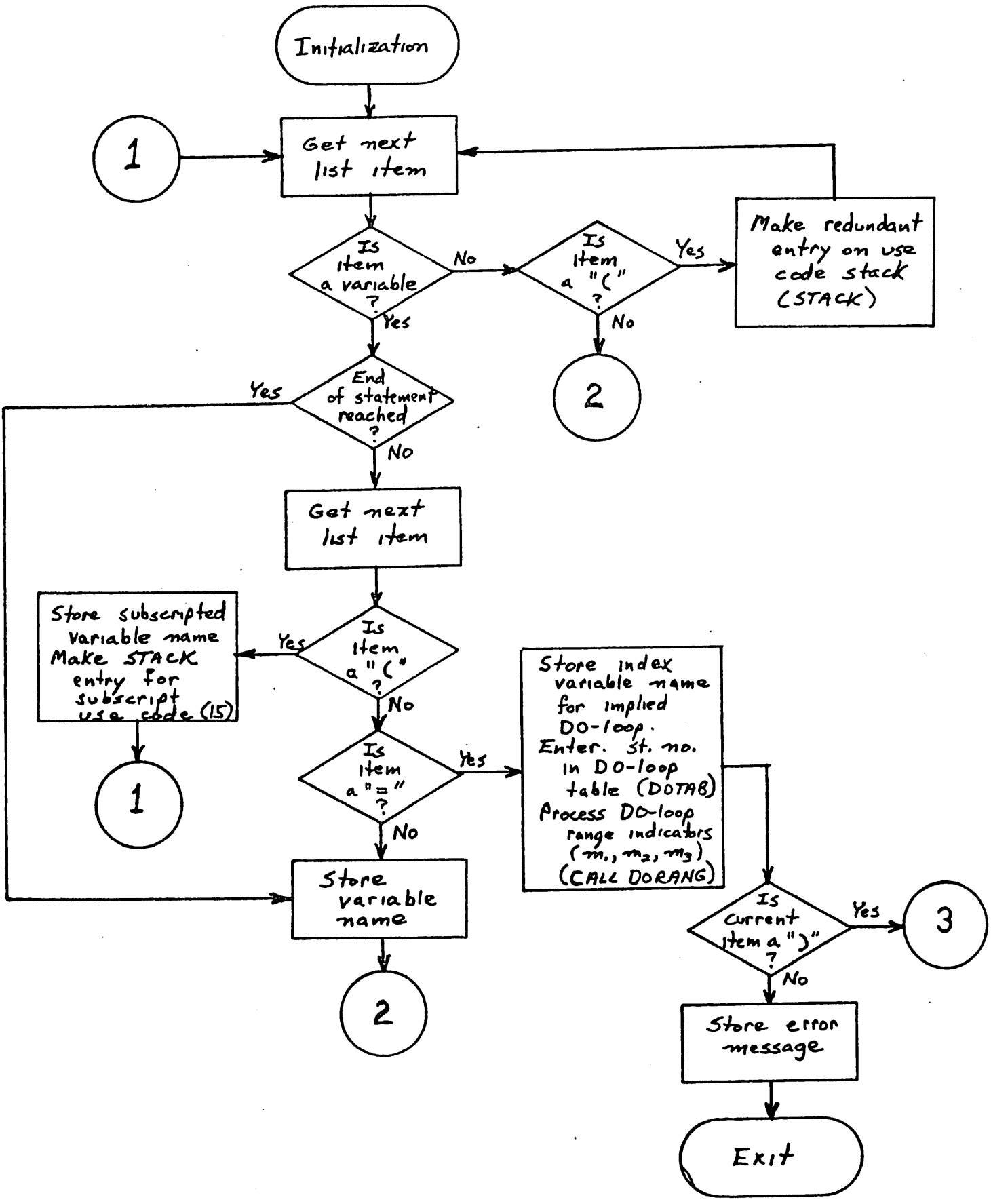
PRINT 10, (((ARRAY(L,M,N), L=1, 10), M = 1, 10), N = 1, 10).

When the first L is reached,	When the second L is reached
STACK (1) = 3 initialize to value of IOF	STACK (1) = 3
STACK (2) = 3	STACK (2) = 3
STACK (3) = 3	STACK (3) = 3
STACK (4) = 3	STACK (4) = 3
STACK (5) = 15	PC = 4
PC = 5	

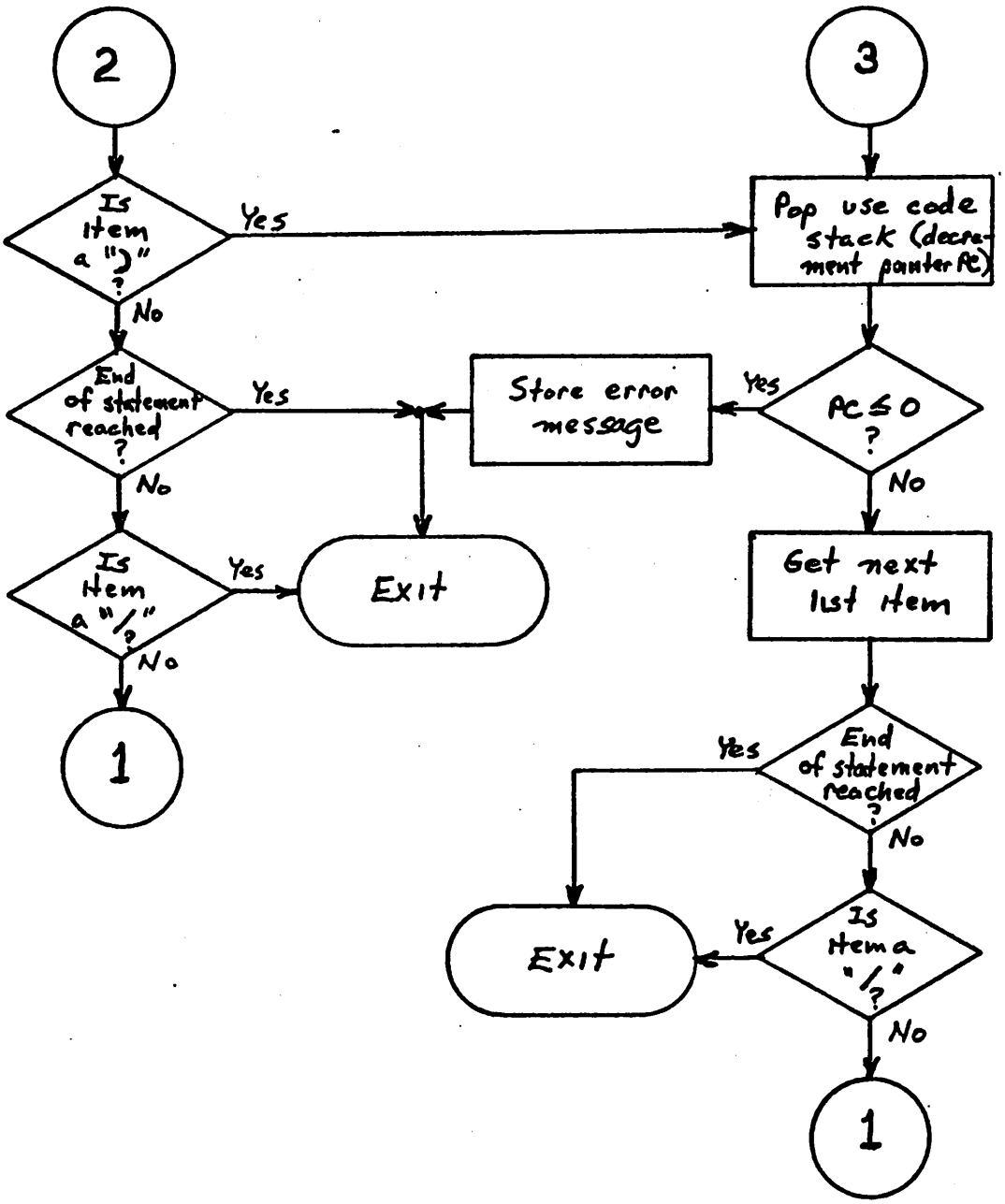
Implied DO-loops are assumed to be of the form $(L_j, i=m_1,m_2,m_3)$, where L_j is an I/O list element and $i=m_1,m_2,m_3$ is the same as in a DO statement (see documentation for FDO, DORANG). Identification depends on the recognition of "=" following a simple variable name. When this condition occurs, the variable represented by i is passed to subroutine TABS with class code 6 (variable), type code 0 (integer), use code 5 (DO-loop index); The FACES assigned number for the I/O statement is then stored in the DO-loop table (DOTAB) and subroutine DORANG is called to process the string m_1,m_2,m_3.

Exit from subroutine IOLIST occurs upon recognition of the end of the I/O statement (if called from FREAD, FWRITE, FPRPU) or a slash (/) (if called from FDATA).

IOLIST


Initialization
1
Get next list item
Is item a variable ?
No
Is item a "(" ?
Yes
Make redundant entry on use code stack (STACK)
Yes
No
Yes
End of statement reached ?
No
2
Get next list item
Store subscripted variable name Make STACK entry for subscript use code (15)
Yes
Is item a "(" ?
No
1
Is item a "=" ?
Yes
Store index variable name for implied DO-loop. Enter. st. no. in DO-loop table (DOTAB) Process DO-loop range indicators (m_1, m_2, m_3) (CALL DORANG)
No
Store variable name
2
Is current item a ")" ?
Yes
3
No
Store error message
Exit

IOLIST (cont.)


2
3
Is item a ")" ?
Yes
No
Pop use code stack (decrement pointer ptr)
End of statement reached ?
Yes
No
Store error message
Yes
PC ≤ 0 ?
No
Is Item a "/" ?
Yes
No
Exit
Get next list item
End of statement reached ?
Yes
No
Exit
Is Item a "/" ?
Yes
No
1
1

SUBROUTINE LISTP (X, KTT, KUU).

Subroutine LISTP is a parsing routine which processes variable lists found in COMMON, DIMENSION and TYPE (INTEGER, REAL, etc.) statements.

X, KTT and KUU are input parameters. Variable from different statement types are identified by KUU:

KUU = 11 variable defined in a COMMON block

= 13 " " " " DIMENSION statement

= 14 " " " " TYPE statement

X is a three word array containing the first name in the list to be processed. KTT is the type code for the variables found in the list.

LISTP extracts the variable names in the list and pass the names to subroutine TABS for storage in the FACES tables. Dimensions of arrays are recognized and passed to TABS in array DIM (in common block TAPAR); the current version of FACES allows only three dimensions for arrays. Variables used as dimensions are recognized and stored with use code 15 (subscripts). Literal lists are currently ignored.

If a slash is encountered during parsing, one of the following actions will be taken:

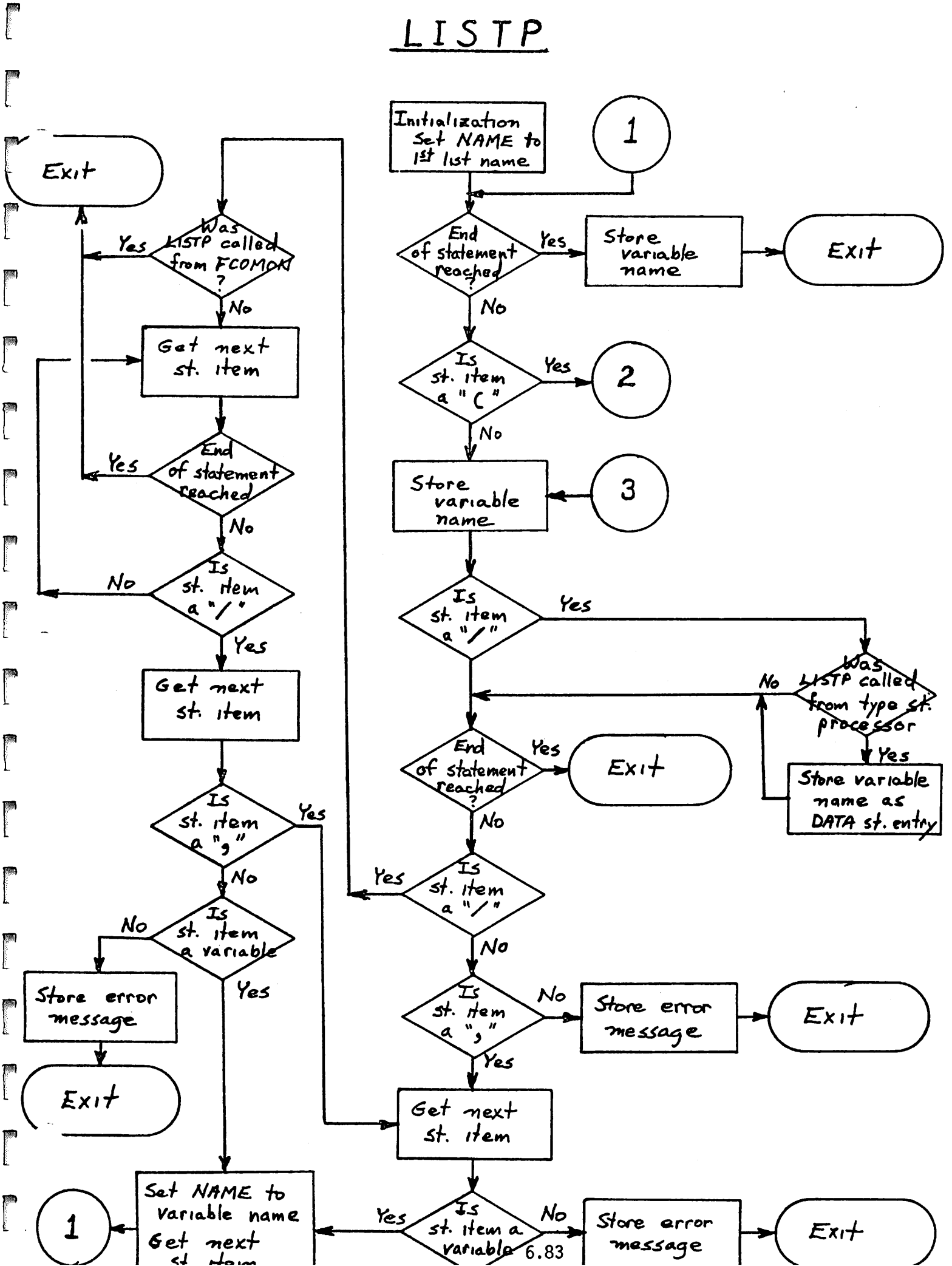
i). If the variable list of a type statement is being processed, call TABS with class code 6 (variable), type code KTT and use code 12 (entry in a data statement) to indicate that the last variable or array has been initialized. Scan over the literal list and process the next variable list if there is one.

ii). If the variable list of a common block is being processed, return.

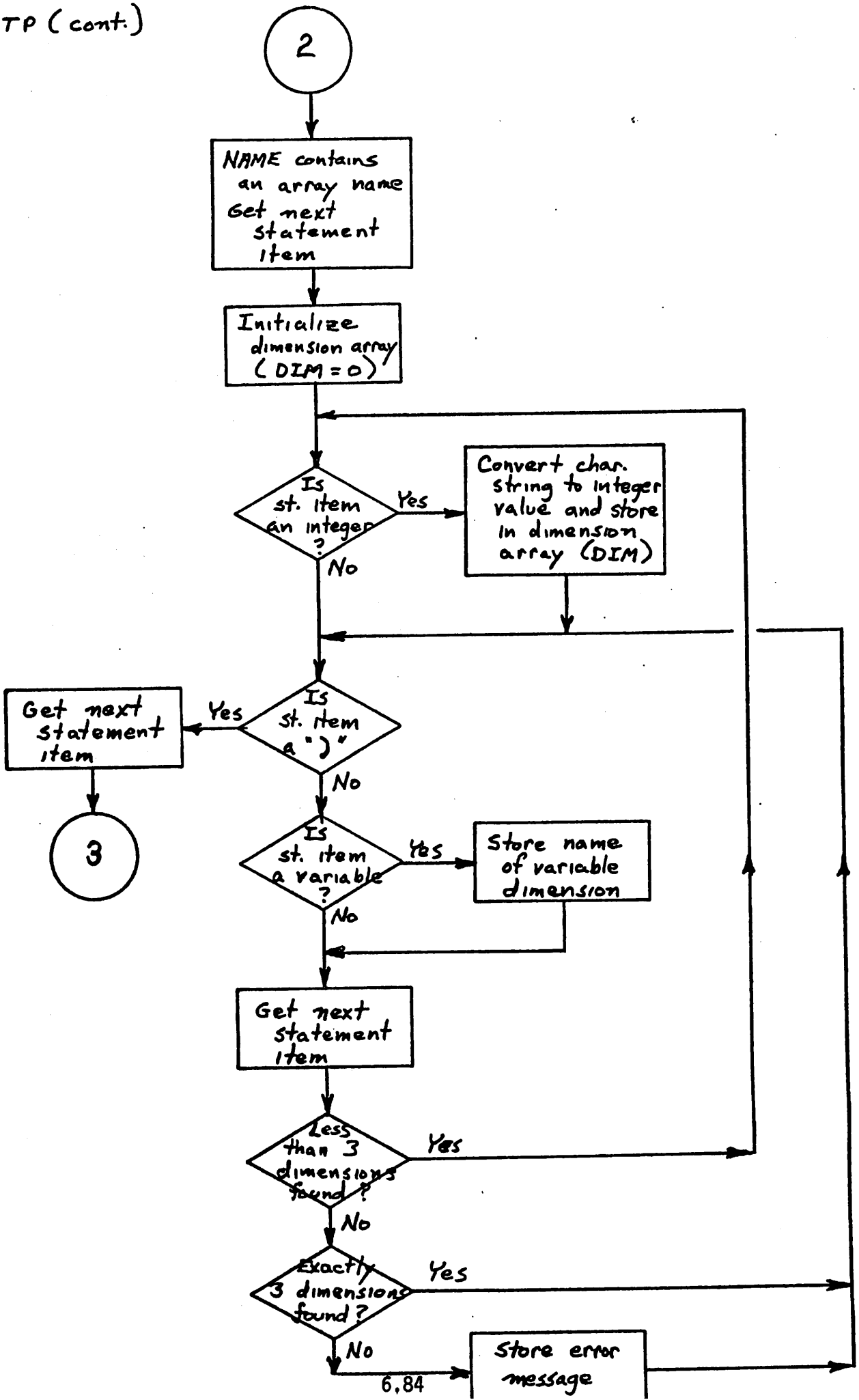
found, it is assumed to be a label and is passed to TABS with class code 5 (label), type code 5 (neutral), and use code 27 (format reference); if a variable is found, it is passed to TABS with class code 6 (variable), type code 5 (neutral), and use code 27 (format reference).

The argument list and I/O variable list are processed by subroutines IOARGS and IOLIST, respectively (see documentation for these routines).

LISTP


Exit
Was LISTP called from FCOMON ?
Yes
No
Get next st. item
End of statement reached
Yes
No
Is st. item a "/"
No
Yes
Get next st. item
Is st. item a ","
Yes
No
Is st. item a variable
No
Yes
Store error message
Exit
1
Set NAME to variable name
Get next st. item
Initialization Set NAME to 1st list name
1
End of statement reached ?
Yes
Store variable name
Exit
No
Is st. item a "C"
Yes
2
No
Store variable name
3
Is st. item a "/"
Yes
No
Was LISTP called from type st. processor
Yes
Store variable name as DATA st. entry
End of statement reached ?
Yes
Exit
No
Is st. item a "/"
Yes
No
Is st. item a ","
No
Store error message
Exit
Yes
Get next st. item
Is st. item a variable
Yes
No
Store error message
Exit

LISTP (cont.)

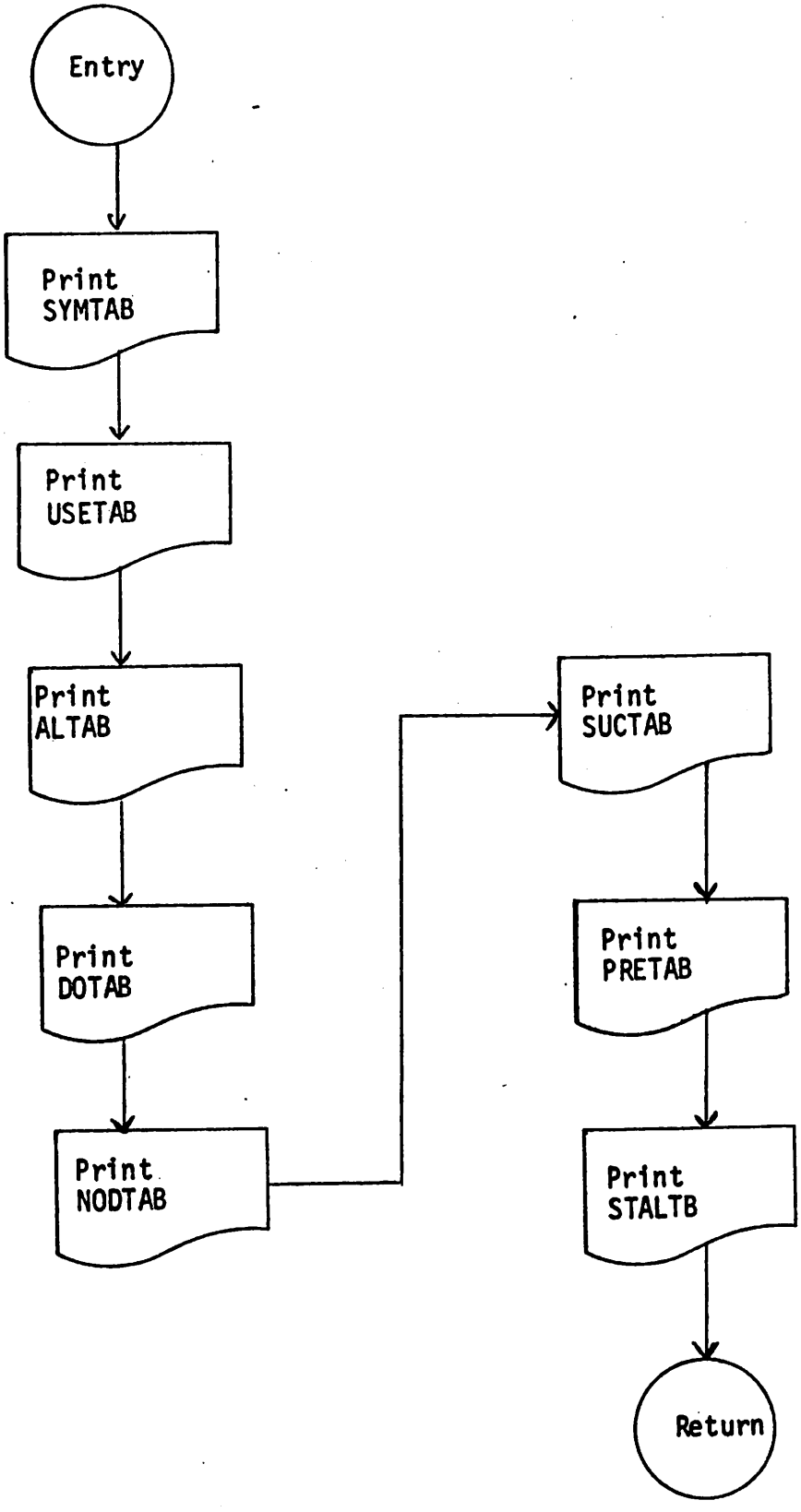

2
NAME contains an array name Get next statement item
Initialize dimension array (DIM = 0)
Is st. item an integer ?
Yes
Convert char. string to integer value and store in dimension array (DIM)
No
Is st. item a ")"
Yes
Get next statement item
3
No
Is st. item a variable ?
Yes
Store name of variable dimension
No
Get next statement item
Less than 3 dimensions found ?
Yes
No
Exactly 3 dimensions found ?
Yes
No
Store error message

SUBROUTINE LOCPRT (MODNO)

This routine prints the local tables: SYMTAB, USETAB, ALTAB, DOTAB, NODTAB, SUCTAB, PRETAB, and STALTB for the module specified by MODNO.

SYMTAB is a hash coded table, only non-empty entries are printed.

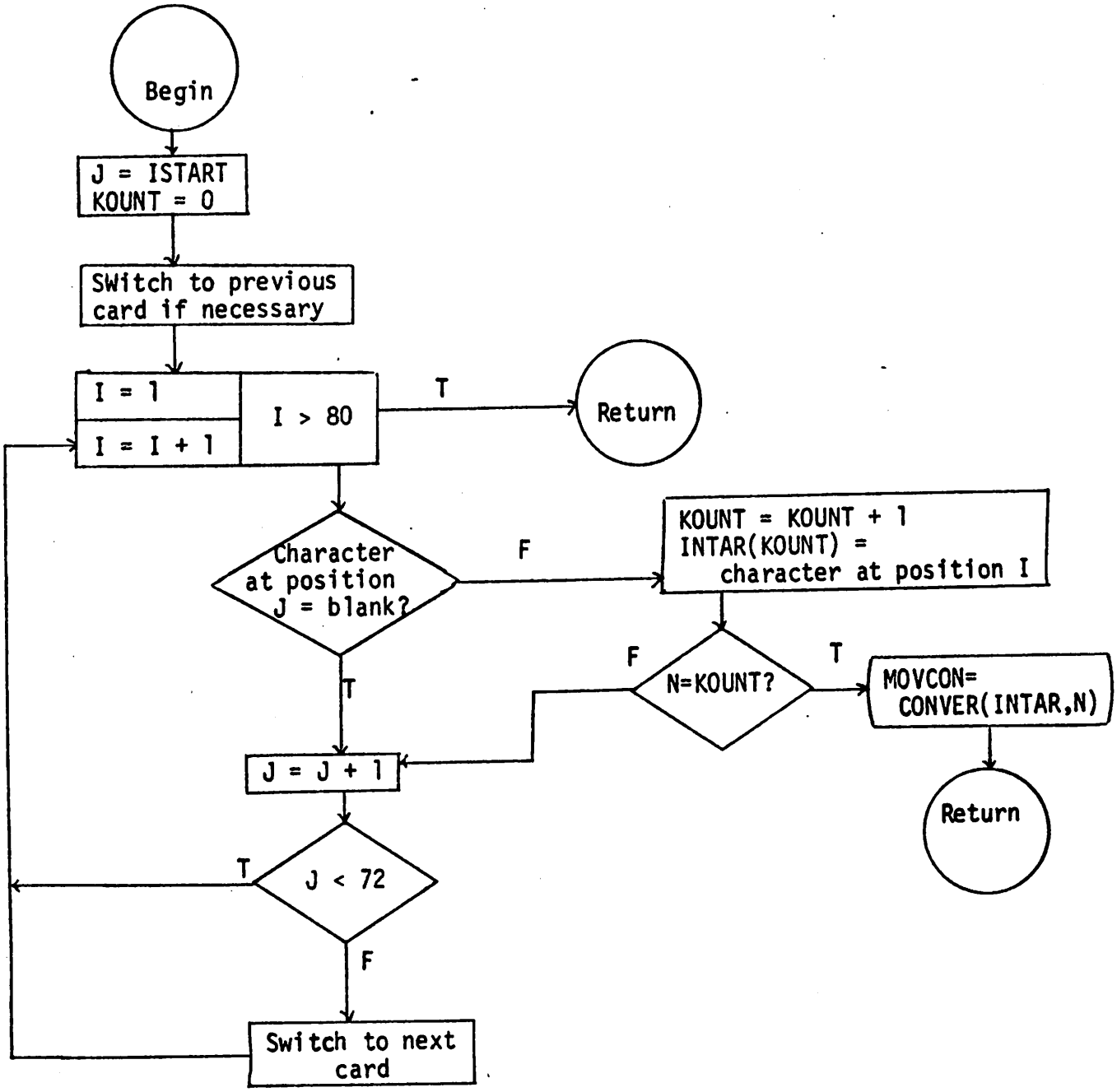
Flowchart for LOCPRT



Function MOVCON(ISTART,N)

This function moves a string of digits (maximum 8) from a FORTRAN input statement to a temporary buffer (INTAR) and calls function CONVER to convert the string of digits to its integer value. Input parameter ISTART indicates the starting point from which to move and convert the digits. The number of digits to be converted is the value of input parameter N. The converted integer value is returned in the value of MOVCON.

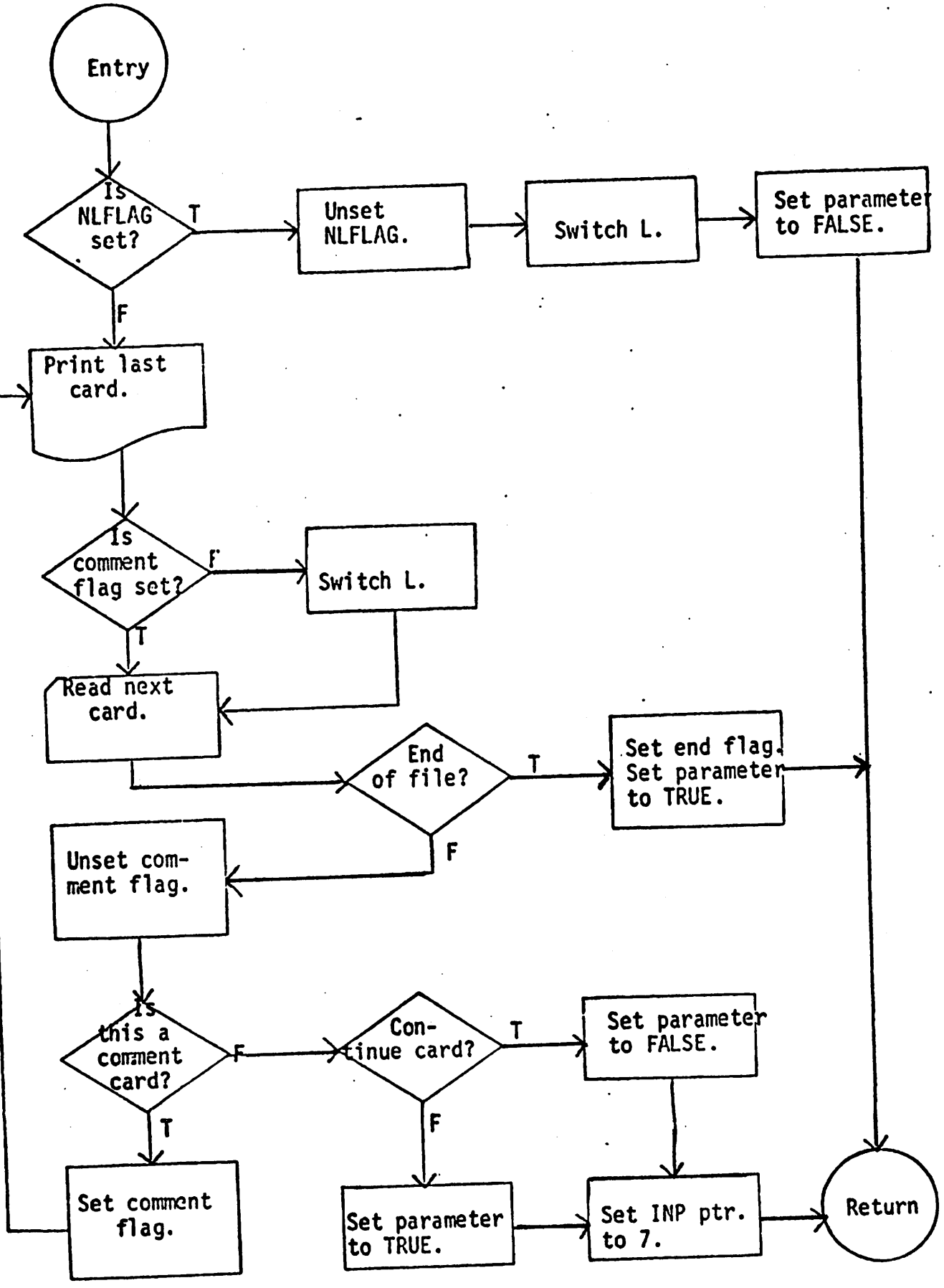
Flowchart for MOVCON



Subroutine NEWCAR(B)

This routine is called to read a new card and print the old one. When a new card is read, the line pointer L to the two-card buffer STORE(2,80) is switched to the new line, either 1 or 2. If NLFLAG is set to one upon entry, NEWCAR does not read a new card but simply switches the line pointer. (NLFLAG is set by other routines that switch back to the previous card to process information. When the end of this previous card is reached, the next card should not be read since it has already been read.) NEWCAR skips all comment cards. The logical output parameter B is set to .TRUE. if the new card read is the beginning of a new statement. If the new card is a continuation of the same statement, B is set to .FALSE. . In both cases, the pointer to the input buffer is set to 7.

Flowchart for NEWCAR

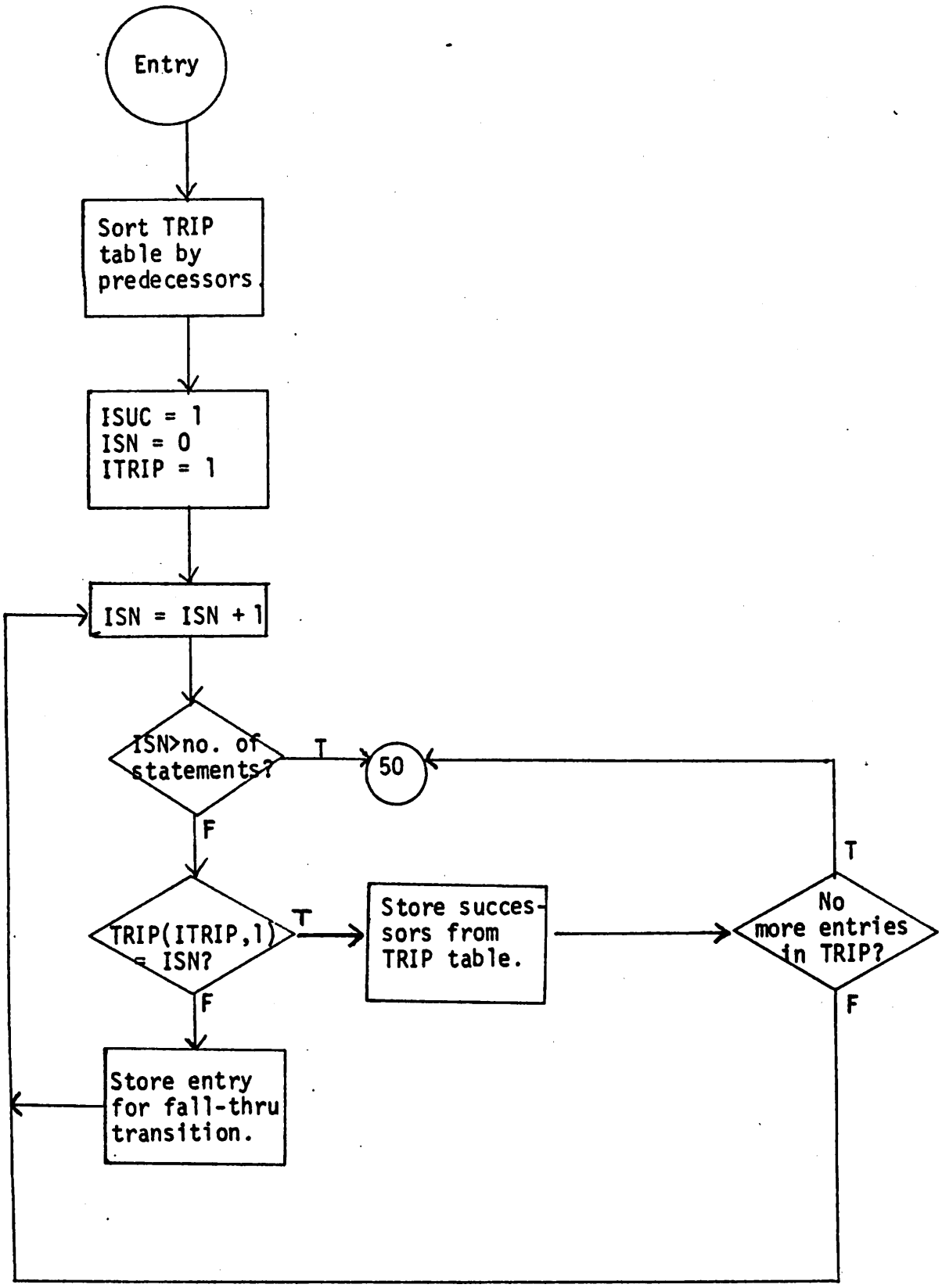


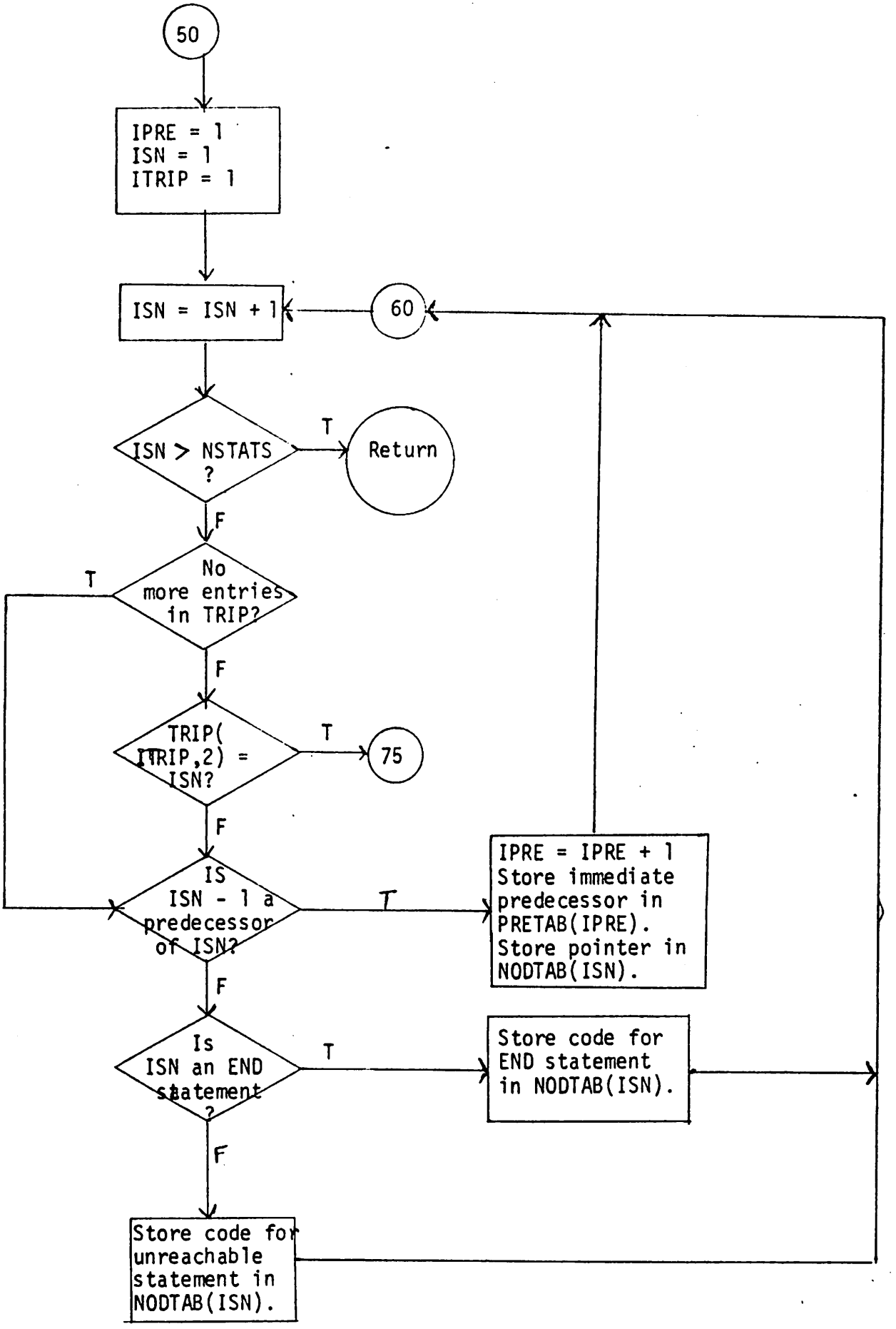
Subroutine NODGEN (NSTATS)

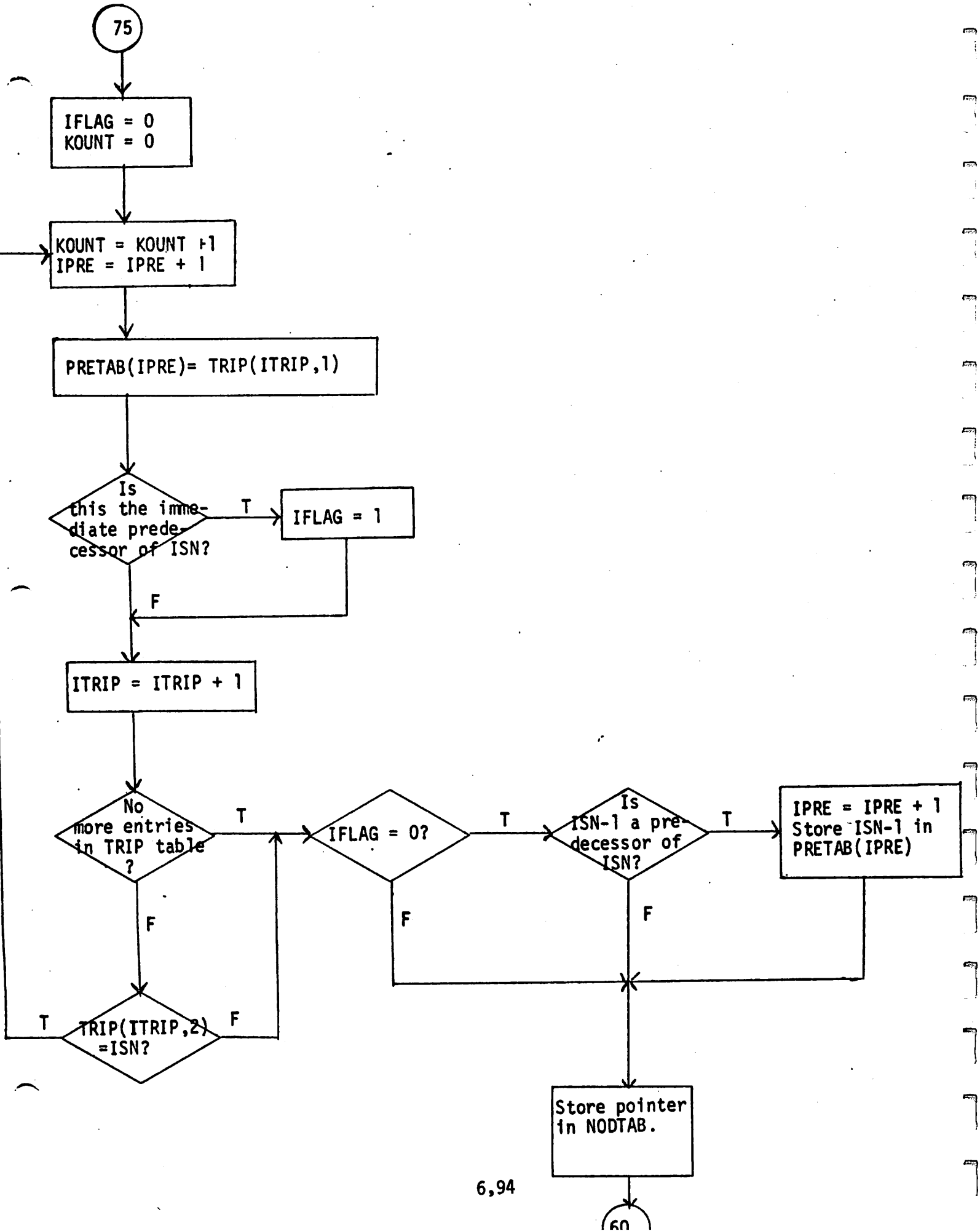
This routine is called from the parser as a post-processing routine after all statements of a module have been analyzed. It forms the node table (NODTAB), successor table (SUCTAB), and predecessor table (PRETAB) from the information stored in the TRIP table.

First, the TRIP table is sorted by predecessors and the successor table is formed. At this time the successor pointers are stored in NODTAB. Then the TRIP table is sorted by successors and the predecessor table is formed. At this time the predecessor pointers are stored in NODTAB.

Flowchart for NODGEN




50
IPRE = 1
ISN = 1
ITRIP = 1
ISN = ISN + 1
60
ISN > NSTATS ?
T
Return
F
No more entries in TRIP?
T
F
TRIP(ITRIP,2) = ISN?
T
75
F
IS ISN - 1 a predecessor of ISN?
T
IPRE = IPRE + 1
Store immediate predecessor in PRETAB(IPRE).
Store pointer in NODTAB(ISN).
F
Is ISN an END statement ?
T
Store code for END statement in NODTAB(ISN).
F
Store code for unreachable statement in NODTAB(ISN).


75
IFLAG = 0
KOUNT = 0
KOUNT = KOUNT +1
IPRE = IPRE + 1
PRETAB(IPRE)= TRIP(ITRIP,1)
Is this the immediate predecessor of ISN?
T
IFLAG = 1
F
ITRIP = ITRIP + 1
No more entries in TRIP table ?
T
IFLAG = 0?
T
Is ISN-1 a predecessor of ISN?
T
IPRE = IPRE + 1
Store ISN-1 in PRETAB(IPRE)
F
F
F
TRIP(ITRIP,2) =ISN?
T
F
Store pointer in NODTAB.
60

SUBROUTINE OUTGT

This is a machine dependent routine that writes the global tables to a permanent file. Detailed description of the file handling routines can be found in section V of the User Mannual.

SUBROUTINE OUTLT

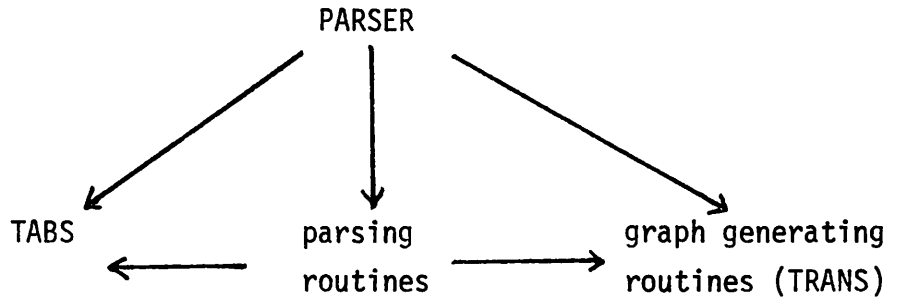
This is a machine dependent routine that writes the local tables of a specific module to a permanent file. Detailed description of the file handling routines can be found in section V of the User Mannual.

SUBROUTINE PARSER

Overview

PARSER is the coordinating routine for the analysis of FORTRAN programs. Several modules (consisting of FORTRAN programs and subprograms) may be analyzed consecutively. Information pertaining to module interfaces and interactions is saved in a set of global tables, and information pertaining to an individual module is saved in a set of local tables. Analysis consists generally of a statement-by-statement scan to identify and store the pertinent information in the system tables.

The following diagram depicts the relationships between PARSER and other system routines.



The following sections describe the organization and function of PARSER in general terms. Detailed descriptions of other system routines will be found in individual descriptions.

Statement Scanning

Statement scanning is performed by subroutine SCAN (and related routines called by SCAN), which is called once for each source statement. SCAN breaks the statement into lexical entities. Output of SCAN is stored in ISS (Intermediate Symbol String) and TSTAB (Temperory Symbol Table). ISS stores character codes to represent lexical entities extracted from rhw statement and TSTAB stores alphanumeric strings.

Subroutine SCAN also recognizes the occurence of a 0-level equal sign (equal sign not enclosed within any parenthesis) and store its position (the position in ISS) in EFLAG.

Parsing

Parsing is done by subroutine PARSER on a statement by statement basis. Since program modules analysed by FACES are assumed to be compilable without syntax errors, syntax checking is limited to aid information extraction. PARSER starts the processing of a statement by calling SCAN to read in the source statement. SCAN breaks the statement into lexical entities. Upon return from SCAN, PARSER will determine the FORTRAN statement type in the following manner. If no 0-level equal sign is found (EFLAG = 0), PARSER eliminates the possibility of an assignment or DO statement, and classifies the statement by matching the first four characters with an entry in a table of statement types (MATCH, Appendix C). If a 0-level equal sign is found, the statement must be an assignment (or function definition), DO, or "IF (<expression>) <assignment st.> ". The DO statement is recognized

by the = , pair in "DO n I = m_1, m_2, m_3". If the first two characters are "IF", a check is made to distinguish between "IF (<expression> <assignment st.> " and "IF (<subscript>) = <expression> ", where the latter represents an assignment statement (or function definition) with subscripted variable (or function name) IF. If DO or IF has not been recognized, the statement is classified as an assignment statement.

Once the statement type has been identified, parsing consists of an examination of the ISS and TSTAB arrays by subroutines designed to accept the syntax for that particular statement type and to extract the information to be stored in the system tables. The following example illustrates the method used.

FORTRAN statement : DO 100 INDEX = J, 10000

ISS content : V V = I , I

TSTAB contents :

```
DOIO  OIND
EX
J
1000  0
```

The ISS string contains two V's (for "variable") indicating an alphanumeric string spread over two entries (4 words) of the TSTAB array. Since this string contains several syntactical entities (namely, DO, 100, INDEX, J, 10000), it is necessary to identify and extract each pertinent entity. A general shifting routine (SHIFTY) is used for this purpose. Given the first 3 words of TSTAB above, position = 6, and length = 7, SHIFTY will return the character string INDEX. The label "100" is extracted in a like manner.

Generation of System Tables

Each occurence of a symbol (name or label) in a statement is recorded in system tables by TABS (and related routines called by TABS).

The structure of the system tables and the operation of TABS is essentially transparent to the parsing routines. TABS is called with the symbol name, class, type and use code* passed as parameters. Additional information is passed through common block TAPAR.

Class code identifies the symbol type (label, variable, subroutine name etc.) Type code is assigned to variable or function name to designate variable type (integer, logical etc.) Use code determines how the symbol is referenced in the statement (subscript, DO-loop index etc.) Based on this information, TABS will make the appropriate entries in the system tables. For the entity "100" in the above example, the call is of the form "CALL TABS (5, 5, 25)", indicating a label, neutral type, used in a DO statement.

Ambiguities sometimes occur during parsing. For example, SUM in

```
TOT = SUM (I)
```

can be an array name or function name. Such ambiguities are resolved by TABS.

* lists of class, type and use code can be found in section III of User Mannual

Structural Analysis

The FACES system represents program structure as a directed graph with nodes representing program statements and edges representing lines of program flow. Accordingly, every statement is given a (pseudo) statement number as it is processed. A compound statement (i.e. logical IF) is given two statement numbers so that the "IF test" (IF (<exp.>)) is represented by a node, and the executable statement is represented by an additional node.

Program transfers are recognized by the parsing routines and recorded in a transition table (TRIP) by subroutine TRANS. Two additional routines (PPTRIP and NODGEN) are called from PARSER at the end of a program module to generate the final form of the transition and node tables.

Processing of Multiple Modules

The FACES system is designed to process several modules at one execution. Presumably these will be a main program and related subroutines, functions, and block data. An END statement is assumed to be present for each module and an end-of-file following the last module. Prior to processing a new module, the local system tables are reset by subroutine ZTABS. The first card of a new module is examined by a special subroutine (FIRST) to determine the module type (program, subroutine, function, block data).

Print Routines

Printing of system tables is handled by LOCPRT (local tables) and GLOPRT (global tables). Additionally, an error routine (ERROR)

is utilized to record the occurrence of errors during processing and to list these errors after processing each module.

Currently ERROR prints only the message numbers and corresponding statement numbers for a given module. A particular error message and originating subroutine may be obtained from a list of error messages included in this documentation.

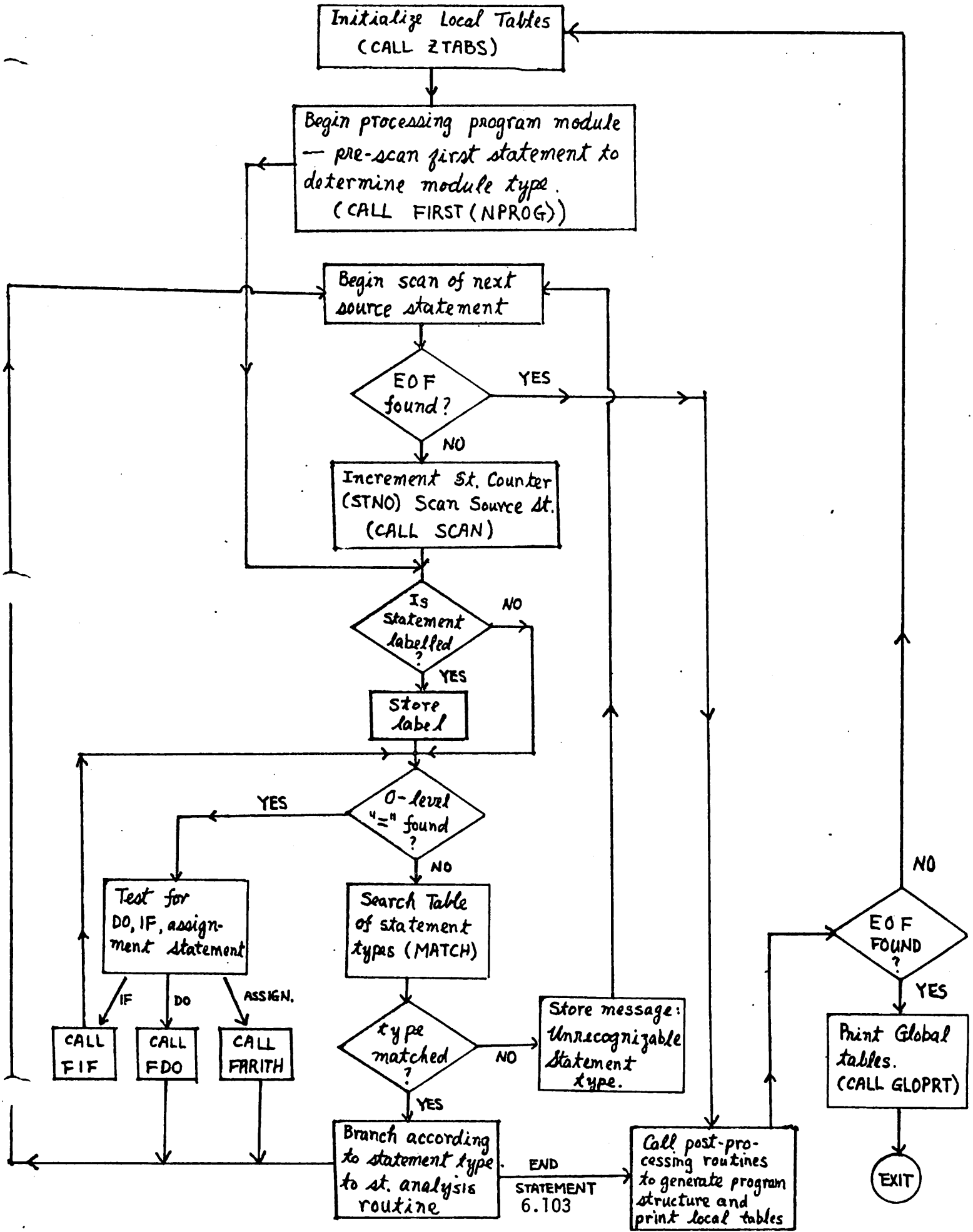
Print and Store Options

FACES requires 1 input card to be supplied by the user immediately preceding the FORTRAN source decks. This card allows the user to suppress the printing of system tables and to allow the transfer of system tables to an external file. Two variables, ISTOR and IPRT in PARSER, receive values from the input card (read in 2I1 FORMAT) which indicate the following:

ISTOR = 0	Suppress transfer of tables to external file (subroutines FILSET, OUTLT, OUTGT are not called)
ISTOR ≠ 0	Transfer tables to external file (call subroutines FILSET, OUTLT, OUTGT)
IPRT = 0	Suppress printing of tables (subroutines LOCPRT, GLOPRT not called)
IPRT ≠ 0	Print tables (call subroutines LOCPRT, GLOPRT)

Subroutines FILSET, OUTLT, OUTGT open a file, transfer local tables to that file, and transfer global tables to that file, respectively. Since the file created is required by the FACES diagnostic routines, the value of ISTOR must not be 0 if the diagnostic routines are to be executed.

PARSER


Initialize Local Tables (CALL ZTABS)
Begin processing program module — pre-scan first statement to determine module type. (CALL FIRST(NPROG))
Begin scan of next source statement
EOF found?
YES
NO
Increment St. Counter (STNO) Scan Source St. (CALL SCAN)
Is Statement labelled?
NO
YES
Store label
0-level "=" found?
YES
NO
Test for DO, IF, assignment statement
IF
DO
ASSIGN.
CALL FIF
CALL FDO
CALL FARITH
Search Table of statement types (MATCH)
type matched?
NO
YES
Store message: Unrecognizable Statement type.
Branch according to statement type to st. analysis routine
END STATEMENT
Call post-processing routines to generate program structure and print local tables
EOF FOUND?
NO
YES
Print Global tables. (CALL GLOPRT)
EXIT

SUBROUTINE PPTRIP

Subroutine PPTRIP performs a post-processing on the TRIP table as generated by the parser using subroutine TRANS. Upon the completion of parsing of a FORTRAN module, the TRIP table contains entries of five types, identified by a three-bit code in the high order bits of the second word of each two word record. PPTRIP examines each record and performs the necessary operations so that each record will contain only the appropriate pair of statement numbers to indicate program control flow, or statement number-special code pair to detail program control boundaries : entry points, exits, external program references, etc. The original three-bit code field is eliminated from the TRIP table.

The mapping used by PPTRIP for the five record types is detailed below :

First: Examine CODE field to determine record type.

Second: Reset CODE field of record to zero.

Third: Depending upon value of CODE, take indicated action:

CODE = 0: Second word of record contains pointer to a label name in the symbol table. Using this pointer, determine from the symbol table the statement number of the statement having this label and replace pointer in TRIP record with this statement number.

CODE = 1: First word of TRIP record contains a pointer to a label name in the symbol table. Replace this pointer with the statement number of the statement having this label.

CODE = 2: First word of the TRIP record is set to the value of the first word of the previous TRIP record. The second word is set to this value incremented by one.

CODE = 3: No action.

CODE = 4: External subroutine reference. Set the second word to 10000.

Subroutine SCAN

The lexical scanner of the FORTRAN Analysis Package scans FORTRAN statements character by character and organizes the general format of the statements in a coded form to be interpreted by the parsing routine. The scanner recognizes entities in a statement and forms a list of alphabetic codes for the entities in an array called ISS (Intermediate Symbol String). The entities that are recognized and their codes are as follows:

statement label	S
alphanumeric string	V
integer constant	I
floating point constant	F
double precision constant	D
complex constant	C
hollerith constant	H
logical constant	T
logical operator	L
relational operator	R

All special characters such as "=", "+", "-", etc., are represented in ISS by their respective character codes.

The scanner also stores all alphanumeric strings and integer constants in a table called TSTAB, a 2 by 300 word array. Alphanumeric strings are stored in blocks of eight characters or less. That is, if a string contains eight or less characters, it will be stored 4 characters per word (left-justified, blank filled) in two words of TSTAB. If a string contains more than eight characters, each block of eight characters will be stored in

TSTAB and the corresponding code will be entered into ISS.

The following examples will illustrate the results of some calls to SCAN.

Example 1

Source statement:

```
SUM = SUM + IFUNC1 (10, M).
```

TSTAB (1, N)	TSTAB (2, N)
SUM	
SUM	
IFUN	C1
10	
I	

ISS : V = V + V (I , V)

Example 2

INTEGER FUNCTION INVERT (INPUT 1, INPUT 2)

TSTAB (1, N)	TSTAB (2, N)
INTE	GERF
UNCT	IONI
NVER	T
INPU	T1
INPU	T2

ISS : V V V (V , V)

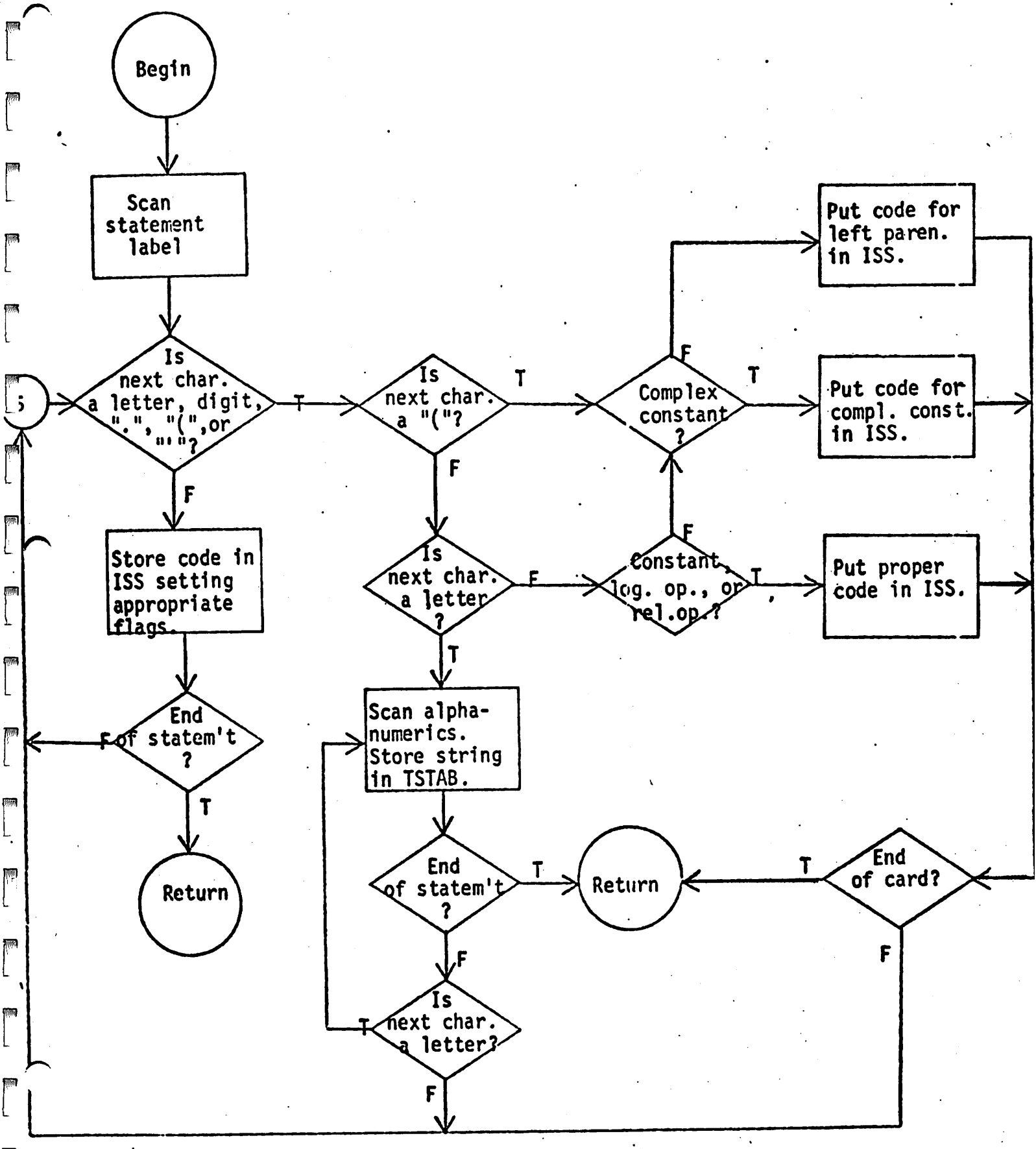
Example 3

```
20  IF (.NOT. (A.EQ.2HON))  GO TO 10.
```

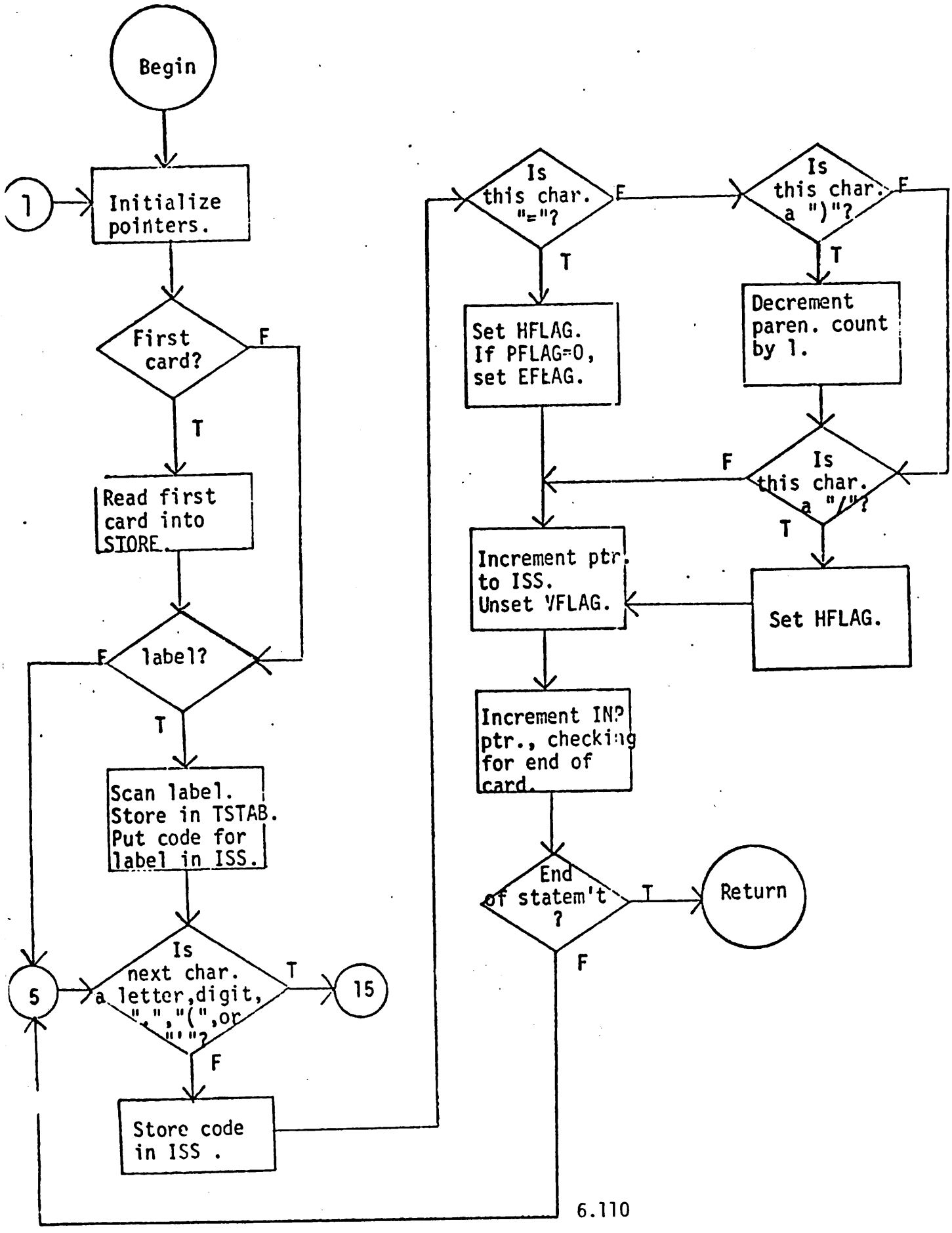
TSTAB (1, N)	TSTAB (2, N)
20	
IF	
A	
GOTO	10

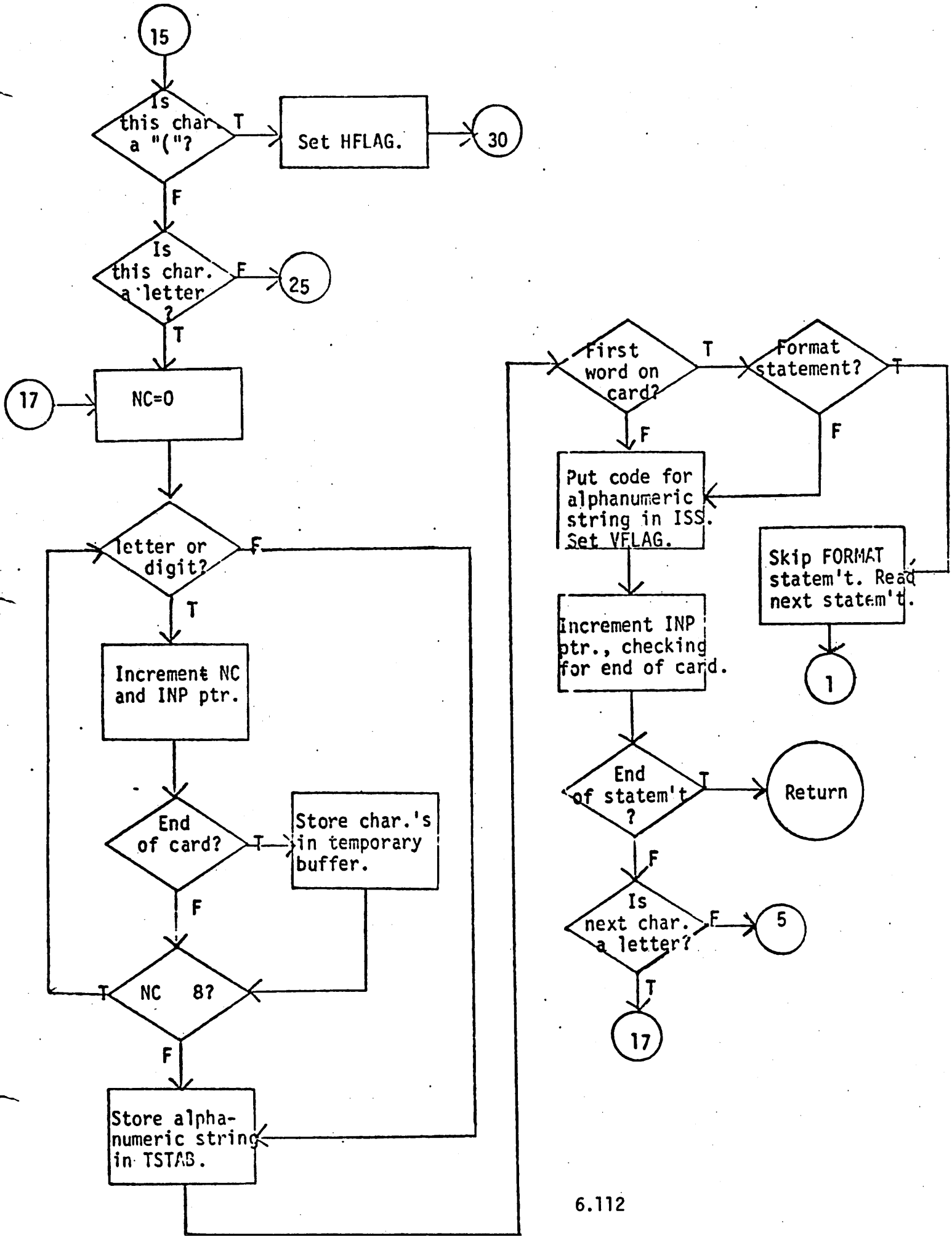
ISS : S V (L (V R H)) V

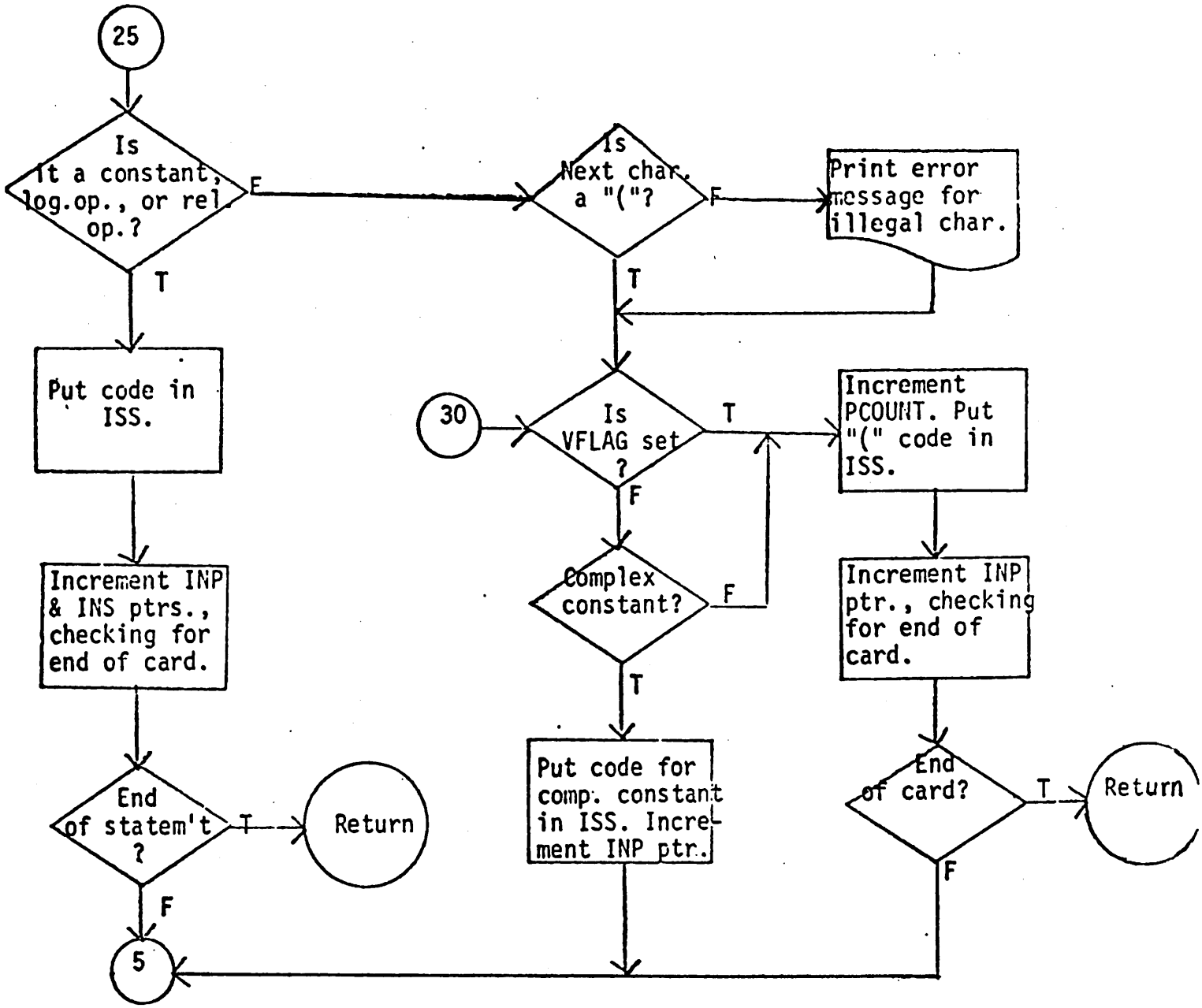
Brief Flowchart of SCAN



Detailed Flowchart of SCAN




15
Is this char a "("?
T
Set HFLAG.
30
F
Is this char. a letter ?
F
25
T
17
NC=0
letter or digit?
F
T
Increment NC and INP ptr.
End of card?
T
Store char.'s in temporary buffer.
F
NC 8?
T
F
Store alpha-numeric string in TSTAB.
First word on card?
T
Format statement?
T
F
F
Put code for alphanumeric string in ISS. Set VFLAG.
Skip FORMAT statem't. Read next statem't.
1
Increment INP ptr., checking for end of card.
End of statem't ?
T
Return
F
Is next char. a letter?
F
5
T
17


25
Is it a constant, log.op., or rel. op.?
E
T
Is Next char. a "("?
F
Print error message for illegal char.
T
Put code in ISS.
30
Is VFLAG set ?
T
Increment PCOUNT. Put "(" code in ISS.
F
Complex constant?
F
T
Increment INP & INS ptrs., checking for end of card.
Increment INP ptr., checking for end of card.
End of statem't ?
T
Return
F
Put code for comp. constant in ISS. Increment INP ptr.
End of card?
T
Return
F
5

SUBROUTINE SEXP

SEXP is a generalized parsing routine designed to process subscript (or parameter) lists and expressions (arithmetic and relational) when encountered by one of the parsing routines. The list or expression may be delimited by the end of the statement or a ")". Since parentheses may be nested within the list or expression, a parentheses count (PC) is kept to identify the current level of nesting. PC is initialized to 1, incremented for each "(", and decremented for each ")". Thus, if PC reaches 0, the ")" is the terminator of the list or expression.

Information passed to SEXP includes common blocks SCAPA and PNTRS containing the ISS and TSTAB arrays and current pointers (JISS, JTST) to these arrays.

MODE specify the type of list for processing:

= 2 process exp. in assignment st

= 15 process subscript exp.

= 19 process subroutine actual parameters

= 20 process conditional branch decision variables

Upon entry to SEXP, the pointers point to the first symbol and character string of the list or expression. Upon return, if no end of statement condition is detected, JISS points to the ")" in ISS and JTST points to the first character string (in TSTAB) following the list or expression.

In order to give SEXP recursive properties (i.e. process lists (or expressions) within lists (or expressions)), a stack (STACK) is maintained to record the type of list or expression at each level of nesting. The i^{th} entry in STACK is actually the use code (passed to subroutine TABS) associated with names at the i^{th} level of nesting, and

the index to STACK is PC, the parentheses count. The initial entry in STACK is set to the value of MODE, the input parameter to SEXP.

The occurrence of a subroutine or function name with parameter list requires special handling by SEXP. In order to effect the proper storage of parameters in the system parameter table (PARTAB), the position number (stored in ARGNO) for each parameter is passed to subroutine TABS through common block TAPAR. Additionally, since nesting may be allowed, a separate argument stack (ARGSTK) is maintained to record the current argument (parameter) number when a new argument list is encountered. Accordingly, each occurrence of a function or subroutine name followed by an argument list causes the current argument number (ARGNO) to be placed on the argument stack (ARGSTK) and ARGNO to be reset to 1. The argument number is incremented when a "," is encountered within an argument list.

The following example illustrates the use of the use code stack (STACK) and the argument stack (ARGSTK). Consider the following assignment statement.

```
VAR1 = FUN1 (I, 100, FUN2 (X, Z(L)), VAR3) + VAR2.
```

PARSER calls FARITH which in turn calls SEXP with MODE = 2 to process the expression to the right of the equal sign. While SEXP is processing this statement, the following events take place:

STACK (1)	= 2	initialized to value of MODE
PC	= 1	parenthesis count
ASI	= 0	pointer to ARGSTK
ARGNO	= 0	

FUN1: Store FUN1 as a function name in tables

(: increment PC (PC = 2)

increment ASI (ASI = 1)

STACK (2) = 18

ARGSTK (1) = ARGNO = 0

reset ARGNO to one (1)

I : store I as a function actual parameter in table

use code = STACK(PC) = 18

, : increment ARGNO (ARGNO = 2)

100 : Store 100 as a constant parameter in table

, : increment ARGNO (ARGNO = 3)

FUN2: Store FUN2 as a function name in tables

(: increment PC (PC = 3)

STACK (3) = 18

ASI = 2

ARGSTK (2) = ARGNO = 3

reset ARGNO to 1 (one)

X : Store X in table as a function actual parameter

, : increment ARGNO (ARGNO = 2)

Z : Store Z as an array in tables

(: increment PC (PC = 4)

STACK (4) = 15 (subscripts)

L : Store L as a subscript in table

) : Since STACK(PC) = 15 (subscripts), pop use code stack

PC = 3

) : Since STACK(PC) = 18 (function actual parameter), this is

the end of a parameter list. Restore the argument number of

previous parameter list.

ARGNO = ARGSTK(ASI) = 3

ASI = ASI - 1 = 1.

Pop use code stack PC = 2

, : increment argument number (ARGNO = 4)

VAR3: Store VAR3 as a function actual parameter in tables

) : Since STACK(PC) = 18 (function actual parameter), this is the end of a parameter list.

Restore the argument number of previous list.

ARGNO = ARGSTK(ASI) = 0

ASI = ASI - 1 = 0

Pop use code stack. PC = 1

+ : no action

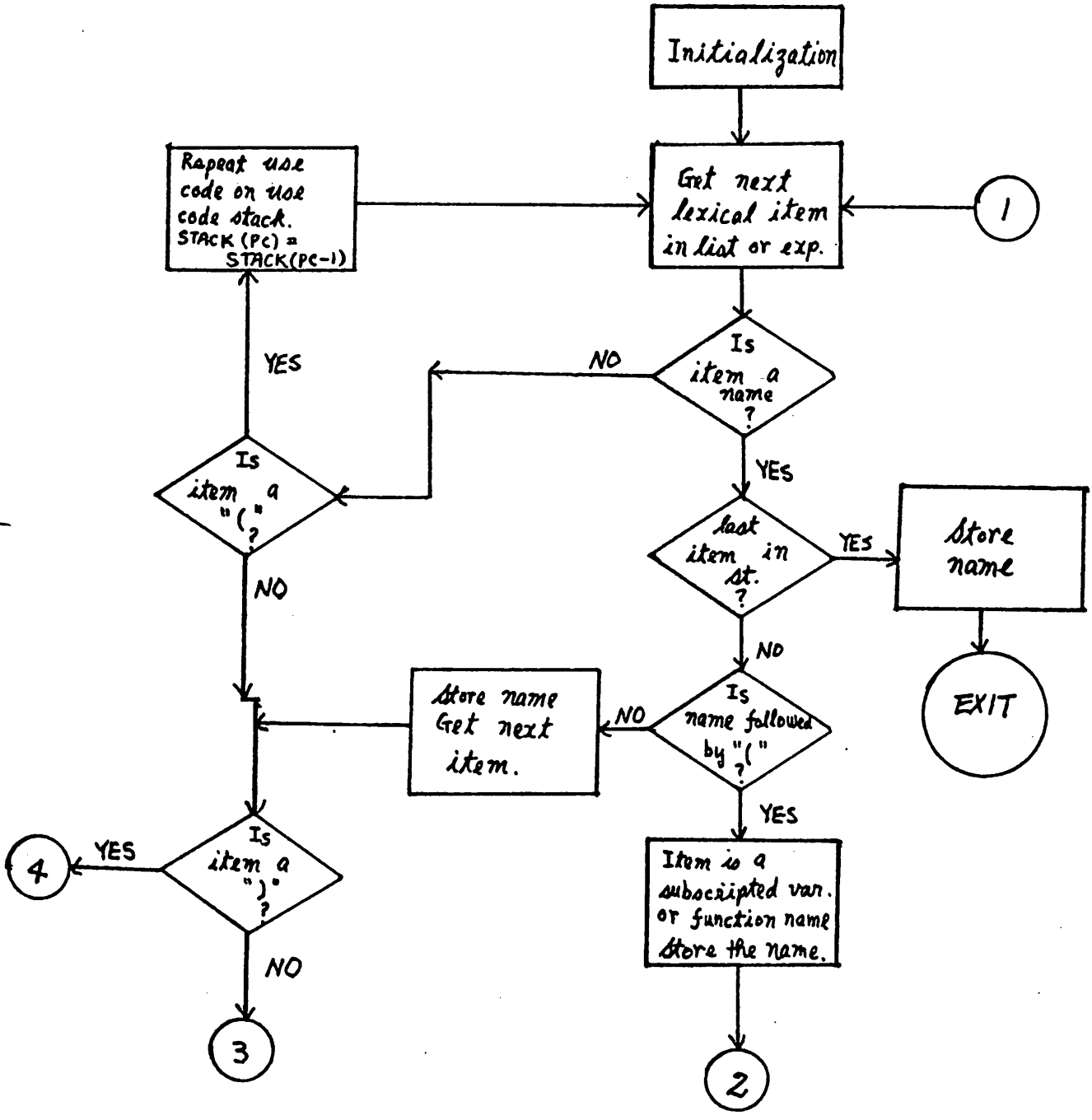
VAR2: Store VAR2 as an input variable

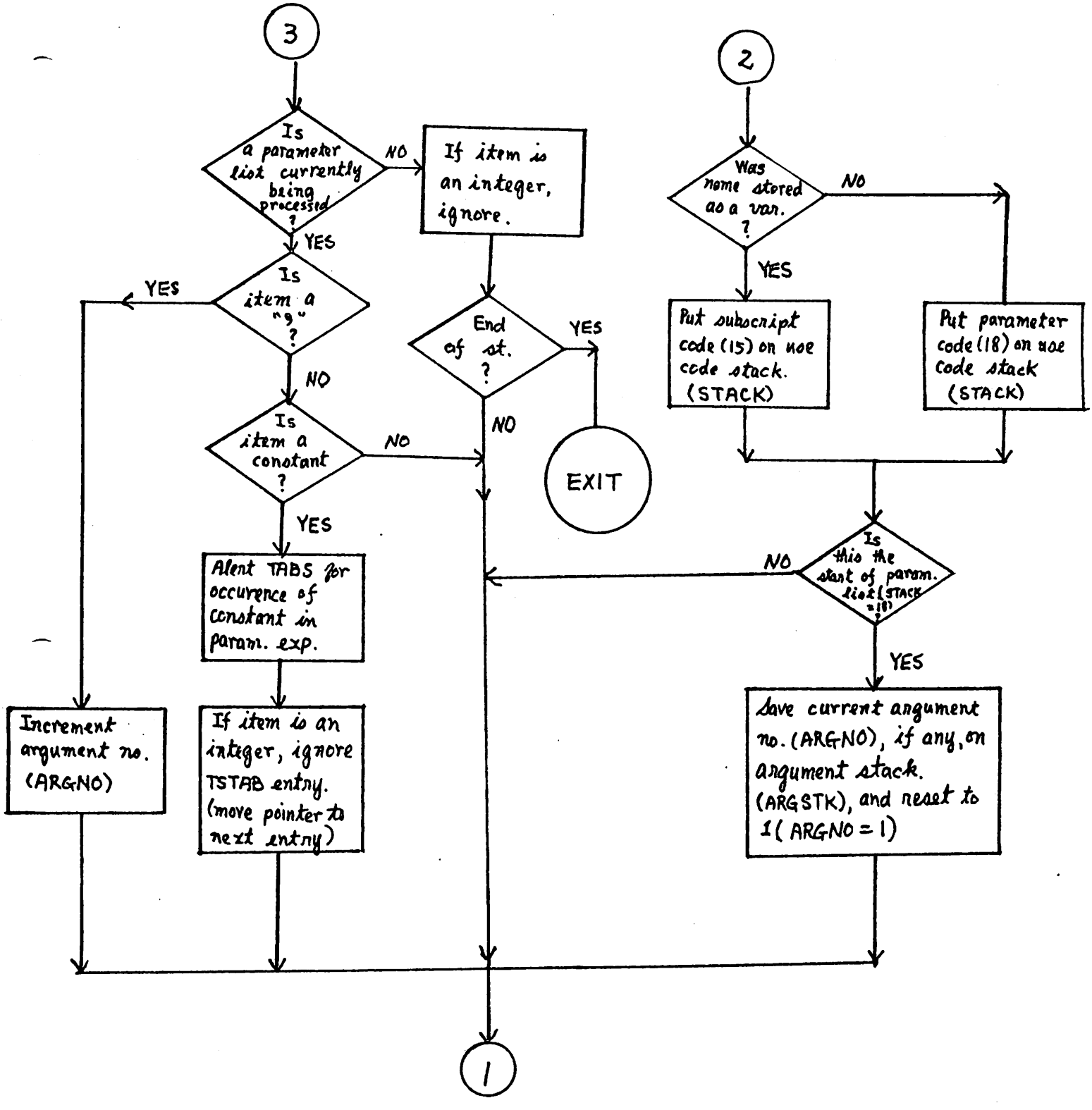
Use code = STACK(PC) = 2.

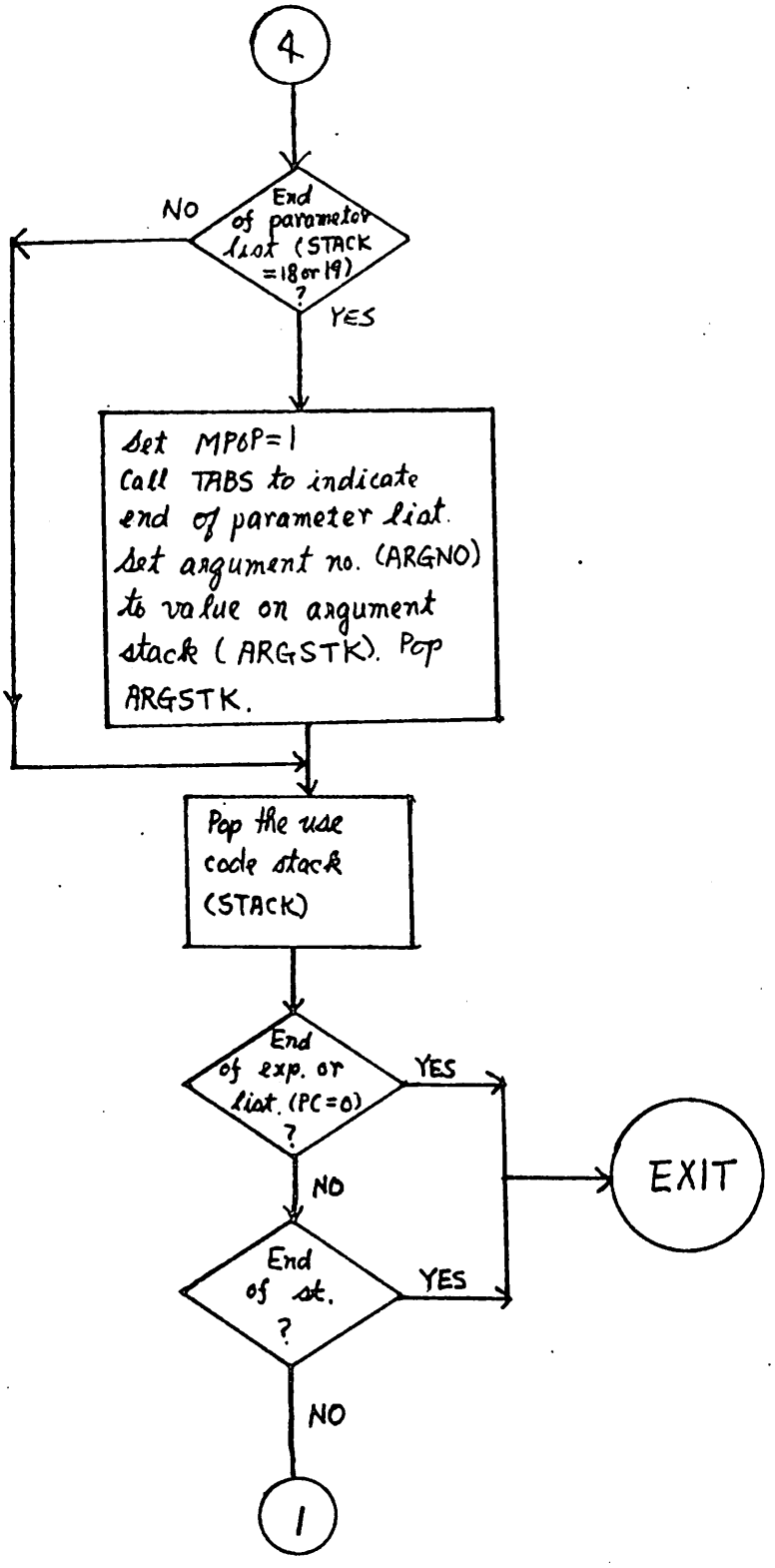
RETURN.

One further note concerning parameter lists should be made. At the end of a parameter list a special call to TABS is made (with MPOP = 1) to alert TABS of this occurrence. Additionally, a call to TABS (with MPOP = 2) is made when a constant is found as a parameter. This differs from the usual practice of ignoring constants in expressions.

SEXP


Initialization
Get next lexical item in list or exp.
1
Is item a name ?
NO
YES
Repeat use code on use code stack. STACK (PC) = STACK(PC-1)
Is item a "(" ?
YES
NO
last item in st. ?
YES
Store name
EXIT
NO
Is name followed by "(" ?
NO
Store name Get next item.
YES
Item is a subscripted var. or function name Store the name.
Is item a ")" ?
YES
4
NO
3
2


3
Is a parameter list currently being processed ?
NO
YES
If item is an integer, ignore.
Is item a "9" ?
YES
NO
Is item a constant ?
NO
YES
End of st. ?
YES
NO
EXIT
Alert TABS for occurence of constant in param. exp.
If item is an integer, ignore TSTAB entry. (move pointer to next entry)
Increment argument no. (ARGNO)
2
Was name stored as a var. ?
NO
YES
Put subscript code (15) on use code stack. (STACK)
Put parameter code (18) on use code stack (STACK)
Is this the start of param. list (STACK = 18) ?
NO
YES
Save current argument no. (ARGNO), if any, on argument stack. (ARGSTK), and reset to 1 (ARGNO = 1)
1


4
End of parameter list (STACK =18 or 19) ?
NO
YES
Set MPOP=1
Call TABS to indicate end of parameter list.
Set argument no. (ARGNO) to value on argument stack (ARGSTK). Pop ARGSTK.
Pop the use code stack (STACK)
End of exp. or list. (PC=0) ?
YES
NO
EXIT
End of st. ?
YES
NO
1

SUBROUTINE SHIFTY(X,Y,IDUMMY,JDUMMY)

Subroutine SHIFTY is called by the various parsing routines to extract and left-justify a portion of a given character string. Parameters required are output array X, input array Y, character position IDUMMY, and number of characters JDUMMY. Character strings are stored in X and Y with 4 characters per word, left justified and blank filled. The following example illustrates the use of SHIFTY.

Input Y: ONET WOTH REE

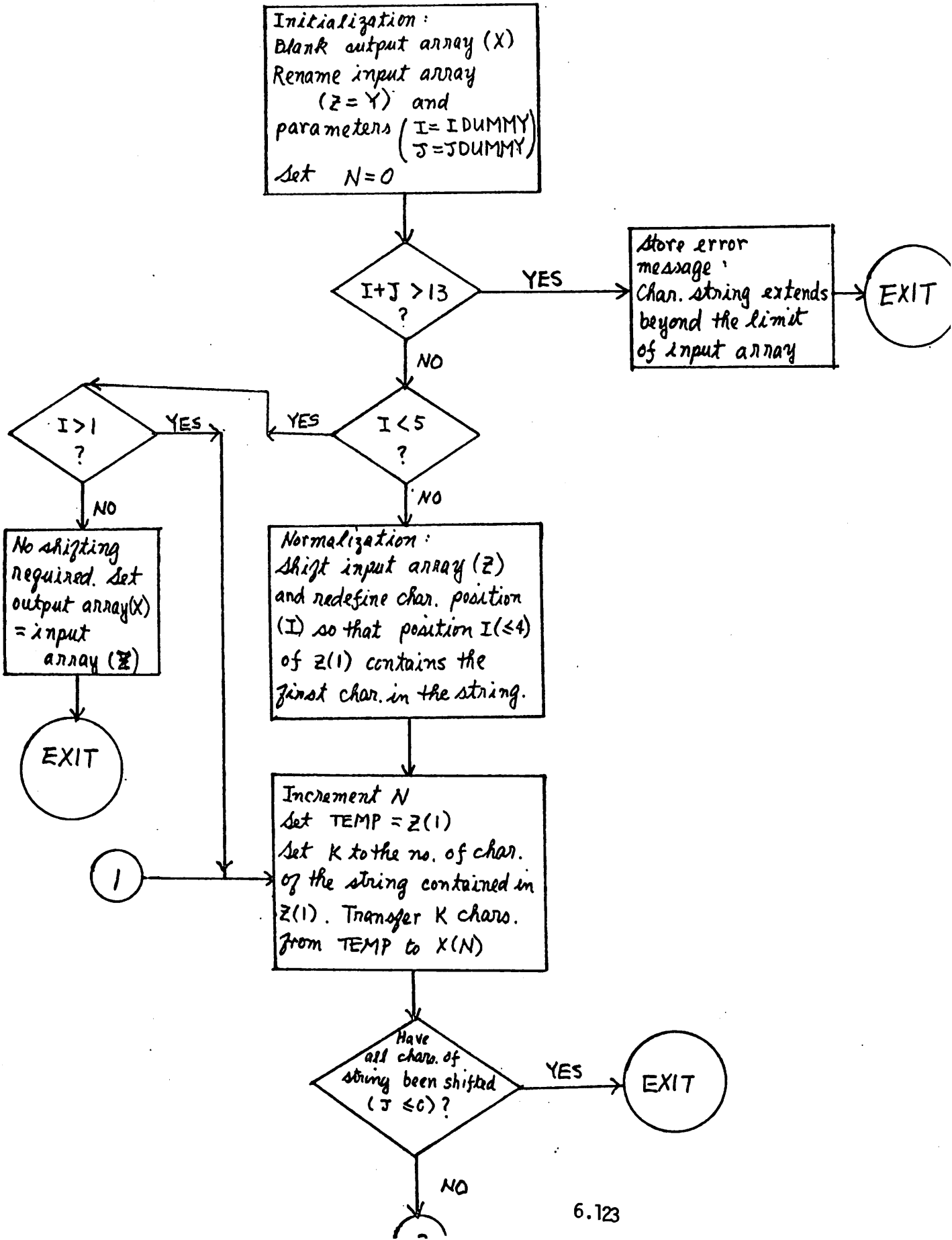
Output X: THRE E

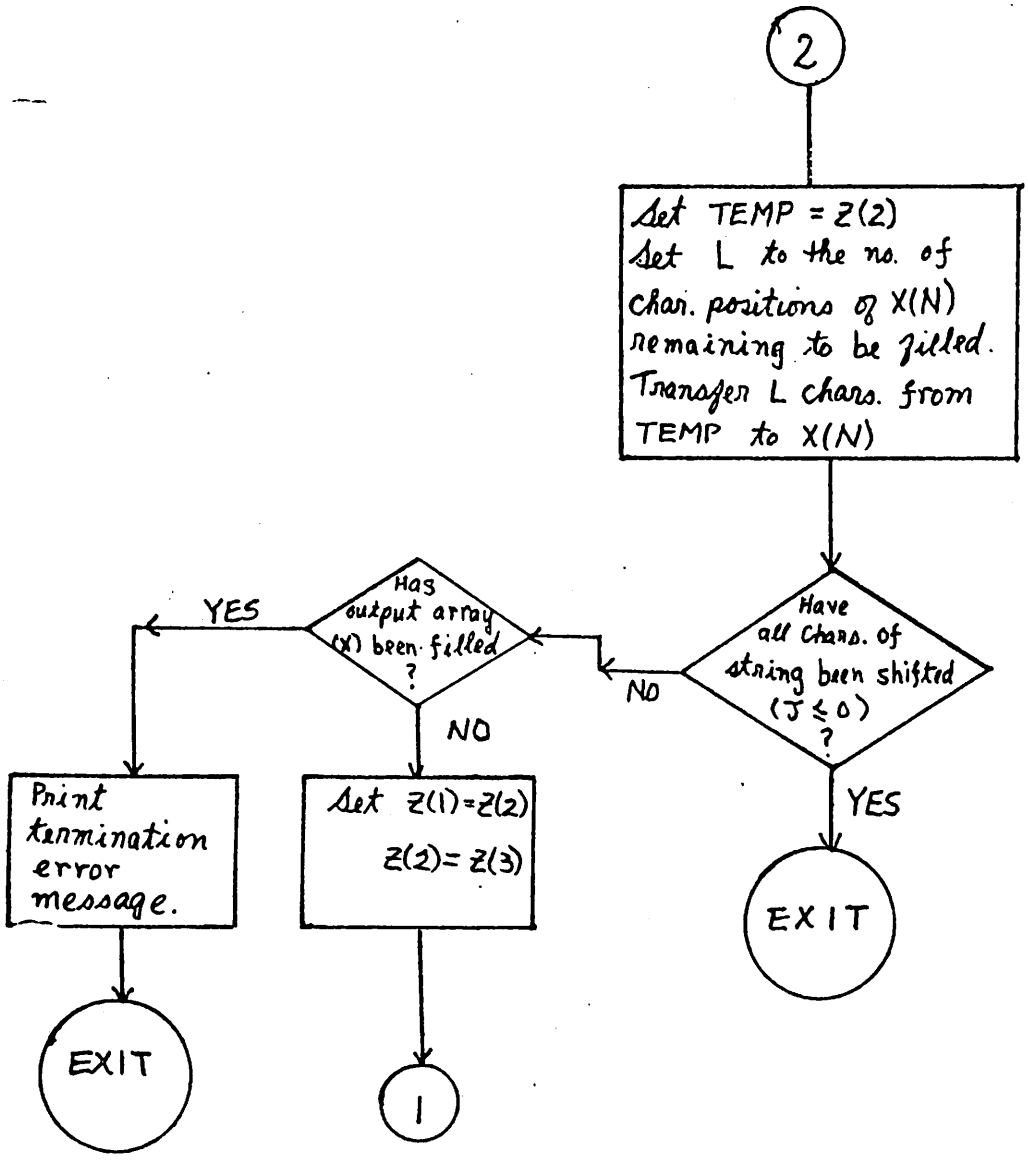
The above results are effected by "CALL SHIFTY(X,Y,7,5)", where 7 indicates the 7^{th} character of the 12 character string in Y, and 5 indicates the length of the desired character string to be transferred to X.

Inputs Y,IDUMMY,JDUMMY are saved at the outset so that they maintain the same values on return as they have upon entering SHIFTY.

Shifting is performed by the FLD function.

SHIFTY

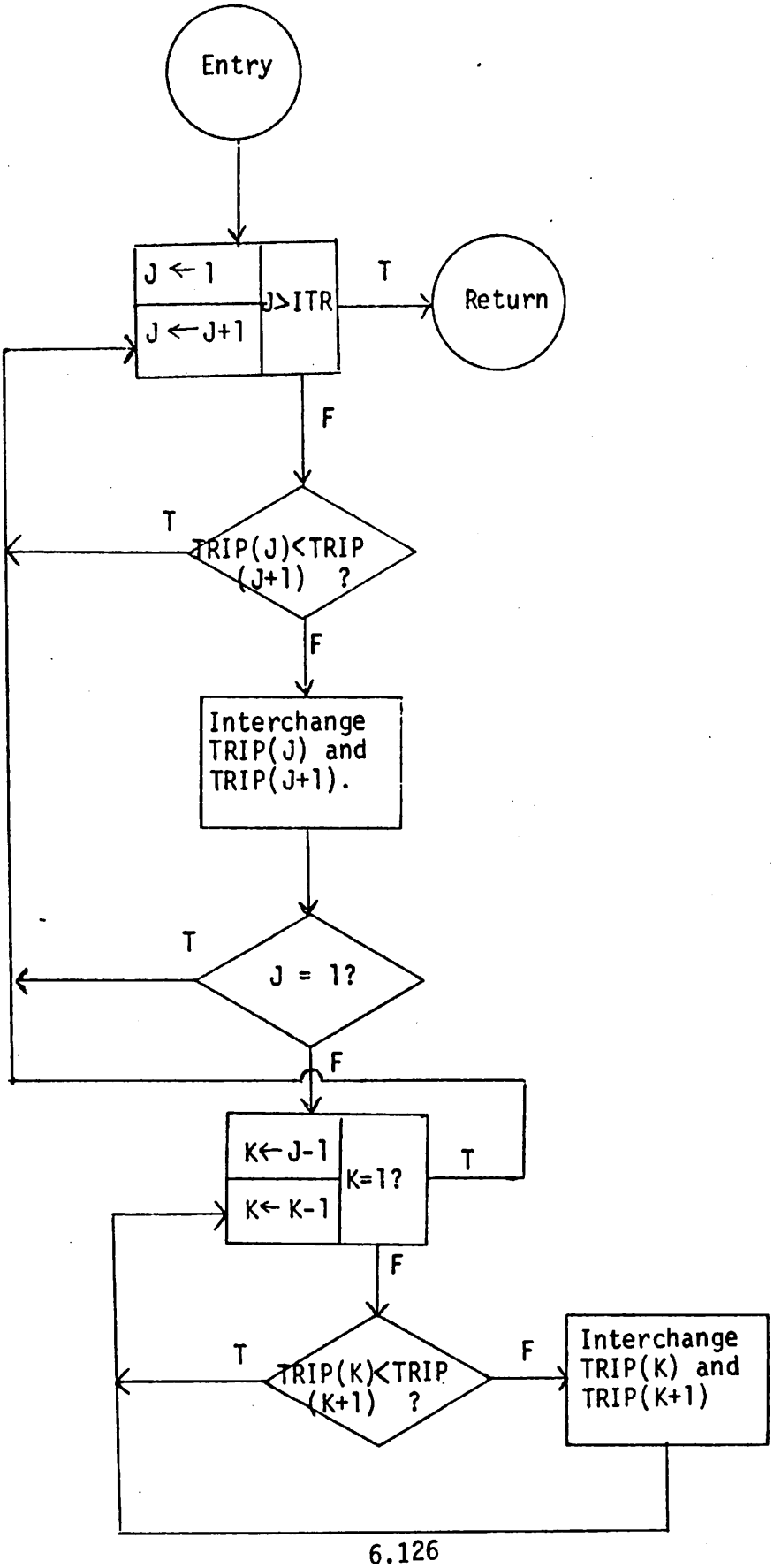

Initialization:
Blank output array (X)
Rename input array
(Z=Y) and
parameters (I=IDUMMY, J=JDUMMY)
Set N=0
I+J >13 ?
YES
Store error message:
Char. string extends beyond the limit of input array
EXIT
NO
I<5 ?
YES
NO
Normalization:
Shift input array (Z) and redefine char. position (I) so that position I(≤4) of Z(1) contains the first char. in the string.
I>1 ?
YES
NO
No shifting required. Set output array(X) = input array (Z)
EXIT
1
Increment N
Set TEMP = Z(1)
Set K to the no. of char. of the string contained in Z(1). Transfer K chars. from TEMP to X(N)
Have all chars. of string been shifted (J ≤ C)?
YES
EXIT
NO


2
Set TEMP = Z(2)
Set L to the no. of char. positions of X(N) remaining to be filled.
Transfer L chars. from TEMP to X(N)
Have all chars. of string been shifted (J ≤ 0) ?
NO
YES
Has output array (X) been filled ?
YES
NO
Print termination error message.
EXIT
Set Z(1)=Z(2)
Z(2)=Z(3)
1
EXIT

Subroutine SORT (ICOL)

This routine sorts the TRIP table in monotonically increasing order according to the column indicated by parameter ICOL. (First column contains predecessors of the non-normal transition pairs; second column contains successors.) A shuttle-interchange sorting algorithm is used in which the list is searched from the beginning until a number is found that is out of place. A pointer is set to this place and the number is moved to its appropriate place. Then the process is repeated starting at the place where the pointer was set.

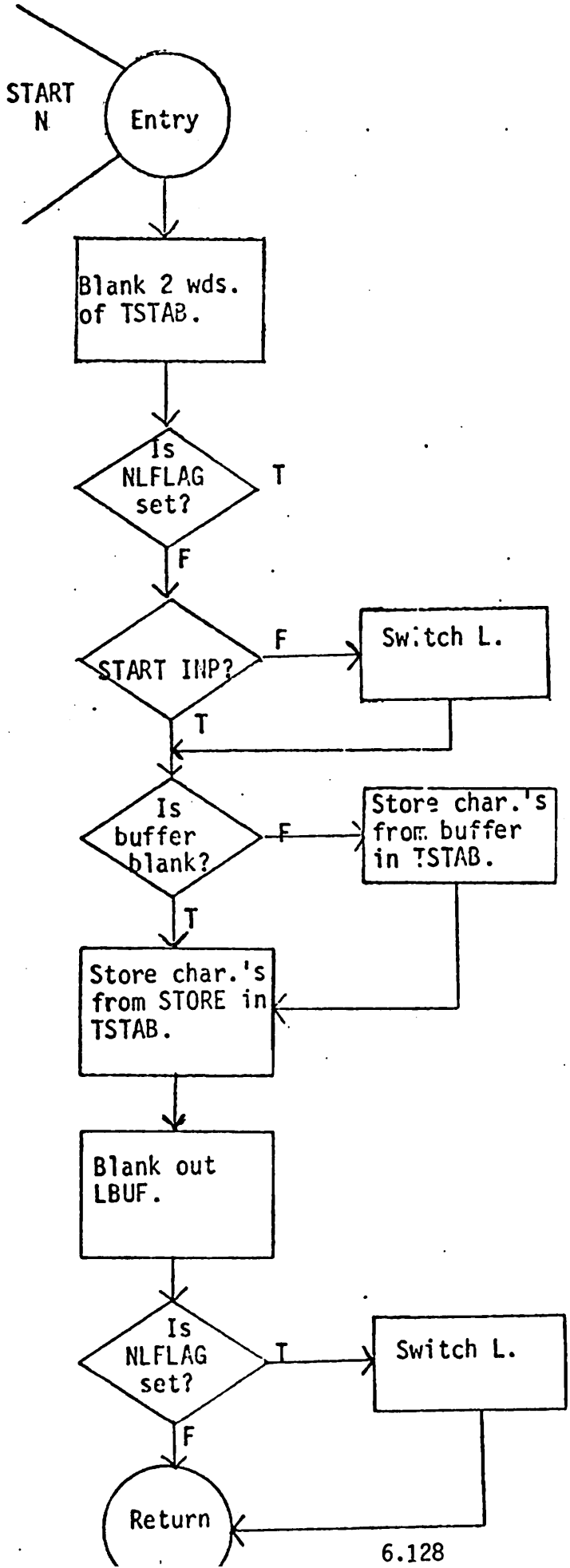
Flowchart for SORT



Subroutine STOW (ISTART,N)

This routine stores a string of alphanumeric characters (maximum 8) from the STORE input buffer into TSTAB - the table of names that is sent to the parser. Input parameter ISTART indicates the starting point of the string in the input buffer and N denotes the number of characters in the string. The characters are stored in TSTAB four characters per word, left justified, blank filled.

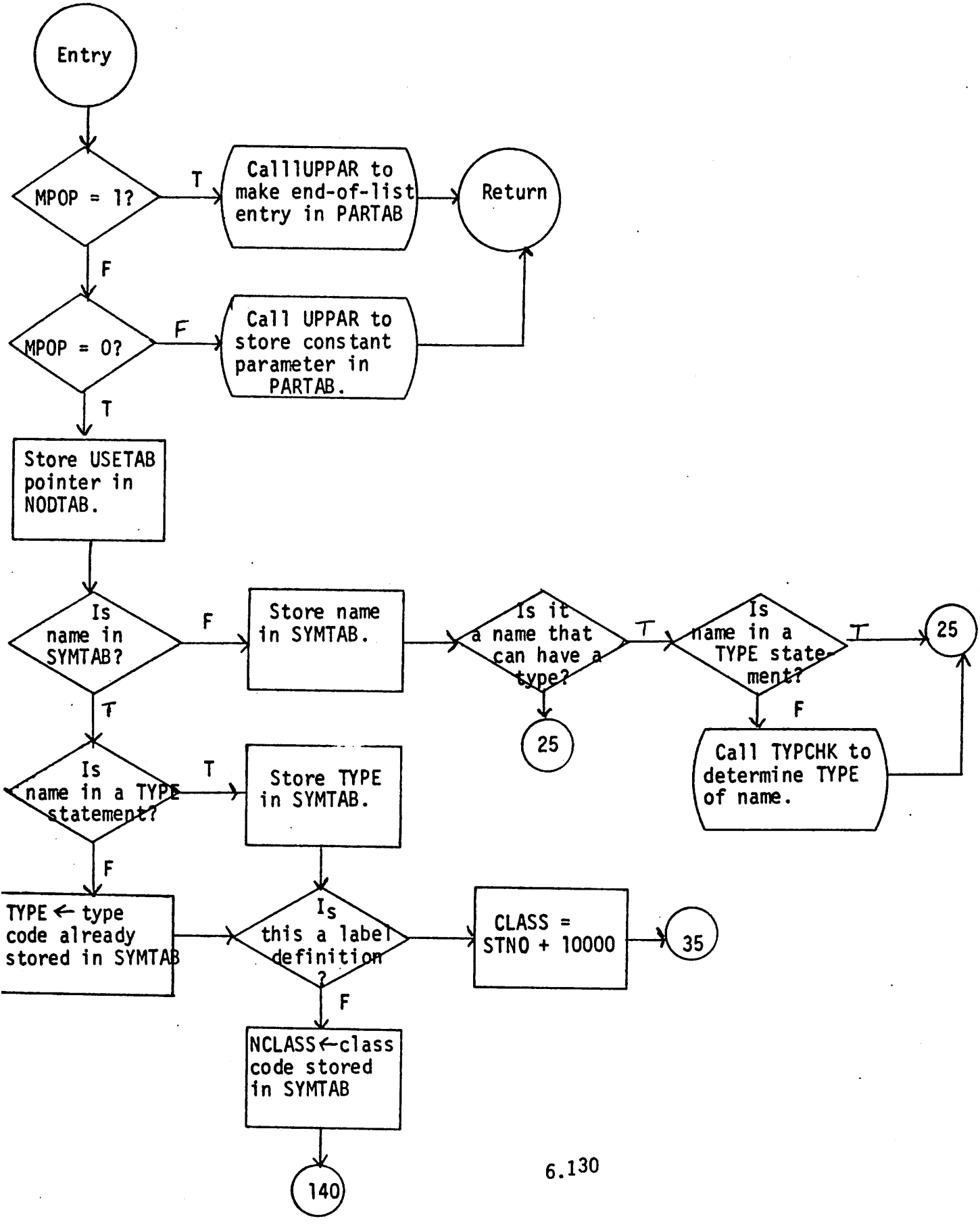
Flowchart for STOW

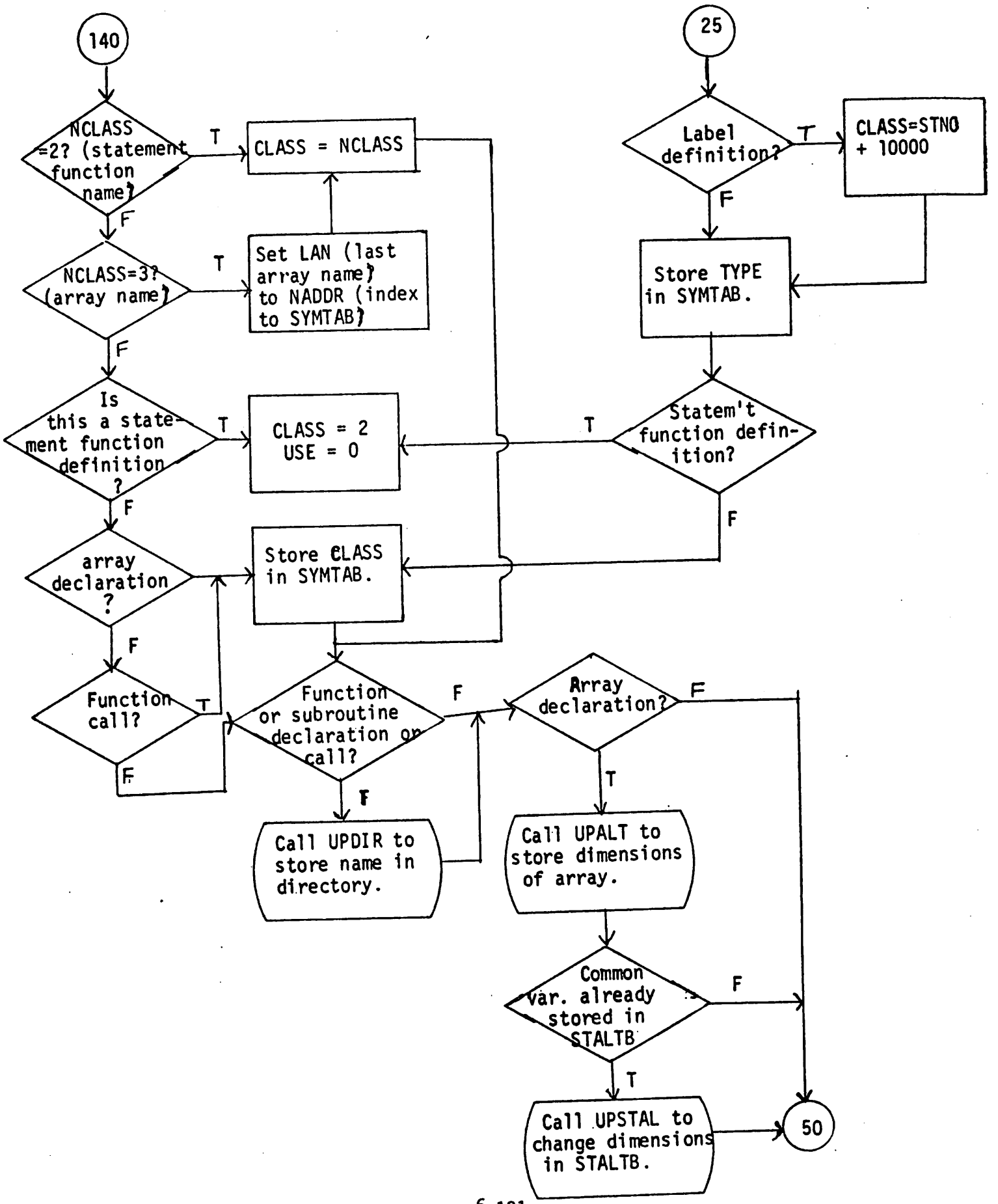


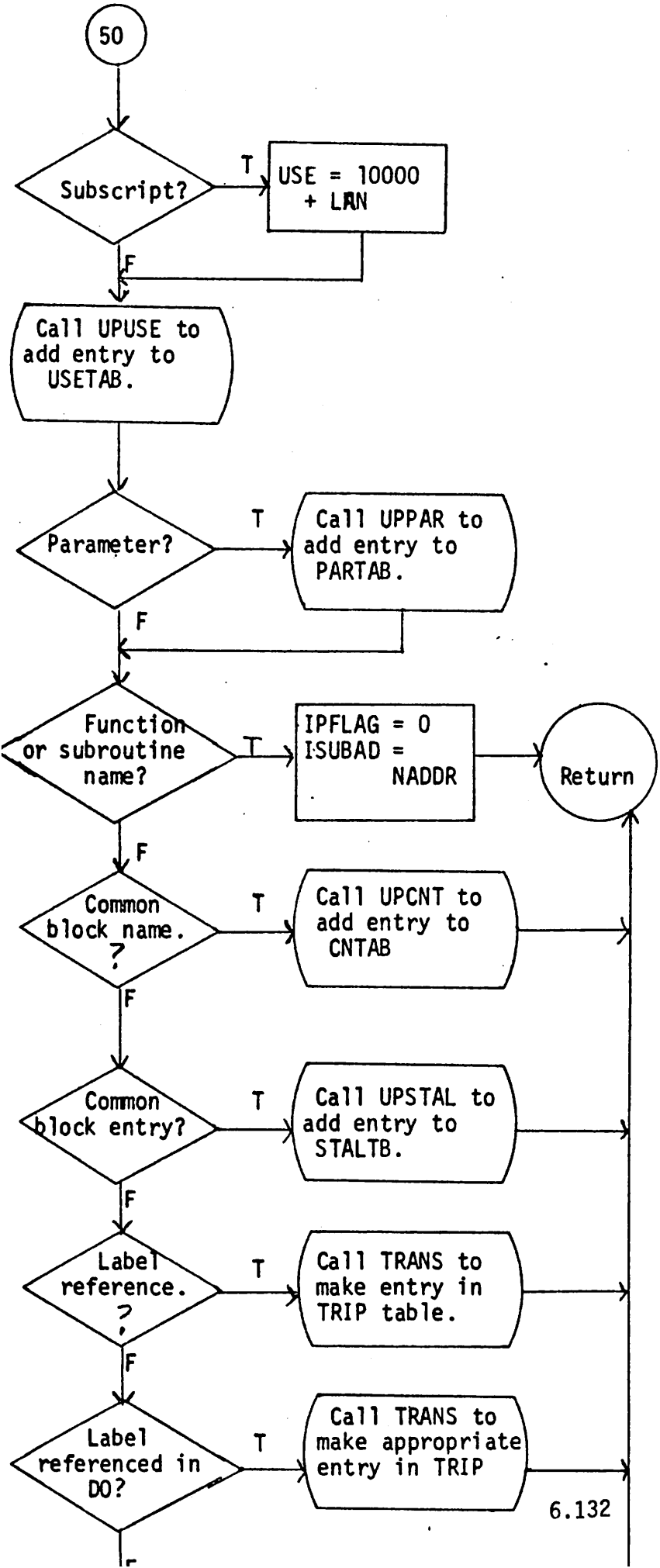
Subroutine TABS (CLASS, TYPE, USE)

TABS is the main table-managing routine. It controls storage into all tables except TRIP and DOTAB. Any time an entry needs to be made in a table(s), TABS is called by the PARSER (or one of its aubroutines) to make the entry. TABS makes a decision as to what kind of entry and into which table an entry is to be made by analyzing the parameters indicating class, type, and use codes and the information stored in the common variables of common block TAPAR. Class code identifies the symbol type (label, variable, subroutine name etc.) Type code is assigned to variable or function name to designate variable type (real, logical etc.) Use code determines how the symbol is referenced in the statement (subscript, Do-loop index etc.) (Detailed description of the class, type and use code can be found in SEC.III of User Manual). After a decision has been made, TABS usually calls one or more of the table-updating subroutines to actually do the storing.

Flowchart for TABS

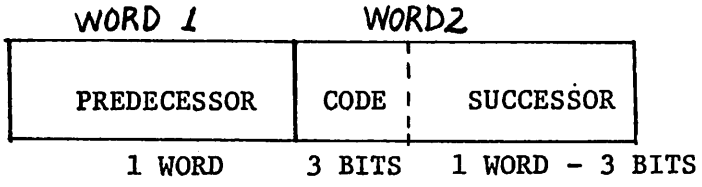



140
NCLASS =2? (statement function name)
T
CLASS = NCLASS
F
NCLASS=3? (array name)
T
Set LAN (last array name) to NADDR (index to SYMTAB)
F
Is this a statement function definition ?
T
CLASS = 2
USE = 0
F
array declaration ?
Store CLASS in SYMTAB.
F
Function call?
T
F
Function or subroutine declaration or call?
F
F
Call UPDIR to store name in directory.
Array declaration?
F
T
Call UPALT to store dimensions of array.
Common var. already stored in STALTB
F
T
Call UPSTAL to change dimensions in STALTB.
50
25
Label definition?
T
CLASS=STNO + 10000
F
Store TYPE in SYMTAB.
Statem't function definition?
T
F


50
Subscript?
T
USE = 10000 + LAN
F
Call UPUSE to add entry to USETAB.
Parameter?
T
Call UPPAR to add entry to PARTAB.
F
Function or subroutine name?
T
IPFLAG = 0
ISUBAD = NADDR
Return
F
Common block name. ?
T
Call UPCNT to add entry to CNTAB
F
Common block entry?
T
Call UPSTAL to add entry to STALTB.
F
Label reference. ?
T
Call TRANS to make entry in TRIP table.
F
Label referenced in DO?
T
Call TRANS to make appropriate entry in TRIP
F

SUBROUTINE TRANS (LOC, MOD)

Subroutine TRANS is called by the various parser subroutines to make entries in the non-normal transition pairs table, TRIP. The TRIP table stores program control transfers other than the standard fall through transitions. The TRIP table records are each two words long as shown below.



TRIP Table Record

The predecessor field will contain the statement number of the originating statement of the transition. The successor field will contain the statement number of the target statement. The three bit code field is used temporarily by TRANS and subroutine PPTRIP to identify the type of record stored.

TRANS creates TRIP records for six transition types. The type of transition, and hence the mode of operation of TRANS, is indicated by the input argument, MOD. The values of MOD and the corresponding transitions are given below.

MOD Cause of transition

1 - Explicit transfer to label

2 - Label referenced in DO statement

3 - Terminal statement (STOP, RETURN)

4 - One Way Logical IF statement

5 - External subroutine reference

6 - END statement.

In the case of references to labels, the input argument LOC contains a pointer to the symbol table entry for the appropriate label. The common variable STNO contains the value of the current statement number. There follows a brief description of the actions taken by TRANS for the possible values of MOD:

If MOD equals 1: Condition indicates transfer to statement label indicated by LOC. predecessor = STNO, CODE = 0, Successor = LOC.

If MOD equals 2: Condition indicates label referenced in a DO statement. predecessor = LOC, CODE = 1, Successor = STNO. Second TRIP table entry is made with CODE = 2 and statement successor left undefined.

If MOD equals 3: Condition indicates a RETURN or STOP statement. predecessor = STNO, CODE = 3, successor = 20,000.

If MOD equals 4: Condition indicates a one way logical IF statement. Two TRIP table entries made. First: predecessor = STNO, CODE = 3, successor = STNO + 1; predecessor = STNO, CODE = 3, successor = STNO + 2.

If MOD = 5: Condition indicates external subroutine reference. Two entries made. First: predecessor = STNO, CODE = 4, successor = LOC = 0 Second: predecessor = STNO, CODE = 3, successor = STNO + 1.

If MOD = 6: Condition indicates end statement.

predecessor = STNO, CODE = 3, successor = 30,000.

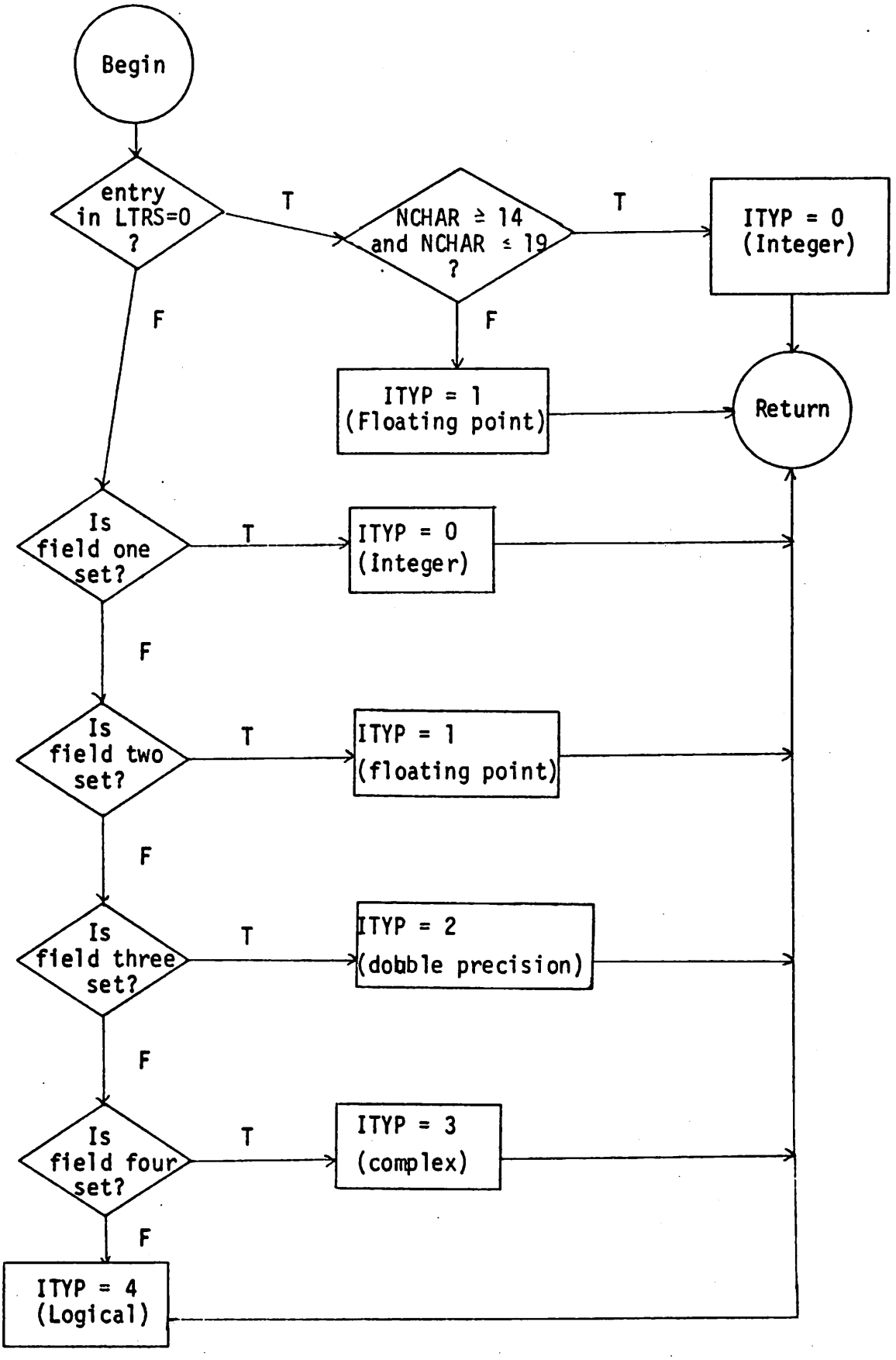
If MOD = 7: Condition indicates program entry point.

predecessor = 40,000, CODE = 3, successor = STNO.

Subroutine TYPCHK (NAME, ITYP)

This is a machine-dependent routine that checks the type of a FORTRAN variable. The first four characters of the name are passed through input parameter NAM and the typ of the variable is passed through output parameter ITYP. TYPCHK determines the type of the variable by checking the entry in the LTRS table that corresponds to the first letter of the variable. If the word is zero, then the FORTRAN defined implicit typing is used. Otherwise, the type is determined according to which bit is set in the word in LTRS.

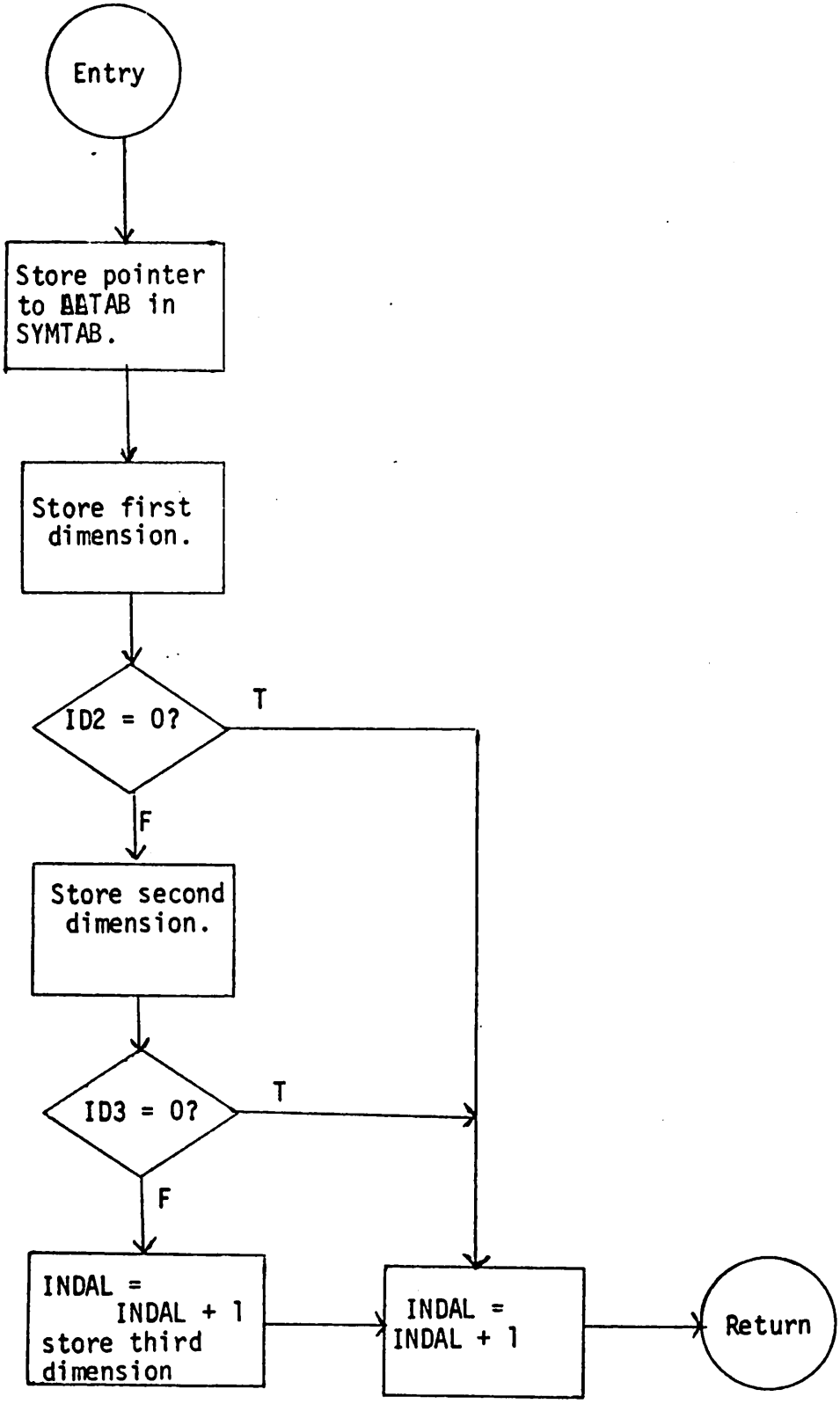
Flowchart for TYPCHK



Subroutine UPALT (ID1, ID2, ID3)

This routine updates ALTAB (Array Length Table) each time an array is dimensioned. The dimensions to be stored are passed through parameters ID1, ID2, and ID3. If the array is linear, ID2 = ID3 = 0. If the array has two dimensions, ID3 = 0. UPALT also stores the pointer in SYMTAB to ALTAB.

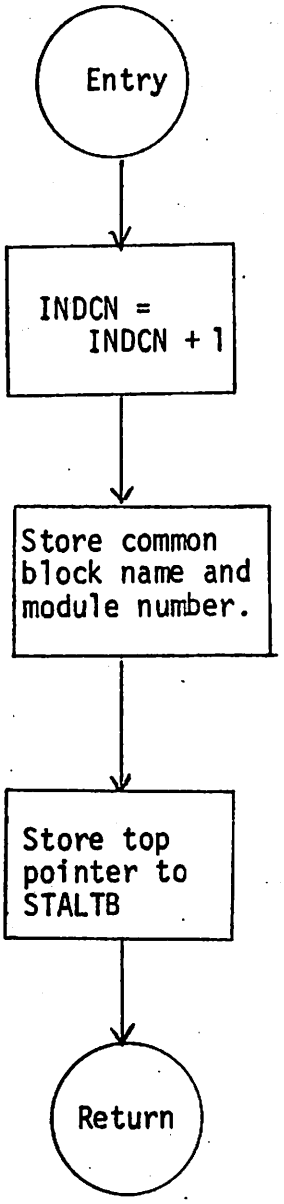
Flowchart for UPALT



Subroutine UPCNT (NAM1, NAM2, NADNO)

This routine makes an entry in CNTAB (Common Name Table) when-ever a common block is declared. NAM1 and NAM2 contain the name of the common block (four characters/word) and MODNO contains the module number in which the common block is being declared. UPCNT also stores the STALTB top pointer in CNTAB. STALTB top pointer points to the starting location of STALTB where the common block entries will be stored.

Flowchart for UPCNT



Subroutine UPDIR (NAM1, NAM2, NF, MODNO)

This routine updates DIREC (global directory) each time a subroutine, function, or program name is encountered. The hash code* is calculated using the module name (stored in NAM1 and NAM2) and the name is entered in DIREC if it is not already there. If the name is already in the directory and it is now being defined (NF ≠ 0), the file number (NF) and module number (MODNO) are changed. (When a module is undefined, its module number

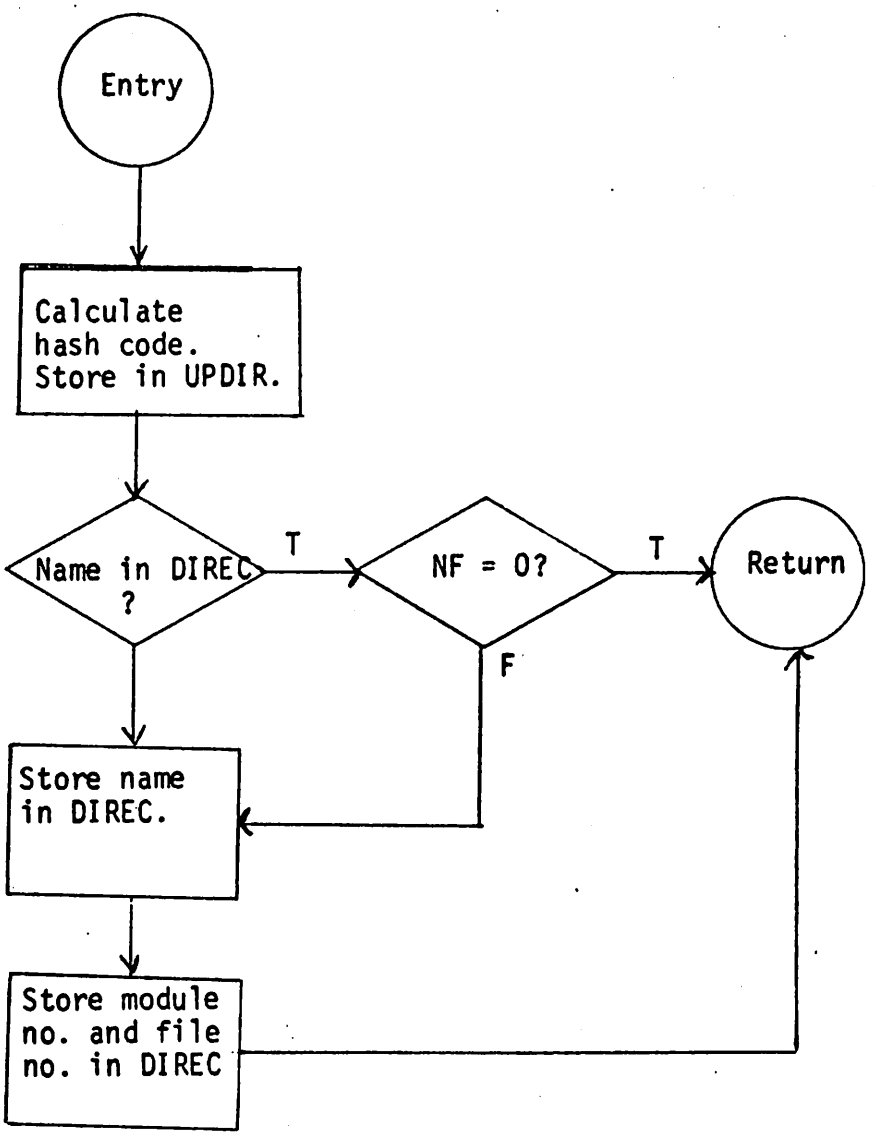
*The hash code formula is:

MOD ((NAM1 + NAM2)/2, JPRIME) + 1

where NAM1 and NAM2 has been converted into R format (right justified, zero filled).

JPRIME is the greatest prime number smaller than the length of the Directory table (JPRIME is initialized in the BLOCK DATA program).

Flowchart for UPDIR



Subroutine UPPAR (MCOD, MODNO, US, IAN)

This routine makes all entries into PARTAB (Parameter table). Depending upon the value of input parameter MCOD, UPPAR does one of the following things:

(a) MCOD = 0. This is an actual or dummy parameter (but not a constant), store the SYMTAB address of this parameter in PARTAB.

(b) MCOD = 1. Makes end-of-list entry in PARTAB.

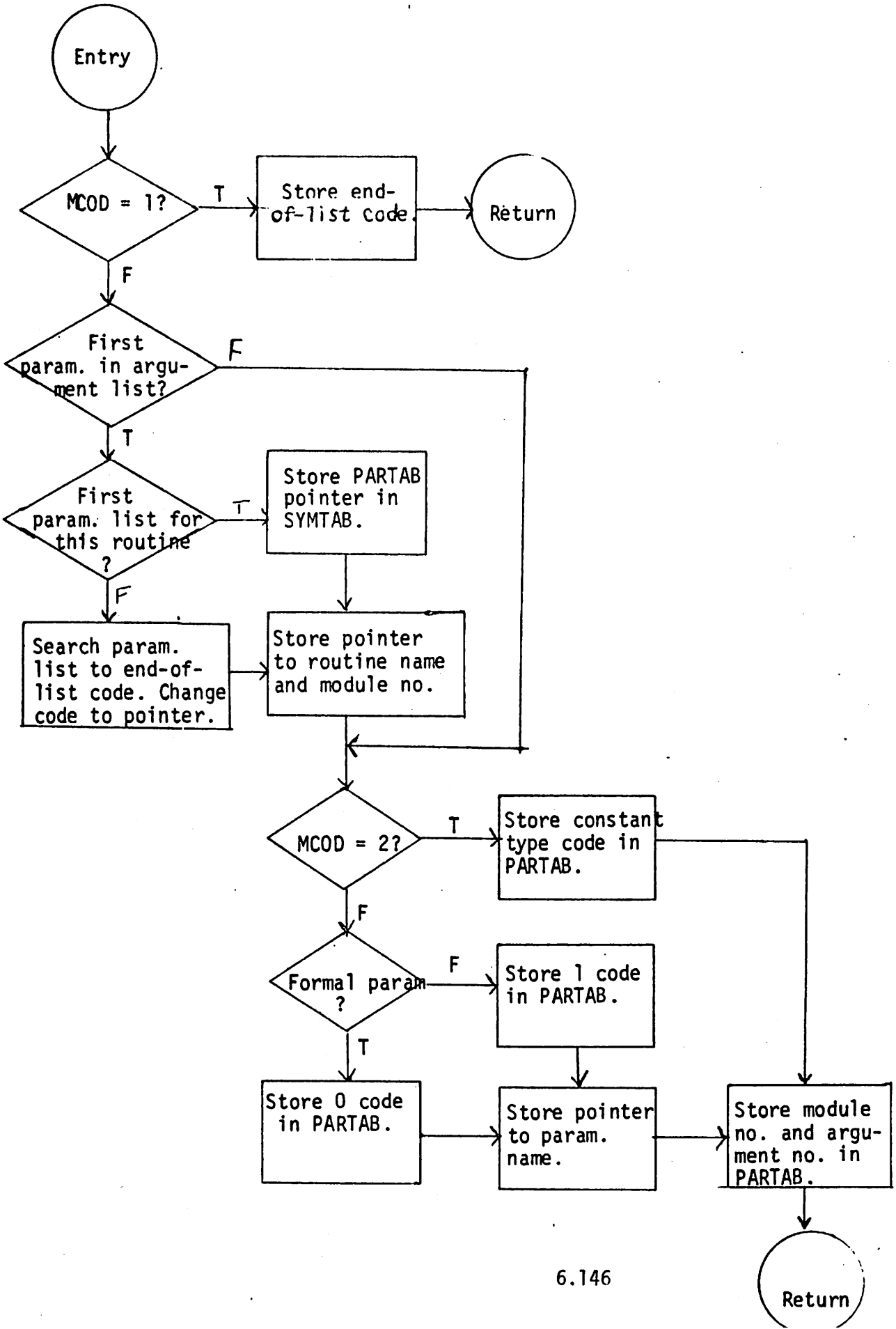
(c) MCOD = 2. Stores code for constant parameter and its type code in PARTAB.

When a parameter is to be stored (MCOD $\neq$ 1), a check is also made to determine whether this is the first parameter list for the corresponding subroutine or function. This involves checking the SYMTAB entry for the routine name to see if the PARTAB pointer is 0. If it is, the pointer to PARTAB is stored in SYMTAB. If this is not the first parameter list for this routine, the list of parameter pointers in PARTAB that correspond to the routine is traversed until the end-of-list code is found. The end-of-list code is changed to be a pointer to the next entry in PARTAB, and then the new list begins.

The SYMTAB address for the parameter being processed is stored in PARTAB. The parameter may be an actual parameter, a dummy parameter or a subroutine or function name to which the parameter list corresponds. From the use code, the CODE field of PARTAB is set accordingly.

CODE	
0	pointer to dummy parameter in SYMTAB
1	pointer to actual parameter in SYMTAB
2	pointer to parameter list for next call of same routine
3	pointer to subroutine or function name in SYMTAB
4	this word marks end of current parameter list
5	this is a constant parameter, pointer field contains constant type

Flowchart for UPPAB

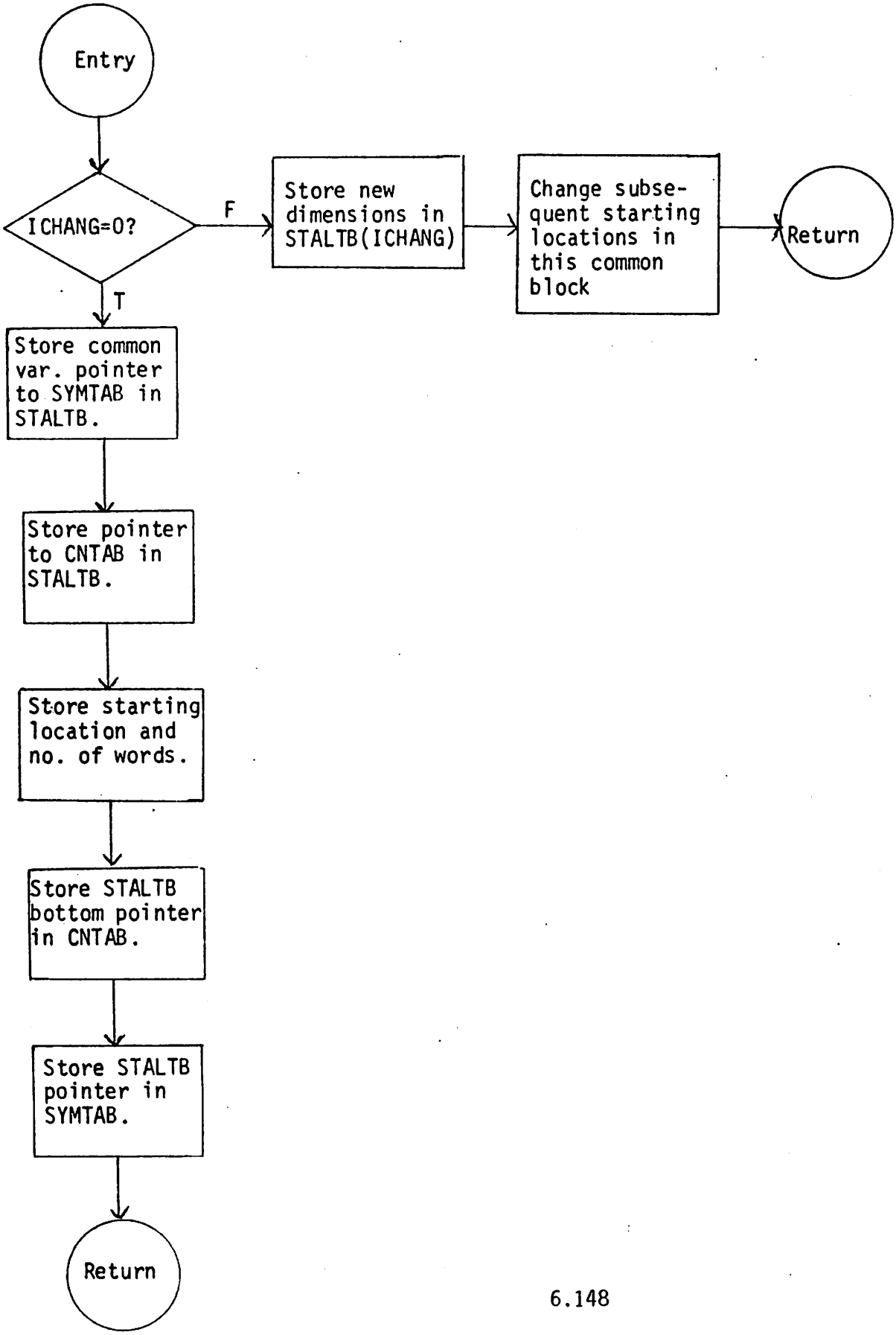


Subroutine UPSTAL (IDIM, ICHANG)

This routine updates STALTB (Storage Allocation Table) whenever a common variable is declared or the dimensions of a previously declared common variable are changed by a DIMENSION statement. When input parameter ICHANG is zero, an entry is made in STALTB which contains a pointer to the common variable name in SYMTAB and the size of the array variable. If ICHANG $\neq$ 0, then the dimension field of STALTB of the entry pointed to by ICHANG is changed and the subsequent starting locations for variables in that common block are updated. In either case, input parameter IDIM contains the size of the array variable.

Whenever a new entry is added to STALTB, the STALTB bottom pointer in CNTAB and the STALTB pointer of the variable in SYMTAB has to be updated.

Flowchart for UPSTAL

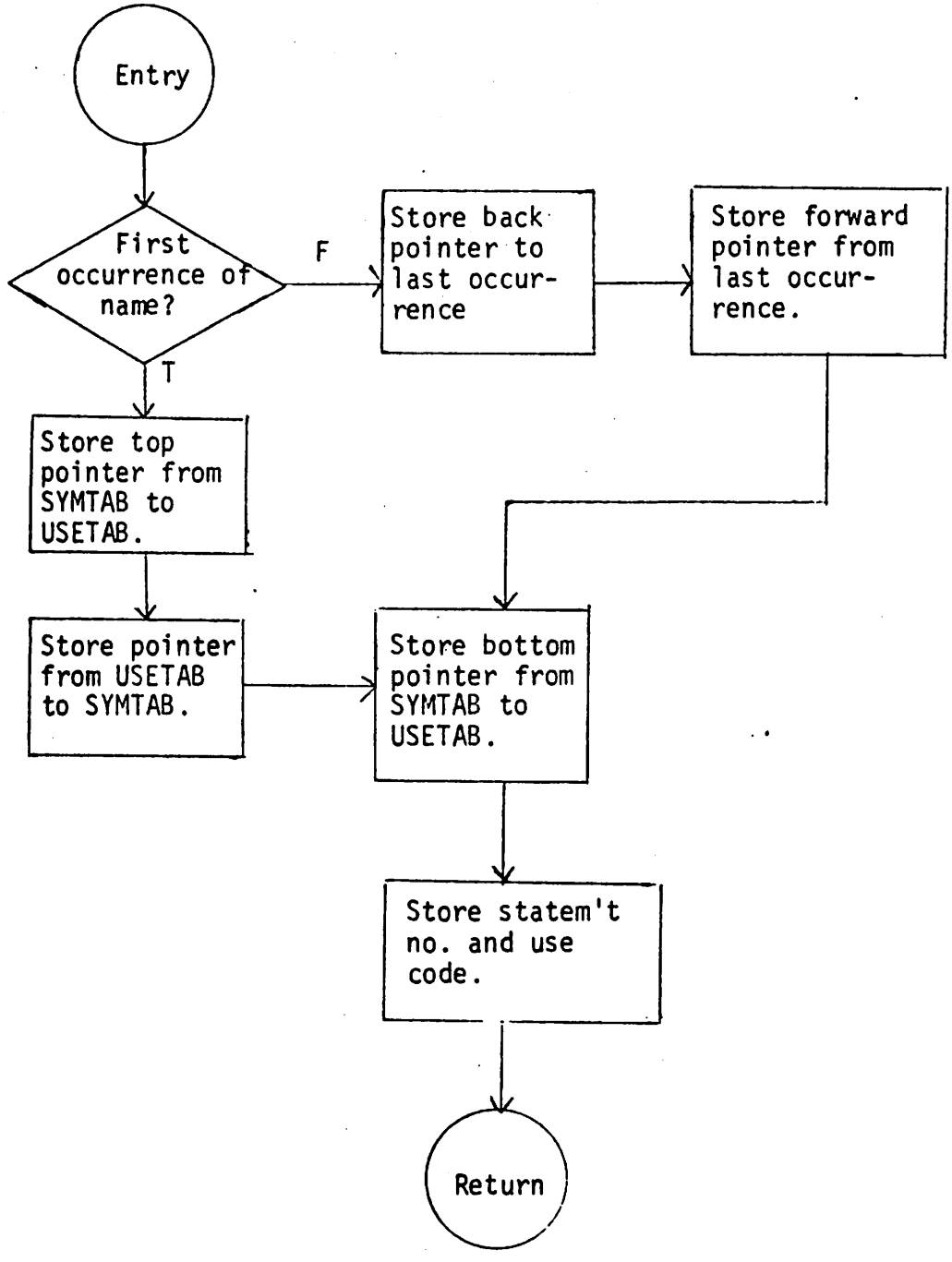


Subroutine UPUSE (IUSE, ISTNO)

This routine updates USETAB (Variable Usage Table) each time a name is found in the source program. Each entry in USETAB is linked to the previous entry and next entry for that name. The first entry for a name is linked to the name in SYMTAB, and the last entry has no forward pointer. Each entry in the table consists of the statement number in which the name occurs (input parameter ISTNO) and the use code for the name in that statement (input parameter IUSE).

UPUSE also update the USETAB bottom pointer in SYMTAB when a new entry is made in USETAB.

Flowchart for UPUSE



Subroutine ZTABS

This routine initializes all local tables to zero and their pointers to the appropriate starting value. ZTABS is called by the parser before analysis of each new module is begun.

VII. DOCUMENTATION FOR THE DIAGNOSTIC ROUTINES

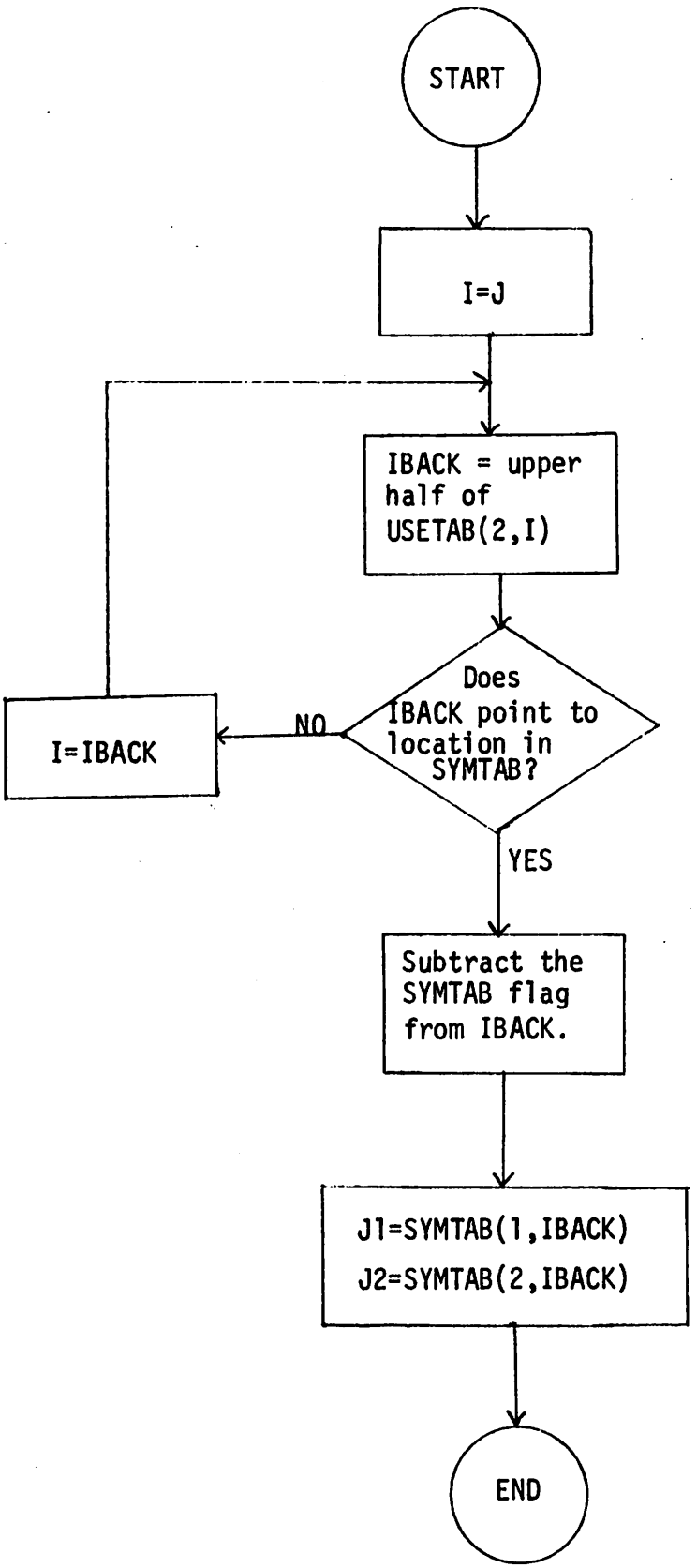
A set of diagnostic routines are included in the FACES package which analyze information stored in the tables produced by the Front End. Documentation for the four currently available diagnostic routines (PARAL, COMBAL, TRACY, and PATHS) follow. The four main routines and their subroutines are separate units and can be called independently by the user, depending on his needs. (The purposes of these main routines are explained in the FACES manual.)

SUBROUTINE BACKUP

Parameters: J,J1,J2

Purpose: This routine uses the Jth entry of the common array USETAB to determine from the common array SYMTAB the name of the variable referred to by that entry. This name is returned to the calling routine in the parameters J1 and J2 (four characters per word, left-justified and blank-filled).

FLOWCHART FOR BACKUP(J,J1,J2)



SUBROUTINE BWVT

Parameters: IV1, IV2, NODE

Purpose: This routine performs a backward variable trace for the value of the subject variable specified by IV1, IV2 at statement NODE. The goal of the trace is to discover all variables which may have had an effect on this value. An obvious use of this knowledge would be to aid in locating all possible sources of an erroneous value of the subject variable at a specific node. The output of BWVT consists of each such variable source, coupled with the node at which the value may be affected. Used in the derivation are the following tables produced by the FACES system:

SYMTAB - local symbol table

USETAB - table of the usages of each symbol in SYMTAB

NODTAB - local node table

PRETAB - table of the predecessor nodes for each node in NODTAB.

Algorithm: It is determined from NODTAB if the statement at NODE can result in an assignment of a new value. This is considered to include only assignment statements and DO statements. If the statement is not one of these types, then all predecessor statements are found in PRETAB and their node numbers placed on the stack NODSTK, unless flagged in UNSTCK. Each is then flagged in UNSTCK by call to ENTER. The last entry to NODSTK is then removed, assigned to NODE and processing begins anew with this value.

When a statement of one of the above types is reached, the name of the variable that receives the new value is determined by call to BACKUP. If this variable is not the subject variable, processing resumes as if the statement had been of the wrong type. If the subject variable is revalued at this node, the identity of each variable involved is derived by separate calls to BACKUP. Each is then placed by references to routine INCLUD on the arrays VARIB and OUT, along with the node number. If the subject variable has no effect on its own new value, the trace for the variable along this path need not continue. In other words no predecessor statements need be entered on NODSTK. Otherwise processing again resumes exactly as if the statement had been of the wrong type.

Whenever NODSTK is empty, the last variable entry to VARIB becomes the new "subject variable" and its associated node is assigned to NODE. The variable trace continues with these values. When all entries to VARIB have been processed, the complete variable trace is concluded. SRTOUT is called to sort the contents of OUT alphabetically by name and then by node number. Using the method outlined above it is probable that a variable will appear in OUT more than once (but with different node values) and each entry will be printed. Also the original subject variable will in rare instances be included in the output, indicating somewhat complex revaluation circumstances.

Possible sources of irregularities:

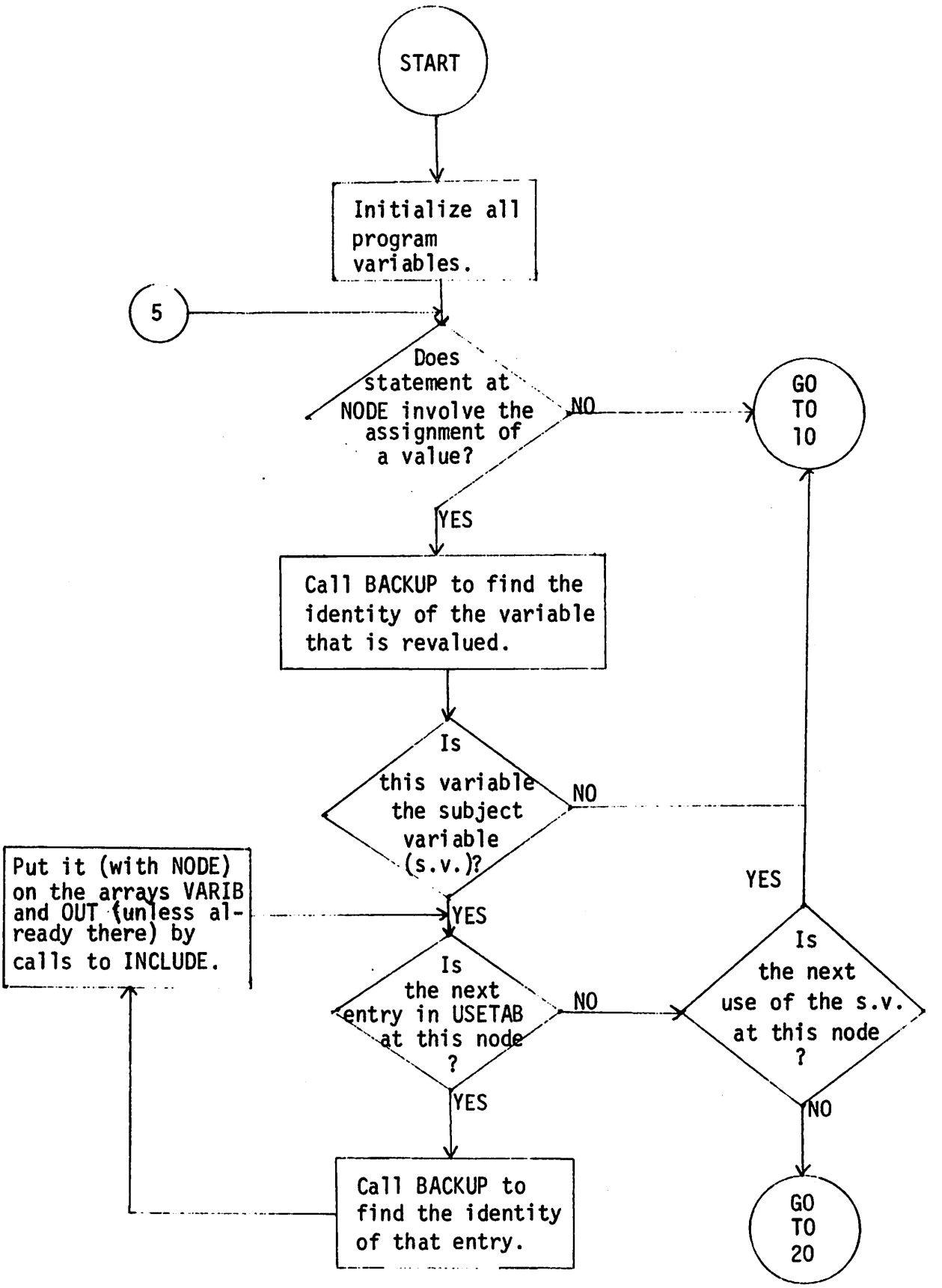
(1) Arrays are handled as if they were simple variables, i.e., during a trace there is no differentiation between different words of an array.

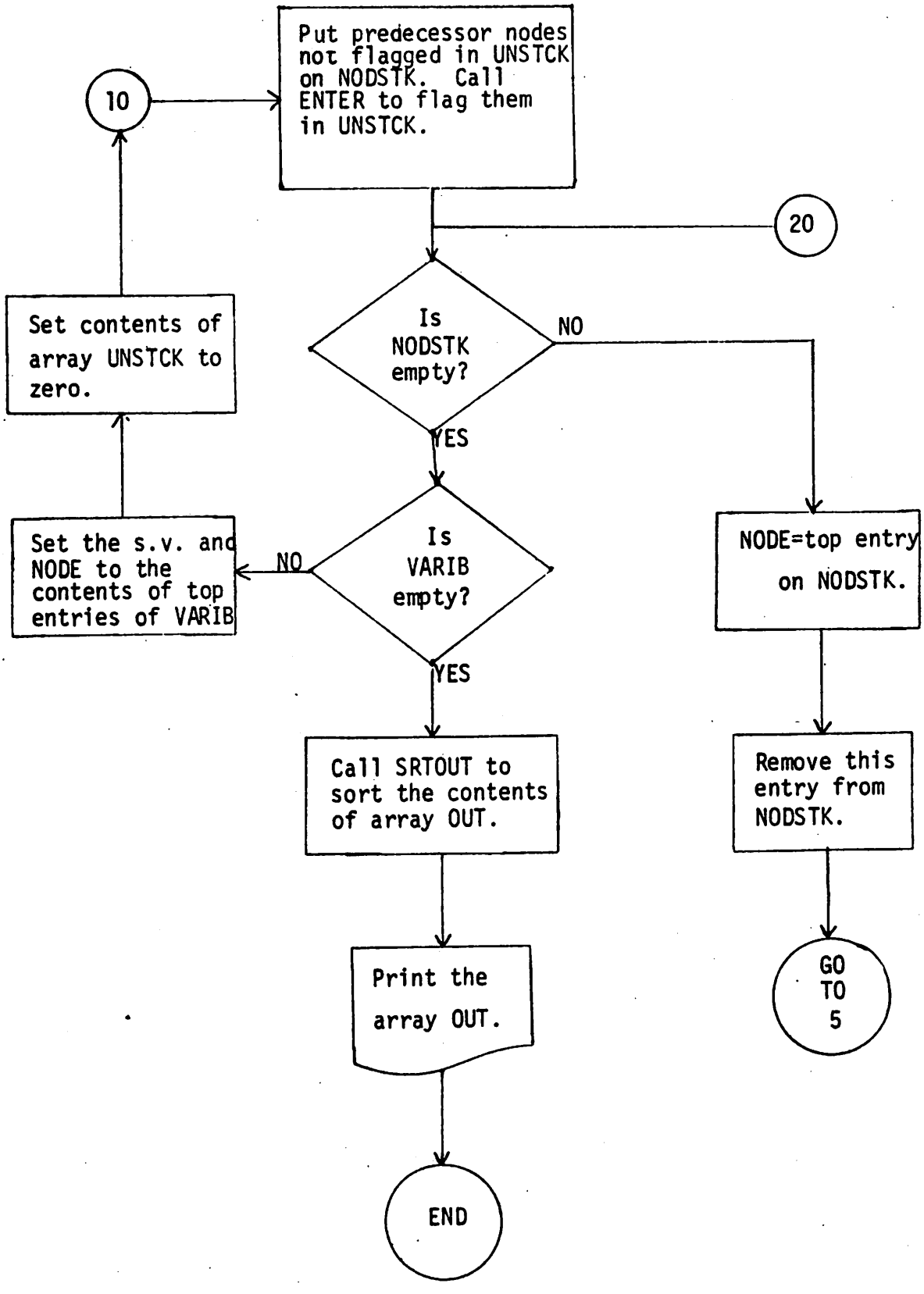
(2) The index of an array is processed as among the variables affecting the value of the array.

(3) I/O statements are ignored.

(4) The effects of routine references are unknown at the modular level so such statements are generally ignored. Function names are treated as variable names.

FLOWCHART FOR BWVT(IV1,IV2,NODE)



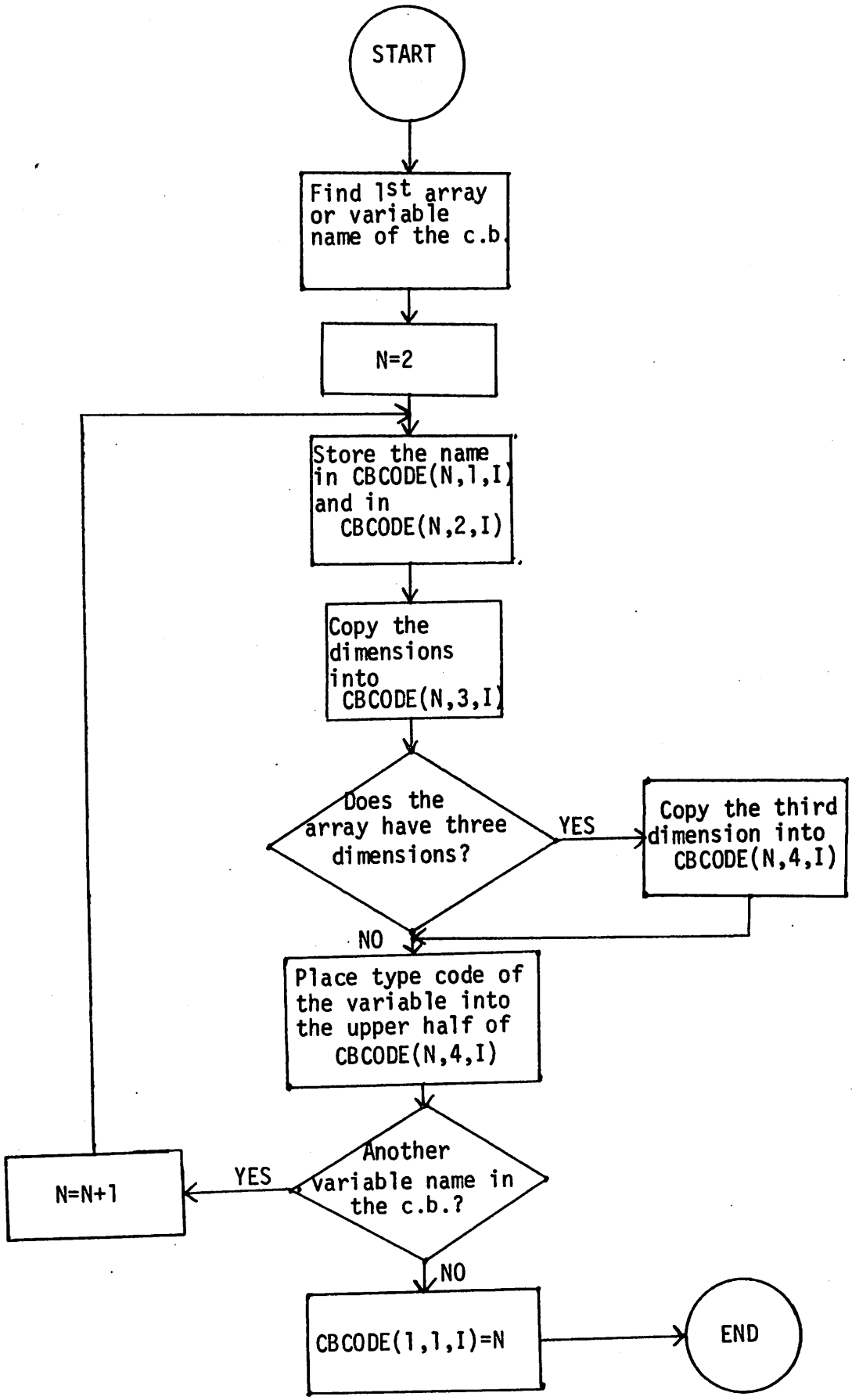

10
Put predecessor nodes not flagged in UNSTCK on NODSTK. Call ENTER to flag them in UNSTCK.
20
Set contents of array UNSTCK to zero.
Is NODSTK empty?
NO
YES
Set the s.v. and NODE to the contents of top entries of VARIB
NO
Is VARIB empty?
NODE=top entry on NODSTK.
YES
Call SRTOUT to sort the contents of array OUT.
Remove this entry from NODSTK.
Print the array OUT.
GO TO 5
END

SUBROUTINE CODER

Parameters: LL, JTOP, JBTM

This routine develops the variables of a common block, as specified in the common arrays SYMTAB, ALTAB, and STALTB, into condensed or coded representation for display in CBCODE. This code, to be stored in the LLth segment of CBCODE(i.e., CBCODE(i,j,LL)), consists of the names of the variables, their dimensions, and their types. JTOP indicates the first entry in STALTB of the desired common block and JBTM the last entry. For the Nth entry of the common block, CBCODE(N,1,LL) and CBCODE(N,2,LL) contain a copy of the symbol's name as found in SYMTAB. CBCODE(N,3,LL) contains the first dimension of the variable in its lower half and the second dimension (if any) in its upper half. CBCODE(N,4,LL) contains the third dimension (if required) in its lower half and the type code of the variable in its upper half. The total number of variables in the common block is then stored in CBCODE(1,1,LL).

FLOWCHART FOR CODER(I)



SUBROUTINE COMBAL(NMCK)

Parameter: NMCK

Purpose: COMBAL is designed to determine if the common blocks of the routines analyzed by the FORTRAN Front End are properly aligned. Each of the common blocks bearing the same name must display the following characteristics:

(1) the same number of variable and/or array entries.

(2) the ith entries of each common block must have the identical dimensions.

(3) the ith entries of each must be of the same type.

If the parameter NMCK = 1, then the blocks must also:

(4) have corresponding entries with identical names.

COMBAL produces output for each common block utilized in the routines that have been analyzed. The output consists of the common block name, the routines it appears in, and diagnostics (if any) for each of these. If the characteristics (1) and/or (2) are not fulfilled, the diagnostic is 'SIZE'. For characteristic (3) the associated diagnostic is 'TYPE' and for (4) it is 'NAME'. Unless otherwise indicated in the output message, the first routine listed without a diagnostic was considered the "standard" common block for the alignment check. The following tables produced by the FORTRAN Front End are read from file and used in the analysis:

CNTAB - global common name table

SYMTAB - local symbol table

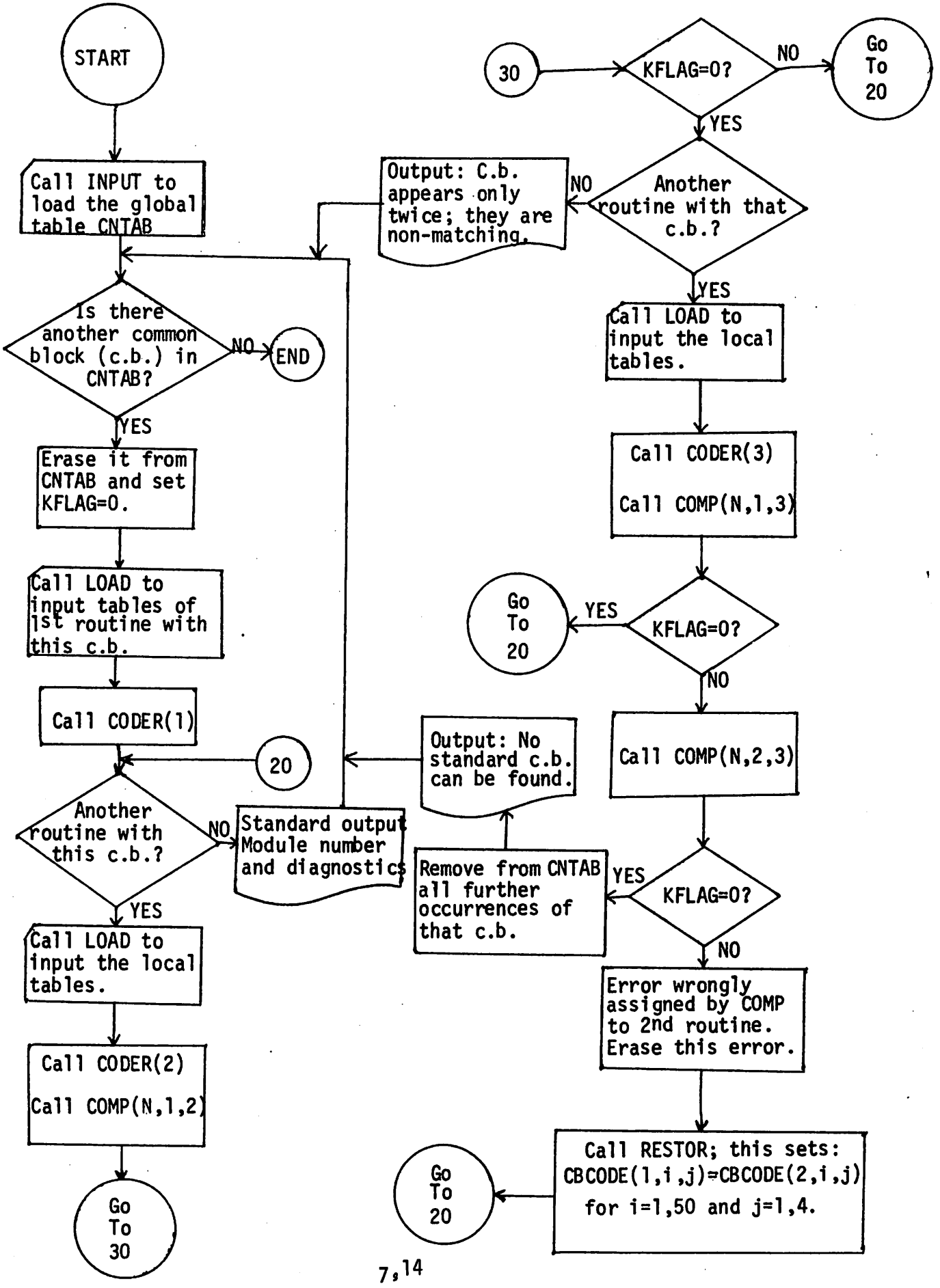
ALTAB - local array length table

STALTB - local storage allocation table

Algorithm: Through CNTAB the name and location of each common block may be obtained. Basically, the concept employed is to locate each routine that contains a particular common block, compare the block's composition in the different routines, and print out indications of the irregularities found, if any. To determine a block's composition in a particular routine, it is necessary to load several of that routine's local tables (as mentioned above). Analysis of these tables by CODER results in the production in CBCODE of the block's composition in condensed, or coded, form. (The code consists of name, dimensions, and type for each variable in the common block.) Then the same process is performed for the next routine in which the common block appears. The two coded forms are compared (by call to COMP). If there is any variation it is determined if there is another routine with that common block. If there is not another use of the block, then the output process is initiated with the irregularity associated with both routines. If there is another appearance of the block, then its coded form is derived and it is compared with first one and then the other of the earlier codes. If it matches neither of the other codes, the module numbers of all routines with that common block are printed with the indication that there is misalignment but with no specification as to which are the offenders. If the code is identical to one of the other codes, the routine associated with the non-matching

code is assigned the diagnostic. The first of the matching codes is then used as the standard for further alignment tests. When each use of the common block has been checked against the standard, output for that block is made and processing begins for the next block.

FLOWCHART FOR COMBAL(N)

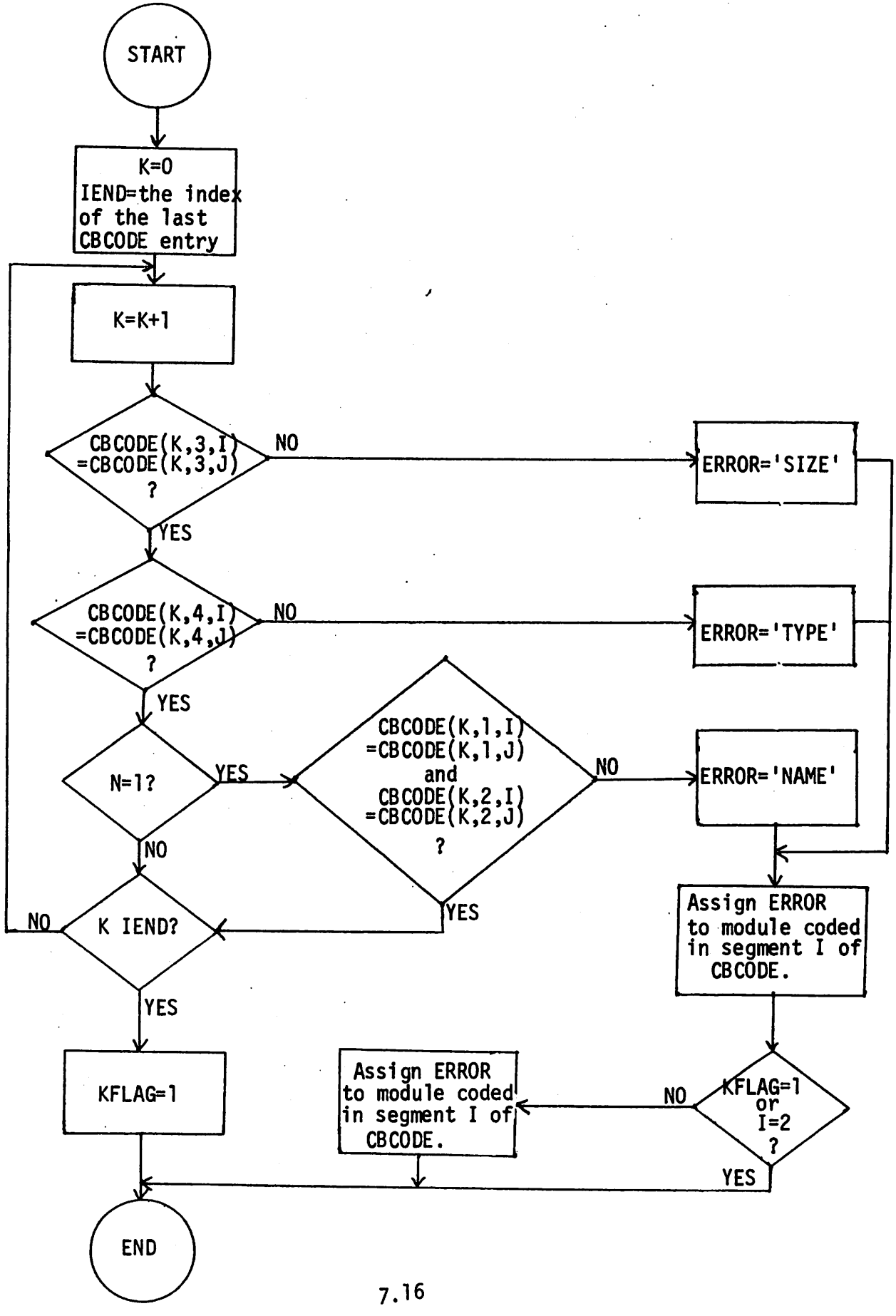


SUBROUTINE COMP

Parameters: NMCK,J1,J2

This routine compares the contents of two segments (identified by J1 and J2) of the array CBCODE. Each segment represents one module's version of a common block. The parameter NMCK, if not equal to one, specifies that certain words of the two segments are not compared (namely those which contain the names of the variables in the block). Only if NMCK = 1 are the words CBCODE(N,1,J1) and CBCODE(N,2,J1) compared to CBCODE(N,1,J2) and CBCODE(N,2,J2), respectively. Next, CBCODE(N,3,J1) is matched against CBCODE(N,3,J2) and CBCODE(N,4,J1) against CBCODE(N,4,J2). These compares are repeated for each relevant value of N. If at any point a variation is noted, the proper diagnostic is inserted into the words in the array ERROR associated with segments J1 and J2 and the routine ends. If no irregularity is found, KFLAG is set to one as an indication that two common blocks have been processed that are properly aligned.

FLOWCHART FOR COMP(N,I,J)



SUBROUTINE ENTER

Parameters: NODE, IRAY

ENTER sets a bit in the array IRAY to 1. The parameter NODE is an index specifying the bit which is to be so handled. The index refers to the consecutive bit locations (with 36 bits included per word) in the array IRAY. For example, if NODE = 1, the left-most bit of IRAY(1) is set to one and if NODE = 37, the left-most bit of IRAY(2) is set to one.

SUBROUTINE FWVT

Parameters: IV1, IV2, NODE

Purpose: This routine performs a forward variable trace for the value of the subject variable specified by IV1 and IV2 upon entering the statement numbered NODE. The goal of the trace is to locate each variable whose value is affected by the value of the subject variable in the stated point in the program. This should allow the programmer to better realize the ramifications of assigning the variable a new value at NODE. The output consists of each affected variable along with the node at which the revaluation occurs. The following tables produced by the FACES system are used in the analysis:

SYMTAB - local symbol table

USETAB - table of the usages of each symbol in SYMTAB

NODTAB - local node table

SUCTAB - table of the successor nodes for each node in NODTAB

Algorithm: It is determined from NODTAB if the statement at NODE can result in an assignment of a new value. This is considered to include only assignment statements and DO statements. If the statement is not one of these types, then all successor statements are found in SUCTAB and their node numbers placed on the stack NODSTK, unless the node is flagged in the array UNSTK. Each is then flagged in UNSTCK by call to ENTER. The

last entry to NODSTK is then removed, assigned to NODE, and processing begins anew with this value.

When a statement of one of the above types is reached, the name of the variable which receives the new value is determined by referencing the routine BACKUP. If this variable is the subject variable, and it is not involved in its own revaluation, no further trace down this path is necessary and the next entry on NODSTK is assigned to NODE and used to continue the trace. Otherwise, the variable name is retained for future reference and the identity of each of the other variables used at this node is discovered by similar calls to BACKUP. If none of these are the subject variable, then processing continues as if the statement had been the wrong type. If indeed the subject variable does appear at this node, the earlier noted "input" variable (and NODE) is entered into both of the arrays VARIB and OUT by separate calls to INCLUD. Processing again continues as if the statement had been of the wrong type.

Whenever NODSTK is empty, the last variable entry in VARIB becomes the "subject variable" and its associated node value is assigned to NODE. The variable trace continues with these values. When all entries to VARIB have been processed, the complete variable trace is concluded. SRTOUT is called to sort the contents of OUT alphabetically by name and then by node number. Using the method outlined above it is probable

that a variable will appear in OUT more than once (but with different node values) and each will be printed. Also the original subject variable will in rare instances be included in the output list, indicating complex revaluation circumstances.

Possible sources of irregularities:

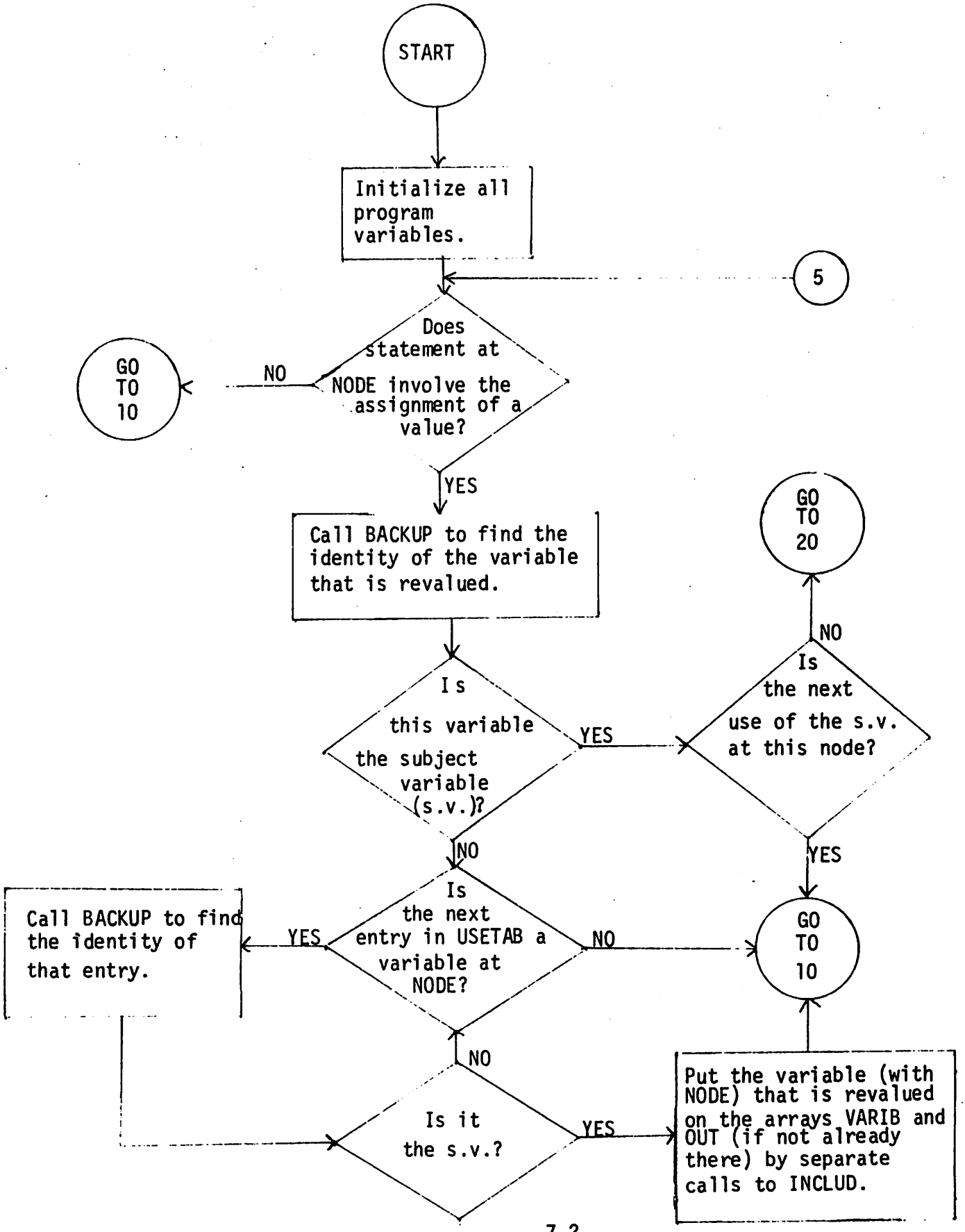
(1) Arrays are handled as if they were simple variables, i.e., during a trace there is no differentiation between different words of an array.

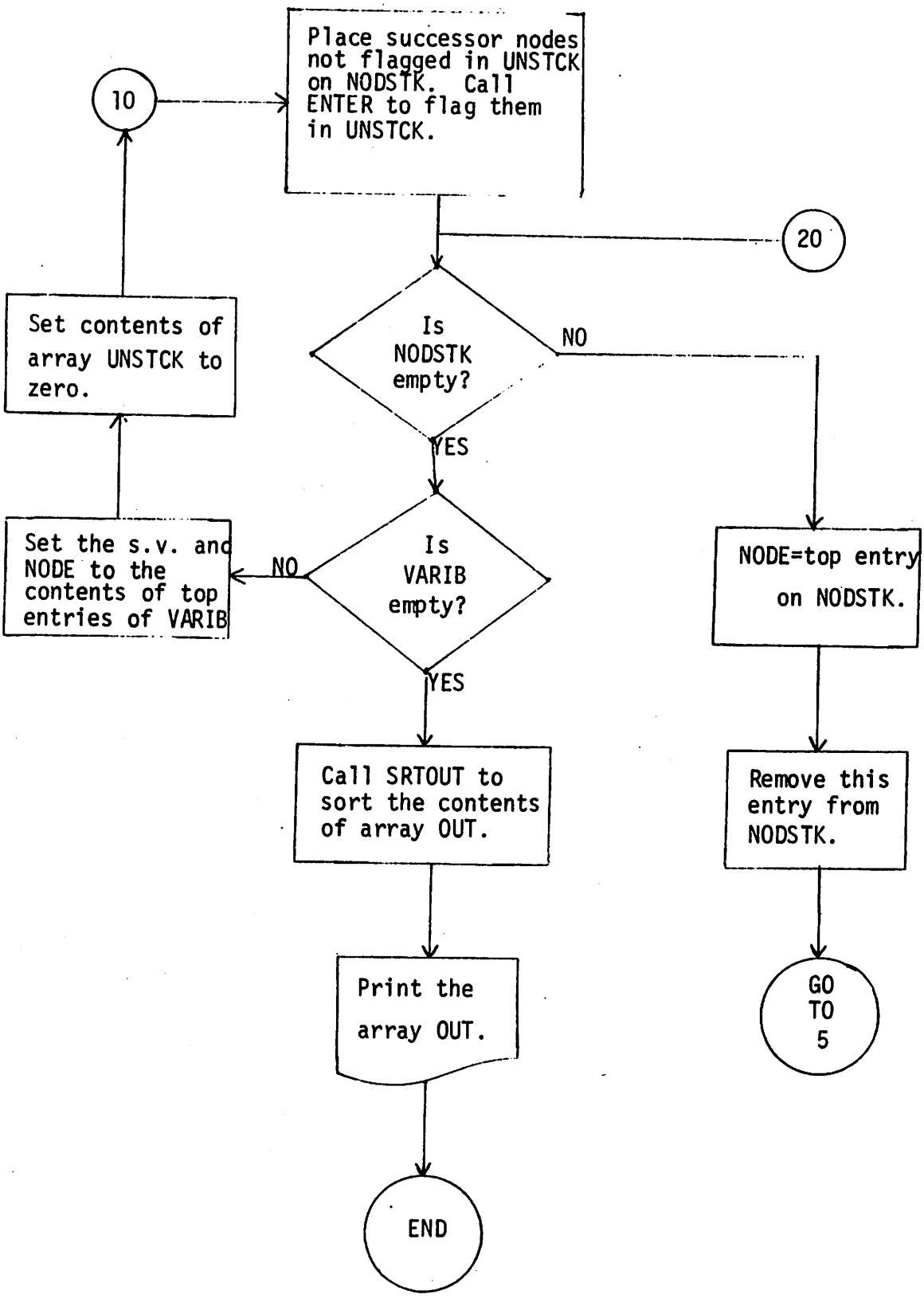
(2) The index of an array is processed as among the variables affecting the value of the array.

(3) I/O statements are ignored.

(4) The effects of routine references are unknown at the modular level so such statements are generally ignored. Function names are treated as variable names.

FLOWCHART FOR FWVT(IV1,IV2,NODE)




10
Place successor nodes not flagged in UNSTCK on NODSTK. Call ENTER to flag them in UNSTCK.
20
Is NODSTK empty?
NO
YES
Set contents of array UNSTCK to zero.
Is VARIB empty?
NO
YES
Set the s.v. and NODE to the contents of top entries of VARIB
NODE=top entry on NODSTK.
Call SRTOUT to sort the contents of array OUT.
Remove this entry from NODSTK.
Print the array OUT.
GO TO 5
END

Subroutine HASHMN

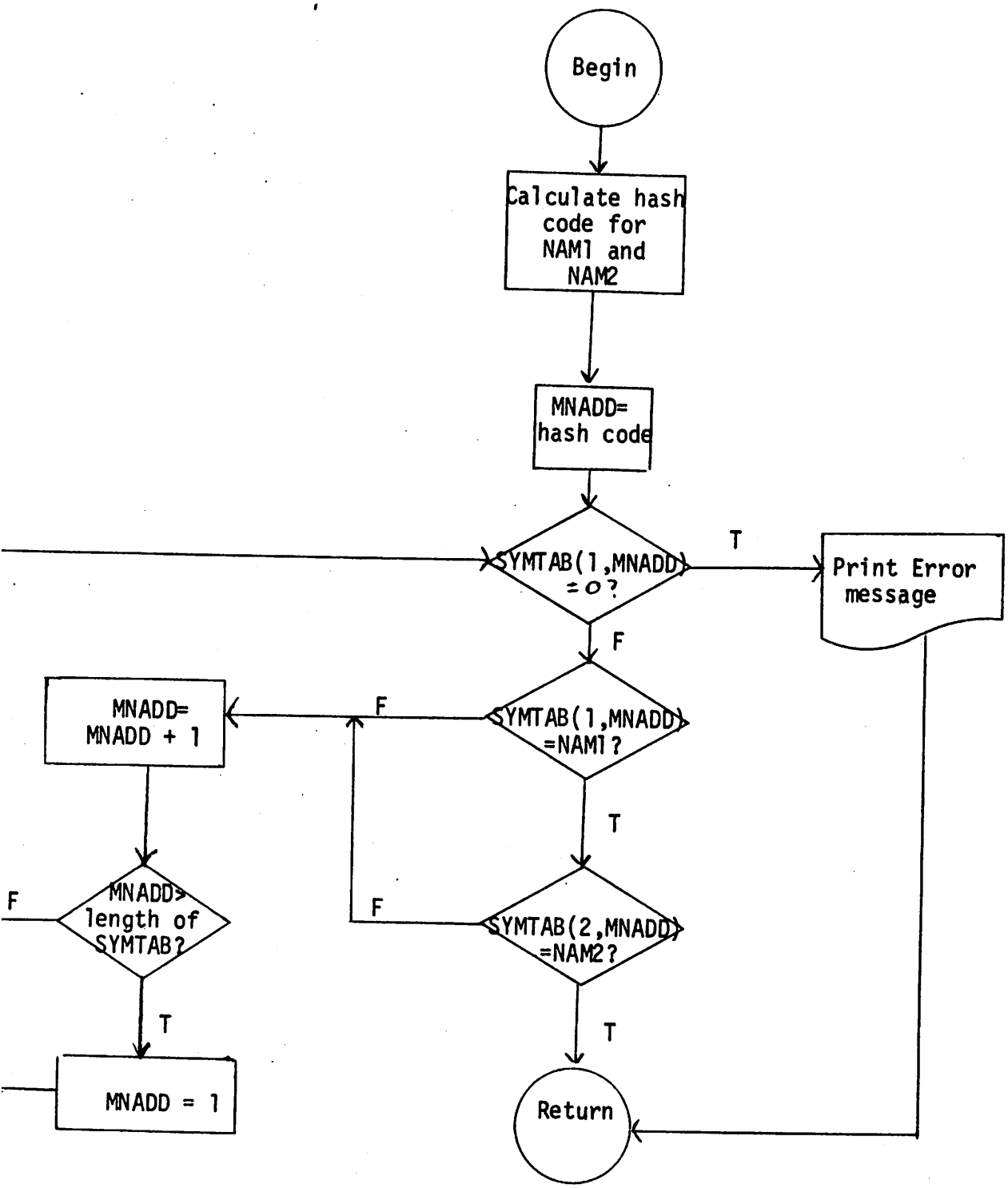
Parameters: NAM1, NAM2, MNADD

This routine calculates the hash code for the module named by parameters NAM1 and NAM2 and returnsthe address of the module name in SYMTAB through the parameter MNADD. NAM1 contains the first four characters of the module name, and NAM2 contains the last three. Suppose CHAR(NAM1) is the character code of NAM1. Thentthe hashing formula is as follows:

$$\text{Hash code} = \text{MOD}\left(\frac{\text{CHAR(NAM1)} + \text{CHAR(NAM2)}}{2}, \text{IPRIME}\right) + 1$$

where IPRIME is the largest prime number less than the table size. If the module name is not found at the hash code address of SYMTAB, then SYMTAB is searched linearly from that point until the name is found.

Flowchart for Subroutine HASHMN (NAM1, NAM2, MNADD)

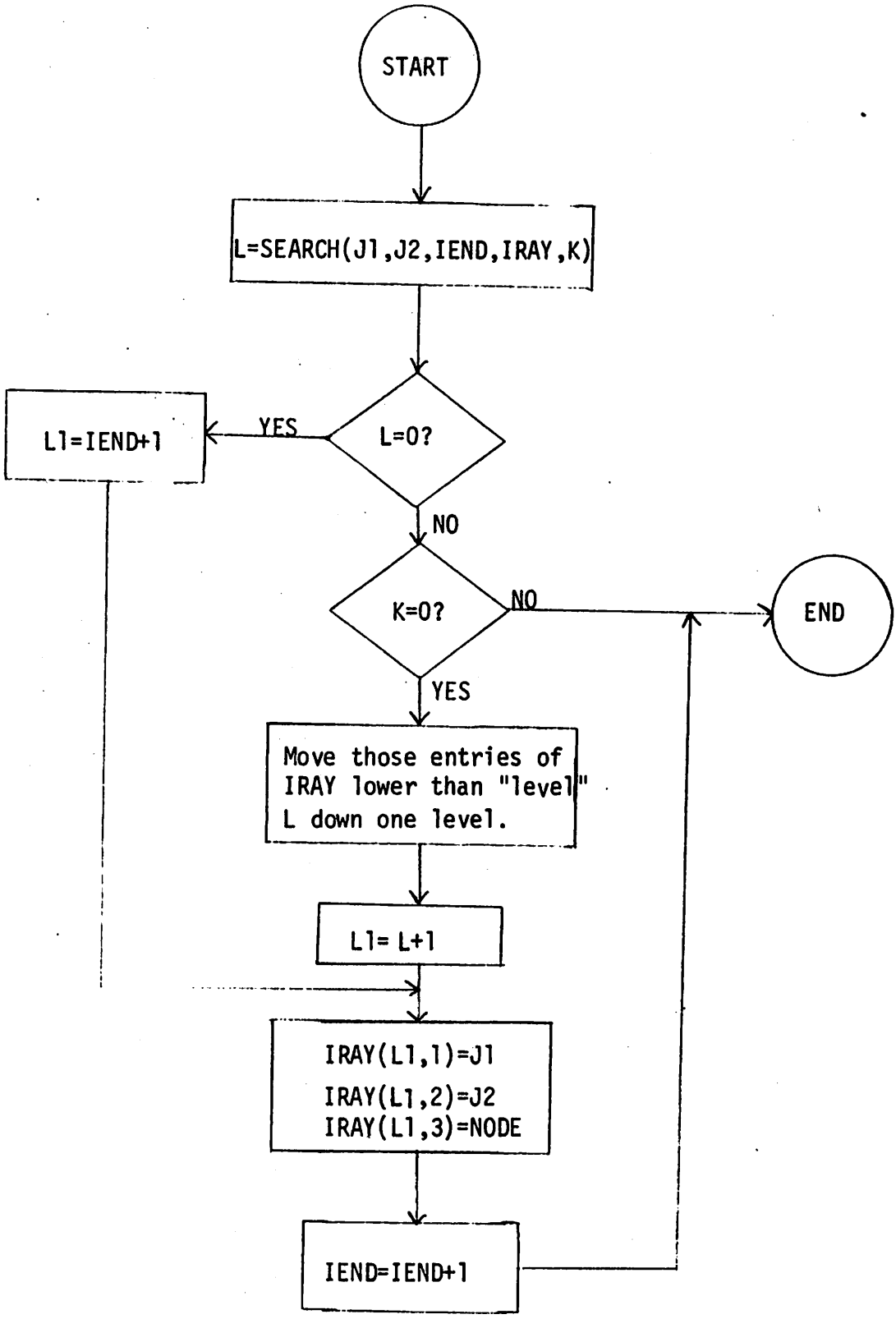


SUBROUTINE INCLUD

Parameters: IRAY,IEND,J1,J2,NODE,K

Purpose: INCLUD is used to enter a variable, named in J1 and J2 (four characters per word) and a related node number NODE into the array IRAY. The entry is performed only if the variable and node are not already entered into the array. The parameter IEND refers to the current length of IRAY. K is used primarily for reference to the integer function SEARCH. If K = 0, SEARCH determines the node in IRAY at which the variable appears. INCLUD then makes the entry at the location following that node. For K = NODE, SEARCH determines the node at which the variable appears with the associated value NODE. If such a node is found, no entry is made by INCLUD. In all other cases, the entry is made at node IEND+1.

FLOWCHART FOR INCLUD(IRAY,IEND,J1,J2,NODE,K)



SUBROUTINE INPUT

INPUT loads from file the global table CNTAB, as produced by FACES.

SUBROUTINE INPUT2

Parameters: MDNAM1, MDNAM2

This routine loads from file several of the tables produced by the FACES system for the routine specified by the parameters. These parameters contain the routine's name, four characters per word, left justified and blank filled. The first table loaded is the global table DIREC, in which may be found the name of the routine. Associated with the name is the assigned module number. Using this number, the required local tables for the routine may be located and loaded. These tables are SYMTAB, NODTAB, PRETAB, USETAB, and SUCTAB.

SUBROUTINE LOAD

Parameter: M

This routine transfers from file to main storage the local tables SYMTAB, ALTAB, and STALTB for the module identifiable by the parameter M. M refers to a storage location in the global table CNTAB (which is held in common). The module number is found in CNTAB(4,M). Through the module number it is possible to determine the local tables' location on file.

INTEGER FUNCTION NCHECK

Parameters: NODE, IRAY

This function returns as its value the contents of the bit specified by the index NODE in the array IRAY. NODE refers to the consecutive bit locations (with 36 bits included per word) in IRAY.

Subroutine PARAL

Parameter: IATOPT

This subroutine checks the parameter alignment of calls made to all subroutines and functions that have been analyzed by the FORTRAN Front End. The following tables formed by the Fortran Front End are read from a file and used in the analysis:

DIREC -- global directory of module names

PARTAB -- global table of parameters

SYMTAB -- local symbol tables

ALTAB -- local array length tables

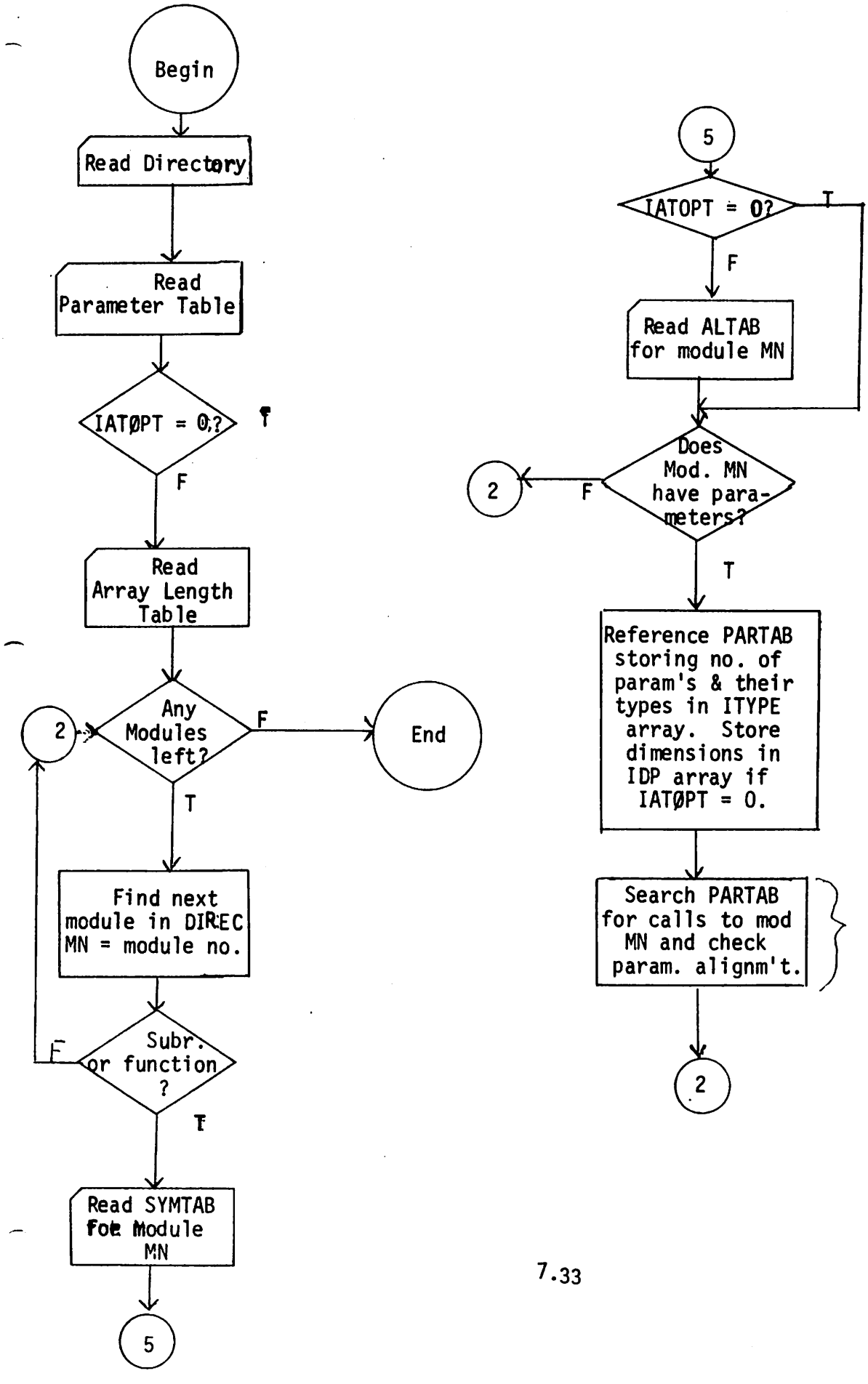
The parameter to this routine, IATØPT, is used as an option for checking array dimensions in parameter lists. If the user wishes to omit the checking of array dimensions, IATØPT should be passed as zero. Otherwise, array dimensions will be checked.

First the directory is scanned to obtain the name and module number of a subroutine or function that has been analyzed (for which tables have been made). Then the symbol table for the routine is read to determine whether the routine has parameters. If it does, the parameters, their types, and dimensions are obtained from the parameter table. The number of formal parameters is stored in variable NP. The formal parameter type codes (as stored in SYMTABØ) are stored in array ITYPE in order of appearance in the parameter list. If a formal parameter is an array (and if IATØPT $\neq$ 0), a pointer to the IDP array will be stored in the high order half of the word in ITYPE. The dimensions ofthe parameter are then stored in the IDP array in the same form in which dimensions are stored in ALTAB.

After all the information described above is stored for a formal parameter list, the parameter table is searched linearly for calls to this routine. The actual parameter lists are then checked against the information stored for the formal parameter list, and warning messages are printed when length of list, types of parameters, or dimensions of parameters do not agree.

This algorithm is repeatdd for all subroutines and functions that have been analyzdd by the Fortran Front End.

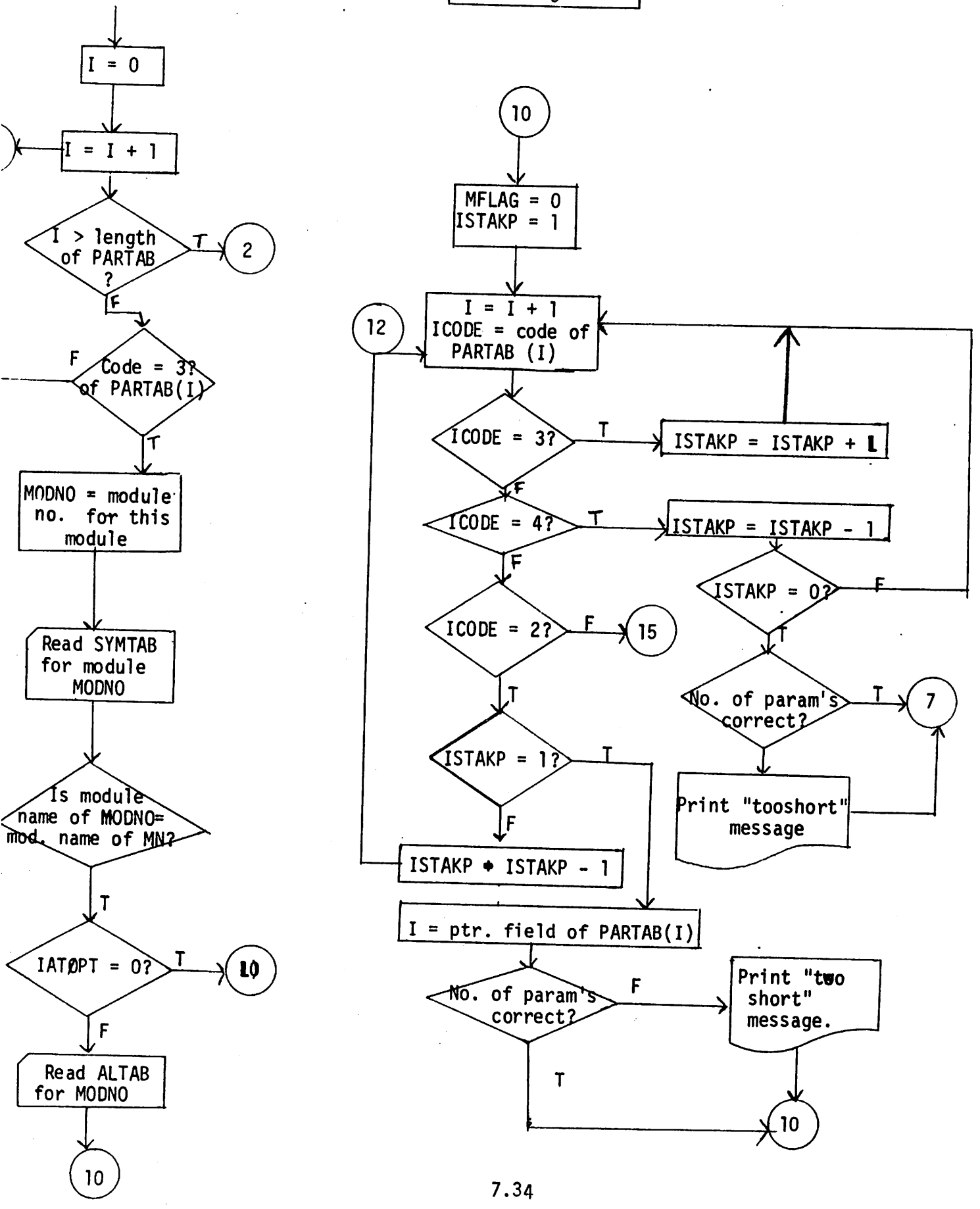
Flowchart for Subroutine PARAL (IATØPT)

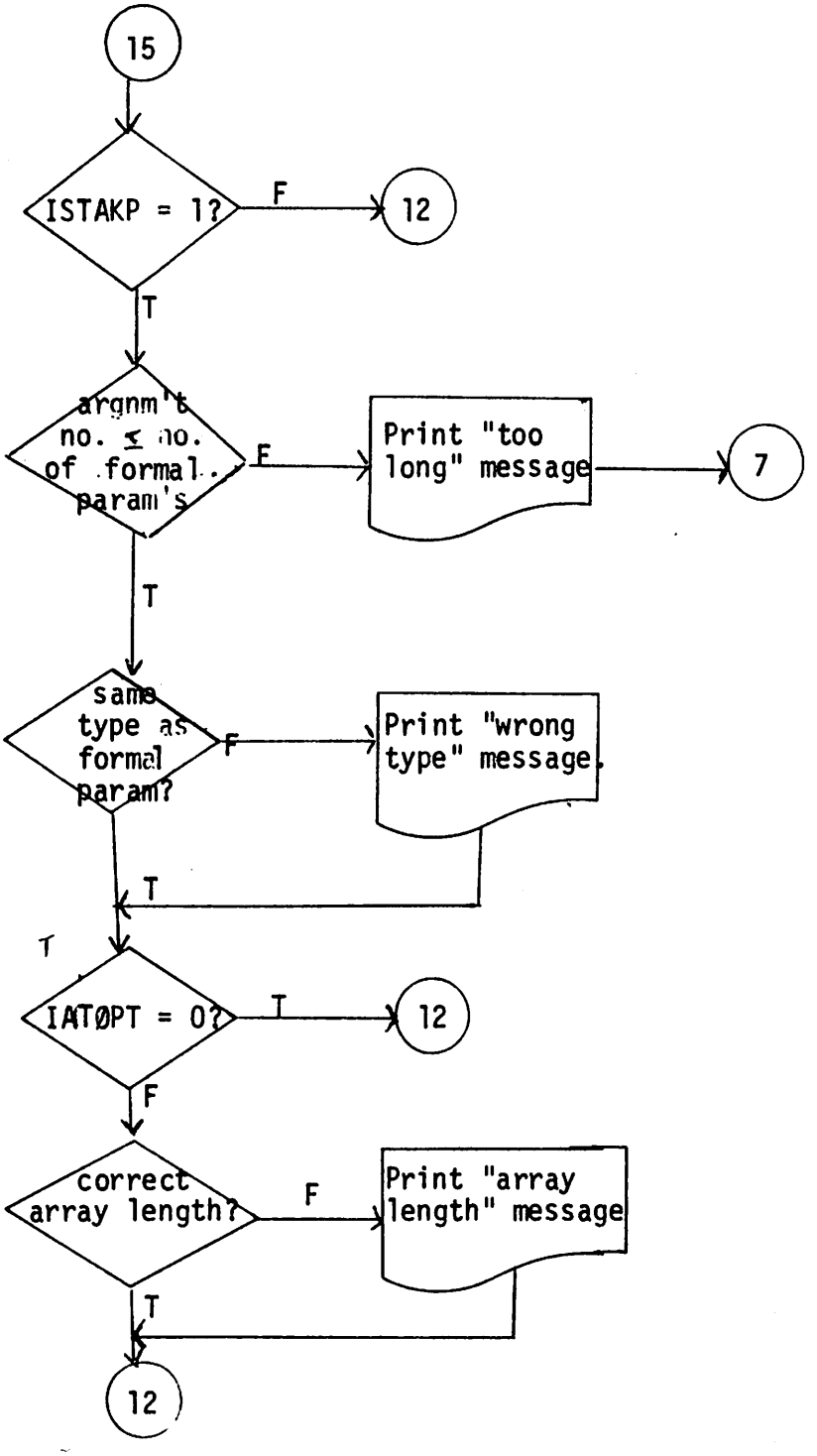


This block expanded on next page.

Flowchart for expansion of:

Search PARTAB for calls to mod. MN and check param. algnm't



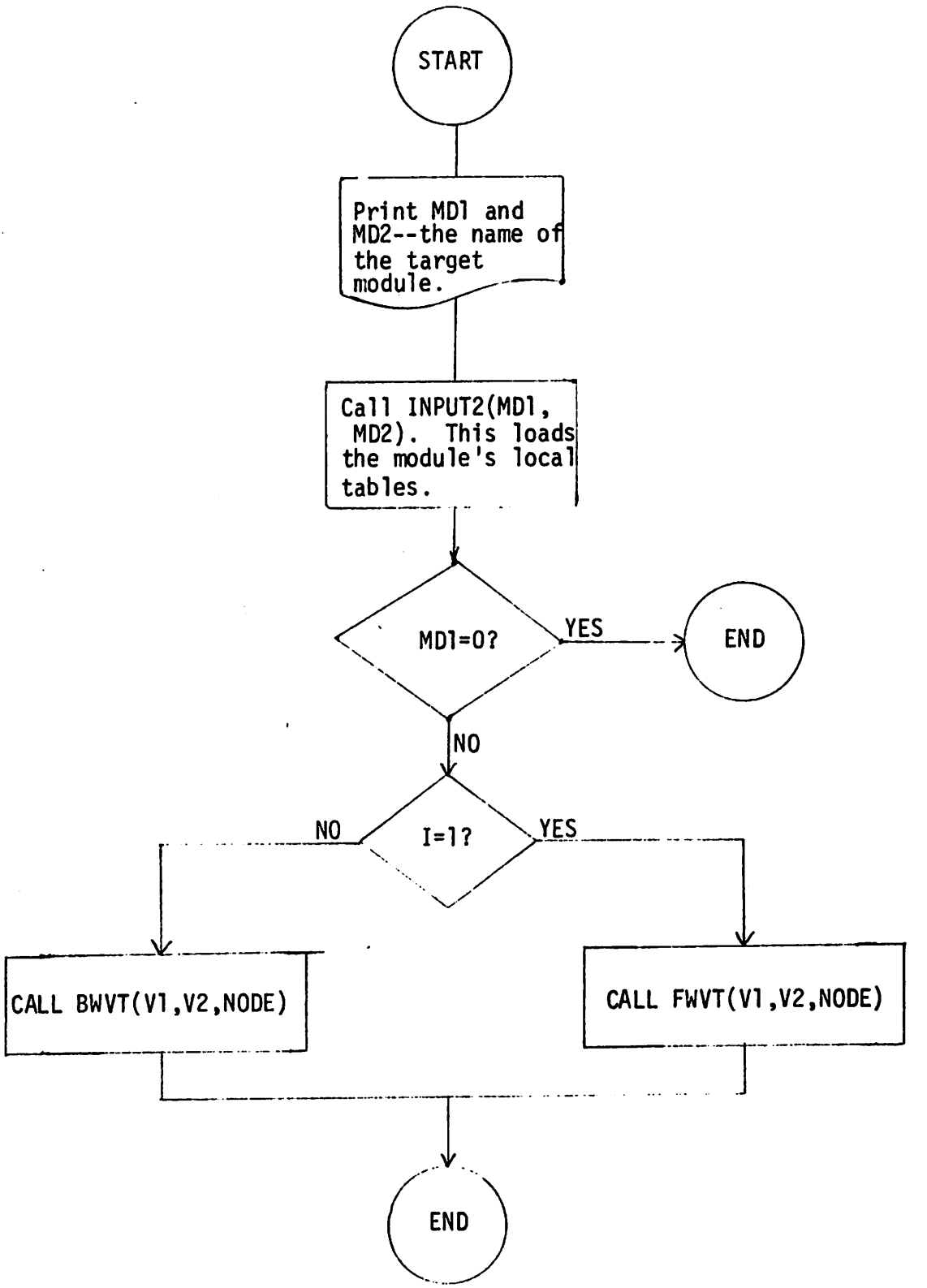

15
ISTAKP = 1?
F
12
T
argnm't no. ≤ no. of formal param's
F
Print "too long" message
7
T
same type as formal param?
F
Print "wrong type" message
T
T
IATØPT = 0?
T
12
F
correct array length?
F
Print "array length" message
T
12

SUBROUTINE PATHS

Parameters: I,MD1,MD2,V1,V2,NODE

Purpose: PATHS is composed of two different but intimately related variable tracing capabilities. If I ≠ 1, the trace, performed by the routine BWVT, is a "backward trace", discovering all those variables that might have affected the value of the subject variable at the specified statement. If I = 1, FWVT is called to perform a "forward variable trace", deriving all variables which might be affected by the value of the subject variable at the specified statement. A further description of the trace methods and results appears in the documentation for BWVT and FWVT. The subject variable is named in the parameters V1, V2 (stored 4 characters per word, left-justified and blank filled), and the statement number in NODE. The trace is performed within the confines of the module identified by MD1,MD2 (stored 4 characters per word, left-justified and blank-filled).

FLOWCHART FOR PATHS(I,MD1,MD2,V1,V2,NODE)



SUBROUTINE RESTOR

This routine transfers the contents of one segment of the common array CBCODE to another of its segments. The statement

```
CBCODE(I,J,1) = CBCODE(I,J,3)
```

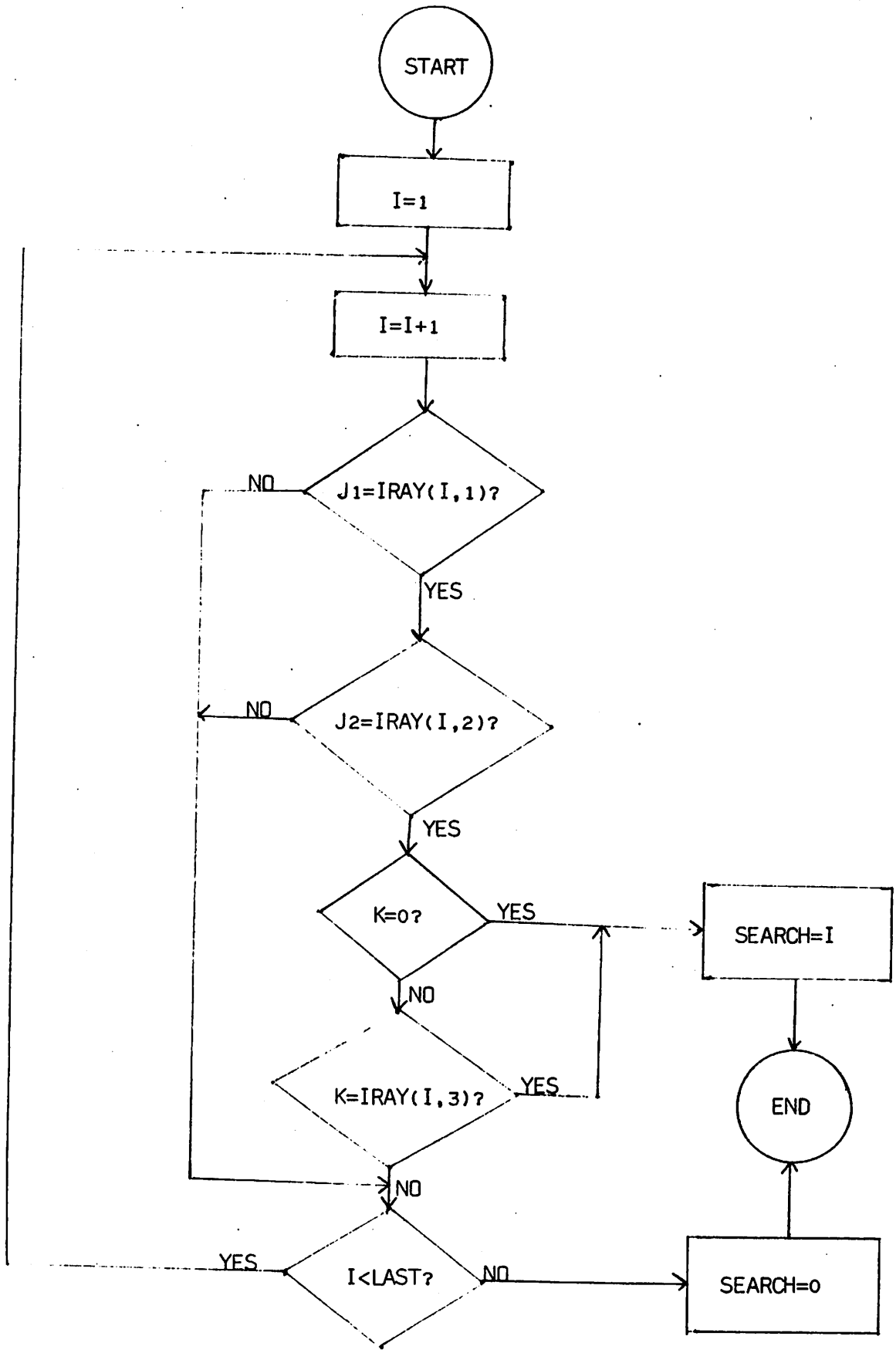
is performed for J equal from one to four and for I equal from one to the contents of CBCODE(1,1,3).

INTEGER FUNCTION SEARCH

Parameters: J1,J2,LAST,IRAY,K

Purpose: This function determines if the name specified by J1 and J2 (four characters per word) appears in the array IRAY. If $K \neq 0$, the number associated in IRAY with the name must also be equal to K. If the name (and number) are not found, the value of SEARCH is zero. Otherwise, SEARCH equals the index of the matched entry in IRAY. The parameter LAST refers to the current length of IRAY.

FLOWCHART FOR SEARCH(J1,J2,LAST,IRAY,K)



SUBROUTINE SRTOUT

Parameters: LOUT

Purpose: This routine is designed to sort the contents of the array OUT so that the entries appear in alphabetical order. The algorithm used is the standard bubble sort with the words OUT(I,1),OUT(I,2), and OUT(I,3) treated as if they were a single string of characters. LOUT is the number of entries in OUT.

SUBROUTINE TRACE

PARAMETERS: MDNAM1, MDNAM2

PURPOSE: This subroutine determines if each variable of a routine that is neither a formal parameter, held in common, or used in a DATA statement is initialized before any possible use that presumes prior initialization. (An array is handled as if it were a simple variable.) Such a use, hencefotth referred to as an input usage, may involve three types of usages, namely those (1) affecting the value of any other variable, (2) regulating the flow of control in the routine, and (3) creating input or output. The parameters specify the name of the routine for which the initialization trace is performed -- four-characters per word, left-justified and blank filled. The following tables, produced by the FACES system, are read from a file and used in the analysis:

DIREC - global directory of names of analyzed modules

SYMTAB - local symbol table

NODTAB - local node table

PRETAB - table of the predecessor nodes for each node in NODTAB

USETAB - table of the usages of each symbol in SYMTAB

ALGORITHM: The table DIREC is loaded in order to determine from it the module number of the routine, as specified by the parameters, to be analyzed. When this number is found the local tables can be located and loaded from file. A search is made through SYMTAB to locate the stored symbols. For each symbol that is an array or variable

name, it is determined (from SYMTAB) if the symbol is held in common or (from USETAB) if it is initialized by a DATA statement. In both instances the symbol is assumed to be handled correctly and is not processed futther. Otherwise, analysis for that symbol continues.

While checking for the symbol's use in a DATA statement, two other processes are ascomplished. First, the nodes associated with each of its usages are flagged in the array LIST. Second, a flag is set when a use is found that involves initialization. If no initialization has occurred when all usages have been recorded, a diagnostic is produced and the next symbol is processed.

If none of the above criteria have succeeded, then the complete initialization trace is begun for the symbol. This involves the backward trace, from each input usage, along all of its possible entry paths (except the loop from a DO loop's terminal node to the DO statement). Each such path must pass through a node that initializes the variable. To assume this the following sequential steps are applied:

(1) Locate on input usage. If no (further) input use exists, processing terminates for that symbol.

(2) The node of this usage is marked, through a flag, in UNSTCK and all of its predecessor nodes that are not flaggqd in UNSTCK are placed on the stack NODSTK. If any of these predecessor nodes are outside the range of the routine, proceed to step 7.

(3) If NODSTK is empty go to step 1.

(4) Determine from the flags in LIST if the top entry of NODSTK uses the variable. If it does not, remove it from NODSTK and return to step 2.

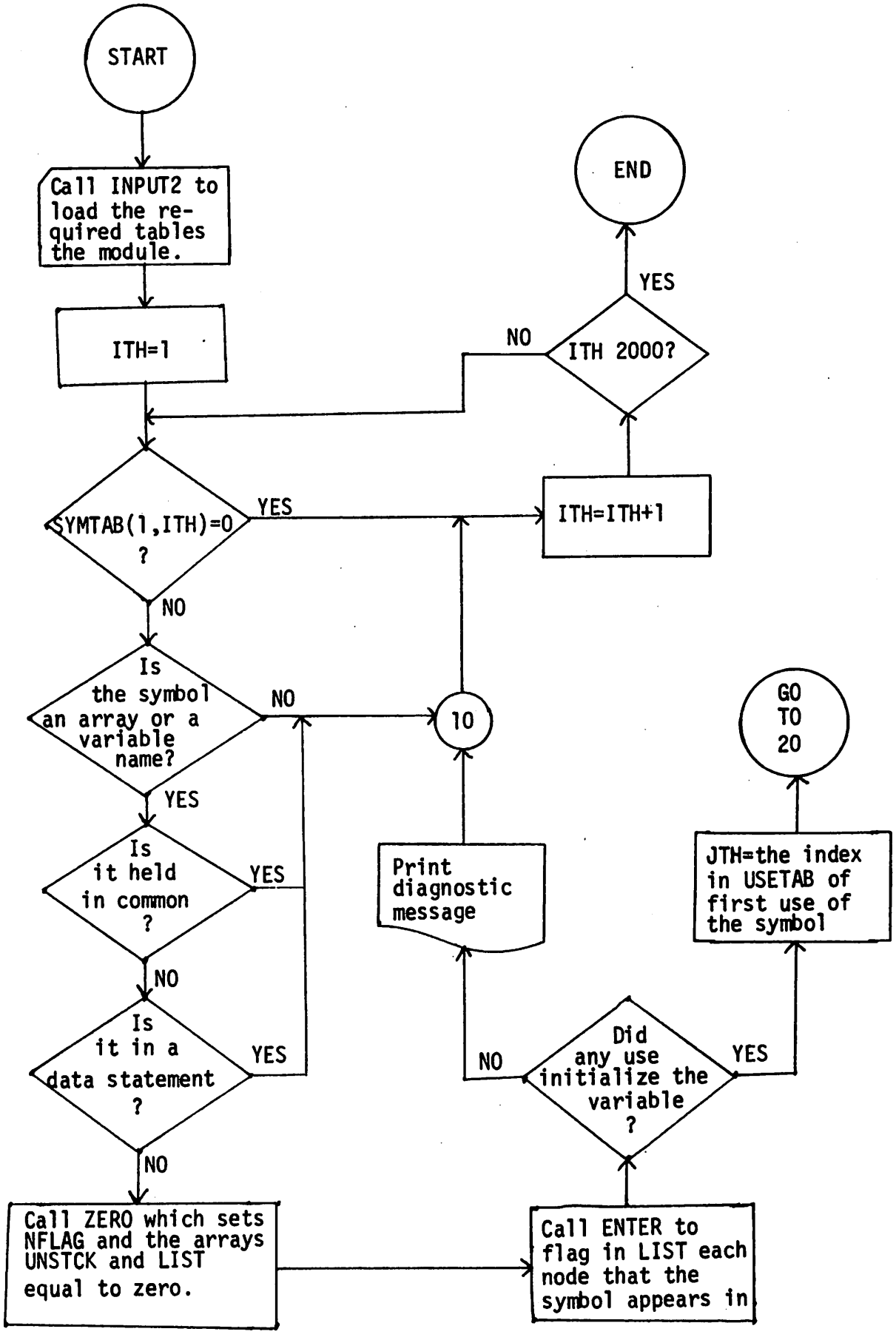
(5) Determine if the variable is initialized in its use at this node. If it is not, remove the node from NODSTK and go to step 2.

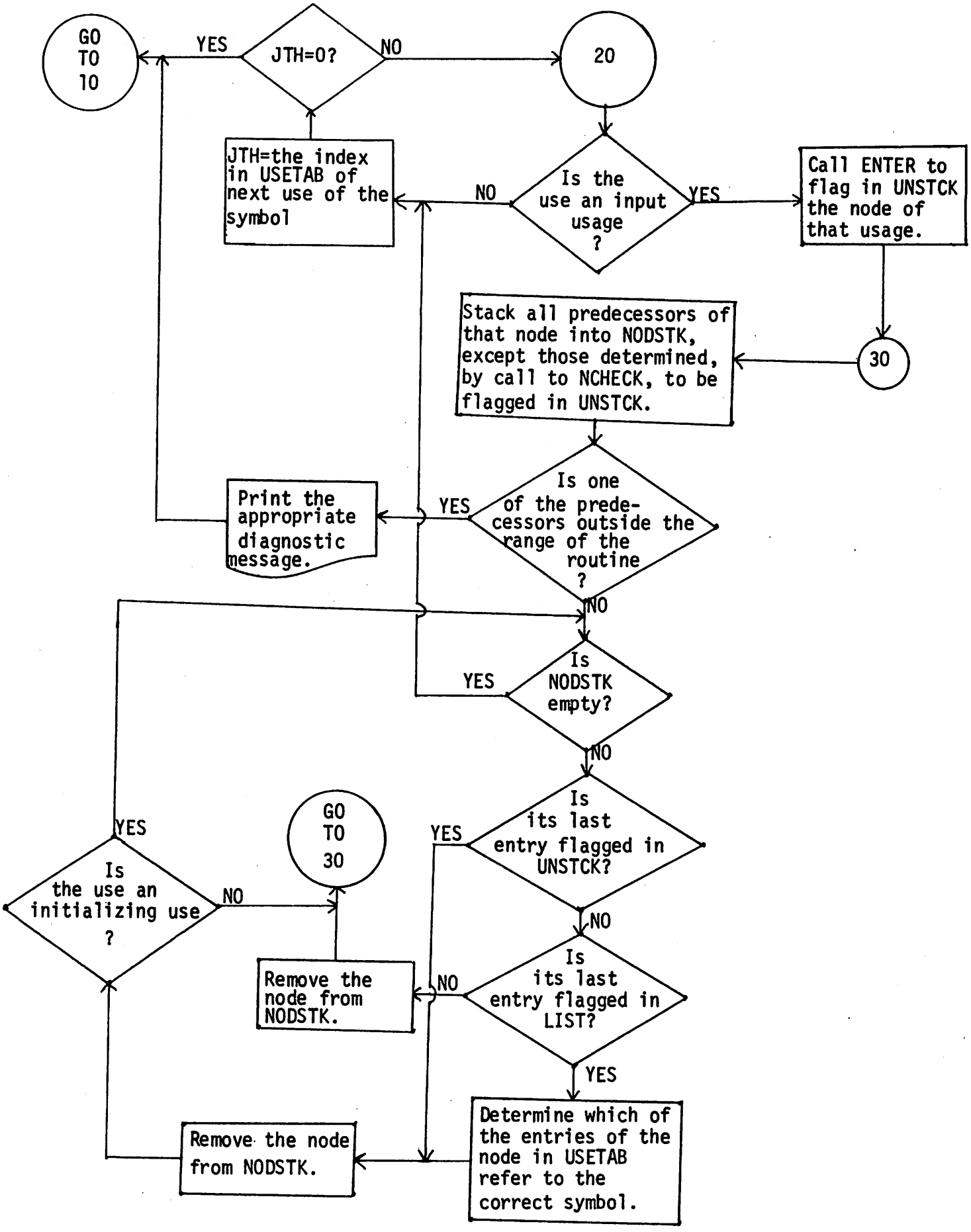
(6) Remove the node from NODSTK and go to step 3.

(7) Print a diagnostic for that symbol.

IRREGULARITIES: Routine calls are assumed to never involve initialization of their actual parameters. Thus a routine call that does in fact initialize one or more of its parameters could result in superfluous diagnostics being produced for those parameters.

FLOWCHART FOR TRACE




GO TO 10
YES
JTH=0?
NO
20
JTH=the index in USETAB of next use of the symbol
NO
Is the use an input usage ?
YES
Call ENTER to flag in UNSTCK the node of that usage.
Stack all predecessors of that node into NODSTK, except those determined, by call to NCHECK, to be flagged in UNSTCK.
30
Print the appropriate diagnostic message.
YES
Is one of the predecessors outside the range of the routine ?
NO
YES
Is NODSTK empty?
NO
YES
Is its last entry flagged in UNSTCK?
NO
GO TO 30
YES
Is the use an initializing use ?
NO
Remove the node from NODSTK.
NO
Is its last entry flagged in LIST?
YES
Remove the node from NODSTK.
Determine which of the entries of the node in USETAB refer to the correct symbol.

SUBROUTINE TRISET

Parameters: I,JJ

This is a master routine whose principal function is to reference the routines LOAD and CODER. It first enters into the next available location in ERROR (determined by the common variable NPNTR) the module number as found in CNTAB(4,JJ). Then the call to LOAD is made with the parameter JJ. From CNTAB(3,JJ), NTOP and NBOT are given values. The following statement is then executed:

```
CALL CODER(I,NTOP,NBOT)
```

SUBROUTINE ZERO

This routine sets the value of NFLAG and each word of the arrays LIST and UNSTCK equal to zero.

Appendix

A. System library routines used by FACES.

FUNCTION LSHIFT (WORD, I)

LSHIFT is called by various bit manipulating routines to perform shifting. If I is positive, the value returned is WORD left shifted I bits, otherwise, the value returned is WORD right shifted I bits with sign extension.

LEFT is identical with LSHIFT.

IDENTIFICATION

Code Designation:	M2 CAL LEFT
Title:	Nominal Left Shift Function
Author:	Thos Sumner, Computer Center, University of California, Berkeley
Date:	July 1972
Environment:	Machine: CDC 6000 Series Operating System: CALIDOSCOPE Coding Language: COMPASS Linkage Convention: RUN2 FORTRAN
Program Name:	LEFT
Deck Name:	LEFT
Entry Points:	LEFT
Availability:	Public Fileset SUBRLIB

PURPOSE

LEFT provides a simple shifting function for FORTRAN use equivalent to the 6000 series machine instruction LXi Bj,X_k . This function should be used only by programmers who fully understand binary data representations. The resulting program may not be readily transferred to other makes of computers.

USAGE

LEFT is called with two arguments, returning as its value the result of shifting its first argument the number of bits indicated by its second argument, e.g.,

```
J = LEFT(M,N)
```

sets J to the result of performing an N place shift of M.

If N is positive, the shift will be left circular, i.e., bits shifted out of the sign bit reenter at the least significant bit and each other bit is shifted N positions toward the sign bit.

If N is negative, the shift will be right arithmetic, i.e., bits are shifted N places toward the least significant bit and all positions vacated on the left are filled by replicating the sign bit. Bits shifted past the least significant bit position are discarded.

Restriction: $|N| < 64$

Refer to the CDC 6000 series Reference Manual for the result generated by LXi Bj,X_k when this restriction is violated.

Examples:

1. The statements

```
M = 5
N = LEFT(M,3)
```

result in N having the value 40 (i.e., 5*2**3)

2. The statements

```
I = 17
J = -3
K = LEFT(I,J)
```

result in K having the value 2 (i.e., 17*(2**-3) with loss of the fraction).

2. The statements

```
L = 5372
L = LEFT(L,60)
```

result in L having the value 5372 since a circular shift of 60 bits (the 6000 series machine word length) is the same as a shift of 0 bits.

SUBROUTINE TSDISK

A CAL system library routine used by FACES to perform random access disk input and output.

IDENTIFICATION

Code Designation:	I9 CAL TSDISK
Title:	Random Access Disk Input/Ouput Subroutine
Author:	Thos Sumner, Computer Center, University of California, Berkeley
Date:	Revision: February 1972
Environment:	Machine: CDC 6000 Series Operating System: CALIDOSCOPE Coding Language: COMPASS Linkage Convention: RUN2 Fortran
Program Name:	TSDISK
Entry Points:	CLDISK, LRDISK, MTDISK, OPDISK, RDDISK, WRDISK, NMDISK
Availability:	CLDR System Library (Public Fileset SUPRLIB)

PURPOSE

To provide random access disk storage for the Fortran programmer.

TSDISK replaces the previous library package SSDISK.[1] Any program which uses SSDISK (as specified in the writeup I9 CAL SSDISK) should function correctly using TSDISK unless the TSDISK deck name or new entry points conflict with those of the user. Use of TSDISK results in improved performance and reduced storage requirement (reduction of about 600B words of code plus the elimination of the user-supplied buffer). Except as provided in this writeup, no usage of SSDISK in a manner other than that described in its writeup is supported by TSDISK. A number of undocumented restrictions in SSDISK are eliminated in TSDISK. TSDISK does not use the CDC SCOPE 3 macros nor the library routines they invoke.

USAGE

The user must provide an index array whose size is greater than the number of distinct records to be written on the random access fileset. The fileset RANDOM is normally used by TSDISK and should not be accessed by any other routine. TSDISK may only be used to read a fileset which it has created. If a random fileset is to be passed to another job, it may be passed as a common fileset (declared by a COMMON control card or library subroutine C4 CAL KOMMON) and properly closed by the job which creates it (see CLDISK below).

---

[1] SSDISK by Marilyn Mahan, Computer Center, and George Pitt, Space Sciences Lab., University of California, Berkeley.

To open fileset RANDOM

and provide access to the index array by TSDISK, OPDISK is normally called exactly once before any other call is made to TSDISK in the job step. There is an exception to this if the NMDISK option is being used (see NMDISK below).

CALL OPDISK(N,INDEX)

N - the dimension of the array INDEX, N ≥ 2. At most N-1 indexed records (identified by the integers from 1 to N-1) may be written on RANDOM. If RANDOM was written in a previous job or job step, N must be at least equal to the value used when it was written.

INDEX - the array declared in the user's program to hold the random file index.

To write on fileset RANDOM

CALL WRDISK(IW,ARRAY,KW)

IW - the record identification number for the record to be written. IW is an integer and 1 ≤ IW < N. The values allowed for IW may, but need not, be used in their natural order (IW = j, followed by IW = j+1).

ARRAY - the first word of the block of data to be stored on the disk as the record identified by IW.

KW - the number of words to be written in the record identified by IW.

If a record has already been written for the given value of IW, it will be superseded by the new record, and the disk space used by the old record will be reused provided the new record is not longer than the old one. If the new record is longer, the disk space occupied by the old record is wasted.

Writing is done with recall (i.e., the central processor is relinquished by the job until the operation is complete) and control returns to the calling program after the operation is complete.

To read from fileset RANDOM

CALL RDDISK(IR,ARRAY,KR)

IR - the identification number of the record to be read.

ARRAY - first word of the block into which the data in the indicated record are to be read.

KR - the number of words to be read from the indicated record. This must be less than or equal to the number of words written in the record.

Reading is done with recall and control returns to the calling program after the operation is complete.

To determine the length of a record on fileset RANDOM

J = LRDISK(IR) (0 ≤ IR < N)

sets J to the length of record number IR. If IR is zero, LRDISK returns the index length. If record IR has not yet been written, LRDISK returns zero. If IR is out of range, LRDISK returns minus one.

To close the fileset RANDOM

CALL CLDISK

This call is used to close the fileset RANDOM after it is written if it is to be preserved for use in a subsequent job or job step. CLDISK need not be called if RANDOM is only being read in the current job step.

To clear the previous contents of fileset RANDOM

CALL MTDISK

This clears the index array and releases all disk storage assigned to the fileset RANDOM. The fileset is still open and is ready to be written.

To assign a different name to the fileset RANDOM

CALL NMDISK (NAME)

NAME - the name to be assigned to the random access fileset. It must be an L-type literal which is a legal fileset name or a variable whose value is such a literal.

If NMDISK is used, it must be called before OPDISK is called.

If more than one random fileset is created using this feature, each should be closed before the name assignment and file opening calls are made for the next, i.e., the sequence of calls is CLDISK, NMDISK, OPDISK. Only one TSDISK fileset may be open at a time.

ERRORS

The following errors will produce a diagnostic and abort the job:

BAD RECORD NO. REQUESTED.

Attempting to use a record number less than one or greater than the index allows (RDDISK, WRDISK).

FILESET RANDOM IS SEQUENTIAL

The fileset named RANDOM already exists as a sequential fileset (OPDISK).

INDEX LENGTH BAD.

Attempting to declare index length less than two. This error may also occur if the array given is too small to contain the index of an existing RANDOM fileset (OPDISK).

READ OF UNWRITTEN RECORD.

Attempting to read a record which has not yet been written (RDDISK).

RECORD TOO SHORT.

Attempting to read a longer record than was written for that record number (RDDISK).

All of these abort with a SYSTEM error code of 99.

RESTRICTION

TSDISK may not be used to read or write the system communication area, i.e., the first 64 (100B) words assigned to the job. This area is not allocated to Fortran programs or data, so this restriction does not apply to anyone doing pure Fortran programming.

METHOD

When the fileset is opened, the declared index is cleared to zero and, if the fileset already exists, the old index is read in. When record number IW is written, INDEX(IW+1) is set to a non-zero value showing the length and disk address of the record. When record number IR is to be read, INDEX(IR+1) is used to retrieve it. INDEX(1) is used for error checking. The other elements of INDEX may be moved as needed.

MEMORY REQUIREMENT

Approximately 300B words of memory are required. Only SYSTEM is required from the library in addition (SYSTEM is always present when using Fortran programs).

TIMING

The central processor time required for a write operation is independent of the record length.

The central processor time required for a read operation is independent of the record length when the amount being read is a multiple of 64 words (the disk physical record unit size). Additional time is required for those words which are part of a fractional PRU at the end of the block being read in. Additional PPU time is also required for a fractional PRU.

Optimization:

TSDISK will run substantially faster if the user confines himself to reading blocks which are integral multiples of 64 words. It requires much more CP time to read a 63 word block than it does a 64 word block.

In order to minimize the PP time required, data should be moved to or from the disk in blocks as large as the program logic and convenience will allow. To illustrate this, two jobs were run, each writing and reading back 100 records, but the record sizes in the second case were ten times those in the first. The second job moved ten times the data of the first, but required only 12% more CP time and only 66% more PP time.

Comparison to SSDISK central processor times:

In the worst case for TSDISK (63 word blocks being read), it is about three times as fast as SSDISK.

In the best cases for TSDISK (reads which call for multiples of 64 words, and all writes), it is ten to one hundred times as fast.

Much of this time is saved by avoiding use of the CDC library routine CPC which was used by SSDISK.

B. Sample file handling routine using TSDISK.

```
      SUBROUTINE FILSET
C*****************************************************************C
C     THIS ROUTINE OPENS FILSET RANDOM AND DEFINE INDEX ARRAY USED    C
C     BY TSDISK                                                       C
C     TSDISK IS A MACHINE DEPENDENT ROUTINE                           C
C*****************************************************************C
      DIMENSION INDEX (1000)
      CALL OPDISK (100 , INDEX)
      RETURN
      END
```

```
      SUBROUTINE OUTLT(MODNO)
      IMPLICIT INTEGER (A-Z)
C*************************************************************C
C     THIS ROUTINE WRITES THE LOCAL TABLE FOR A SINGLE MODULE TO FILE  C
C                                                               C
C     NLTBS = NUMBER OF LOCAL TABLES IN A MODULE.               C
C     NGTBS = NUMBER OF GLOBAL TABLES.                          C
C                                                               C
C     SUBROUTINE WRDISK (IW, ARRAY, KW) 9 A CAL SYSTEM ROUTINE FOR      C
C     HANDLING RANDOM ACCESS FILES.                             C
C     IW = THE RECORD NUMBER TO BE WRITTEN                      C
C     ARRAY = THE FIRST WORD OF THE BLOCK OF DATA TO BE STORED ON       C
C             THE DISK AS THE RECORD IDENTIFIED BY IW.          C
C     KW = THE NUMBER OF WORDS TO BE WRITTEN IN THE RECORD IDENTIFIED   C
C          BY IW.                                               C
C*************************************************************C
      COMMON /GLOBAL/ DIREC(3,10 ), LDIR, INDIR, CNTAB(4,300),
     C LCN, INDCN, PARTAB (2000), LPAR, IPAR, JPRIME
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     C ALTAB (56), LAL, INDAL, DOTAB (3,50), LDO, INDDO, IPRIME,
     C NODTAB (3,1000), LNOD, SUCTAB (1500), ISUC, PRETAB (1500), IPRE,
     C LPS, TRIP (1000,2), LTRIP, ITR, STALTB (2,100), LSTAL, INDST
      DATA NLTBS /8/, NGTBS /3/
C*************************************************************C
C     EVALUATE THE STARTING RECORD NUMBER FOR MODULE MODNO      C
C*************************************************************C
      M = MODNO
      IBSN = ( MOD(M, 100) - 1 ) *NLTBS + NGTBS    1
C*************************************************************C
C     STORE SYMTAB.                                             C
C*************************************************************C
      KOUNT = LSYM * 5
      CALL WRDISK (IBSN, SYMTAB, KOUNT)
C*************************************************************C
C     STORE NODTAB.                                             C
C*************************************************************C
      IBSN = IBSN + 1
      KOUNT = LNOD * 3
      CALL WRDISK (IBSN, NODTAB, KOUNT)
C*************************************************************C
C     SAVE SUCTAB LAST ENTRY POINTER IN SUCTAB (1)              C
C     STORE SUCTAB IN FILE.                                     C
C*************************************************************C
      IBSN = IBSN + 1
      KOUNT = LPS
      SUCTAB (1) = ISUC
      CALL WRDISK (IBSN, SUCTAB, KOUNT)
C*************************************************************C
C     SAVE PRETAB LAST ENTRY POINTER IN PRETAB (1).             C
C     STORE PRETAB IN FILE                                      C
C*************************************************************C
      IBSN = IBSN + 1
      KOUNT = LPS
      PRETAB(1) = IPRE
      CALL WRDISK (IBSN, PRETAB, KOUNT)
C*************************************************************C
C     SAVE USETAB CURRENT INDEX IN USETAB (1,1).                C
```

```
C      STORE USETAB IN FILE.                                            C
C*****************************************************************C
       IBSN = IBSN + 1
       KOUNT = LUSE * 2
       USETAB(1,1) = INDUT - 1
       CALL WRDISK (IBSN, USETAB, KOUNT)
C*****************************************************************C
C      SAVE ALTAB CURRENT INDEX IN ALTAB (1).                           C
C      STORE ALTAB ON FILE.                                             C
C*****************************************************************C
       IBSN = IBSN + 1
       KOUNT = LAL
       ALTAB (1) = INDAL - 1
       CALL WRDISK (IBSN, ALTAB, KOUNT)
C*****************************************************************C
C      SAVE DOTAB CURRENT INDEX IN DOTAB (1,1).                         C
C      STORE DOTAB IN FILE.                                             C
C*****************************************************************C
       IBSN = IBSN + 1
       KOUNT = LDO * 3
       DOTAB (1,1) = INDDO
       CALL WRDISK (IBSN, DOTAB, KOUNT)
C*****************************************************************C
C      SAVE STALTB CURRENT INDEX IN STALTB (1,1).                       C
C      STORE STALTB IN FILE.                                            C
C*****************************************************************C
       IBSN = IBSN + 1
       KOUNT = LSTAL * 2
       STALTB (1,1) = INDST - 1
       CALL WRDISK (IBSN, STALTB, KOUNT)

       PRINT 100, MODNO
  100  FORMAT (1HO, *LOCAL TABLES FOR MODULE *, I4,* HAVE BEEN STORED ON
     CFILE*)
       RETURN
       END
```

```
      SUBROUTINE OUTGT
      IMPLICIT INTEGER (A-Z)
C*****************************************************************C
C     THIS ROUTINE WRITES THE GLOBAL TABLES TO FILE               C
C                                                                 C
C     SUBROUTINE WRDISK (IW, ARRAY, KW) 9 A CAL SYSTEM ROUTINE FOR C
C     HANDLING RANDOM ACCESS FILES.                               C
C     IW = THE RECORD NUMBER TO BE WRITTEN                        C
C     ARRAY = THE FIRST WORD OF THE BLOCK OF DATA TO BE STORED ON C
C             THE DISK AS THE RECORD IDENTIFIED BY IW.            C
C     KW = THE NUMBER OF WORDS TO BE WRITTEN IN THE RECORD IDENTIFIED C
C          BY IW.                                                 C
C                                                                 C
C     THIS SUBROUTINE CALLS MACHINE DEPENDENT ROUTINES            C
C*****************************************************************C
      COMMON /GLOBAL/ DIREC(3, 1 0), LDIR, INDIR, CNTAB(4, 3 0),
     C LCN, INDCN, PARTAB (2000), LPAR, IPAR, JPRIME
C*****************************************************************C
C     STORE DIREC ON FILE.                                        C
C*****************************************************************C
      KOUNT = LDIR * 3
      IW = 1
      CALL WRDISK (IW, DIREC, KOUNT)
C*****************************************************************C
C     SAVE THE LAST ENTRY POINTER OF CNTAB IN CNTAB (1,1).        C
C     STORE CNTAB IN FILE.                                        C
C*****************************************************************C
      KOUNT = LCN * 4
      IW = 2
      CNTAB(1,1) = INDCN
      CALL WRDISK (IW, CNTAB, KOUNT)
C*****************************************************************C
C     SAVE THE PARTAB LAST ENTRY POINTER IN PARTAB (1).           C
C     STORE PARTAB IN FILE.                                       C
C*****************************************************************C
      KOUNT = LPAR
      IW = 3
      PARTAB(1) = IPAR - 1
      CALL WRDISK (IW, PARTAB, KOUNT)

      PRINT 100
  100 FORMAT (1H1, *GLOBAL TABLE HAS BEEN STORED ON FILE*)
      RETURN
      END
```

```
      SUBROUTINE EOFRIT
C*****************************************************************C
C     THIS ROUTINE CLOSES FILSET RANDOM AND PRESERVE IT FOR USE IN    C
C     A SUBSEQUENT JOB OR JOB STEP                                    C
C                                                                     C
C     THIS SUBROUTINE CALLS MACHINE DEPENDENT ROUTINES                C
C*****************************************************************C
      CALL CLDISK
      RETURN
      END
```

In order to prepare filset RANDOM for accessing by the Diagnostic Routines, subroutine OPDISK (a routine of TSDISK) must be called before any read is executed. OPDISK is usually called by the main program (DRIVER) provided by the user. For example :

```
PROGRAM DRIVER ( OUTPUT )
IMPLICIT INTEGER (A-Z)
DIMENSION INDEX (1000)
CALL OPDISK (1000, INDEX)
      :
calls to Diagnostic Routines
      :
      :
      :
STOP
END
```

```
      SUBROUTINE READGTS(NTABLE)
C*****************************************************************C
C     ROUTINE READGTS READS IN GLOBAL TABLES FROM FILE            C
C     NTABLE = 1    DIREC                                         C
C            = 2    CNTAB                                         C
C            = 3    PARTAB                                        C
C                                                                 C
C     NGTBS = NUMBER OF GLOBAL TABLES                             C
C                                                                 C
C     SUBROUTINE RDDISK (IR, ARRAY, KR) 9 A CAL SYSTEM ROUTINE FOR C
C     READING RANDOM FILE RLCORDS.                                C
C     IR = THE IDENTIFICATION NUMBER OF THE RECORD TO BE READ     C
C     ARRAY = FIRST WORD OF THE BLOCK INTO WHICH THE DATA IN THE  C
C             INDICATED RECORD ARE TO BE READ.                    C
C     KR = THE NUMBER OF WORDS TO BE READ FROM THE INDICATED RECORD. C
C                                                                 C
C     RDDISK IS A MACHINE DEPENDENT ROUTINE.                      C
C*****************************************************************C
      IMPLICIT INTEGER (A-Z)
      COMMON /GLOBAL/ DIREC (3,1  ), LDIR, INDIR, CNTAB (4,3 0), LCN,
     1 INDCN, PARTAB (2000), LPAR, IPAR, JPRIME
      DATA NGTBS /3/
      IF ( (NTABLE.GE.1) .AND. (NTABLE.LE.NGTBS) ) GO TO 20
      PRINT 10, NTABLE
   10 FORMAT (////, 1H , *SUBROUTINE READGTS*,/, 1H ,
     1 *COMPUTED GO TO STATEMENT INDEX OUT OF RANGE*,/,
     2 1H , *NTABLE = *, I4)
      STOP
   20 GO TO (1000, 200 , 3000), NTABLE
C*****************************************************************C
C     READ GLOBAL TABLE.                                          C
C*****************************************************************C
 1000 KOUNT = LDIR * 3
      CALL RDDISK (NTABLE, DIREC, KOUNT)
      RETURN
C*****************************************************************C
C     READ CNTAB.                                                 C
C     CNTAB (1,1) CONTAINS CNTAB CURRENT INDEX.                   C
C*****************************************************************C
 2000 KOUNT = LCN * 4
      CALL RDDISK (NTABLE, CNTAB, KOUNT)
      INDCN = CNTAB (1,1)
      RETURN
C*****************************************************************C
C     READ PARTAB                                                 C
C     PARTAB (1) CONTAINS PARTAB CURRENT POINTER.                 C
C*****************************************************************C
 3000 KOUNT = LPAR
      CALL RDDISK (NTABLE, PARTAB, KOUNT)
      IPAR = PARTAB (1)
      RETURN
      END
```

```
      SUBROUTINE READLTS(MODNO,NTABLE)
C*****************************************************************C
C     ROUTINE READLTS READS IN LOCAL TABLES FROM FILE             C
C     MODNO = MODULE NUMBER                                       C
C     NTABLE =      LOCAL TABLE NUMBER                            C
C            = 1    SYMTAB                                        C
C            = 2    NODTAB                                        C
C            = 3    SUCTAB                                        C
C            = 4    PRETAB                                        C
C            = 5    USETAB                                        C
C            = 6    ALTAB                                         C
C            = 7    DOTAB                                         C
C            = 8    STALTB                                        C
C                                                                 C
C     NLTBS = NUMBER OF LOCAL TABLES IN A MODULE                  C
C     LGTBS = NUMBER OF GLOBAL TABLES                             C
C                                                                 C
C     SUBROUTINE RDDISK (IR, ARRAY, KR) 9 A CAL SYSTEM ROUTINE FOR C
C     READING RANDOM FILE RECORDS.                                C
C     IR = THE IDENTIFICATION NUMBER OF THE RECORD TO BE READ     C
C     ARRAY = FIRST WORD OF THE BLOCK INTO WHICH THE DATA IN THE  C
C             INDICATED RECORD ARE TO BE READ.                    C
C     KR = THE NUMBER OF WORDS TO BE READ FROM THE INDICATED RECORD. C
C                                                                 C
C     RDDISK IS A MACHINE DEPENDENT ROUTINE.                      C
C*****************************************************************C
      IMPLICIT INTEGER (A-Z)
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB (56), LAL, INDAL, DOTAB (3,50), LDO, INDDO, IPRIME,
     2 NODTAB (3,1000), LNOD, SUCTAB (1500), ISUC, PRETAB (1500), IPRE,
     3 LPS, STALTB (2,100), LSTAL, INDST
      DATA NLTBS /8/, NGTBS /3/
C*****************************************************************C
C     EVALUATE THE STARTING RECORD NUMBER OF MODULE.              C
C*****************************************************************C
      M = MODNO
      IBSN = ( MOD(M,1   ) - 1 ) * NLTBS + NGTBS
      IF ( (NTABLE.GE.1) .AND. (NTABLE.LE.NLTBS) ) GO TO 20
      PRINT 10, NTABLE
   10 FORMAT (////, 1H , *SUBROUTINE READLTS*,/, 1H ,
     1 *COMPUTED GO TO STATEMENT INDEX OUT OF RANGE*,/,
     2 1H , *NTABLE = *, I4)
      STOP
   20 IBSN = IBSN + NTABLE
      GO TO (1 00, 200 , 30  , 4   , 5000, 600 , 7000, 8000), NTABLE
C*****************************************************************C
C     READ SYMTAB.                                                C
C*****************************************************************C
 1000 KOUNT = LSYM * 5
      CALL RDDISK (IBSN, SYMTAB, KOUNT)
      RETURN
C*****************************************************************C
C     READ  NODTAB.                                               C
C*****************************************************************C
 2000 KOUNT = LNOD * 3
      CALL RDDISK (IBSN, NODTAB, KOUNT)
      RETURN
```

```
C*************************************************************************C
C      READ SUCTAB.                                                       C
C      SUCTAB (1) CONTAINS SUCTAB LAST ENTRY POINTER.                     C
C*************************************************************************C
 3000 KOUNT = LPS
      CALL RDDISK (IBSN, SUCTAB, KOUNT)
      ISUC = SUCTAB (1)
      RETURN
C*************************************************************************C
C      READ PRETAB.                                                       C
C      PRETAB (1) CONTAINS POINTER TO THE LAST ENTRY OF PRETAB.           C
C*************************************************************************C
 4000 KOUNT = LPS
      CALL RDDISK (IBSN, PRETAB, KOUNT)
      IPRE = PRETAB (1)
      RETURN
C*************************************************************************C
C      READ USETAB.                                                       C
C      USETAB (1,1) CONTAINS USETAB LAST ENTRY POINTER.                   C
C*************************************************************************C
 5000 KOUNT = LUSE * 2
      CALL RDDISK (IBSN, USETAB, KOUNT)
      INDUT = USETAB (1,1)
      RETURN
C*************************************************************************C
C      READ ALTAB                                                         C
C      ALTAB (1) CONTAINS ALTAB LAST ENTRY POINTER.                       C
C*************************************************************************C
 6000 KOUNT = LAL
      CALL RDDISK (IBSN, ALTAB, KOUNT)
      INDAL = ALTAB (1)
      RETURN
C*************************************************************************C
C      READ DOTAB.                                                        C
C      DOTAB (1,1) CONTAINS POINTER TO LAST ENTRY OF DOTAB.               C
C*************************************************************************C
 7000 KOUNT = LDO * 3
      CALL RDDISK (IBSN, DOTAB, KOUNT)
      INDDO = DOTAB (1,1)
      RETURN
C*************************************************************************C
C      READ STALTB.                                                       C
C      STALTB (1,1) CONTAINS POINTER TO THE LAST ENTRY OF STALTB.         C
C*************************************************************************C
 8000 KOUNT = LSTAL * 2
      CALL RDDISK (IBSN, STALTB, KOUNT)
      INDST = STALTB (1,1)
      RETURN
      END
```

C. Common Block Description.

/BLANK/

SYMTAB (5,1800) - Symbol Table.

LSYM - length of SYMTAB.

NADDR - Contains the SYMTAB address in which the symbol processed is found or should be inserted if this is the first appearance of the symbol.

USETAB - Use Table.

LUSE - length of USETAB.

INDUT - pointer to the next available entry of USETAB.

ALTAB - array length table.

LAL - length of ALTAB.

INDAL - pointer to the next available entry of ALTAB.

DOTAB - DO Table.

LDO - length of DOTAB.

INDDO - pointer to the next entry of DOTAB.

IPRIME - prime number for SYMTAB hash coding.

NODTAB - Node Table.

LNOD - length of NOTAB.

SUCTAB - successor table.

ISUC - pointer to the last/next entry of SUCTAB.

PRETAB - predecessor table.

IPRE - pointer to the last/next entry of PRETAB

LPS - length of predecessor and successor table.

TRIP - TRIP Table.

LTRIP - length of TRIP table.

ITR - pointer to the next available entry of TRIP.

STALTB - storage allocation table.

LSTAL - length of STALTB.

INDST - pointer to the next available space of STALTB.

/GLOBAL/

DIREC - directory

LDIR - length of DIREC.

INDIR - contains the hash coded DIREC address for name being processed.

CNTAB - common name table.

LCN - length of CNTAB.

INDCN - pointer to the last entry of CNTAB.

PARTAB - parameter table

LPAR - length of PARTAB

IPAR - pointer to the last used entry of PARTAB.

JPRIME - prime number for DIREC hash coding.

/DOTAB/

LABFLG - Flag indicating label has already appeared in a DO statement.
(Set by TABS and referenced by TRANS).

/INSTUF/

STORE (2,80) - circular input buffer, hold card image of statement being analyzed.

INP - pointer to the character being analyzed. STORE (2, INP).

L - pointer to the buffer being used, STORE (L, 80).

ISAV - pointer to starting position of an integer in STORE.

ND - number of digits in an integer in STORE.

LFLAG - indicates which line store buffer the present statement is in.

When LFLAG = 1, L = 1

LFLAG = -1, L = 2.

NLFLAG - = 1 indicates that a line of code has been read and that some routines have switched back to the previous card to process information.

= 0 otherwise

HFLAG - = 1 indicates that a hollerith constant may occur because a 1, =, or (has been found).

= 0 otherwise.

LBUF (10) - buffer holding first line of characters of a variable or integer that occur on one line and are continued on another.

/SCAPPA/

TSTAB (2,300) - temporary symbol table, containing alphanumeric strings.

ISS (600) - contains character codes to represent lexical entities extracted from Fortran Statements.

S statement label

V alphanumeric string

I	integer	constant
F	floating point	constant
D	double precision	constant
C	complex	constant
H	hollerith	constant
T	logical	constant
L	logical	operator
R	relational	operator

ITS - pointer to the next unused entry in TSTAB

INS - pointer to the last entry in ISS.

JFLAG - indicates that statement being scanned is the first statement of the input system (program) and card must be read in. Value is 0 for first card and 1 for all others.

EFLAG - indicates to parser that = has been found in statement. Value is the location of = in ISS if = is found, otherwise 0.

ENDFL = 1 if end of file found

= 0 otherwise.

/TABSUB/

ISTP - marker to mark the starting location of STALTB in which variables defined within the currently processed common block is stored.

NSAV - contains statement number in which last symbol processed by TABS occurred.

LAN - contains SYMTAB address of last array name processed.

ISUBAD - contains the SYMTAB address of the subroutine or function name being processed

IPFLAG - = 1 a parameter list is being processed

= 0 otherwise

ICTYPE - contains type code (as used in scanning routine) of constant in a parameter list.

/TAPAR/ (TABS).

NAME (2) - stores name being processed.
4 chars./word.

STNO - statement number being processed.

DIM (3) - contains dimensions of an array.

MN - module number of the module being processed.

MPOP - special communication flag between parser and TABS. Normally MPOP = 0.
If MPOP = 1., a special call is being made to TABS to indicate a right paren. is found in parameter list. (End of parameter list)
If MPOP = 2, a special call is being made to indicate a constant in a parameter list.

NFILE - file number of the module being processed.

ARGNO - contains argument number of variable or constant in a parameter list.

/TYPES/

LTRS (26) - each word corresponds to one letter of the alphabet and contains a 5 - 1 bit fields corresponding to the possible types of variables. The appropriate bit is set to 1 if an implicit type statement has so indicated.

bit set	
1	integer
2	real
3	double precision
4	complex
5	logical

/TMATCH/

BIAS - Bias the starting location of MATCH.

MATCH(51) - Contains keywords to identify statement types.

INDEX	MATCH ENTRY	FORTRAN St. Type
1		
2		
3	FORM	* FORMAT
4	PRIN	PRINT
5	IMPL	IMPLICIT
6	NAME	* NAMELIST
7	ENCO	* ENCODE
8	DECO	* DECODE
9	PUNC	PUNCH
10	IF	IF
11	COMP	COMPLEX
12	EXTE	EXTERNAL
13	BLOC	BLOCK DATA
14	END	END
15	READ	READ
16		
17		
18		
19		
20	ENDF	* ENDFILE
21		
22		
23	REAL	REAL
24	BACK	* BACKSPACE
25		
26	LOGI	LOGICAL
27	FUNC	FUNCTION
28	DIME	DIMENSION
29		
30	SUBR	SUBROUTINE
31	DATA	DATA
32	PROG	PROGRAM
33	DOUB	DOUBLE PRECISION
34	CALL	CALL
35	INTE	INTEGER
36	COMM	COMMON
37		
38	ASSI	ASSIGN

39	WRIT		WRITE
40	EQUI	*	EQUIVALENCE
41	STOP		STOP
42	REWI	*	REWIND
43			
44	CONT		CONTINUE
45	GOTO		GOTO
46			
47			
48	ENTR		ENTRY
49			
50	PAUS	*	PAUSE
51	RETU		RETURN

* - statement types currently ignored by PARSER.

Appendix D

STATEMENT TYPES

INDEX	MATCH ENTRY	FORTRAN St. Type
1		
2		
3	FORM	* FORMAT
4	PRIN	PRINT
5	IMPL	IMPLICIT
6	NAME	* NAMELIST
7	ENCO	* ENCODE
8	DECO	* DECODE
9	PUNC	PUNCH
10	IF	IF
11	COMP	COMPLEX
12	EXTE	EXTERNAL
13	BLOC	BLOCK DATA
14	END	END
15	READ	READ
16		
17		
18		
19		
20	ENDF	* ENDFILE
21		
22		
23	REAL	REAL
24	BACK	* BACKSPACE
25		
26	LOGI	LOGICAL
27	FUNC	FUNCTION
28	DIME	DIMENSION
29		
30	SUBR	SUBROUTINE
31	DATA	DATA
32	PROG	PROGRAM
33	DOUB	DOUBLE PRECISION
34	CALL	CALL
35	INTE	INTEGER
36	COMM	COMMON
37		
38	ASSI	ASSIGN
39	WRIT	WRITE
40	EQUI	* EQUIVALENCE
41	STOP	STOP
42	REWI	* REWIND
43		
44	CONT	CONTINUE
45	GOTO	GOTO
46		
47		
48	ENTR	ENTRY
49		
50	PAUS	* PAUSE
51	RETU	RETURN
52		ASSIGNMENT STATEMENT
53		DO STATEMENT

Note: Index also indicates statement type code.

* - currently ignored by PARSER

Appendix E Some Limited Capabilities of FACES

1. FACES does not handle some complex FORTRAN constructions (e.g. NAMELIST and EQUIVALENCE) and some machine dependent constructions (e.g. ENCODE and test for EOF). Appendix D contains a list of FORTRAN statement types processed by FACES.

2. The system will not handle function calls in which one of the parameters require the evaluation of the same function. For example, if LEFT is a function, FACES will bomb out on the following statement :

```
IWORD = LEFT (LEFT (IDUMMY, 2), 4)
```

3. Some of the warning messages issued by the diagnostic routines are incorrect, especially in parameter alignment (PARAL).

Appendix F. USER EXPERIENCE

The FACES system, written in FORTRAN, consists fo two parts: the FORTRAN Front End and the Diagnostic Routines. The system made up of over 130 subroutines and functions which require approximately 8000 card images (6000 cards for F.F.E. and 2000 cards for Diagnostic Routines).

FACES has been successfully used at NASA Marshall Space Flight Center. At the Berkeley campus, FACES has been used to debug and validate itself during the installation process. For example, an uninitiallized variable was detected by FACES during installation. Operational experience on the CDC 6400 indicates a processing rate ranging from 6 cards/sec to 14 cards/sec.

One of the limitations of FACES is the large amount of memory required by the system tables. The current configuration allows the analysis of up to 100 program modules with a maximum of 1000 statements in each module. The current table sizes are capable to handle all the software systems FACES has analyzed so far. In fact, some users have suggested reducing the table sizes in order to have better turn around time. This proposal is being considered.

Some users have also complained about the output format of the diagnostic routines. In some error messages, modules are referenced by module number (assigned by FACES) instead of module name. Also, the exact location of the error detected and the statement in question are not printed. Some thoughts have been given to improve the output format.

FACES does not handle some complex FORTRAN statements (e.g. EQUIVALENCE and NAMELIST) and machine dependent statements (e.g. ENCODE and DECODE). FACES may be extended to analyse these statements in the future.

In general, FACES is a powerful automatic software validation system and users are quite satisfied with its performance.

Appendix G ERROR MESSAGES

Messg.#	Subroutine	Error Message
1	PARSER	Statement type not recognized
2	FARITH	Syntax error in assignment statement
3	FASS	Syntax error in ASSIGN statement
4	FDO	Index variable of DO statement not recognized
5	FDO	Error in limits or increment of DO statement
6	FGOTO	No comma after switch variable of assigned GOTO st.
7	FGOTO	No left parenthesis in assigned GOTO statement
8	FGOTO	No left parenthesis in computed GOTO statement
9	FGOTO	Syntax error in label list of GOTO statement
10	FGOTO	No comma following list in computed GOTO statement
11	FIF	Transfer label or variable not recognized in IF st.
12	FIF	No left parenthesis in IF statement
13	FCALL	No left parentheses in CALL statement
14	FTYPE	Syntax error in TYPE statement
15	FCOMON	Syntax error in COMMON statement
16	FCOMON	Common block name not recognized
17	FCOMON	Error in delimeter of common block name
18	FCOMON	Variable not recognized following common block name
19	LISTP	Error in delimeter in list
20	LISTP	Variable not recognized in list
21	FUNCI	No left parenthesis in FUNCTION statement
22	FSUB	No left parenthesis in subroutine statement
23	FDO	Equals sign not recognized in proper sequence
24	FDO	Value of limit or increment too large
25	FGOTO	No index variable in completed GOTO statement
26	FREAD	No comma in "READ F, list"
27	FREAD	Syntax error in "READ TAPE unit, list"
28	FREAD	Syntax error in "READ fvar, list"
29	FREAD	Syntax error in "READ INPUT TAPE unit, F, list"
30	FWRITE	Type of WRITE statement unrecognizable
31	FWRITE	Syntax error in "WRITE TAPE unit, list"
32	FWRITE	Syntax error in "WRITE OUTPUT TAPE unit, F, list"

Messg.#	Subroutine	Error Message
33	IOARGS	Unit unrecognizable in READ/WRITE statement
34	IOARGS	Syntax error in arguments of READ/WRITE statement
35	IOARGS	ERR or END condition unrecognizable in READ/WRITE st.
36	IOLIST	Unbalanced parentheses in I/O list
37	FPRPU	No comma after format in PRINT/PUNCH statement
38	FENTRY	No left parenthesis in ENTRY statement
39	LISTP	Subscripted variable has more than 3 dimenensions

SUBROUTINES FREAD

Subroutines FREAD is called by PARSER to process FORTRAN I/O statements. The main function is to recognize the various forms for READ statements and to call subroutines IOARGS and IOLIST when argument lists and I/O lists are encountered. The following forms are recognized:

READ (argument list) list

READ INPUT TAPE unit, f, list

READ f, list

READ TAPE unit, list

WRITE (argument list) list

WRITE OUTPUT TAPE unit, f, list

WRITE TAPE unit, list

where argument list has the form

unit, f, ERR=ℓ_1, END=ℓ_2

where ℓ_1 is the next statement for execution if error is detected in READ/WRITE

ℓ_2 is the next statement for execution if EOF

Entities extracted by FREAD, are the unit reference (unit) and the format reference (f). The unit reference is assumed to be either an integer constant or a PARAMETER variable. If a constant is found, it is simply ignored; if a variable is found, it is passed to TABS with class code 10 (PARAMETER variable), type code 0 (integer) and use code 26 (unit reference). The format reference is assumed to be the label of a FORMAT statement or the name of an array that contains FORMAT specifications. If an integer is

found, it is assumed to be a label and is passed to TABS with class code 5 (label), type code 5 (neutral), and use code 27 (format reference); if a variable is found, it is passed to TABS with class code 6 (variable), type code 5 (neutral), and use code 27 (format reference).

The argument list and I/O variable list are processed by subroutines IOARGS and IOLIST, respectively (see documentation for these routines).

SUBROUTINES FWRITE

Subroutines FWRITE is called by PARSER to process FORTRAN I/O statements. The main function is to recognize the various forms for WRITE statements and to call subroutines IOARGS and IOLIST when argument lists and I/O lists are encountered. The following forms are recognized:

READ (argument list) list

READ INPUT TAPE unit, f, list

READ f, list

READ TAPE unit, list

WRITE (argument list) list

WRITE OUTPUT TAPE unit, f, list

WRITE TAPE unit, list

where argument list has the form

unit, f, ERR=ℓ_1, END=ℓ_2

where ℓ_1 is the next statement for execution if error is detected in READ/WRITE

ℓ_2 is the next statement for execution if EOF

Entities extracted by FWRITE are the unit reference (unit) and the format reference (f). The unit reference is assumed to be either an integer constant or a PARAMETER variable. If a constant is found, it is simply ignored; if a variable is found, it is passed to TABS with class code 10 (PARAMETER variable), type code 0 (integer) and use code 26 (unit reference). The format reference is assumed to be the label of a FORMAT statement or the name of an array that contains FORMAT specifications. If an integer is

found, it is assumed to be a label and is passed to TABS with class code 5 (label), type code 5 (neutral), and use code 27 (format reference); if a variable is found, it is passed to TABS with class code 6 (variable), type code 5 (neutral), and use code 27 (format reference).

The argument list and I/O variable list are processed by subroutines IOARGS and IOLIST, respectively (see documentation for these routines).

SUBROUTINES FDIME

Subroutines FDIME is called from PARSER to process FORTRAN DIMENSION statements:

DIMENSION $v_1/\ell_1/v_2/\ell_2/ \ldots, v_n/\ell_n/$

where v_i represents a list of array declarations, and ℓ_i represents a literal list. Since the v_i and ℓ_i lists appear the same as those in type statements, they are processed by subroutine LISTP, the list processor for type statements. Thus the only function of FDIME is the extraction of the first variable in the list and setting of the use code (13 for use in DIMENSION statement). This information is passed to LISTP, which processes the remainder of the statement.

SUBROUTINE FCOMON

$$\text{COMMON}/X_1/a_1, a_2, \ldots, a_n/X_2/a_n+1, \ldots, a_m$$

where X_i represents a block name and a_i represents a variable or array name or array declaration. This form may appear somewhat different when variables are assigned to blank common; first, the string "/X_1/" may be omitted so that the list $a_1, a_2, \ldots, a_n$ is assigned to blank common, and secondly the appearance of two slashes with no block name between them assigns the variables which follow to blank common. Recognition of the various combinations of these forms is the major function of FCOMON. If blank common is being assigned, a block name consisting of all blanks is passed to subroutine TABS, indicating that the succeeding variables are assigned to blank common.

Each variable or array list ($a_1, a_2, \ldots, a_n$) is processed by subroutine LISTP. FCOMON extracts the first variable of a list, sets the use code (11 for common block entry) and passes control to LISTP, which processes the list and returns to FCOMON when a slash or end of statement is reached.

Common block names are recognized by FCOMON and passed to subroutine TABS with class code 7 (common block name) and use code 0 (declaration).

FORTRAN FRONT END DOCUMENTATION

General Description

There are three main tasks for the FORTRAN Front End; parsing, table generation and program graph generation. The main routines for each task are PARSER, TABS and TRANS respectively. The main routine for the entire Front End is PARSER which also drives TABS and TRANS.

PARSING

Parsing is done on a statement by statement basis. Since program modules analysed by FACES are assumed to be compilable without any syntax error, syntax checking is limited to aid information extraction. Statement types* are identified by matching keywords (e.g. PRIN for PRINT statement). Once the statement type is identified, special purpose parsing routine designed to accept the syntax for that particular statement type is called to extract and pass information to TRANS and TABS for making table entries.

Generation of System Tables

Each occurence of a logical symbol (name or label) is recorded in system tables by TABS (and related routines call by TABS).

TABS is called with the symbol name, the class, type and use code passed as parameters. Class code identifies the symbol type (label, variable, subroutine name etc.). Type code is assigned to variable or function name to designate variable type (integer, logical etc.). Use code determines how the symbol is referenced in the statement (subscript, DO-loop index etc.). Based on this information, TABS will make the appropriate entries in the system tables. Detailed descriptions of table

*Statement types are listed in Appendix D.

structures can be found in section III of User Mannual.

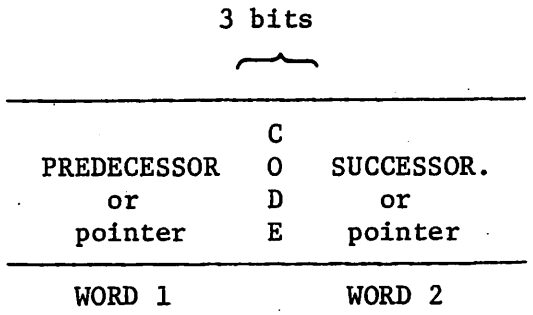
Program Graph Generation

The FACES system represents program structure as a directed graph with nodes representing program statements and edges representing lines of program flow. Each statement is treated as a node and is given a (pseudo) statement number as it is processed.

"Normal transitions" (i.e. the only successor of statement i is statement i+1) are not recorded in the transition table (TRIP). Therefore, if statement is is not found in the TRIP table as a predecessor, normal transition is assumed. Program transfers recognized by the parsing routines are recorded in the TRIP table as "non-normal transition pairs" by subroutine TRANS. "Non-normal transition pair" (i,j) represent permissable transition from statement i to statement j.

While program transfers are parsed, their destination (statement label) may be undefined at that point and i, j is recorded as undefined labels. Post-processing of TRIP table by PPTRIP will convert these undefined labels into statement numbers for the generation of NODTAB, SUCTAB, and PRETAB.

TRIP (2,1000) - (Transition Pair Table).



TRIP is a temporary local table generated by the Front End to record all program transfers.

Since the predecessor and successor may be undefined when a program transfer is parsed, the predecessor and successor field may contain pointers instead of statement numbers. The contents are identified by the code field. Post-processing will convert pointers into statement numbers for the generation of NODTAB, SUCTAB and PRETAB. The above configuration is a layout of the TRIP table before post-processing.

Code

0 predecessor field contains predecessor statement number. successor field contains a pointer to a label name in the symbol table.

1 predecessor field contains a pointer to a label name in the symbol table. successor field contains successor statement number.

2 predecessor and successor field are empty (zero). reserve space for DO-loop exit transition.

3 both predecessor and successor field contain statement numbers.

4 external subroutine reference

predecessor field is the predecessor statement number.

successor field is empty (zero).

Special codes are stored in the successor and predecessor field for transitions where the successor or predecessor does not exist

Predecessor

40,000	entry statement

Successor

20,000	RETURN or STOP statement.
30,000	END statement.

After post processing by PPTRIP, the code field is eliminated and all pointers are converted into statement numbers. The TRIP table printed by Front End is the TRIP table after post-processing.

SUBROUTINE FPRPU

Subroutine FPRPU is called by PARSTER to process FORTRAN PRINT and PUNCH statements.

```
PRINT  f, list
PUNCH  f, list
```

where f = format label.

list = a list of variables or arrays separated by commas.

The format label is extracted by FPRPU. The format label is assumed to be the label of a FORMAT statement or the name of an array that contains FORMAT specifications. If an integer is found, it is assumed to be a label and is passed to TABS with class code 5 (label), type code 5 (neutral), and use code 27 (format reference); if a variable is found, it is passed to TABS with class code 6 (variable), type code 5 (neutral), and use code 27 (format reference).

The list of variables to be printed or punched are processed by a call to subroutine IOLIST with IOF=3 to indicate this is a call from FPRPU.

SUBROUTINE LISTP (X, KTT, KUU).

Subroutine LISTP is a parsing routine which processes variable lists found in COMMON, DIMENSION and TYPE (INTEGER, REAL, etc.) statements.

X, KTT and KUU are input parameters. Variable from different statement types are identified by KUU:

KUU = 11 variable defined in a COMMON block
= 13 " " " " DIMENSION statement
= 14 " " " " TYPE statement

X is a three word array containing the first name in the list to be processed. KTT is the type code for the variables found in the list.

LISTP extracts the variable names in the list and pass the names to subroutine TABS for storage in the FACES tables. Dimensions of arrays are recognized and passed to TABS in array DIM (in common block TAPAR); the current version of FACES allows only three dimensions for arrays. Variables used as dimensions are recognized and stored with use code 15 (subscripts). Literal lists are currently ignored.

If a slash is encountered during parsing, one of the following actions will be taken:

i). If the variable list of a type statement is being processed, call TABS with class code 6 (variable), type code KTT and use code 12 (entry in a data statement) to indicate that the last variable or array has been initialized. Scan over the literal list and process the next variable list if there is one.

ii). If the variable list of a common block is being processed, return.

C. Common Block Description.

/BLANK/

SYMTAB (5,1800) - Symbol Table.

LSYM - length of SYMTAB.

NADDR - Contains the SYMTAB address in which the symbol processed is found or should be inserted if this is the first appearance of the symbol.

USETAB - Use Table.

LUSE - length of USETAB.

INDUT - pointer to the next entry of USETAB.

ALTAB - array length table.

LAL - length of ALTAB.

INDAL - pointer to the next entry of ALTAB.

DOTAB - DO Table.

LDO - length of DOTAB.

INDDO - pointer to the next entry of DOTAB.

IPRIME - prime number for SYMTAB hash coding.

NODTAB - Node Table.

LNOD - length of NOTAB.

SUCTAB - successor table.

ISUC - pointer to the last/next entry of ISUC.

PRETAB - predecessor table.

IPRE - pointer to the last/next entry of IPRE.

LPS - length of predecessor and successor table.

TRIP - TRIP Table.

LTRIP - length of TRIP table.

LOITR - pointer to the next available entry of TRIP.

STALTB - storage allocation table.

LSTAL - length of STALTB.

INDST - pointer to the next available space of STALTB.

/GLOBAL/

DIREC - directory

LDIR - length of DIREC.

INDIR - contains the hash coded DIREC address for name being processed.

CNTAB - common name table.

LCN - length of CNTAB.

INDCN - pointer to the last entry of CNTAB.

PARTAB - parameter table

LPAR - length of PARTAB

IPAR - pointer to the last used entry of PARTAB.

JPRIME - prime number for DIREC hash coding.

/DOTAB/

LABFLG - Flag indicating label has already appeared in a DO statement.
(Set by TABS and referenced by TRANS).

/INSTUF/

STORE (2,80) - circular input buffer, hold card image of statement being analyzed.

INP - pointer to the character being analyzed. STORE (2, INP).

L - pointer to the buffer being used, STORE (L, 80).

ISAV - pointer to starting position of an integer in STORE.

ND - number of digits in an integer in STORE.

LFLAG - indicates which line store buffer the present statement is in.

When LFLAG = 1, L = 1

LFLAG = -1, L = 2.

NLFLAG - = 1 indicates that a line of code has been read and that some routines have switched back to the previous card to process information.

= 0 otherwise

HFLAG - = 1 indicates that a hollerith constant may occur because a 1, =, or (has been found).

= 0 otherwise.

LBUF (10). - buffer holding first line of characters of a variable or integer that occur on one line and are continued on another.

/SCAPPA/

TSTAB (2,300). - temporary symbol table, containing alphanumeric strings.

ISS (600) - contains character codes to represent lexical entities extracted from Fortran Statements.

S statement label

V alphanumeric string

I	integer	constant
F	floating point	constant
D	double precision	constant
C	complex	constant
H	hollerith	constant
T	logical	constant
L	logical	operator
R	relational	operator

ITS - pointer to the next unused entry in TSTAB

INS - pointer to the last entry in ISS.

JFLAG - indicates that statement being scanned is the first statement of the input system (program) and card must be read in. Value is 0 for first card and 1 for all others.

EFLAG - indicates to parser that = has been found in statement. Value is the location of = in ISS if = is found, otherwise 0.

ENDFL = 1 if end of file found

= 0 otherwise.

/TABSUB/ TABS.

ISTP - marker to mark the starting location of STALTB in which variables defined within the currently processed common block is stored.

NSAV - contains statement number in which last symbol processed by TABS occurred.

LAN - contains SYMTAB address of last array name processed.

ISUBAD - contains the SYMTAB address of the subroutine or function name being processed

IPFLAG - = 1 a parameter list is being processed

= 0 otherwise

ICTYPE - contains type code (as used in scanning routine) of constant in a parameter list.

/TAPAR/ (TABS).

NAME (2) - stores name being processed.
4 chars./word.

STNO - statement number being processed.

DIM (3) - contains dimensions of an array.

MN - module number of the module being processed.

MPOP - special communication flag between parser and TABS. Normally MPOP = 0.
If MPOP = 1., a special call is being made to TABS to indicate a right paren is found in parameter list.
If MPOP = 2, a special call is being made to indicate a constant in a parameter list.

NFILE - file number of the module being processed.

ARGNO - contains argument number of variable or constant in a parameter list.

/TYPES/ TYRCHK.

LTRS (26) - each word corresponds to one letter of the alphabet and contains a 5 - 1 bit fields corresponding to the possible types of variables. The appropriate bit is set to 1 if an implicit type statement has so indicated.

bit set	
1	integer
2	real
3	double precision
4	complex
5	logical

NODTAB (3,1000) - (Node Table)

1/2 wd.	1/2 wd.	1/2 wd.	1/2 wd.	1/2 wd.	1/2 wd.
Statem't. Type Cd.	USETAB Ptr.	Success. Ptr.	No. of Success.	Pred. Ptr.	No. of Pred.
Word 1		Word 2		Word 3	

SUCTAB (1500)

1 wd.
Statement no.

PRETAB (1500)

1 wd.
Statement no.
Statement no.

NODTAB, SUCTAB, and PRETAB together serve to link each node (each statement is treated as a node) to its immediate successors and predecessors. The index of NODTAB identifies the statement number of a node. The statement type code for that statement is stored in word 1 along with a pointer to the beginning of the USETAB list for this statement. Word 2 of NODTAB contains a pointer to the list of immediate successors in SUCTAB and the number of successors the node has. Entries in SUCTAB (SUCCESSOR PTR) to SUCTAB (SUCCESSOR PTR + NUM of SUCCESSORS) points to statements that are successors of the node specified. Word 3 contains the same information for immediate predecessors of the node which are stored in PRETAB.

Special codes are stored in SUCTAB and PRETAB for certain transitions where the successor or predecessor does not exist.

SUCTAB

10,000	SUBROUTINE reference
20,000	RETURN or STOP statement
30,000	END statement
60,000	transfer to an undefined label
70,000	transfer to an undefined DO-LOOP label

PRETAB

40,000	ENTRY statement
50,000	PROGRAM, SUBROUTINE or FUNCTION entry point
70,000	Predecessor is an undefined DO-LOOP label
90,000	Predecessor not found

SUBROUTINE FDATA

Subroutine FDATA is called from PARSER to process FORTRAN DATA statements, which are assumed to have the general form

$$\text{DATA } v_1/\ell_1/v_2/\ell_2/, \ldots, /v_n/\ell_n/$$

$$\text{or DATA } v_1/\ell_1, v_2/\ell_2/, \ldots, v_n/\ell_n/$$

where v_i is a variable list and ℓ_i is a literal list. The variable lists are processed by subroutine IOLIST (IOLIST is called once for each variable list). Literal lists are simply ignored.

In order to interface with IOLIST, the string DATA stored in the first row of TSTAB is eliminated. The first data variable is shifted left to fill the empty space. When IOLIST is called, JTST and JISS (pointers associated with TSTAB and ISS resp.) point to the first data variable in the list.

For example: DATA NUMBER, ALPHA /1,2.0/

UPON ENTRY		BEFORE IOLIST IS CALLED	
TSTAB	ISS	TSTAB	ISS
DATA NUMB	V	NUMB ER	V
ER	V		V
ALPHA A	,	ALPHA A	,
	V		V
	/		/
	I		I
	,		,
	F		F
	/		/

SUBROUTINE FENTRY

Subroutine FENTRY is called from PARSER to process ENTRY statements, which have the form:

ENTRY ff(b_1,b_2,...,b_n)

where ff is the symbolic name of the entry point and b_1,b_2,...,b_n is a list of formal (dummy) parameters.

The entry point name is extracted from the symbol string in TSTAB by subroutine SHIFTY and passed to subroutine TABS with class code 9 (entry point name), type code 5 (neutral), and use code 0 (declaration).

Formal parameters are recognized by FENTRY and passed to TABS with class code 6 (variable), type code 5 (neutral) and use code 16, 17 (function or subroutine dummy parameters). A special call to TABS is made after the last name is processed with MPOP = 1 to indicate the end of the parameter list.

Finally a call is made from FENTRY to subroutine TRANS (with LOC=0 and MOD=7) to record a transition from an external point to the ENTRY statement.

SUBROUTINE FIF(PARAM)

Subroutine FIF is called from PARSER to process FORTRAN IF statements, which have one of the following general forms:

Arithmetic IF: IF (⟨arithmetic expression⟩) n_1,n_2,n_3

One Way Logical IF: IF (⟨ logical expression ⟩) ⟨statement⟩

Two Way Logical IF: IF (⟨ logical expression ⟩) n_1,n_2

where n_1,n_2,n_3 are statement numbers or switch variables, or are omitted.

The arithmetic or logical expression is processed by subroutine SEXP; the parameter (MODE) to SEXP is set to 20, indicating names in the expression are stored with use code 20 (conditional branch decision variable).

The distinction between an executable statement and statement labels following an IF expression is made by the presence or absence of an integer following the ")" in the above forms, i.e. n_1 is an integer. If n_1 is a switch variable, FIF will currently mistakenly recognize the statement as a logical IF. Since the occurrence was judged to be rare, and since a more involved recognition process is required to resolve the situation, the current version of FACES restricts n_1 to be an integer. However, the necessary modifications could be made if required by more frequent use of switch variables.

The transfer labels n_2 and n_3 may be statement numbers or switch variables, or may be omitted. If a transfer label is a statement number, it is passed to TABS with class code 5 (label), type code 5 (neutral), use code 10 (label reference). If a transfer label is a switch variable, it is passed to TABS with class code 6 (variable), type code 5 (neutral), and use code 29 (switch variable in IF statement).

If a logical IF statement is recognized, a call (with MOD=4) is made to subroutine TRANS to allow transitions in the program graph from the IF statement to the following two statements. Additionally, the value of parameter PARAM is set to 1 to indicate to PARSER that a logical IF was detected; PARSER then assigns a new number to executable statement S, effectively splitting the logical IF into two nodes in the program graph.

Output parameter PARAM remains 0 if an arithmetic IF is detected by FIF.

SUBROUTINE FIMP

This routine processes IMPLICIT type statements. A table (array LTRS) of twenty six entries is used to store the information in the IMPLICIT statement. Each entry in the table corresponds to one letter of the alphabet and each contains five one-bit fields:

bit set	
1	integer
2	real
3	double precision
4	complex
5	logical

When an IMPLICIT statement is encountered, FIMP scans the statement, determining the type specification and the letters to be implicitly typed. It then marks the appropriate fields in the LTRS table. For example, for the statement

IMPLICIT INTEGER (A-C)

the integer fields of the entries for A, B, and C in the LTRS table would be marked to indicate that the letters are implicitly typed integer.

SUBROUTINE FIRST

Subroutine FIRST is called from PARSER to perform a pre-scan of the first statement of a FORTRAN program module. Its purpose is to prepare the parser to parse the next module. Cards that are not module declaration statements (program, subroutine, function or block data statements) are ignored by FIRST.

Module identification is accomplished in the following manner. Subroutines, functions, and block data are recognized by the key words SUBROUTINE, FUNCTION, and BLOCK DATA in the first statement. Since a FUNCTION statement may also contain a type specification, a check is made for the key words INTEGER, LOGICAL, COMPLEX, DOUBLE PRECISION, REAL preceding FUNCTION.

If a subroutine, function, or block data module is recognized, no action is taken, since module identification information is assigned in FSUB, FUNCI, or PARSER for these module types. If none of these modules is recognized, the subsequent statements are ignored (skipped) until a module declaration statement is reached.

SUBROUTINES FREAL, FTYPE(KT)

Subroutines FREAL and FTYPE are called from PARSER to process FORTRAN type statements. The general form for type statements is

$$t\ v_1/\ell_1/v_2/\ell_2/\ \ldots\ /v_n/\ell_n/$$

where t stands for INTEGER, REAL, DOUBLE PRECISION, COMPLEX, or LOGICAL: v_i represents a list of variables, arrays (with or without dimensions), function names, and ENTRY names; ℓ_i represents a literal list, i.e., a list of constants which are to be assigned to the last variable or array in the variable list.

An additional function of FTYPE and FREAL is the recognition of statements of the form

$$\text{type FUNCTION f } (a_1, a_2, \ldots, a_n).$$

Thus a check is made for the presence of the character string "FUNCTION" and, if found, the function name (f) is extracted and passed to subroutine FUNCI for processing of the remainder of the statement.

Subroutine LISTP is called by FREAL and FTYPE to process variable and literal lists, i.e. that portion of the statement following the type (t) in the first form.

The various type statements are handled in the following manner. PARSER recognizes the keyword of particular type statement (INTE, REAL, DOUB, COMP), sets the type code (KT) accordingly, and passes control to subroutine FTYPE. Since there is no break character between the type specification and the first name in the list (e.g. INTEGER X appears as one continuous character string in array TSTAB), FTYPE extracts the first name (using subroutine

SHIFTY) and passes it, along with the type code and use code (14 for type statement entry), to LISTP for processing of the remainder of the statement. REAL statements are an exception, since the 4 characters of "REAL" are stored in exactly one word of TSTAB: consequently, no shifting is required to extract the first list item, and REAL statements are processed be FREAL and LISTP rather than FTYPE.

FUNCTION HASHIN(N)

Function HASHIN computes a hash index based on the sum of the third and fourth characters of a given character string. The character string is contained in input parameter N, 4 characters per word, left-justified.

The formula for the hash index is

HASHIN = (3^{rd} char of N + 4^{th} char of N + 10

The value returned is used by PARSER to index the table of key words (MATCH, Appendix C).

The value returned depends on the internal character representation of the machine.

SUBROUTINE IOARGS

Subroutine IOARGS is called from FREAD and FWRITE to process the argument list for READ and WRITE statements. The form for the argument list (see documentation for FREAD, FWRITE, FPRPU) is the following:

(unit, F, ERR=ℓ_1, END=ℓ_2) where unit = I/O unit number

F = format label

ℓ_1 = next statement for execution if error detected in the I/O operation

ℓ_2 = next statement for execution if EOF

The unit reference (unit) is assumed to always be present while any combination of the remaining arguments may be omitted in the I/O statement.

The unit reference is assumed to be either an integer constant or a PARAMETER variable. If a constant is found, it is simply ignored; if a variable is found, it is passed to TABS with class code 10 (PARAMETER variable), type code 0 (integer), and use code 26 (unit reference).

The format reference is assumed to be the label of a FORMAT statement or the name of an array that contains FORMAT specifications. If an integer is found, it is assumed to be a label and is passed to TABS with class code 5 (label), type code 5 (neutral), and use code 27 (format reference); if a variable is found, it is passed to TABS with class code 6 (variable), type code 5, and use code 27. Reference to a namelist name is not currently recognized, since FACES does not currently recognize NAMELIST statements.

For the arguments ERR=ℓ_1 and ERR=ℓ_2, ℓ_1 and ℓ_2 represent labels of statements to which control can be transferred. Thus the label names for ℓ_1 and ℓ_2 are passed to TABS with use code 10 (label reference), type code 5 (neutral), and class code 5 (label).

Control is returned from IOARGS to FREAD or FWRITE upon recognition of the ")" representing the end of the argument list.

SUBROUTINE IOLIST(IOF)

Subroutine IOLIST is called from an I/O statement processor or the DATA statement processor to process I/O variable lists. Input parameter IOF is a use code indicator which may have the following values:

IOF = 3 If called from FWRITE or FPRPU

Indicates names in list are external output variables

IOF = 4 if called from FREAD

Indicates names in list are external input variables

IOF = 12 if called from FDATA

Indicates names in list are assigned data in a

DATA statement

List elements are assumed to be subscripted or unsubscripted variable, array names, or implied DO-loops.

Since nesting is allowed in the I/O list (e.g. implied DO Loops), a use code stack (STACK) is maintained to record the use code for names found at the various levels of nesting. The initial entry in the stack is the value of IOF (namely, 3, 4, or 12 as described above), since names at the highest level are list elements. When a (is found after a variable name, the STACK is pushed and STACK (TOP) = 15 (use code for subscripts). If a (is found and is not after a variable name, then the STACK is pushed and STACK (TOP) = STACK (TOP-1). The variable PC is used as the index to STACK as well as representing the parentheses count. Each variable name found in the I/O list is passed to subroutine TABS for storage in the FACES table.

The following example illustrate the use of the use code stack (STACK) and PC. Consider the following PRINT statement.

PRINT 10, (((ARRAY(L,M,N), L=1, 10), M = 1, 10), N = 1, 10).

When the first L is reached.

STACK (1) = 3 initialize to value of IOF

STACK (2) = 3

STACK (3) = 3

STACK (4) = 3

STACK (5) = 15

PC = 5

When the second L is reached

STACK (1) = 3

STACK (2) = 3

STACK (3) = 3

STACK (4) = 3

PC = 4

Implied DO-loops are assumed to be of the form $(L_j, i=m_1,m_2,m_3)$, where L_j is an I/O list element and $i=m_1,m_2,m_3$ is the same as in a DO statement (see documentation for FDO, DORANG). Identification depends on the recognition of "=" following a simple variable name. When this condition occurs, the variable represented by i is passed to subroutine TABS with class code 6 (variable), type code 0 (integer), use code 5 (DO-loop index); The FACES assigned number for the I/O statement is then stored in the DO-loop table (DOTAB) and subroutine DORANG is called to process the string m_1,m_2,m_3.

Exit from subroutine IOLIST occurs upon recognition of the end of the I/O statement (if called from FREAD, FWRITE, FPRPU) or a slash (/) (if called from FDATA).

Subroutine SCAN

The lexical scanner of the FORTRAN Analysis Package scans FORTRAN statements character by character and organizes the general format of the statements in a coded form to be interpreted by the parsing routine. The scanner recognizes entities in a statement and forms a list of alphabetic codes for the entities in an array called ISS (Intermediate Symbol String). The entities that are recognized and their codes are as follows:

statement label	S
alphanumeric string	V
integer constant	I
floating point constant	F
double precision constant	D
complex constant	C
hollerith constant	H
logical constant	T
logical operator	L
relational operator	R

All special characters such as "=", "+", "-", etc., are represented in ISS by their respective character codes.

The scanner also stores all alphanumeric strings and integer constants in a table called TSTAB, a 2 by 300 word array. Alphanumeric strings are stored in blocks of eight characters or less. That is, if a string contains eight or less characters, it will be stored 4 characters per word (left-justified, blank filled) in two words of TSTAB. If a string contains more than eight characters, each block of eight characters will be stored in

TSTAB and the corresponding code will be entered into ISS.

The following examples will illustrate the results of some calls to SCAN.

Example 1

Source statement:

SUM = SUM + IFUNC1 (10, M).

TSTAB (1, N)	TSTAB (2, N)
SUM	
SUM	
IFUN	C1
10	
I	

ISS: V = V + V (I, V)

Example 2

INTEGER FUNCTION INVERT (INPUT 1, INPUT 2)

TSTAB (1, N)	TSTAB (2, N)
INTE	GERF
UNCT	IONI
NVER	T
1 NPU	T1
1 NPU	T2

ISS: V V V (V, V)

Example 3

```
20  IF (.NOT. (A.EQ.2HON))  GO TO 10.
```

TSTAB (1, N)	TSTAB (2, N)
20	
IF	
A	
GOTO	10

ISS: S V (L(V R H))V

SUBROUTINE SEXP

SEXP is a generalized parsing routine designed to process subscript (or parameter) lists and expressions (arithmetic and relational) when encountered by one of the parsing routines. The list or expression may be delimited by the end of the statement or a ")". Since parentheses may be nested within the list or expression, a parentheses count (PC) is kept to identify the current level of nesting. PC is initialized to 1, incremented for each "(", and decremented for each ")". Thus, if PC reaches 0, the ")" is the terminator of the list or expression.

Information passed to SEXP includes the common blocks SCAPA and PNTRS containing the ISS and TSTAB arrays and current pointers (JISS, JTST) to these arrays.

MODE specify the type of list for processing:

= 2 process exp in assignment st

= 15 process subscript exp

= 19 process subroutine actual parameters

= 20 process conditional branch decision variables

Upon entry to SEXP, the pointers point to the first symbol and character string of the list or expression. Upon return, if no end of statement condition is detected, JISS points to the ")" in ISS and JTST points to the first character string (in TSTAB) following the list or expression.

In order to give SEXP recursive properties (i.e. process lists (or expressions) within lists (or expressions)), a stack (STACK) is maintained to record the type of list or expression at each level of nesting. The i^{th} entry in STACK is actually the use code (passed to subroutine TABS) associated with names at the i^{th} level of nesting, and

the index to STACK is PC, the parentheses count. The initial entry in STACK is set to the value of MODE, the input parameter to SEXP.

The occurrence of a subroutine or function name with parameter list requires special handling by SEXP. In order to effect the proper storage of parameters in the system parameter table (PARTAB), the position number (stored in ARGNO) for each parameter is passed to subroutine TABS through common block TAPAR. Additionally, since nesting may be allowed, a separate argument stack (ARGSTK) is maintained to record the current argument (parameter) number when a new argument list is encountered. Accordingly, each occurrence of a function or subroutine name followed by an argument list causes the current argument number (ARGNO) to be placed on the argument stack (ARGSTK) and ARGNO to be reset to 1. The argument number is incremented when a "," is encountered within an argument list.

The following example illustrates the use of the use code stack (STACK) and the argument stack (ARGSTK). Consider the following assignment statement.

```
VAR1 = FUN1 (I, 100, FUN2 (X, Z(L)), VAR3) + VAR2.
```

PARSER calls FARITH which in turn calls SEXP with MODE = 2 to process the expression to the right of the equal sign. While SEXP is processing this statement, the following events take place:

```
STACK (1) = 2        initialized to value of MODE
PC        = 1        parenthesis count
ASI       = 0        pointer to ARGSTK
ARGNO     = 0
```

```
FUN1:   Store FUN1 as a function name in tables
(   :   increment PC (PC = 2)
        increment ASI (ASI = 1)
        STACK (2) = 18
        ARGSTK (1) = ARGNO = 0
        reset ARGNO to one (1)
I   :   store I as a function actual parameter in table
        use code = STACK(PC) = 18
,   :   increment ARGNO  (ARGNO = 2)
100 :   Store 100 as a constant parameter in table
,   :   increment ARGNO  (ARGNO = 3)
FUN2:   Store FUN2 as a function name in tables
(   :   increment PC  (PC = 3)
        STACK (3) = 18
        ASI = 2
        ARGSTK (2) = ARGNO = 3
        reset ARGNO to 1 (one)
X   :   Store X in table as a function actual parameter
,   :   increment ARGNO (ARGNO = 2)
Z   :   Store Z as an array in tables
(   :   increment PC  (PC = 4)
        STACK (4) = 15 (subscripts)
L   :   Store L as a subscript in table
)   :   Since STACK(PC) = 15 (subscripts), pop use code stack
        PC = 3
)   :   Since STACK(PC) = 18 (function actual parameter), this is
        the end of a parameter list.  Restore the argument number of
```

previous parameter list.

ARGNO = ARGSTK(ASI) = 3

ASI = ASI - 1 = 1.

Pop use code stack PC = 2

, : increment argument number (ARGNO = 4)

VAR3: Store VAR3 as a function actual parameter in tables

) : Since STACK(PC) = 18 (function actual parameter), this is the end of a parameter list.

Restore the argument number of previous list.

ARGNO = ARGSTK(ASI) = 0

ASI = ASI - 1 = 0

Pop use code stack. PC = 1

+ : no action

VAR2: Store VAR2 as an input variable

Use code = STACK(PC) = 2.

RETURN.

One further note concerning parameter lists should be made. At the end of a parameter list a special call to TABS is made (with MPOP = 1) to alert TABS of this occurrence. Additionally, a call to TABS (with MPOP = 2) is made when a constant is found as a parameter. This differs from the usual practice of ignoring constants in expressions.

SUBROUTINE TRANS (LOC, MOD)

Subroutine TRANS is called by the various parser subroutines to make entries in the non-normal transition pairs table, TRIP. The TRIP table stores program control transfers other than the standard fall through transitions. The TRIP table records are each two words long as shown below.

PREDECESSOR	CODE	SUCCESSOR
1 WORD	3 BITS	1 WORD - 3 BITS

TRIP Table Record

The predecessor field will contain the statement number of the originating statement of the transition. The successor field will contain the statement number of the target statement. The three bit code field is used temporarily by TRANS and subroutine PPTRIP to identify the type of record stored.

TRANS creates TRIP records for six transition types. The type of transition, and hence the mode of operation of TRANS, is indicated by the input argument, MOD. The values of MOD and the corresponding transitions are given below.

MOD Cause of transition

1 - Explicit transfer to label

2 - Label referenced in DO statement

3 - Terminal statement (STOP, RETURN)

4 - One Way Logical IF statement

5 - External subroutine reference

6 - END statement.

In the case of references to labels, the input argument LOC contains a pointer to the symbol table entry for the appropriate label. The common variable STNO contains the value of the current statement number. There follows a brief description of the actions taken by TRANS for the possible values of MOD:

If MOD equals 1: Condition indicates transfer to statement label indicated by LOC. predecessor = STNO, CODE = 0, Successor = LOC.

If MOD equals 2: Condition indicates label referenced in a DO statement. predecessor = LOC, CODE = 1, Successor = STNO. Second TRIP table entry is made with CODE = 2 and statement successor left undefined.

If MOD equals 3: Condition indicates a RETURN or STOP statement. predecessor = STNO, CODE = 3, successor = 20,000.

If MOD equals 4: Condition indicates a one way logical IF statement. Two TRIP table entries made. First: predecessor = STNO, CODE = 3, successor = STNO + 1; predecessor = STNO, CODE = 3, successor = STNO + 2.

If MOD = 5: Condition indicates external subroutine reference. Two entries made. First: predecessor = STNO, CODE = 4, successor = LOC = 0 Second: predecessor = STNO, CODE = 3, successor = STNO + 1.

If MOD = 6: Condition indicates end statement.

predecessor = STNO, CODE = 3, successor = 30,000.

If MOD = 7: Condition indicates program entry point.

predecessor = 40,000, CODE = 3, successor = STNO.

Subroutine UPCNT (NAM1, NAM2, NADNO)

This routine makes an entry in CNTAB (Common Name Table) when ever a common block is declared. NAM1 and NAM2 contain the name of the common block (four characters/word) and MODNO contains the module number in which the common block is being declared. UPCNT also stores the STALTB top pointer in CNTAB. STALTB top pointer points to the starting location of STALTB where the common block entries will be stored.

Subroutine UPDIR (NAM1, NAM2, NF, MODNO)

This routine updates DIREC (global directory) each time a subroutine, function, or program name is encountered. The hash code* is calculated using the module name (stored in NAM1 and NAM2) and the name is entered in DIREC if it is not already there. If the name is already in the directory and it is now being defined (NF $\neq$ 0), the file number (NF) and module number (MODNO) are changed. (When a module is undefined, its module number is the number of the module in which the routine was called.)

*The hash code formula is:

MOD ((NAM1 + NAM2)/2, JPRIME) + 1

where NAM1 and NAM2 has been converted into R format (right justified, zero filled).

JPRIME is the greatest prime number smaller than the length of the Directory table (JPRIME is initialized in the BLOCK DATA program).

Subroutine UPPAR (MCOD, MODNO, US, IAN)

This routine makes all entries into PARTAB (Parameter table). Depending upon the value of input parameter MCOD, UPPAR does one of the following things:

(a) MCOD = 0. This is an actual or dummy parameter (but not a constant), store the SYMTAB address of this parameter in PARTAB.

(b) MCOD = 1. Makes end-of-list entry in PARTAB.

(c) MCOD = 2. Stores code for constant parameter and its type code in PARTAB.

When a parameter is to be stored (MCOD $\neq$ 1), a check is also made to determine whether this is the first parameter list for the corresponding subroutine or function. This involves checking the SYMTAB entry for the routine name to see if the PARTAB pointer is 0. If it is, the pointer to PARTAB is stored in SYMTAB. If this is not the first parameter list for this routine, the list of parameter pointers in PARTAB that correspond to the routine is traversed until the end-of-list code is found. The end-of-list code is changed to be a pointer to the next entry in PARTAB, and then the new list begins.

The SYMTAB address for the parameter being processed is stored in PARTAB. The parameter may be an actual parameter, a dummy parameter or a subroutine or function name to which the parameter list corresponds. From the use code, the CODE field of PARTAB is set accordingly.

CODE	
0	pointer to dummy parameter in SYMTAB
1	pointer to actual parameter in SYMTAB
2	pointer to parameter list for next call of same routine
3	pointer to subroutine or function name in SYMTAB
4	this word marks end of current parameter list
5	this is a constant parameter, pointer field contains constant type

Subroutine UPSTAL (IDIM, ICHANG)

This routine updates STALTB (Storage Allocation Table) whenever a common variable is declared or the dimensions of a previously declared common variable are changed by a DIMENSION statement. When input parameter ICHANG is zero, an entry is made in STALTB which contains a pointer to the common variable name in SYMTAB and the size of the array variable. If ICHANG $\neq$ 0, then the dimension field of STALTB of the entry pointed to by ICHANG is changed and the subsequent starting locations for variables in that common block are updated. In either case, input parameter IDIM contains the size of the array variable.

Whenever a new entry is added to STALTB, the STALTB bottom pointer in CNTAB and the STALTB pointer of the variable in SYMTAB has to be updated.

Subroutine UPUSE (IUSE, ISTNO)

This routine updates USETAB (Variable Usage Table) each time a name is found in the source program. Each entry in USETAB is linked to the previous entry and next entry for that name. The first entry for a name is linked to the name in SYMTAB, and the last entry has no forward pointer. Each entry in the table consists of the statement number in which the name occurs (input parameter ISTNO) and the use code for the name in that statement (input parameter IUSE).

UPUSE also update the USETAB bottom pointer in SYMTAB when a new entry is made in USETAB.

FORTRAN FRONT END SOURCE LISTING.

```
*DECK ALPHA
      LOGICAL FUNCTION ALPHA(CHAR)
C*****************************************************************C
C  FUNCTION ALPHA DETERMINES WHETHER THE INPUT PARAMETER CHAR IS A LETTER A--Z.
C  IF IT IS A LETTER, THE VALUE .TRUE. IS RETURNED.  OTHERWISE, .FALSE. IS    C
C  RETURNED.                                                                  C
C  CHAR HOLDS AN ALPHANUMERIC CHARACTER IN A1 FORMAT.                         C
C  THIS FUNCTION DEPENDS ON THE INTERNAL CHARACTER REPRESENTATION OF          C
C  THE MACHINE USED.                                                          C
C*****************************************************************C
      INTEGER CHAR,FLD
      ITEMP=0
      ITEMP=FLD(0,6,CHAR)
      IF((ITEMP.LT.1).OR.(ITEMP.GT.26)) GO TO 5
      ALPHA=.TRUE.
      RETURN
5     ALPHA=.FALSE.
      RETURN
      END
*DECK BITRET
      SUBROUTINE BITRET(IWD,ICD,IPM)
C*****************************************************************C
C  SUBROUTINE BITRET MANIPULATES THE LEFT-MOST 2 BITS OF THE RIGHT            C
C  HALF-WORD OF IWD.                                                          C
C  ICD (OUTPUT) IS SET TO THE VALUE OF THE FIRST BIT (0 OR 1)                 C
C  IMP (OUTPUT) IS SET TO PLUS (+) OR MINUS (-) (A1 FORMAT), DEPENDING        C
C  ON THE VALUE OF THE SECOND BIT (0, 1 RESP.)                                C
C  IWD (INPUT-OUTPUT) IS SET TO THE RIGHT HALF WORD OF IWD WITH ZEROS         C
C  IN THE FIRST TWO BITS ON RETURN.                                           C
C  THIS SUBROUTINE DEPENDS ON THE WORD LENGTH OF THE MACHINE USED.            C
C*****************************************************************C
      IMPLICIT INTEGER(A-Z)
      DATA HPLUS,HMINUS/1H+,1H-/
      ICD=FLD(30,1,IWD)
      IB2=FLD(31,1,IWD)
      IPM=HPLUS
      IF(IB2.NE.0)IPM=HMINUS
      IWD=FLD(32,28,IWD)
      RETURN
      END
*DECK BITSET
      INTEGER FUNCTION BITSET(I,KB1,KB2)
C*****************************************************************C
C  FUNCTION BITSET SETS ONE-BIT INDICATORS IN THE PARAMETER VARIABLE I.C
C  IF KB1=1, THE LEFT-MOST BIT OF THE RIGHT HALF-WORD OF I IS SET TO 1.C
C  IF KB2=1, THE SECOND BIT OF THE RIGHT HALF-WORD OF I IS SET TO 1.  C
C                                                                     C
C  BITSET DEPENDS ON THE WORD LENGTH OF THE MACHINE USED.             C
C*****************************************************************C
      J=I
      IF(KB1.EQ.1) J=J.OR.400000000000B
      IF(KB2.EQ.1)J=J.OR.200000000000B
      BITSET=J
      RETURN
      END
*DECK BLKDATA
```

```
      BLOCK DATA
      IMPLICIT INTEGER(A-Z)
C*****************************************************************
C     BLOCK DATA INITIALIZS THE FOLLOWING:                       C
C  COMMON BLOCK /TMATCH/ : EACH ELEMENT OF MATCH CONTAINS KEYWORD C
C                          FOR CLASSIFYING FORTRAN STATEMENTS     C
C     COMMON BLOCK /H/ : EACH ELEMENT OF H CONTAINS AN ALPHANUMERIC
C                        CHARACTER                                C
C     INITIALIZE GLOBAL TABLES AND POINTERS                      C
C         DIREC  = DIRECTORY                                     C
C         LDIR   = LENGTH OF DIRECTORY.                          C
C         INDIR  = POINTS TO THE FIRST UNUSED ENTRY IN DIREC.    C
C         CNTAB  = COMMON NAME TABLE                             C
C         LCN    = LENGTH OF COMMON NAME TABLE                   C
C         INDCN  = POINTS TO THE FIRST UNUSED ENTRY IN CNTAB     C
C         PARTAB = PARAMETER TABLE                               C
C         LPAR   = LENGTH OF PARTAB.                             C
C         IPAR   = POINTS TO THE FIRST UNUSED ENTRY IN PARTAB.   C
C         JPRIME = PRIME NUMBER FOR THE HASH CODED DIREC.        C
C*****************************************************************
      COMMON/TMATCH/BIAS,MATCH(51)
      DATA BIAS/27,MATCH/     0,     0,4HDOEM,4HPRIN,4HIMPL,4HNAME,
     1    4HENCO,4HDECO,4HPUNC,4HIF  ,4HCOMP,4HEXTE,4HELSC,4HEND ,
     2    4HREAD,     0,     0,     0,     0,4HENDF,     0,     0,
     3    4HREAL,4HBACK,     0,4HLOGI,4HFUNC,4HDIME,     0,4HSUBR,
     4    4HDATA,4HPROG,4HDOUB,4HCALL,4HINTE,4HCOMM,     0,4HASSI,
     5    4HWRIT,4HEQUI,4HSTOP,4HREWI,     0,4HCONT,4HGOTO,     0,
     6          0,4HENTR,     0,4HPAUS,4HRETU/
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      COMMON/INSTUF/STORE(2,80),INP,L,ISAV,ND,LFLAG,NLFLAG,HFLAG
     1,LBUF(10)
      COMMON/TABSUB/ISTP,NSAV,LAN,ISUBAD,IPFLAG,ICTYPE
      DATA HBL,HA,HB,HC,HD,HE,HF,HG,HH,HI,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,
     1 HT,HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT/1H ,1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,
     3 1HI,1HJ,1HK,1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,
     4 1HY,1HZ,1H),1H=,1H*,1H(,1H,,1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,
     5 1H9,1H ,1H/,1H.,4HFORM,2HAT/
      DATA LBUF/10*6H      /
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      DATA JFLAG/0/,ENDFL/0/
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
      COMMON/GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAR, IPAR, JPRIME
      DATA LDIR, LCN, LPAR /100,300, 2000/, DIREC /300*0/,
     C INDCN, IPAR/1, 2/, JPRIME /97/, CNTAB /1200*0/, PARTAB /2000*0/
      END
*DECK INIT
      PROGRAM INIT (INPUT, OUTPUT, TAPE1 = INPUT, FACES)
C*****************************************************************
C     THIS ROUTINE INITIALIZE THE LENTH OF LOCAL TABLES AND CALLS C
```

```
C     PARSER TO PROCESS THE INPUT PROGRAMS.
C
C          LSYM   = LENGTH OF SYMTAB.
C          LUSE   = LENGTH OF USETAB
C          LAL    = LENGTH OF ALTAB
C          LDO    = LENGTH OF DOTAB
C          LSTAL  = LENGTH OF STALTB
C          LNOD   = LENGTH OF DOTAB
C          LTRIP  = LENGTH OF TRIP
C          LPS    = LENGTH OF PREDECESSOR AND SUCCESSOR TABLE.
C          IPRIME = PRIME NUMBER FOR THE HASH CODED SYMTAB.
C*****************************************************************
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     C ALTAB (56), LAL, INDAL, DOTAB (3,50), LDO, INDDO, IPRIME,
     C NODTAB (3,1000), LNOD, SUCTAB (1500), ISUC, PRETAB (1500), IPRE,
     C LPS, TRIP (1000,2), LTRIP, ITP, STALTB (2,100), LSTAL, INDST
      LSYM = 1800
      LUSE = 2500
      LAL = 56
      LDO = 50
      LSTAL = 100
      INDDO = 0
      IPRIME = 1799
      LNOD = 1000
      LTRIP = 1000
      LPS = 1500
      CALL PARSER
      END
*DECK BUFF
      SUBROUTINE BUFF(ISTART)
      IMPLICIT INTEGER (A-Z)
C*****************************************************************
C  SUBROUTINE BUFF STORES AN ALPHANUMERIC STRING FROM THE END OF A CARD INTO
C  THE TEMPORARY BUFFER LBUF SO THAT THE NEXT CARD MAY BE READ BY SCAN WITHOUT
C  LOSING THE STRING.  THE STRING IS STORED ONE CHARACTER PER WORD AND AT
C  MOST 10 CHARACTERS MAY BE STORED.  BLANKS ARE NOT STORED.
C  PARAMETER
C  ISTART-(INPUT)--POINTER TO STARTING POSITION OF STRING IN STORE.
C  COMMON VARIABLES
C  SEE SUBROUTINE SCAN.
C  LOCAL VARIABLES
C  INDEX--POINTER TO CHARACTER IN STORE
C  NL--LINE OF STORE
C*****************************************************************
      COMMON/INSTUF/STORE(2,80),INP,L,ISAV,ND,LFLAG,NLFLAG,HFLAG
     1,LBUF(10)
      COMMON/H/H8L,H4,H8,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
C*****************************************************************
C  GO BACK TO PREVIOUS CARD.
C*****************************************************************
      IF(L.EQ.1) NL=2
      IF(L.EQ.2) NL=1
C*****************************************************************
C  STORE CHARACTERS IN LBUF.
```

```
C*************************************************************************C
      K=0
      DO 5 I=1,60
      INDEX=ISTART-1+I
C*************************************************************************C
C  IGNORE 73 TO 80 COLUMNS.                                               C
C  IGNORE BLANK COLUMNS                                                   C
C*************************************************************************C
      IF(INDEX.GT.72) GO TO 10
      IF(STORE(NL,INDEX) .EQ.HBL) GO TO 5
      K=K+1
      IF(K.LE.10) GO TO 3
      PRINT 2
2     FORMAT(* FATAL ERROR:  ATTEMPT TO OVERSTORE TEMPORARY BUFFER IN=,
     1 * SCANNING ROUTINE.*)
      STOP
3     LBUF(K)=STORE(NL,INDEX)
5     CONTINUE
10    RETURN
      END
*DECK COMPLX
      SUBROUTINE COMPLX(B)
      IMPLICIT INTEGER(A-Z)
C*************************************************************************C
C  SUBROUTINE COMPLX IS CALLED BY SCAN TO CHECK FOR A COMPLEX CONSTANT WHEN  C
C  A LEFT PARENTHESIS IS FOUND.  UPON ENTRY, INP IS POINTING TO THE LEFT     C
C  PAREN. IN STORE.  IF A COMPLEX CONSTANT IS FOUND, INP WILL BE POINTING TO C
C  THE RIGHT PAREN. AT THE END OF THE CONSTANT, AND B WILL HAVE THE VALUE    C
C  .TRUE..  IF A COMPLEX CONSTANT IS NOT FOUND, INP WILL BE POINTING TO THE  C
C  FIRST LEFT PARENTHESIS AND B WILL BE .FALSE..                             C
C  PARAMETER                                                                 C
C  B-(OUTPUT)--LOGICAL VARIABLE INDICATING WHETHER A COMPLEX CONSTANT IS FOUND C
C  GLOBAL VARIABLE :                                                         C
C  NLFLAG - (OUTPUT) -- SET TO 1 IF A NEW CARD HAS BEEN READ AND SOME        C
C                       ROUTINES HAVE SWITCHED BACK TO THE PREVIOUS          C
C                       CARD FOR PROCESSING                                  C
C  LOCAL VARIABLES                                                           C
C  NFLAG--VALUE SET TO 1 WHEN NEW CARD IS READ                               C
C  ISAVP--POINTER TO POSITION OF LEFT PAREN. IN STORE ARRAY.                 C
C*************************************************************************C
      COMMON/INSTUF/STORE(2,80),INP,L,ISAV,ND,LFLAG,NLFLAG,HFLAG
     1,LBUF(10)
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HCU,HSL,HPE,HFORM,HAT,HG,HH,HI
      LOGICAL B
      NFLAG=0
C*************************************************************************C
C  INP POINTS TO THE LEFT PAREN. IN STORE                                    C
C  SAVE POSITION OF LEFT PARENTHESIS.                                        C
C*************************************************************************C
      ISAVP=INP
C*************************************************************************C
C  INCREMENT POINTER AND CHECK FOR NEW CARD.                                 C
C*************************************************************************C
5     INP=INP+1
```

```
      IF(INP.LE.72) GO TO 7
      CALL NEWCAR (B)
      INP=7
      NFLAG=1
C*****************************************************************************C
C  CHECK FOR PAREN. OR COMMA.                                                  C
C*****************************************************************************C
7     IF(STORE(L,INP).EQ.HLP) GO TO 20
      IF(STORE(L,INP).EQ.HPP) GO TO 20
      IF(STORE(L,INP).NE.HCO) GO TO 5
C*****************************************************************************C
C  COMMA FOUND.  SCAN TO RIGHT PAREN. AND RETURN .TRUE..                       C
C*****************************************************************************C
10    INP=INP+1
      IF(INP.LE.72) GO TO 15
      CALL NEWCAR(B)
      INP=7
15    IF(STORE(L,INP).NE.HPP) GO TO 10
      B=.TRUE.
      RETURN
C*****************************************************************************C
C  NOT A COMPLEX CONSTANT. RESET POINTER AND SET B TO  .FALSE.                 C
C*****************************************************************************C
20    INP=ISAVP
      B=.FALSE.
C*****************************************************************************C
C  IF A NEW CARD HAS BEEN READ, SWITCH BACK TO THE PREVIOUS CARD AND           C
C  SET INDICATOR NLFLAG TO 1.                                                  C
C*****************************************************************************C
      IF(NFLAG.EQ.0) GO TO 30
      NLFLAG=1
      IF(L.EQ.1) GO TO 25
      L=1
      RETURN
25    L=2
30    RETURN
      END
*DECK CONS
      SUBROUTINE CONS(IP)
      IMPLICIT INTEGER(A-Z)
C*****************************************************************************C
C  SUBROUTINE CONS IS CALLED BY SCAN TO CHECK FOR INTEGER, FLOATING POINT,     C
C  DOUBLE PRECISION, HOLLERITH, AND LOGICAL CONSTANTS AND FOR LOGICAL AND      C
C  RELATIONAL OPERATORS.  UPON ENTRY TO CONS, THE INP POINTER SHOULD BE        C
C  POINTING TO A DIGIT OR PERIOD IN THE STORE BUFFER.  IF NO CONSTANT OR       C
C  OPERATOR IS DETECTED, THE INP REMAINS POINTING TO THE INITIAL CHARACTER     C
C  AND IP HAS VALUE 0.  IF A CONSTANT OR OPERATOR IS DETECTED, INP POINTS TO   C
C  THE LAST CHARACTER IN THE CONSTANT OR OPERATOR, AND IP WILL BE SET TO       C
C  ONE OF THE VALUES LISTED BELOW:                                             C
C          1  INTEGER                                                          C
C          2  FLOATING POINT                                                   C
C          3  DOUBLE PRECISION                                                 C
C          4  HOLLERITH CONSTANT                                               C
C          5  LOGICAL OPERATOR                                                 C
C          6  LOGICAL CONSTANT                                                 C
C          9  RELATIONAL OPERATOR                                              C
```

```
C  COMMON VARIABLES                                                    C
C  SEE SUBROUTINE SCAN.                                               C
C  LOCAL VARIABLES                                                    C
C  FPFLAG--VALUE 1 FOR FLOATING POINT CONSTANT, VALUE 0 OTHERWISE     C
C  PFLAG--VALUE 1 WHEN DECIMAL ENCOUNTERED. VALUE 0 OTHERWISE         C
C  DFLAG--VALUE 1 WHEN DOUBLE PRECISION CONSTANT FOUND, VALUE 0 OTHERWISE  C
C  ISPER--POINTER TO LOCATION OF PERIOD IN STORE BUFFER               C
C*****************************************************************C
      COMMON/INSTUF/STORE(2,80),INP,L,ISAV,ND,LFLAG,NLFLAG,HFLAG
     1,LBUF(10)
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HS,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,H$P,H$O,H$S,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,H$,HH,HI
      LOGICAL ALPHA,DIGIT,B
      DFLAG=0
      FPFLAG=0
C*****************************************************************C
C  CHECK FOR DIGIT.                                                   C
C*****************************************************************C
      ICHAR=STORE(L,INP)
      IF(.NOT.(DIGIT(ICHAR))) GO TO 50
C*****************************************************************C
C  DIGIT FOUND, MARK STARTING POSITION OF INTEGER                     C
C*****************************************************************C
      ISAV=INP
      ND=1
5     PFLAG=0
C*****************************************************************C
C  SCAN UNTIL NON-DIGIT FOUND.                                        C
C*****************************************************************C
7     INP=INP+1
      IF(INP.LE.72) GO TO 10
      CALL NEWCAR (B)
      IF(.NOT.B) GO TO 8
      NLFLAG=1
      GO TO 40
C*****************************************************************C
C  END OF CARD FOUND.  STORE DIGITS IN BUFFER IF INTEGER.             C
C*****************************************************************C
8     ISAVX=ISAV
      IF(FPFLAG.EQ.0) CALL BUFF(ISAVX)
      INP=7
10    IF(STORE(L,INP).EQ.HBL) GO TO 7
      ICHAR=STORE(L,INP)
      IF(.NOT.(DIGIT(ICHAR))) GO TO 15
      ND=ND+1
      GO TO 7
15    NLFLAG=0
      IF(PFLAG.NE.0) GO TO 20
      IF(STORE(L,INP).NE.HPE) GO TO 21
C*****************************************************************C
C  PERIOD FOUND.  MARK PLACE.                                         C
C*****************************************************************C
      ISPER=INP
      PFLAG=1
      FPFLAG=1
```

```
C*************************************************************************C
C  INCREMENT POINTER AND CHECK FOR END OF CARD.                           C
C*************************************************************************C
16     INP=INP+1
       IF(INP.LE.72) GO TO 17
       CALL NEWCAR (B)
       IF(B)GO TO 40
       INP=7
17     IF(STORE(L,INP).EQ.HBL) GO TO 16
       ICHAR=STORE(L,INP)
       IF(DIGIT(ICHAR))GO TO 54
       GO TO 21
20     DFLAG=0
C*************************************************************************C
C  CHECK FOR EXPONENTIAL CONSTANT.                                        C
C*************************************************************************C
21     IF(STORE(L,INP).NE.HE) GO TO 30
       EFLAG=1
22     INP=INP+1
       IF(INP.LE.72) GO TO 25
       CALL NEWCAR (B)
       INP=7
25     IF(STORE(L,INP).EQ.HBL) GO TO 22
       IF(STORE(L,INP).NE.HQ) GO TO 5
C*************************************************************************C
C  .EQ. FOUND INSTEAD OF EXPONENTIAL.  MOVE POINTER BACK TO FIRST PERIOD AND C
C  RETURN INTEGER CODE.                                                   C
C*************************************************************************C
       IF(INP.GT.ISPER)GO TO 27
       NLFLAG=1
       IF(L.EQ.1)GO TO 26
       L=1
       GO TO 27
26     L=2
27     INP=ISPER
       GO TO 40
C*************************************************************************C
C  CHECK FOR DOUBLE PRECISION CONSTANT.                                   C
C*************************************************************************C
30     IF(STORE(L,INP).NE.HD) GO TO 35
       DFLAG=1
       GO TO 22
C*************************************************************************C
C  CHECK FOR HOLLERITH CONSTANT.                                          C
C*************************************************************************C
   35  IF(STORE(L,INP).NE.HH)GO TO 391
       IF(HFLAG.EQ.0) GO TO 40
C*************************************************************************C
C  HOLLERITH CONSTANT FOUND.  CONVERT INTEGER AND SCAN HOLLERITH FIELD.   C
C*************************************************************************C
       ISTART=ISAV
       N=ND
       NH=MOVCON(ISTART,N)
       DO 38 M=1,NH
       INP=INP+1
       IF(INP.LE.72) GO TO 38
```

```
      CALL NEWCAR (8)
      INP=7
36    CONTINUE
39    IP=4
      RETURN
 391  ICHAR=STORE(L,INP)
      IF((PFLAG.EQ.0).OR.(.NOT.(ALPHA(ICHAR))))GO TO 40
      INP=INP-1
40    IF(DFLAG.EQ.0) GO TO 42
C*****************************************************************
C  DOUBLE PRECISION CONSTANT FOUND.                              C
C*****************************************************************
      INP=INP-1
      IP=3
      RETURN
42    IF(FPFLAG.EQ.0) GO TO 44
C*****************************************************************
C  FLOATING POINT CONSTANT FOUND.                                C
C*****************************************************************
      IP=2
      INP=INP-1
      RETURN
C*****************************************************************
C  INTEGER FOUND.                                                C
C*****************************************************************
44    IP=1
      INP=INP-1
      RETURN
C*****************************************************************
C  CHECK FOR PERIOD AND INCREMENT POINTER.                       C
C*****************************************************************
50    IF(STORE(L,INP) .NE.HPE) GO TO 100
51    INP=INP+1
      IF(INP.LE.72) GO TO 53
      CALL NEWCAR (8)
      INP=7
C*****************************************************************
C  SCAN DIGITS PAST DECIMAL.                                     C
C*****************************************************************
53    IF(STORE(L,INP).EQ.HBL) GO TO 51
      ICHAR=STORE(L,INP)
      IF(.NOT.(DIGIT(ICHAR))) GO TO 60
54    ND=ND+1
      PFLAG=1
      FPFLAG=1
      GO TO 7
60    NSAVE=INP
C*****************************************************************
C  CHECK FOR LOGICAL CONSTANT.                                   C
C*****************************************************************
      IF(.NOT.((STORE(L,INP).EQ.HT).OR.(STORE(L,INP).EQ.HF)))GOTO 85
      INP=INP+1
      IF(INP.LE.72) GO TO 65
      CALL NEWCAR (8)
      INP=7
65    IF(STORE(L,INP).NE.HPE) GO TO 86
```

```
      IP=6
      RETURN
C*****************************************************************C
C  CHECK FOR LOGICAL OPERATOR.                                    C
C*****************************************************************C
85    INP=NSAVE
      IF(.NOT.((STORE(L,INP).EQ.HA).OR.(STORE(L,INP).EQ.HO).OR.
     1  (STORE(L,INP).EQ.HN))) GO TO 88
86    INP=INP+1
      IF(INP.LE.72) GO TO 87
      CALL NEWCAR (B)
      INP=7
87    IF(STORE(L,INP).EQ.HE) GO TO 88
      IF(STORE(L,INP).NE.HPE) GO TO 86
      IP=5
      RETURN
C*****************************************************************C
C  RELATIONAL OPERATOR FOUND.                                     C
C*****************************************************************C
88    IP=9
89    INP=INP+1
      IF(INP.LE.72) GO TO 90
      CALL NEWCAR (B)
      INP=7
90    IF(STORE(L,INP).NE.HPE) GO TO 89
      RETURN
C*****************************************************************C
C  CHECK FOR QUOTE.                                               C
C*****************************************************************C
100   IF(STORE(L,INP).EQ.HQU) GO TO 110
C*****************************************************************C
C  NO CONSTANT OR OPERATOR FOUND.                                 C
C*****************************************************************C
      INP=INP-1
      IP=0
      RETURN
C*****************************************************************C
C  HOLLERITH IN QUOTES FOUND. SCAN IT.                            C
C*****************************************************************C
110   INP=INP+1
      IF(INP.LE.72) GO TO 115
      CALL NEWCAR (B)
      INP=7
115   IF(STORE(L,INP).NE.HQU) GO TO 110
      INP=INP+1
      IF(INP.LE.72) GO TO 118
      CALL NEWCAR (B)
      IF(.NOT.(B)) GO TO 118
117   IP=4
      RETURN
118   IF(STORE(L,INP).EQ.HQU) GO TO 110
      INP=INP-1
      GO TO 117
      END
*DECK CONST
      INTEGER FUNCTION CONST(N1,N2)
```

```
      IMPLICIT INTEGER(A-Z)
C*****************************************************************
C  FUNCTION CONST PERFORMS TRANSFORMATION FROM THE CHARACTER CODE      C
C  OF AN INTEGER TO THE VALUE OF THAT INTEGER.   INPUT PARAMETERS      C
C  N1,N2 CONTAIN AN 8 CHARACTER STRING - 4 CHAR/WORD, LEFT-JUSTIFIED,  C
C  BLANK-FILLED - AND THE INTEGER VALUE IS RETURNED AS THE VALUE       C
C  OF THE FUNCTION.                                                    C
C*****************************************************************
      DATA HBL/1H /
      J=0
      VALUE=0
      IF(N2.EQ.HBL)GO TO 2
      DO 1 I=1,4
      DIGIT=FLD((4-I)*6,6,N2)
      IF(DIGIT.EQ.55B) GO TO 1
      VALUE=VALUE+(DIGIT-33B)*10**J
      J=J+1
    1 CONTINUE
    2 DO 3 I=1,4
      DIGIT=FLD((4-I)*6,6,N1)
      IF(DIGIT.EQ.55B) GO TO 3
      VALUE=VALUE+(DIGIT-33B)*10**J
      J=J+1
    3 CONTINUE
      CONST=VALUE
      RETURN
      END
*DECK CONVER
      INTEGER FUNCTION CONVER(IARRAY,N)
C*****************************************************************
C  FUNCTION CONVER CONVERTS A STRING OF DIGITS (MAX. 8) TO THEIR INTEGER
C  VALUE AND RETURNS THIS VALUE.
C  PARAMETERS
C  IARRAY-(INPUT)--BUFFER FROM WHICH DIGITS ARE TO BE CONVERTED.
C  N--(INPUT)--NO. OF DIGITS TO BE CONVERTED.
C*****************************************************************
      DIMENSION IARRAY(8)
      INTGR=0
      DO 20 I=1,8
      IDIGIT=(IARRAY(I)-33B)*(10**(N-I))
      INTGR=INTGR+IDIGIT
      IF(I.EQ.N) GO TO 25
20    CONTINUE
25    CONVER=INTGR
      RETURN
      END
*DECK DIGIT
      LOGICAL FUNCTION DIGIT(CHAR)
C*****************************************************************
C  FUNCTION DIGIT DETERMINES WHETHER THE INPUT PARAMETER CHAR IS A DIGIT 0--9  C
C  IF IT IS A DIGIT, THE VALUE .TRUE. IS RETURNED.  OTHERWISE, .FALSE. IS     C
C  RETURNED.                                                                  C
C  CHAR CONTAINS AN ALPHANUMERIC CHARACTER IN A1 FORMAT                C
C                                                                      C
C  DIGIT DEPENDS ON THE INTERNAL CHARACTER REPRESENTATION OF THE       C
C  MACHINE USED.                                                       C
```

```
 8001  FORMAT(4X,I4,9X,I4,11X,I4,11X,I1,6X,A1,I7,10X,I1,6X,A1,I7,10X,
     1 I1,4X,A1,I7)
  300 CONTINUE
C*****************************************************************************3
C  PRINT STORAGE ALLOCATION TABLE                                              C
C*****************************************************************************C
902   PRINT 9050
9050  FORMAT(1H1,16X,24HSTORAGE ALLOCATION TABLE//15H THIS TABLE CON,
     1 60HTAINS LISTS OF ALL COMMON VARIABLES IN ALL MODULES AND THEIR,
     2 16H ASSOCIATED WORD/41H LENGTHS.  ONE ENTRY IS MADE EACH TIME A ,
     3 42HCOMMON VAR. IS DECLARED IN A COMMON BLOCK.//13H CNTAB PTR.--,
     4 51HPOINTS TO NAME OF COMMON BLOCK IN COMMON NAME TABLE/
     5 53H SYMTAB PTR.--POINTS TO VARIABLE NAME IN SYMBOL TABLE/
     6 60H STARTING LOCATION--INDICATES STORAGE STARTING LOCATION RELA,
     7 28HTIVE TO CURRENT COMMON BLOCK/26H NO. OF WORDS--TOTAL DIMEN,
     8 16HSION OF VARIABLE//6H INDEX,5X,11H CNTAB PTR.,5X,11HSYMTAB PTR.
     9 ,5X,10HSTART LOC.,5X,11HNO. OF WDS./)
      ISTAL=INDST-1
      DO 950 I=1,ISTAL
      IWD1=STALTB(1,I)
      IWD2=STALTB(2,I)
      ICNP=HIFECH(IWD1)
      ISP=LOFECH(IWD1)
      ISL=HIFECH(IWD2)
      NW=LOFECH(IWD2)
      PRINT 9051,I,ICNP,ISP,ISL,NW
 9051 FORMAT(1X,I4,8X,I3,13X,I4,12X,I5,13X,I5)
  950 CONTINUE
      RETURN
      END
*DECK MIN
      FUNCTION MIN(I1,I2)
      IF(I1.LT.I2) GO TO 20
      MIN=I2
      RETURN
20    MIN=I1
      RETURN
      END
*DECK MOVCON
      FUNCTION MOVCON(ISTART,N)
      IMPLICIT INTEGER (A-Z)
C*****************************************************************************C
C  FUNCTION MOVCON MOVES A STRING OF MAX. 8 DIGITS TO A TEMPORARY BUFFER      C
C  (INTAR) AND CALLS FUNCTION CONVER TO CONVERT THE CHARACTER CODE STRING TO  C
C  THE PROPER INTEGER.  THE VALUE OF MOVCON IS THE CONVERTED INTEGER.         C
C  COMMON VARIABLES                                                           C
C  SEE SUBROUTINE SCAN.                                                       C
C  PARAMETERS                                                                 C
C  ISTART--(INPUT)--POINTER TO STARTING POSITION IN STORE OF STRING TO BE     C
C  CONVERTED.                                                                 C
C  N-(INPUT)--NO. OF DIGITS IN INTEGER TO BE CONVERTED.                       C
C  LOCAL VARIABLES                                                            C
C  INTAR(8)--BUFFER FOR INTEGERS                                              C
C  KOUNT--COUNTER FOR DIGITS BEING MOVED                                      C
C  J--POSITION OF DIGIT IN STORE                                              C
C*****************************************************************************C
```

```
      IFIRST = TRIP(I,1)
      ISEC=TRIP(I,2)
      PRINT 1092,IFIRST,ISEC
 1092 FORMAT(6X,I5,5X,I5)
 1095 CONTINUE
C*****************************************************************C
C  PRINT MODE TABLE                                               C
C*****************************************************************C
      PRINT 2010
 2010 FORMAT(1H1,44X,*MODE TABLE*//1X,*STATEMT NO.*,3X,*STATEMT TYPE*,
     1 3X,*USETAB PTR.*,3X,*SUCCESS. PTR.*,3X,*SUCCESS. NO.*,3X,
     2 *PRED. PTR.*,3X,*PRED. NO.*/)
      DO 2020 I=1, STNO
      IWD1=NODTAB(1,I)
      IWD2=NODTAB(2,I)
      IWD3=NODTAB(3,I)
      ISTYP=HIFECH(IWD1)
      IUP=LOFECH(IWD1)
      ISUPT=HIFECH(IWD2)
      ISN=LOFECH(IWD2)
      IPRPT=HIFECH(IWD3)
      IPN=LOFECH(IWD3)
      PRINT 2015,I,ISTYP,IUP,ISUPT,ISN,IPRPT,IPN
 2015 FORMAT(5X,I4,6(9X,I5))
 2020 CONTINUE
C*****************************************************************3
C  PRINT PRECEDESSOR AND SUCCESSOR TABLE.                         C
C*****************************************************************3
      PRINT 2030
 2030 FORMAT(1H1,1X,* INDEX*,5X,*SUCCESSOR TABLE*,5X,
     1 *PREDECESSOR TABLE*/)
      IF(IPRE.GT.ISUC) GO TO 2035
      N=ISUC
      GOTO 2036
 2035 N=IPRE
 2036 DO 2040 I=1,N
      PRINT 2038,I,SUCTAB(I),PRETAB(I)
 2038 FORMAT(1X,I4,9X,I5,17X,I5)
 2040 CONTINUE
      IF(INDOO.EQ.0) GOTO 902
      PRINT 8000
 8000 FORMAT(1H1,10X,*DO LOOP TABLE*,//,2X,*ST. NO.*,5X,*LABEL PTR.*,5X,
     1 *INDEX PTR.*,5X,*VP-CODE*,2X,*INIT. VALUE*,5X,*VP-CODE*,2X,
     2 *TERM. VALUE*,3X,*VP-CODE*,2X,*INCREMENT*)
      DO 300 I=1,INDOO
      IWD1=HIFECH(DOTAB(1,I))
      IWD2=LOFECH(DOTAB(1,I))
      IWD3=LOFECH(DOTAB(2,I))
      IWD4=LOFECH(DOTAB(3,I))
      CALL BITRET(IWD4,ICD1,IPM1)
      IWD5=HIFECH(DOTAB(3,I))
      CALL BITRET(IWD5,ICD2,IPM2)
      IWD6=HIFECH(DOTAB(2,I))
      CALL BITRET(IWD6,ICD3,IPM3)
      PRINT 8001,IWD1,IWD2,IWD3,ICD1,IPM1,IWD4,ICD2,IPM2,IWD5,ICD3,
     1 IPM3,IWD6
```

```
     3 45H SOURCE LISTING AND A USE CODE INDICATING HOW/12H THE NAME IS,
     4 57H USED IN THAT STATEMENT.  EACH ENTRY IS LINKED TO THE PRE,
     5 25HVIOUS AND NEXT ENTRIES OF/30H THE SAME NAME BY BACKWARD AND,
     6 57H FORWARD POINTERS.  THE FIRST ENTRY OF A NAME CONTAINS A ,
     7  7HPOINTER/33H TO THE NAME IN THE SYMBOL TABLE.//12H BACK POINTE,
     8 61HR--POINTS TO PREVIOUS ENTRY OF NAME IN USAGE TABLE OR TO NAME,
     9 16H IN SYMBOL TABLE/41H FORWARD POINTER--POINTS TO NEXT ENTRY OF,
     1 46H NAME IN USAGE TABLE.  FOR LAST ENTRY, FORWARD/5X,7HPOINTER,
     2 6H IS 0./49H BP-CODE--1 WHEN BACK PTR. POINTS TO SYMBOL TABLE/
     3 10X, 41HO WHEN BACK POINTER POINTS TO USAGE TABLE//6H INDEX,5X,
     4 11HSTATEMT NO., 5X,8HUSE CODE,5X,7HBP-CODE,5X,9HBACK PTR.,5X,
     5 12HFORWARD PTR./)
      IUT=INDUT-1
      DO 200 K=1,IUT
      IWD1=USETAB(1,K)
      IWD2=USETAB(2,K)
      ISN=HIFECH(IWD1)
      IUC=LOFECH(IWD1)
      ICD=HCFECH(IWD2)
      IBP=HMFECH(IWD2)
      IFP=LOFECH(IWD2)
      PRINT 5000,K,ISN,IUC,ICD,IBP,IFP
5000  FORMAT(1X,I4,11X,I3,8X,I5,12X,I1,10X,I4,10X,I4)
200   CONTINUE
C*****************************************************************************C
C  PRINT ARRAY LENGTH TABLE                                                     C
C*****************************************************************************C
      PRINT 1080
 1080 FORMAT(1H1,10X,18HARRAY LENGTH TABLE//21H THIS TABLE CONTAINS ,
     1 57HDIMENSIONS FOR ALL ARRAYS THAT APPEAR IN A MODULE.  EACH ,
     2  8HENTRY IN/47H THIS TABLE IS POINTED TO BY THE ALTAB POINTER ,
     3 38HOF THE ARRAY NAME IN THE SYMBOL TABLE.//6H INDEX,5X,6HDIM. 1,
     4 5X,6HDIM. 2,5X,6HDIM. 3/)
      I=1
  201 K=I
      IWD=ALTAB(K)
      ID1=LOFECH(IWD)
      ID2=HIFECH(IWD)
      IF (ID2.EQ.0)GOTO 205
      IWD=ALTAB(K+1)
       IF(HIFECH(IWD).EQ.0)GOTO 210
      I=I+1
      ID3=LOFECH(IWD)
      GOTO 220
  205 ID2=1
  210 ID3=1
  220 PRINT 7000,K,ID1,ID2,ID3
 7000 FORMAT(1X,I3,3(7X,I4))
      I = I + 1
      IF (I .LT. INDAL ) GO TO 201
C*****************************************************************************C
C  PRINT TRANSITION PAIR TABLE.                                                 C
C*****************************************************************************C
  225 PRINT 1090
 1090 FORMAT(1H1,1X,*TRANSITION PAIRS TABLE*/)
      DO 1095 I=1,ITR
```

```
C*****************************************************************C
C  THIS ROUTINE PRINTS THE LOCAL TABLES   SYMTAB,USETAB,ALTAB,DOTAB,  C
C  NODTAB, SUCTAB, PRETAB, AND STALTB.                               C
C*****************************************************************C
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1 ,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
      COMMON/TABSUB/ISTP,NSAV,LAN,ISUBAD,IPFLAG,ICTYPE
      PRINT 1000, MODNO
 1000 FORMAT(1H1,10X,33HLOCAL TABLES GENERATED FOR MODULE,I5/)
C*****************************************************************C
C  PRINT SYMBOL TABLE                                               C
C*****************************************************************C
      PRINT 1060
1060  FORMAT(1HO,40X,12HSYMBOL TABLE//30H THIS TABLE CONTAINS ALL NAMES,
     139H AND LABELS THAT APPEAR IN EACH MODULE./16H NAMES ARE HASH-,
     251HCODED AND APPEAR WITH FOLLOWING CODES AND POINTERS //
     311H TYPE CODES,14X,11HCLASS CODES/11H 0--INTEGER,14X,
     4 15HO--PROGRAM NAME/18H 1--FLOATING POINT,7X,18H1--SUBROUTINE NAME
     5,/20H 2--DOUBLE PRECISION,5X,26H2--STATEMENT FUNCTION NAME/
     6 11H 3--COMPLEX,14X,13H3--ARRAY NAME/11H 4--LOGICAL,14X,
     7 16H4--FUNCTION NAME/11H 5--NEUTRAL,14X, 8H5--LABEL/25X,
     8 11H6--VARIABLE/25X,20H7--COMMON BLOCK NAME//
     9 60H USETAB TOP POINTER--POINTS TO FIRST ENTRY OF NAME IN USAGE ,
     1 5HTABLE/53H USETAB BOTTOM POINTER--POINTS TO FIRST ENTRY OF NAME,
     2 15H IN USAGE TABLE/41H STALTB PTR.--POINTS TO LOCATION OF NAME ,
     3 57HIN STORAGE ALLOCATION TABLE IF NAME IS A COMMON VARIABLE./
     3 14X,*OR TO ENTRY IN PARTAB IF SUB. OR FUNCTION NAME*/
     4 61H ALTAB PTR.--POINTS TO LOCATION OF NAME IN ARRAY LENGTH TABLE,
     5 21H IF NAME IS AN ARRAY.//6H INDEX,5X,4HNAME,5X,4HTYPE,5X,
     6 5HCLASS,5X,15HUSETAB TOP PTR.,4X,16HUSETAB BOT. PTR.,4X,
     2 11HSTALTB PTR.,4X,10HALTAB PTR./)
      DO 100 K=1,LSYM
      IF(SYMTAB(1,K).EQ.0) GO TO 100
      IWD3=SYMTAB(3,K)
      IWD4=SYMTAB(4,K)
      IWD5=SYMTAB(5,K)
      ITYPE=HIFECH(IWD3)
      ICLASS=LOFECH(IWD3)
      ITP=HIFECH(IWD4)
      IBP=LOFECH(IWD4)
      ISP=HIFECH(IWD5)
      IAP=LOFECH(IWD5)
      PRINT 3000,K,SYMTAB(1,K),SYMTAB(2,K),ITYPE,ICLASS,ITP,IBP,
     1 ISP,IAP
 3000 FORMAT(1X,I4,7X,2A4,1X,I4,5X,I5,8X,I4,13X,I4,15X,I4,10X,I4)
100   CONTINUE
C*****************************************************************C
C  PRINT USE TABLE                                                  C
C*****************************************************************C
      PRINT 1070
1070  FORMAT(1H1,30X,11HUSAGE TABLE//30H THIS TABLE CONTAINS ONE ENTRY,
     1 59H FOR EACH APPEARANCE OF A NAME IN A MODULE.  EACH ENTRY CON,
     2  9HTAINS THE/47H STATEMENT NO. IN WHICH THE NAME APPEARS IN THE,
```

```
      IF(I.EQ.3)GO TO 4
      CALL ERROR(39)
      GO TO 4
    5 JISS=JISS+1
C*****************************************************************C
C  STORE VARIABLE NAME                                            C
C*****************************************************************C
    6 CALL TABS(KC,KT,KU)
      IF(ISS(JISS).NE.HSL)GO TO 60
      IF(KUU.NE.14)GO TO 60
C*****************************************************************C
C  MAKE AN ENTRY IN TABLE TO INDICATE THAT THE VARIABLE HAS BEEN  C
C  INITIALIZED.                                                   C
C*****************************************************************C
      KU=12
      CALL TABS(KC,KT,KU)
C*****************************************************************C
C  CHECK FOR END OF STATEMENT, END OF LIST                        C
C*****************************************************************C
   60 IF(JISS.GE.INS)RETURN
      IF(ISS(JISS).EQ.HSL)GO TO 9
      IF(ISS(JISS).EQ.HCO)GO TO 7
      CALL ERROR(19)
      RETURN
C*****************************************************************C
C  GET NEXT LIST ITEM                                             C
C*****************************************************************C
    7 JISS=JISS+1
      IF(ISS(JISS).EQ.HV)GO TO 8
      CALL EPROR(20)
      RETURN
    8 NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      JTST=JTST+1
      JISS=JISS+1
      GO TO 1
C*****************************************************************C
C  END OF COMMON BLOCK. RETURN                                    C
C*****************************************************************C
    9 IF(KUU.EQ.11)RETURN
C*****************************************************************C
C  SCAN OVER LITERAL LIST                                         C
C*****************************************************************C
   10 JISS=JISS+1
      IF(JISS.GE.INS)RETURN
      IF(ISS(JISS).EQ.HV.OR.ISS(JISS).EQ.HI)JTST=JTST+1
      IF(ISS(JISS).NE.HSL)GO TO 10
      JISS=JISS+1
      IF(ISS(JISS).EQ.HCO)GO TO 7
      IF(ISS(JISS).EQ.HV)GO TO 8
      CALL ERROR(20)
      RETURN
      END
*DECK LOCPRT
      SUBROUTINE LOCPRT(MODNO)
      IMPLICIT INTEGER (A-Z)
```

```
      DIMENSION X(3)
      NAME(1)=X(1)
      NAME(2)=X(2)
    1 KC=6
      KT=KTT
      KU=KUU
      IF(JISS.LE.INS)GO TO 2
      CALL TABS(KC,KT,KU)
      RETURN
C*****************************************************************C
C  CHECK FOR ARRAY                                                C
C*****************************************************************C
    2 IF(ISS(JISS).NE.HLP)GO TO 6
C*****************************************************************C
C  STORE DIMENSIONS OF ARRAY                                      C
C*****************************************************************C
      KC=3
      JISS=JISS+1
      I=0
      DIM(1)=0
      DIM(2)=0
      DIM(3)=0
    3 IF(ISS(JISS).NE.HI)GO TO 4
      I=I+1
      DIM(I)=CONST(TSTAB(1,JTST),TSTAB(2,JTST))
      JTST=JTST+1
    4 IF(ISS(JISS).EQ.HFP)GO TO 5
      IF(ISS(JISS).NE.HV)GO TO 40
C*****************************************************************C
C  DIMENSION SPECIFIED BY A VARIABLE. SAVE NAME, CLASS, TYPE AND USE   C
C  CODE OF THE ARRAY. MAKE AN ENTRY OF THE SUBSCRIPT IN TABLES.   C
C     CLASS = VARIABLE                                            C
C     TYPE  = NEUTRAL                                             C
C     USE   = SUBSCRIPT                                           C
C*****************************************************************C
      TEMP1=NAME(1)
      TEMP2=NAME(2)
      TEMP3=KC
      TEMP4=KT
      TEMP5=KU
      NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      KC=6
      KT=5
      KU=15
      CALL TABS(KC,KT,KU)
C*****************************************************************C
C  RESTORE THE NAME, CLASS, TYPE AND USE CODE OF THE ARRAY NAME   C
C*****************************************************************C
      NAME(1)=TEMP1
      NAME(2)=TEMP2
      KC=TEMP3
      KT=TEMP4
      KU=TEMP5
   40 JISS=JISS+1
      IF(I.LT.3)GO TO 3
```

```
C  LEFT PARENTHESIS FOUND                                              C
C  PUT DUPLICATE OF TOP OF STACK ENTRY ON STACK                        C
C*********************************************************************C
   10 IF(TEMP.NE.HLP)GO TO 11
      PC=PC+1
      STACK(PC)=STACK(PC-1)
      JISS=JISS+1
      TEMP=ISS(JISS)
      GO TO 1
C*********************************************************************C
C  CHECK FOR INTEGER, RIGHT PARENTHESIS, END OF STATEMENT              C
C*********************************************************************C
   11 IF(TEMP.EQ.HI)JTST=JTST+1
      IF(TEMP.EQ.HRP)GO TO 8
      IF(JISS.GE.INS)RETURN
      IF(TEMP.EQ.HSL)RETURN
C*********************************************************************C
C  GET NEXT LIST ITEM                                                  C
C*********************************************************************C
      JISS=JISS+1
      TEMP=ISS(JISS)
      GO TO 1
      END
*DECK LISTP
      SUBROUTINE LISTP(X,KTT,KUU)
      IMPLICIT INTEGER(A-Z)
C*********************************************************************C
C  SUBROUTINE LISTP IS A PARSING ROUTINE WHICH PROCESSES A VARIABLE    C
C  LIST FOUND IN SPECIFICATION STATEMENTS. (COMMON, DIMENSION AND      C
C  TYPE STATEMENTS).                                                   C
C  UPON ENTRY, POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS ISS,TSTAB    C
C  ,RESP.) POINT TO THE ENTRIES IN ISS AND TSTAB FOLLOWING THE ENTRIES C
C  FOR THE FIRST NAME IN THE LIST.  UPON RETURN, EITHER AN END OF      C
C  STATEMENT CONDITION OR THE DELIMETER  /  HAS BEEN FOUND.            C
C  IF  /  IS FOUND, JISS POINTS TO THE  /  AND JTST POINTS TO          C
C  THE NEXT ENTRY IN TSTAB.                                            C
C                                                                      C
C  X - 2 WORD ARRAY CONTAINING THE CHARACTER STRING FOR THE FIRST      C
C  ITEM IN THE LIST                                                    C
C     KTT - TYPE CODE FOR THE VARIABLES IN THE LIST                    C
C     KUU - USE CODE FOR THE VARIABLES IN THE LIST (COMMON BLOCK ENTRY,C
C           DIMENSION STATEMENT ENTRY, OR  TYPE STATEMENT ENTRY).      C
C  DIM - 3 WORD ARRAY USED TO PASS THE ARRAY DIMENSIONS FOUND FOR      C
C  A GIVEN LIST NAME TO SUBROUTINE TABS .                              C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES        C
C        EXTRACTED FROM FORTRAN STATEMENTS                             C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS      C
C  KC,KT,KU - PARAMETERS INDICATING THE CLASS,TYPE, AND USE CODES      C
C  FOR THE NAME IN QUESTION                                            C
C*********************************************************************C
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,H$P,HEQ,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
```

```
      PC=PC+1
      STACK(PC)=15
      JISS=JISS+1
      TEMP=ISS(JISS)
      GO TO 1
C*****************************************************************C
C  CHECK FOR IMPLIED DO LOOP                                      C
C*****************************************************************C
    2 IF(TEMP.EQ.HEQ)GO TO 4
C*****************************************************************C
C  STORE SIMPLE VARIABLE NAME                                     C
C*****************************************************************C
    3 KC=6
      KT=5
      KU=STACK(PC)
      NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      CALL TABS(KC,KT,KU)
      JTST=JTST+1
      GO TO 11
C*****************************************************************C
C  IMPLIED DO-LOOP FOUND.  STORE NAME OF INDEX VARIABLE.          C
C*****************************************************************C
    4 KC=6
      KT=0
      KU=5
      NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      CALL TABS(KC,KT,KU)
      JTST=JTST+1
C*****************************************************************C
C  ENTER ASSIGNED STATEMENT NUMBER IN DO-LOOP TABLE               C
C*****************************************************************C
      INDDO=INDDO+1
      DOTAB(1,INDDO)=HISTOR(DOTAB(1,INDDO),STNO)
C*****************************************************************C
C  SCAN DO-LOOP RANGE INDICATORS                                  C
C*****************************************************************C
      CALL DORANG
      IF(ISS(JISS).EQ.HRP)GO TO 8
      CALL ERROR(25)
      RETURN
C*****************************************************************C
C  RIGHT PARENTHESIS FOUND                                        C
C  POP THE USE CODE STACK AND CHECK FOR END OF STATEMENT          C
C*****************************************************************C
    8 PC=PC-1
      IF(PC.LE.0)GO TO 9
      JISS=JISS+1
      IF(JISS.GT.INS)RETURN
      TEMP=ISS(JISS)
      IF(TEMP.EQ.HSL)RETURN
      GO TO 1
    9 CALL ERROR(36)
      RETURN
C*****************************************************************C
```

```
      END
*DECK IOLIST
      SUBROUTINE IOLIST(IOF)
      IMPLICIT INTEGER(A-Z)
C*****************************************************************C
C  SUBROUTINE IOLIST IS A PARSING ROUTINE WHICH PROCESSES THE LIST  C
C  OF VARIABLES OF AN I/O OR DATA STATEMENT.                        C
C  UPON ENTRY, POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS ISS,TSTAB C
C  ,RESP.) POINT TO THE FIRST VARIABLE IN THE LIST, UNLESS THE LIST C
C  BEGINS WITH (  , IN WHICH CASE JISS POINTS TO THE (  .           C
C  A USE CODE STACK IS KEPT TO INDICATE THE PROPER USE CODES        C
C  FOR NAMES AT THE VARIOUS LEVELS OF NESTING.  NAMES AT THE HIGHEST C
C  LEVEL ARE EITHER EXTERNAL INPUT OR OUTPUT VARIABLES OR VARIABLES C
C  ASSIGNED  VALUES IN A DATA STATEMENT AS DETERMINED BY PARAMETER IOF.C
C  IOF - USE CODE PARAMETER                                         C
C        IOF = 3  IF CALLED FROM OUTPUT ST. PROCESSOR : EWRITE,EPRPU C
C        IOF = 4  IF CALLED FROM INPUT ST. PROCESSOR : EREAD        C
C        IOF = 12 IF CALLED FROM DATA ST. PROCESSOR : EDATA         C
C  KC,KT,KU - PARAMETERS TO SUBROUTINE TABS INDICATING THE CLASS,TYPE, C
C  AND USE CODES FOR THE NAME IN QUESTION                           C
C*****************************************************************C
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOS,NFILE,ARGNO
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HPP,HEQ,HAS,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INOUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
      DIMENSION STACK(15)
C*****************************************************************C
C  INITIALIZE USE CODE STACK TO I/O CODE                            C
C*****************************************************************C
      STACK(1)=IOF
      PC=1
      TEMP=ISS(JISS)
C*****************************************************************C
C  EXAMINE NEXT LIST ITEM                                           C
C*****************************************************************C
    1 IF(TEMP.NE.HV)GO TO 10
      IF(JISS.GE.INS)GO TO 3
      JISS=JISS+1
      TEMP=ISS(JISS)
      IF(TEMP.NE.HLP)GO TO 2
C*****************************************************************C
C  SUBSCRIPTED VARIABLE FOUND.  STORE NAME, PUT USE CODE            C
C  ON STACK FOR SUCCEEDING NAMES                                    C
C*****************************************************************C
      KT=5
      KU=STACK(PC)
      NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      CALL TABS(KC,KT,KU)
      JTST=JTST+1
```

```
C*************************************************************************C
    4 JTST=JTST+1
      JISS=JISS+1
      IF(ISS(JISS).EQ.HT)GO TO 5
      CALL ERROR(35)
      RETURN
    5 NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      KC=5
      KT=5
      KU=10
      CALL TABS(KC,KT,KU)
C*************************************************************************C
C  CHECK FOR END OF ARGUMENT LIST                                         C
C*************************************************************************C
      JISS=JISS+1
      JTST=JTST+1
      IF(ISS(JISS).EQ.HPP)RETURN
C*************************************************************************C
C  CHECK FOR  END  OR  ERR  CONDITION                                     C
C*************************************************************************C
      JISS=JISS+2
      IF(ISS(JISS).EQ.HEQ)GO TO 4
      CALL ERROR(35)
      RETURN
C*************************************************************************C
C  STORE FORMAT REFERENCE                                                 C
C     CLASS = VARIABLE OR UNDEFINED LABEL                                 C
C     TYPE  = NEUTRAL                                                     C
C     USE   = FORMAT REFERENCE                                            C
C*************************************************************************C
    6 IF(ISS(JISS).EQ.HI)GO TO 7
      IF(ISS(JISS).EQ.HV)GO TO 8
      CALL ERROR(34)
      RETURN
    7 KC=5
      GO TO 9
    8 KC=6
    9 KT=5
      KU=27
      NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      CALL TABS(KC,KT,KU)
      JISS=JISS+1
      JTST=JTST+1
C*************************************************************************C
C  CHECK FOR END OF ARGUMENT LIST                                         C
C*************************************************************************C
      IF(ISS(JISS).EQ.HPP)RETURN
      JISS=JISS+2
C*************************************************************************C
C  CHECK FOR ERR OR END CONDITION                                         C
C*************************************************************************C
      IF(ISS(JISS).EQ.HEQ)GO TO 4
      CALL ERROR(35)
      RETURN
```

```
C             F     = FORMAT LABEL                                          C
C             L1    = NEXT STATEMENT FOR EXECUTION IF ERROR DETECTED        C
C             L2.   = NEXT STATEMENT FOR EXECUTION IF EOF                   C
C  THE UNIT REFERENCE IS ASSUMED TO BE ALWAYS PRESENT.WHILE ANY             C
C  COMBINATION OF THE REMAINING ARGUMENTS MAY BE OMITTED IN THE             C
C  I/O STATEMENT.                                                          C
C  UPON ENTRY, THE POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS ISS,TSTAB,   C
C  RESP.) POINT TO THE FOLLOWING : JTST POINTS TO THE STRING  READ  OR     C
C   WRITE  IN TSTAB AND JISS POINTS TO THE  (  FOLLOWING THE  READ  OR     C
C   WRITE .  UPON RETURN, JTST POINTS TO THE LAST TSTAB ENTRY FOR THE      C
C  ARGUMENT LIST AND JISS POINTS TO THE FOLLOWING  )  .                    C
C  KC,KT,KU - PARAMETERS TO SUBROUTINE TABS INDICATING THE CLASS,          C
C  TYPE, AND USE CODES FOR THE NAME IN QUESTION                            C
C************************************************************************C
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HOU,HSL,HPE,HFORM,HAT,HC,HH,HI
      JISS=JISS+1
      JTST=JTST+1
C************************************************************************C
C  CHECK FOR UNIT NAME OR NUMBER                                          C
C  IF UNIT NUMBER FOUND , IGNORE                                          C
C************************************************************************C
      IF(ISS(JISS).EQ.HV)GO TO 1
      IF(ISS(JISS).EQ.HI)GO TO 2
      CALL ERROR(33)
      RETURN
C************************************************************************C
C  STORE UNIT NAME                                                        C
C************************************************************************C
    1 NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      KC=10
      KT=0
      KU=26
      CALL TABS(KC,KT,KU)
C************************************************************************C
C  CHECK FOR END OF ARGUMENT LIST                                         C
C************************************************************************C
    2 JISS=JISS+1
      JTST=JTST+1
      IF(ISS(JISS).EQ.HRP)RETURN
      IF(ISS(JISS).EQ.HCC)GO TO 3
      CALL ERROR(34)
      RETURN
C************************************************************************C
C  CHECK FOR  END  OR  ERR  CONDITION                                     C
C************************************************************************C
    3 JISS=JISS+1
      IF(ISS(JISS+1).NE.HEQ)GO TO 6
      JISS=JISS+1
C************************************************************************C
C  STORE LABEL FOR  END  OR  ERR  CONDITION                               C
```

```
      NCHARS=FLD(0,24,NAM2)
      NUM=(NCHARS+ICHARS)/2
      NADDR=MOD(NUM,IPRIME)+1
    2 IF(SYMTAB(1,NADDR).EQ.0) GO TO 10
      IF(SYMTAB(1,NADDR).NE. NAM1) GO TO 5
      IF(SYMTAB(2,NADDR).NE.NAM2) GO TO 5
C*************************************************************************C
C  NAME FOUND                                                             C
C*************************************************************************C
      INFLAG=1
      RETURN
C*************************************************************************C
C  THERE IS A COLLISION, CHECK THE NEXT SYMTAB ENTRY.                     C
C*************************************************************************C
    5 NADDR=NADDR+1
      IF(NADDR.GT.LSYM)NADDR=1
      KOUNT=KOUNT+1
      IF(KOUNT.LT.LSYM) GO TO 2
      PRINT 7
7     FORMAT(* LIMITS OF SYMTAB EXCEEDED*)
      STOP
C*************************************************************************C
C  NAME NOT FOUND IN SYMTAB, INFLAG = 0                                   C
C*************************************************************************C
   10 INFLAG=0
      RETURN
      END
*DECK HASHIN
      INTEGER FUNCTION HASHIN(N)
C*************************************************************************C
C  FUNCTION HASHIN COMPUTES A HASH INDEX BASED ON THE SUM OF THE THIRD    C
C  AND FOURTH CHARACTERS OF A NAME (CHARACTER STRING).                    C
C  THE RETURNED FUNCTION VALUE IS TO BE USED AS AN INDEX TO THE TABLE     C
C  OF KEY WORDS (MATCH)                                                   C
C  N - INPUT NAME (4 CHAR/WORD)                                           C
C  N3 - CHARACTER CODE  VALUE  OF THIRD CHARACTER OF N                    C
C  N4 - CHARACTER CODE  VALUE  OF FOURTH CHARACTER OF N                   C
C                                                                         C
C     HASHIN = N3 + N4 + 10                                               C
C  THE HASHIN VALUE RETURNED DEPENDS ON THE INTERNAL CHARACTER            C
C  REPRESENTATION OF THE MACHINE                                          C
C*************************************************************************C
      INTEGER FLD
      N3=FLD(12,6,N)
      N4=FLD(18,6,N)
      HASHIN=N3+N4+10
      RETURN
      END
*DECK IOARGS
      SUBROUTINE IOARGS
      IMPLICIT INTEGER(A-Z)
C*************************************************************************C
C  SUBROUTINE IOARGS IS A PARSING ROUTINE WHICH PROCESSES THE             C
C  ARGUMENT LISTS OF I/O STATEMENTS.                                      C
C     (UNIT, F, ERR=L1, END=L2)                                           C
C     WHERE UNIT = I/O UNIT NUMBER                                        C
```

```
      IBOT=LOFECH(IWD3)
      PRINT 9001,I,CNTAB(1,I),CNTAB(2,I),ITOP,IBOT,CNTAB(4,I)
 9001 FORMAT(1X,I3,5X,2A4,11X,I4,16X,I4,16X,I3)
  900 CONTINUE
      PRINT 9080
 9080 FORMAT(1H1,8X,*PARAMETER TABLE*//* THIS TABLE CONTAINS ENTRIES *,
     1 *FOR ALL PARAMETERS IN SYSTEM.  THE CODE IN EACH WORD DISTIN*,
     2 *GUISHES*/* THE TYPE OF POINTER STORED IN THAT WORD.  THE MOD. *,
     3 *NO. IS THE MODULE IN WHICH THE PARAMETER APPEARS*//* CODES*//
     4* 0--PTR. TO DUMMY PARAMETER IN SYMTAB*/* 1--PTR TO ACTUAL *,
     5 *PARAMETER IN SYMTAB*/* 2--PTR. TO PARAMETER LIST FOR NEXT CALL*,
     6 * OF SAME ROUTINE*/* 3--PTR. TO SUB. OR FUNCTION NAME IN SYMTAB*/
     7* 4--THIS WORD MARKS END OF CURRENT PARAMETER LIST*/
     8* 5--THIS IS A CONSTANT PARAMETER. POINTER FIELD CONTAINS CONSTAN*
     9,*T TYPE*//* INDEX*,5X,*CODE*,5X,*MODULE NO.*,5X,*ARGUMT NO.*,5X,
     1 *POINTER*/)
      DO 960 I=1,IPAR
      ITEM=PARTAB(I)
      ICODE=HTFECH(ITEM)
      MONO=HFFECH(ITEM)
      IAN=MFFECH(ITEM)
      IPTR=LPFECH(ITEM)
      PRINT 9090,I,ICODE,MONO,IAN,IPTR
 9090 FORMAT(1X,I4,7X,I2,10X,I3,12X,I2,11X,I4)
  960 CONTINUE
  965 RETURN
      END
*DECK HASH
      SUBROUTINE HASH(INFLAG,NAM1,NAM2)
      IMPLICIT INTEGER(A-Z)
C*********************************************************************C
C  THIS ROUTINE CALCULATES THE HASH CODE FOR A NAME TO BE STORED IN   C
C  SYMTAB AND INDICATES WHETHER THE NAME IS ALREADY IN SYMTAB THROUGH  C
C  PARAMETER INFLAG. COLLISIONS ARE RESOLVED BY GOING TO THE NEXT     C
C  BLANK TABLE ENTRY                                                  C
C  VARIABLES                                                          C
C  INFLAG--OUTPUT PARAMETER THAT INDICATES THE NAME IS ALREADY IN     C
C  SYMTAB WHEN VALUE IS ONE.  IF NAME IS NOT IN SYMTAB, VALUE IS ZERO. C
C  NAM1--FIRST FOUR CHARACTERS OF NAME                                C
C  NAM2--LAST THREE CHARACTERS OF NAME                                C
C  NADDR--COMMON VARIABLE SET TO THE ADDRESS IN SYMTAB AT WHICH THE   C
C  NAME IS FOUND OR SHOULD BE STORED IF NAME IS NOT ALREADY IN THE    C
C  TABLE.                                                             C
C  IPRIME = PRIME NUMBER USED FOR SYMTAB ADDRESS HASH CODING.         C
C                                                                     C
C  THE HASH CODE FORMULA IS                                           C
C     HASH = MOD ( (NAM1+NAM2)/2, IPRIME ) + 2                        C
C  THE HASH ADDRESS RETURNED DEPENDS ON THE CHARACTER REPRESENTATION   C
C  OF THE MACHINE USED.                                               C
C*********************************************************************C
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DCTAB(3,50),LDC,INDDC,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
      KOUNT=0
      ICHARS=FLD(0,24,NAM1)
```

```
      NAME(2)=TSTAB(2,JTST)
      KT=5
      KU=27
      CALL TABS(KC,KT,KU)
      JTST=JTST+1
      JISS=JISS+1
C*****************************************************************************C
C  CHECK FOR END OF STATEMENT                                                  C
C*****************************************************************************C
      IF(JISS.GE.INS)RETURN
      IF(ISS(JISS).NE.HCO)GO TO 8
C*****************************************************************************C
C  PROCESS OUTPUT VARIABLE LIST                                               C
C*****************************************************************************C
      JISS=JISS+1
      CALL IOLIST(IOF)
      RETURN
    8 CALL ERROR(32)
      RETURN
      END
*DECK GLOPRT
      SUBROUTINE GLOPRT
      IMPLICIT INTEGER(A-Z)
C*****************************************************************************C
C  THIS ROUTINE PRINTS THE THREE GLOBAL TABLES  DIREC,CNTAB,AND PARTAB.       C
C*****************************************************************************C
      COMMON/GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAR, IPAR, JPRIME
      PRINT 1050
 1050 FORMAT(1H1,10X,13HGLOBAL TABLES//
     1 11X,9HDIRECTORY//42H THIS TABLE LISTS NAMES OF ALL MODULES IN ,
     2 50HSYSTEM ALONG WITH AN ASSIGNED MODULE NO. AND CODE.//
     3 * INDEX*,5X,*MODULE NAME*,5X,*FILE*,5X,*NUMBER*/)
      DO 800 I=1,LDIR
      IF(DIREC(1,I).EQ.0) GO TO 800
      IWD3=DIREC(3,I)
      MCD=HIFECH(IWD3)
      MONO=LOFECH(IWD3)
      PRINT 8001,I,DIREC(1,I),DIREC(2,I),MCD,MONO
8001  FORMAT(1X,I3,7X,2A4,5X,I5,6X,I3)
800   CONTINUE
      PRINT 1090
 1090 FORMAT(1H1,24X,17HCOMMON NAME TABLE//21H THIS TABLE CONTAINS ,
     1 60HNAMES OF ALL COMMON BLOCKS IN THE SYSTEM.  ONE ENTRY IS MADE/
     2 58H FOR EACH DECLARATION OF A COMMON BLOCK.  POINTERS TO THE ,
     3 39HTOP AND BOTTOM OF THE LIST OF VARIABLES/15H IN THE COMMON ,
     4 33HBLOCK ARE STORED WITH EACH ENTRY.//18H STALTB TOP PTR.--,
     5 59HPOINTER TO FIRST VARIABLE OF COMMON BLOCK IN STORAGE ALLOCA,
     6 10HTION TABLE/46H STALTB BOTTOM PTR.--POINTER TO LAST VARIABLE ,
     7 39HOF COMMON BLOCK IN STORAGE ALLOC. TABLE/17H MODULE NO.--NO. ,
     8 52HOF MODULE IN WHICH COMMON BLOCK DECLARATION APPEARS.//6H INDEX,
     9 3X,10HBLOCK NAME,5X,15HSTALTB TOP PTR.,5X,16HSTALTB BOT. PTR.,5X,
     1 10HMODULE NO./)
      DO 900 I=1,INDCN
      IWD3=CNTAB(3,I)
      ITOP=HIFECH(IWD3)
```

```
      I=2
      J=7
      CALL SHIFTY(X,Z,I,J)
      IF(.NOT.ALPHA(X(1)))GO TO 2
      NAME(1)=X(1)
      NAME(2)=X(2)
      KC=6
      KT=5
      KU=27
      CALL TABS(KC,KT,KU)
    2 JISS=JISS+1
      IF(JISS.GE.INS)RETURN
      IF(ISS(JISS).NE.HCO)GO TO 3
      JISS=JISS+1
      JTST=JTST+1
      CALL IOLIST(IOF)
      RETURN
    3 CALL ERROR(31)
      RETURN
    4 IF((TSTAB(2,JTST).EQ.HEOUT).AND.(TSTAB(1,JTST+1).EQ.HPUTT))GO TO 5
      CALL ERROR(30)
      RETURN
C*****************************************************************C
C     STATEMENT FORM:  WRITE OUTPUT TAPE UNIT,F,LIST                C
C  IF VARIABLE UNIT NAME FOUND, STORE IN TABLES                     C
C  IF UNIT NUMBER FOUND, IGNORE                                     C
C*****************************************************************C
    5 JTST=JTST+1
      Z(1)=TSTAB(2,JTST)
      Z(2)=HBL
      Z(3)=HBL
      JTST=JTST+1
      JISS=JISS+1
      IF(ISS(JISS).EQ.HCO)GO TO 6
      Z(2)=TSTAB(1,JTST)
      Z(3)=TSTAB(2,JTST)
      JTST=JTST+1
      JISS=JISS+1
    6 I=4
      J=7
      CALL SHIFTY(X,Z,I,J)
      IF(.NOT.ALPHA(X(1)))GO TO 7
      NAME(1)=X(1)
      NAME(2)=X(2)
      KC=10
      KT=0
      KU=26
      CALL TABS(KC,KT,KU)
C*****************************************************************C
C  STORE FORMAT SPECIFICATION VARIABLE OR FORMAT STATEMENT NUMBER  C
C*****************************************************************C
    7 IF(ISS(JISS).NE.HCO)GO TO  8
      JISS=JISS+1
      IF(ISS(JISS).EQ.HV)KC=6
      IF(ISS(JISS).EQ.HI)KC=5
      NAME(1)=TSTAB(1,JTST)
```

```
C     WRITE (ARGUMENT LIST) LIST                                        C
C     WRITE OUTPUT TAPE UNIT, F, LIST                                   C
C     WRITE TAPE UNIT, LIST                                             C
C       WHERE ARGUMENT LIST HAS THE FOLLOWING FORM:                     C
C             UNIT, F, ERR=L1, END=L2                                   C
C          F    = FORMAT LABEL                                          C
C          LIST = LIST OF OUTPUT VARIABLE                               C
C          L1   = NEXT STATEMENT FOR EXECUTION IF ERROR DETECTED IN     C
C                 WRITE                                                 C
C          L2   = NEXT STATEMENT FOR EXECUTION IF EOF                   C
C  UPON ENTRY, POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS ISS,TSTAB     C
C  ,RESP.) POINT TO THE FIRST SYMBOL OF THE STATEMENT, I.E. THE TSTAB   C
C  ENTRY CONTAINS ONE OF THE FOLLOWING STRINGS :  WRITE    ,            C
C   WRITETAP ,  WRITEOUT                                                C
C                                                                       C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES         C
C        EXTRACTED FROM FORTRAN STATEMENTS                              C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS       C
C  KC,KT,KU - PARAMETERS TO SUBROUTINE TABS INDICATING THE CLASS,TYPE   C
C  AND USE CODES FOR THE NAME IN QUESTION                               C
C  IOF - PARAMETER TO SUBROUTINE IOLIST INDICATING THE USE CODE FOR     C
C  VARIABLES IN THE I/O LIST (IOF=3 INDICATES EXTERNAL OUTPUT VARIABLE)C
C***********************************************************************C
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      DIMENSION X(3),Z(3)
      LOGICAL ALPHA
      DATA HETAP/4HETAP/,HEOUT/4HEOUT/,HPUTT/4HPUTT/
      IOF=3
      JISS=JISS+1
      IF(ISS(JISS).NE.HLP)GO TO 1
C***********************************************************************C
C     STANDARD FORM OF WRITE:  WRITE(UNIT,Z,ERR=L1,END=L2)LIST          C
C  PROCESS ARGUMENTS AND OUTPUT VARIABLE LIST                           C
C***********************************************************************C
      CALL IOARGS
      IF(JISS.GE.INS)RETURN
      JISS=JISS+1
      CALL IOLIST(IOF)
      RETURN
    1 IF(TSTAB(2,JTST).NE.HETAP)GO TO 4
C***********************************************************************C
C     STATEMNET FORM:  WRITE TAPE UNIT,LIST                             C
C  IF VARIABLE UNIT NAME FOUND, STORE IN TABLES                         C
C  IF UNIT NUMBER FOUND, IGNORE                                         C
C  PROCESS OUTPUT VARIABLE LIST                                         C
C***********************************************************************C
      IF(ISS(JISS).NE.HV)GOTO 3
      JTST=JTST+1
      Z(1)=TSTAB(1,JTST)
      Z(2)=TSTAB(2,JTST)
      Z(3)=HBL
```

```
C      TYPE   = INTEGER, REAL, LOGICAL, DOUBLE PRECISION, COMPLEX,       C
C               NEUTRAL                                                  C
C      USE    = DECLARATION OR ENTRY IN TYPE STATEMENT                   C
C***********************************************************************C
       KC=4
       NAME(1)=X(1)
       NAME(2)=X(2)
C***********************************************************************C
C      IF TYPE CODE NOT NEUTRAL (5), THE FUNCTION NAME HAS BEEN         C
C      TYPED, PASS FUNCTION NAME TO TABS WITH USE CODE 14 (ENTRY IN     C
C      TYPE STATEMENT).                                                 C
C***********************************************************************C
       CALL TABS (KC, KT, KU)
       IF (KT.EQ.5) GO TO 101
       KU = 14
       CALL TABS (KC, KT, KU)
  101 CONTINUE
C***********************************************************************C
C  CHECK FOR END OF STATEMENT                                           C
C***********************************************************************C
    1  IF(JISS.GT.INS)RETURN
       IF(ISS(JISS).EQ.HLP)GO TO 2
       CALL ERROR(21)
       RETURN
C***********************************************************************C
C  STORE NAMES OF FORMAL PARAMETERS                                     C
C      CLASS = VAR                                                      C
C      TYPE  = NEUTRAL                                                  C
C      USE   = FUNCTION DUMMY PARAMETER                                 C
C***********************************************************************C
    2  ARGNO=0
       KC=6
       KT=5
       KU=16
    3  JISS=JISS+1
       IF(JISS.GE.INS) GO TO 5
       IF(ISS(JISS).NE.HV)GO TO 3
       ARGNO=ARGNO+1
       NAME(1)=TSTAB(1,JTST)
       NAME(2)=TSTAB(2,JTST)
       CALL TABS(KC,KT,KU)
       JTST = JTST + 1
       GO TO 3
C***********************************************************************C
C  STORE END OF PARAMETER LIST CODE IN PARTAB                           C
C***********************************************************************C
5      MPOP=1
       MODNO=MN
       CALL TABS(KC,KT,KU)
       RETURN
       END
*DECK FWRITE
       SUBROUTINE FWRITE
       IMPLICIT INTEGER(A-Z)
C***********************************************************************C
C  SUBROUTINE FWRITE IS A PARSING ROUTINE WHICH PROCESSES WRITE STMNTS.C
```

```
      IF(X(2).NE.HTICN)GO TO 5
C*****************************************************************C
C  TRANSFER PROCESSING TO SUBROUTINE FUNCI                        C
C*****************************************************************C
      X(1)=X(3)
      X(2)=HBL
      X(3)=HBL
      KU=0
      CALL FUNCI(X,KT,KU)
      RETURN
C*****************************************************************C
C  PROCESS VARIABLE LIST                                          C
C*****************************************************************C
    5 CALL LISTP(X,KT,KU)
      RETURN
      END
#DECK FUNCI
      SUBROUTINE FUNCI(X,KT,KU)
      IMPLICIT INTEGER(A-Z)
C*****************************************************************C
C  SUBROUTINE FUNCI IS A PARSING ROUTINE WHICH PROCESSES FUNCTION STS. C
C  UPON ENTRY, POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS ISS,TSTAB   C
C  ,RESP.) POINT TO THE SYMBOL FOLLOWING THE FUNCTION NAME.           C
C  IF THE STATEMENT CONTAINS A SYMBOL IN THIS POSITION, IT MUST BE  ( .C
C                                                                     C
C  INPUT VARIABLES                                                    C
C     X  - 3 WORD ARRAY CONTAINING THE FUNCTION NAME (PASSED FROM     C
C          PARSER OR TYPE STATEMENT PROCESSOR)                        C
C     KT - TYPE CODE OF FUNCTION NAME                                 C
C     KU - USE CODE OF FUNCTION NAME                                  C
C                                                                     C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES       C
C        EXTRACTED FROM FORTRAN STATEMENTS                            C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS     C
C  NSUB  - MODULE SEQUENCE NUMBER                                     C
C  NFILE - MODULE FILE NUMBER.                                        C
C  MN    - MODULE NUMBER.                                             C
C  KC,KT,KU - PARAMETERS TO SUBROUTINE TABS INDICATING THE CLASS,TYPE C
C  AND USE CODES FOR THE NAME IN QUESTION                             C
C*****************************************************************C
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOS,NFILE,ARGNO
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      DIMENSION X(3)
C*****************************************************************C
C  ASSIGN MODULE AND FILE NUMBERS                                 C
C*****************************************************************C
      NSUB=NSUB+1
      MN=NSUB+300
      NFILE=NFILE+1
C*****************************************************************C
C  STORE FUNCTION NAME                                            C
C     CLASS = EXTERNAL FUNCTION NAME                              C
```

```
      IF(JISS.NE.INS)GO TO 11
      Z(2)=TSTAB(1,JTST)
      Z(3)=TSTAB(2,JTST)
      J=7
   11 CALL SHIFTY(X,Z,I,J)
      KC=6
      NAME(1)=X(1)
      NAME(2)=X(2)
      CALL TABS(KC,KT,KU)
      RETURN
    1 IF(ISS(JISS).EQ.HV)GO TO 2
C*************************************************************************C
C  CALL LISTP TO PROCESS VARIABLE LIST                                    C
C*************************************************************************C
      Z(2)=HBL
      Z(3)=HBL
      I=4
      J=1
      CALL SHIFTY(X,Z,I,J)
      CALL LISTP(X,KT,KU)
      RETURN
C*************************************************************************C
C  CHECK FOR FUNCTION STATEMENT                                           C
C*************************************************************************C
    2 JISS=JISS+1
      IF(ISS(JISS).NE.HV)GO TO 4
      IF(TSTAB(1,JTST).EQ.HUNCT)GO TO 3
      CALL ERROR(14)
      RETURN
C*************************************************************************C
C  FUNCTION NAME IS LONGER THAN ONE CHARACTER                             C
C  EXTRACT FUNCTION NAME AND TRANSFER PROCESSING TO SUBROUTINE FUNCT      C
C*************************************************************************C
    3 Z(1)=TSTAB(2,JTST)
      JTST=JTST+1
      Z(2)=TSTAB(1,JTST)
      Z(3)=TSTAB(2,JTST)
      I=4
      J=7
      CALL SHIFTY(X,Z,I,J)
      JISS=JISS+1
      JTST=JTST+1
      KU=0
      CALL FUNCT(X,KT,KU)
      RETURN
C*************************************************************************C
C  CHECK FOR FUNCTION STATEMENT, FUNCTION NAME IS ONLY  ONE               C
C  CHARACTER LONG                                                         C
C*************************************************************************C
    4 Z(2)=TSTAB(1,JTST)
      Z(3)=TSTAB(2,JTST)
      I=4
      J=9
      CALL SHIFTY(X,Z,I,J)
      JTST=JTST+1
      IF(X(1).NE.HFUNC)GO TO 5
```

```
      NAME(2)=TSTAB(2,JTST)
      CALL TABS(KC,KT,KU)
      JTST=JTST+1
      GO TO 2
C*********************************************************************C
C  STORE END OF PARAMETER LIST CODE IN PARTAB.                        C
C*********************************************************************C
5     MPOP=1
      MCONO=MN
      CALL TABS(KC,KT,KU)
      RETURN
      END
*DECK FTYPE
      SUBROUTINE FTYPE(KT)
      IMPLICIT INTEGER(A-Z)
C*********************************************************************C
C  SUBROUTINE FTYPE IS A PARSING ROUTINE WHICH PROCESSES THE FOLLOWING C
C  TYPE STATEMENTS : INTEGER, LOGICAL, COMPLEX, DOUBLE PRECISION      C
C  UPON ENTRY, POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS ISS,TSTAB   C
C  ,RESP.) POINT TO THE FIRST SYMBOL OF THE STATEMENT, UNLESS THE     C
C  STATEMENT IS A DOUBLE PRECISION STATEMENT, IN WHICH CASE THEY POINT C
C  TO THE SECOND SYMBOL.                                              C
C  IF THIS IS A FUNCTION DECLARATION, PROCESSING IS PASSED TO FUNCI    C
C  OTHERWISE LISTP IS CALLED TO PROCESS THE VARIABLE LIST.            C
C                                                                     C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES       C
C        EXTRACTED FROM FORTRAN STATEMENTS                            C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS     C
C  KT - PARAMETER INDICATING THE TYPE CODE FOR VARIABLES IN THE TYPE  C
C  STATEMENT. (PARSER SETS KT WHEN THE ST. TYPE IS DETERMINED)        C
C  KC,KT,KU - PARAMETERS TO SUBROUTINE TABS INDICATING THE CLASS,TYPE, C
C  AND USE CODES FOR THE NAME IN QUESTION                             C
C*********************************************************************C
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      DIMENSION X(3),Z(3)
      DATA HUNCT/4HUNCT/,HFUNC/4HFUNC/,HTION/4HTION/
      Z(1)=TSTAB(2,JTST)
      JISS=JISS+1
      JTST=JTST+1
      KU=14
      IF(JISS.LT.INS)GO TO 1
C*********************************************************************C
C  EXTRACT AND STORE SINGLE VARIABLE NAME                             C
C     CLASS = VARIABLE                                                C
C     TYPE  = COMPLEX, LOGICAL, DOUBLE PRECISION, OR INTEGER          C
C     USE   = ENTRY IN TYPE STATEMENT                                 C
C*********************************************************************C
      Z(2)=HBL
      Z(3)=HBL
      I=4
      J=1
```

```
      DIMENSION X(3),Z(3)
C*****************************************************************C
C  ASSIGN MODULE AND FILE NUMBERS                                 C
C*****************************************************************C
      NSUB=NSUB+1
      MN=NSUB+200
      NFILE=NFILE+1
C*****************************************************************C
C  EXTRACT AND STORE SUBROUTINE NAME                              C
C      CLASS = SUBROUTINE NAME                                    C
C      TYPE  = NEUTRAL                                            C
C      USE   = DECLARATION                                        C
C*****************************************************************C
      JISS=JISS+1
      JTST=JTST+1
      I=3
      J=6
      Z(1)=TSTAB(1,JTST)
      Z(2)=TSTAB(2,JTST)
      Z(3)=HBL
C*****************************************************************C
C  CHECK TO SEE IF THE S  NAME    SPANS OVER THE NEXT ROW OF TSTAB  C
C*****************************************************************C
      IF (JISS .EQ. INS) GO TO 23
      IF (ISS(JISS+1).NE.HV) GO TO 23
      JTST = JTST + 1
      JISS = JISS +1
      Z(3) = TSTAB(1, JTST)
      J = 7
   23 CALL SHIFTY (X, Z, I, J)
      NAME(1)=X(1)
      NAME(2)=X(2)
      KC=1
      KT=5
      KU=0
      CALL TABS(KC,KT,KU)
      KU=17
      KC=6
      JISS=JISS+1
      JTST=JTST+1
C*****************************************************************C
C  CHECK FOR END OF STATEMENT                                     C
C*****************************************************************C
      IF(JISS.GT.INS)RETURN
      IF(ISS(JISS).EQ.HLP)GO TO 1
      CALL ERROR(22)
      RETURN
C*****************************************************************C
C  STORE NAMES OF FORMAL PARAMETERS                               C
C*****************************************************************C
    1 ARGNO=0
    2 JISS=JISS+1
      IF(JISS.GE.INS) GO TO 5
      IF(ISS(JISS).NE.HV)GO TO 2
      ARGNO=ARGNO+1
      NAME(1)=TSTAB(1,JTST)
```

```
      RETURN
C*****************************************************************C
C  CHECK TO SEE IF THE VARIABLE NAME  SPANS OVER THE NEXT ROW OF TSTAB C
C  AND WHETHER THE STRING FUNCTION IS FOUND                       C
C*****************************************************************C
    2 IF(ISS(JISS).NE.HV)GO TO 1
      IF(X(1).EQ.HFUNC)GO TO 4
    3 X(2)=TSTAB(1,JTST)
      JTST=JTST+1
      JISS=JISS+1
      GO TO 1
    4 IF(TSTAB(1,JTST).NE.HTICN)GO TO 3
C*****************************************************************C
C     STATEMENT FORM:  REAL FUNCTION ...                          C
C  EXTRACT FUNCTION NAME AND TRANSFER PROCESSING TO SUBROUTINE FUNCI.  C
C     USE = DECLARATION                                           C
C*****************************************************************C
      KU=0
      X(1)=TSTAB(2,JTST)
      X(2)=HBL
      JTST=JTST+1
      JISS=JISS+1
      IF(JISS.LE.INS)GO TO 5
      CALL FUNCI(X,KT,KU)
      RETURN
    5 IF(ISS(JISS).NE.HV)GO TO 6
      X(2)=TSTAB(1,JTST)
      JTST=JTST+1
      JISS=JISS+1
    6 CALL FUNCI(X,KT,KU)
      RETURN
      END
*DECK FSUB
      SUBROUTINE FSUB
      IMPLICIT INTEGER(A-Z)
C*****************************************************************C
C  SUBROUTINE FSUB IS A PARSING ROUTINE WHICH PROCESSES SUBROUTINE STS.C
C  UPON ENTRY,POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS ISS,TSTAB     C
C  ,RESP.) POINT TO THE FIRST SYMBOL OF THE STATEMENT, I.E. THE TSTAB  C
C  ENTRY CONTAINS THE STRING  SUBROUTI .                          C
C                                                                 C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES   C
C        EXTRACTED FROM FORTRAN STATEMENTS                        C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS C
C  NSUB - MODULE SEQUENCE NUMBER                                  C
C  NFILE - MODULE FILE NUMBER                                     C
C  MN    - MODULE NUMBER                                          C
C  KC,KT,KU - PARAMETERS TO SUBROUTINE TABS INDICATING THE CLASS,TYPE, C
C  AND USE CODES FOR THE NAME IN QUESTION                         C
C*****************************************************************C
      COMMON/SCARA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TARAS/NAME(2),STNO,DIM(3),MN,MPOS,NFILE,ASGN3
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
```

```
      NAME(2)=TSTAB(2,JTST)
      KT=5
      KU=27
      CALL TAPS(KC,KT,KU)
      JISS=JISS+1
      JTST=JTST+1
      IF(ISS(JISS).EQ.HCO)GO TO 16
      CALL ERROR(29)
      RETURN
C*****************************************************************C
C     PROCESS INPUT LIST
C*****************************************************************C
   16 JISS=JISS+1
      CALL IOLIST(IOF)
      RETURN
      END
*DECK FREAL
      SUBROUTINE FREAL
      IMPLICIT INTEGER(A-Z)
C*****************************************************************C
C  SUBROUTINE FREAL IS A PARSING ROUTINE WHICH PROCESSES REAL STMNTS.  C
C  THE STATEMENT CAN BE A TYPE DECLARATION OR REAL FUNCTION            C
C  DECLARATION                                                         C
C  UPON ENTRY, POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS ISS,TSTAB    C
C  ,RESP.) POINT TO THE FIRST SYMBOL OF THE STATEMENT, I.E. THE TSTAB  C
C  ENTRY CONTAINS THE STRING  REAL N , WHERE N REPRESENTS ALL OR PART  C
C  OF THE CHARACTERS OF THE FIRST VARIABLE IN A LIST, OR  REALFUNC     C
C  WHICH MAY INDICATE THE STATEMENT IS ACTUALLY A FUNCTION STATEMENT.  C
C                                                                      C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES        C
C        EXTRACTED FROM FORTRAN STATEMENTS                             C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS      C
C*****************************************************************C
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HCU,HSL,HPE,HFORM,HAT,HG,HH,HI
      DIMENSION X(3)
      DATA HFUNC/4HFUNC/,HTION/4HTION/
C*****************************************************************C
C  INITIALIZE  TYPE = REAL                                             C
C              USE  = TYPE ST. ENTRY                                   C
C*****************************************************************C
      KU=14
      KT=1
      JISS=JISS+1
      X(1)=TSTAB(2,JTST)
      X(2)=HBL
      JTST=JTST+1
      IF(JISS.LE.INS)GO TO 2
C*****************************************************************C
C  PROCESS TYPE LIST                                                   C
C*****************************************************************C
    1 CALL LISTP(X,KT,KU)
```

```
      KC=6
      KT=5
      KU=27
      CALL TABS(KC,KT,KU)
      JISS=JISS+1
      JTST=JTST+1
      IF(ISS(JISS).EQ.HCO)GO TO 11
      CALL ERROR(28)
      RETURN
   10 NAME(1)=TSTAB(2,JTST)
      NAME(2)=HBL
      KC=6
      KT=5
      KU=27
      CALL TABS(KC,KT,KU)
      JTST=JTST+1
   11 JISS=JISS+1
      CALL IOLIST(IOF)
      RETURN
   12 IF(TSTAB(2,JTST).NE.HINPU)GO TO 9
      IF(TSTAB(1,JTST+1).NE.HTTAP)GOTO 9
C*************************************************************************C
C     STATEMENT FORM:  READ INPUT TAPE UNIT,F,LIST
C     IF VARIABLE UNIT NAME FOUND,STORE IN TABLES
C     IF UNIT NUMBER FOUND,IGNORE
C*************************************************************************C
      JTST=JTST+1
      Z(1)=TSTAB(2,JTST)
      Z(2)=HBL
      Z(3)=HBL
      JTST=JTST+1
      JISS=JISS+1
      IF(ISS(JISS).EQ.HCO)GO TO 13
      Z(2)=TSTAB(1,JTST)
      JTST=JTST+1
      JISS=JISS+1
   13 I=2
      J=7
      CALL SHIFTY(X,Z,I,J)
      IF(.NOT.ALPHA(X(1)))GO TO 14
      NAME(1)=X(1)
      NAME(2)=X(2)
      KC=10
      KT=0
      KU=26
      CALL TABS(KC,KT,KU)
   14 IF(ISS(JISS).EQ.HCO)GO TO 15
      CALL ERROR(29)
      RETURN
C*************************************************************************C
C     STORE FORMAT STATEMENT NUMBER OR FORMAT SPECIFICATION VARIABLE
C*************************************************************************C
   15 JISS=JISS+1
      IF(ISS(JISS).EQ.HV)KC=6
      IF(ISS(JISS).EQ.HI)KC=5
      NAME(1)=TSTAB(1,JTST)
```

```
      IF(ISS(JISS).NE.HI)GO TO 2
      NAME(2)=TSTAB(1,JTST)
      JTST=JTST+1
      JISS=JISS+1
    2 KC=5
      KT=5
      KU=27
      CALL TABS(KC,KT,KU)
      IF(ISS(JISS).EQ.HCO)GO TO 3
      CALL ERROR(26)
      RETURN
    3 JISS=JISS+1
      CALL IOLIST(IOF)
      RETURN
    4 IF(TSTAB(2,JTST).NE.HTAPE)GO TO 8
C*****************************************************************C
C     STATEMENT FORM:  READ TAPE UNIT,LIST                        C
C     IF VARIABLE UNIT NAME FOUND, STORE IN TABLES                C
C     CLASS = PARAMETER VARIABLE                                  C
C     TYPE  = INTEGER                                             C
C     USE   = I/O UNIT REF.                                       C
C     IF UNIT NUMBER FOUND, IGNORE                                C
C  PROCESS INPUT VARIABLE LIST                                    C
C*****************************************************************C
      JTST=JTST+1
      IF(ISS(JISS).EQ.HV)GO TO 5
      IF(ISS(JISS).EQ.HI)GO TO 6
      CALL ERROR(27)
      RETURN
    5 NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      KC=10
      KT=0
      KU=26
      CALL TABS(KC,KT,KU)
    6 JTST=JTST+1
      JISS=JISS+1
      IF(ISS(JISS).EQ.HCO)GO TO 7
      CALL ERROR(27)
      RETURN
    7 JISS=JISS+1
      CALL IOLIST(IOF)
      RETURN
    8 IF(ISS(JISS).EQ.HV)GO TO 12
C*****************************************************************C
C     STATEMENT FORM:  READ F,LIST                                C
C     EXTRACT AND STORE FORMAT SPECIFICATION VARIABLE             C
C  AND PROCESS INPUT VARIABLE LIST                                C
C     CLASS = VARIABLE                                            C
C     TYPE  = NEUTRAL                                             C
C     USE   = FORMAT REFERENCE.                                   C
C*****************************************************************C
      IF(ISS(JISS).EQ.HCO)GO TO 10
    9 NAME(1)=TSTAB(2,JTST)
      JTST=JTST+1
      NAME(2)=TSTAB(1,JTST)
```

```
C  SUBROUTINE FREAD IS A PARSING ROUTINE WHICH PROCESSES READ STMNTS.  C
C     READ (ARGUMENT LIST) LIST                                       C
C     READ INPUT TAPE UNIT, F, LIST                                   C
C     READ F, LIST                                                    C
C     READ TAPE UNIT, LIST                                            C
C          WHERE ARGUMENT LIST HAS THE FORM                           C
C              UNIT, F, ERR=L1, END =L2                               C
C                 F = FORMAT LABEL                                    C
C                 LIST = LIST OF INPUT VARIABLES                      C
C                 L1   = NEXT STATEMENT EXECUTED IF ERROR DETECTED IN C
C                        READ                                         C
C                 L2   = NEXT STATEMENT FOR EXECUTION IF EOF.         C
C                 UNIT = TAPE UNIT NUMBER                             C
C  UPON ENTRY, POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS ISS,TSTAB    C
C  ,RESP.) POINT TO THE FIRST SYMBOL OF THE STATEMENT, I.E. THE TSTAB  C
C  ENTRY IS ONE OF THE FOLLOWING :  READ     ,  READ N ,  READTAPE ,   C
C   READINPU ,  READ F                                                 C
C                                                                      C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES        C
C        EXTRACTED FROM FORTRAN STATEMENTS                            C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS      C
C*********************************************************************C
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/SCRPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,H0P,H0Q,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      DIMENSION X(3),Z(3)
      DATA HTAPE/4HTAPE/,HINPL/4HINPU/,HTTAP/4HTTAP/
      LOGICAL ALPHA
      IOF=4
      JISS=JISS+1
      IF(ISS(JISS).NE.HLP)GO TO 1
C*********************************************************************C
C     STANDARD FORM OF READ:  READ(UNIT,Z,ERR=L1,END=L2)LIST          C
C  PROCESS I/O ARGUMENTS, CHECK FOR END OF STATEMENT, PROCESS INPUT    C
C  VARIABLE LIST                                                       C
C*********************************************************************C
      CALL ICARGS
      IF(JISS.GE.INS)RETURN
      JISS=JISS+1
      CALL IOLIST(IOF)
      RETURN
    1 IF(ALPHA(TSTAB(2,JTST)))GO TO 4
C*********************************************************************C
C     STATEMENT FORM:  READ N,LIST                                     C
C     EXTRACT AND STORE FORMAT STATEMENT NUMBER                        C
C  AND PROCESS INPUT VARIABLE LIST                                     C
C     CLASS = UNDEFINED LABEL                                          C
C     TYPE  = NEUTRAL                                                  C
C     USE   = FORMAT REFERENCE                                         C
C*********************************************************************C
      NAME(1)=TSTAB(2,JTST)
      JTST=JTST+1
      NAME(2)=HBL
```

```
      J=3
C*****************************************************************C
C  IF STATEMENT IN THE FORM :  PRINT FORMAT LABEL                 C
C                              PUNCH FORMAT LABEL                 C
C              GO TO 2                                            C
C*****************************************************************C
      IF(JISS.GE.INS)GO TO 2
      JTST=JTST+1
      JISS=JISS+1
      IF(ISS(JISS).EQ.HCO)GO TO 1
      Z(2)=TSTAB(1,JTST)
      JTST=JTST+1
      JISS=JISS+1
      J=7
      IF(JISS.GE.INS)GO TO 2
      IF(ISS(JISS).EQ.HCO)GO TO 1
      CALL ERROR(37)
      RETURN
C*****************************************************************C
C  EXTRACT AND STORE FORMAT REFERENCE                             C
C     CLASS = UNDEFINED LABEL OR VARIABLE                         C
C     TYPE  = NEUTRAL                                             C
C     USE   = FORMAT REFERENCE                                    C
C*****************************************************************C
    1 CALL SHIFTY(X,Z,I,J)
      KC=5
      IF(ALPHA(X(1)))KC=6
      KT=5
      KU=27
      NAME(1)=X(1)
      NAME(2)=X(2)
      CALL TABS(KC,KT,KU)
      JISS=JISS+1
C*****************************************************************C
C  PROCESS OUTPUT VARIABLE LIST                                   C
C*****************************************************************C
      IOF=3
      CALL IOLIST(IOF)
      RETURN
C*****************************************************************C
C  EXTRACT AND STORE FORMAT REFERENCE                             C
C*****************************************************************C
    2 CALL SHIFTY(X,Z,I,J)
      KC=5
      IF(ALPHA(X(1)))KC=6
      KT=5
      KU=27
      NAME(1)=X(1)
      NAME(2)=X(2)
      CALL TABS(KC,KT,KU)
      RETURN
      END
*DECK FREAD
      SUBROUTINE FREAD
      IMPLICIT INTEGER(A-Z)
C*****************************************************************C
```

```
      IF(ISS(JISS).NE.HV)GOTO 1
      IF(TSTAB(2,JTST).NE.HFUNC)GOTO 1
      IF(TSTAB(1,JTST+1).NE.HTION ) GO TO 1
C*****************************************************************C
C     IF THERE IS A DECLARATION SEQUENCE ERROR, REPRINT THE MODULE  C
C     DECLARATION CARD.                                             C
C*****************************************************************C
  200 IF ( DECERR )  GO TO 300
      IF ( L.EQ.1 ) LPRT = 2
      IF ( L.EQ.2 ) LPRT = 1
      PRINT 250, STNO, ( STORE (LPRT,K ), K = 1, 80)
  250 FORMAT (1H1, I4, 5X, 80A1)
  300 RETURN
C*****************************************************************C
C     NEXT CARD NOT A MODULE DECLARATION CARD, SET DECLARATION      C
C     ERROR FLAG TO FALSE. IGNORE SUBSEQUENT STATEMENTS UNTIL A     C
C     MODULE DECLARATION CARD IS REACHED.                           C
C*****************************************************************C
    1 DECERR = .FALSE.
      GO TO 30
      END
*DECK FPRPU
      SUBROUTINE FPRPU
      IMPLICIT INTEGER(A-Z)
C*****************************************************************C
C  SUBROUTINE FPRPU IS A PARSING ROUTINE WHICH PROCESSES PRINT/PUNCH STS
C          PRINT F, LIST                                          C
C          PUNCH F, LIST                                          C
C     WHERE F = THE FORMAT LABEL                                  C
C           LIST = ARGUMENT LIST                                  C
C  UPON ENTRY, POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS ISS,TSTAB C
C  ,RESP.) POINT TO THE FIRST SYMBOL OF THE STATEMENT, I.E. THE TSTAB C
C  ENTRY IS ONE OF THE FOLLOWING STRINGS :  PRINT N ,  PUNCH N  ,  C
C  WHERE N REPRESENTS ALL OR PART OF THE CHARACTERS MAKING UP A   C
C  FORMAT STATEMENT NUMBER OR SPECIFICATION ARRAY.                C
C                                                                 C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES   C
C        EXTRACTED FROM FORTRAN STATEMENTS                        C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS C
C  KC,KT,KU - PARAMETERS TO SUBROUTINE TABS INDICATING THE CLASS,TYPE, C
C  AND USE CODES FOR THE NAME IN QUESTION                         C
C  IOF - PARAMETER TO SUBRCUTINE IOLIST INDICATING THE USE CODE FOR C
C  VARIABLES IN THE OUTPUT LIST (IOF=3 INDICATES EXTERNAL OUTPUT VAR.) C
C*****************************************************************C
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAP/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      LOGICAL ALPHA
      DIMENSION X(3),Z(3)
      Z(1)=TSTAB(2,JTST)
      Z(2)=HBL
      Z(3)=HBL
      I=2
```

```
      JTST=1
      IF(ISS(1).EQ.HS) GO TO 1
      IF(EFLAG.NE.0)GOTO 1
      TEMP=TSTAB(1,JTST)
C*****************************************************************C
C     INDEX - STATEMENT TYPE CODE                                   C
C*****************************************************************C
      INDEX=HASHIN(TEMP)
      IF(TEMP.EQ.MATCH(INDEX))GO TO 5
      INDEX=BIAS
    6 INDEX=INDEX+1
      IF(TEMP.EQ.MATCH(INDEX))GO TO 5
      IF(MATCH(INDEX).NE.0)GO TO 6
      GO TO 1
    5 CONTINUE
C*****************************************************************C
C     CHECK FOR SUBROUTINE,FUNCTION, BLOCK DATA, OR PROGRAM STATEMENT
C*****************************************************************C
      IF(INDEX.EQ.30.OR.INDEX.EQ.27.OR.INDEX.EQ.13.OR.INDEX.EQ.32)
     1 GO TO 200
      IF(INDEX.EQ.35.OR.INDEX.EQ.26.OR.INDEX.EQ.11)GOTO 2
      IF(INDEX.EQ.33)GOTO 3
      IF(INDEX.EQ.23)GOTO4
C*****************************************************************C
C     INTEGER,LOGICAL,OR COMPLEX
C     CHECK FOR FUNCTION STATEMENT
C*****************************************************************C
    2 JISS=JISS+1
      IF(JISS.GT.INS)GOTO 1
      IF(ISS(JISS).NE.HV)GOTO 1
      Z(1)=TSTAB(2,JTST)
      JTST=JTST+1
C*****************************************************************C
C     CALL SHIFTY TO EXTRACT CHARACTERS 4 TO 8 AND PUT RESULT IN    C
C     X(1) AND X(2), LEFT JUSTIFIED AND FOUR CHAR. PER WORD.        C
C*****************************************************************C
      Z(2)=TSTAB(1,JTST)
      Z(3)=TSTAB(2,JTST)
      I=4
      J=8
      CALL SHIFTY(X,Z,I,J)
      IF ( (X(1) .EQ. HFUNC) .AND. (X(2) .EQ. HTION) )  GO TO 200
      GOTO 1
C*****************************************************************C
C     DOUBLE PRECISION
C     CHECK FOR FUNCTION STATEMENT
C*****************************************************************C
    3 JISS=JISS+1
      JTST=JTST+1
      GOTO 2
C*****************************************************************C
C     REAL
C     CHECK FOR FUNCTION STATEMENT.
C*****************************************************************C
    4 JISS=JISS+1
      IF(JISS.GT.INS)GOTO 1
```

```
      CALL TYPCHK(NAM,ITYPE)
      SYMTAB(3,I)=HISTOR(TEMP,ITYPE)
  200 CONTINUE
      RETURN
      END
*DECK FILSET
      SUBROUTINE FILSET
C*********************************************************************C
C     THIS ROUTINE OPENS FILSET RANDOM AND DEFINE INDEX ARRAY USED    C
C     BY TSDISK                                                       C
C     TSDISK IS A MACHINE DEPENDENT ROUTINE                           C
C*********************************************************************C
      DIMENSION INDEX (1000)
      CALL OPDISK (1000, INDEX)
      RETURN
      END
*DECK FRST
      SUBROUTINE FIRST
      IMPLICIT INTEGER (A-Z)
C*********************************************************************C
C     SUBROUTINE FIRST IS CALLED FROM PARSER TO PERFORM A PRE-SCAN    C
C     OF THE FIRST STATEMENT OF A PROGRAM MODULE.  ITS PURPOSE IS     C
C     TO DETERMINE THE MODULE TYPE (ROUTINE,SUBROUTINE,FUNCTION,      C
C     BLOCK DATA). IF FIRST STATEMENT IS NOT A MODULE DECLARATION,    C
C     PERFORM ERROR RECOVERY. IGNORE SUBSEQUENT STATEMENTS UNTIL A    C
C     PROGRAM, SUBROUTINE OR FUNCTION DECLARATION STATEMENT IS REACHED C
C                                                                     C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES       C
C        EXTRACTED FROM FORTRAN STATEMENTS                            C
C     JISS - POINTER TO THE ISS ENTRY WHICH IS BEING ANALYZED CURRENTLYC
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS     C
C     JTST - POINTER TO TSTAB ENTRY BEING ANALYZED                    C
C     MATCH(51) - AN ARRAY CONTAINING KEYWORDS FOR CLASSIFYING FORTRAN C
C                 STATEMENTS.                                         C
C*********************************************************************C
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,SFLAG,ENDFL
      COMMON/TMATCH/BIAS,MATCH(51)
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPDP,NFILE,ARGNO
      COMMON /INSTUF/ STORE (2,80), INP, L, ISAV, ND, LFLAG, NLFLAG,
     1 HFLAG, LBUF (10)
      LOGICAL DECERR
      DIMENSION X(3),Z(3)
      DATA HS/1HS/,HFUNC/4HFUNC/,HTION/4HTION/,HV/1HV/,HBL/1H /
     6 ,MAIN1/4H$MAI/,MAIN2/2HN$/
      DECERR = .TRUE.
      STNO=1
      PRINT 1000
 1000 FORMAT(1H1)
   30 CALL SCAN
C*********************************************************************C
C     IF EOF READ, RETURN.                                            C
C*********************************************************************C
      IF (ENDFL.GE.1) RETURN
      INS=INS-1
      JISS=1
```

```
      N=5
      GO TO 50
45    CALL ERROR(1)
      N=1
50    JISS=JISS+1
C*****************************************************************C
C  GET NEXT LETTER.                                               C
C  CALL MOVBIT TO CONVERT THE ALPHABET INTO AN INDEX OF LTRS.     C
C  SET THE CORRESPONDING BIT OF LTRS TO INDICATE TYPE             C
C*****************************************************************C
55    JTST=JTST+1
      JISS=JISS+1
      LETR=TSTAB(1,JTST)
      LETR=MOVBIT(LETR)
      TEMP=LTRS(LETR)
      LTRS(LETR)=SETONE(N,TEMP)
65    JISS=JISS+1
      IF(ISS(JISS).EQ.CO) GO TO 55
      IF(ISS(JISS).NE.HY) GO TO 90
C*****************************************************************C
C  STATEMENT OF TYPE :  IMPLICIT TYPE (LETTER-LETTER)             C
C  PROCESS RANGE OF LETTERS.                                      C
C*****************************************************************C
      JISS=JISS+1
      JTST=JTST+1
      ENDLET=TSTAB(1,JTST)
      ENDLET=MOVBIT(ENDLET)
70    LETR=LETR+1
      IF(LETR.GT.ENDLET) GO TO 65
      TEMP=LTRS(LETR)
      LTRS(LETR)=SETONE(N,TEMP)
      GO TO 70
C*****************************************************************C
C  PROCESS END OF TYPE SPECIFICATIONS.                            C
C*****************************************************************C
90    IF(ISS(JISS).NE.RP) GO TO 100
      JISS=JISS+1
      IF(JISS.GT.INS) GO TO 100
      IF(ISS(JISS).NE.CO) GO TO 100
C*****************************************************************C
C  PROCESS THE NEXT IMPLICIT DECLARATION                          C
C*****************************************************************C
      JISS=JISS+1
      JTST=JTST+1
      GO TO 5
C*****************************************************************C
C  CHANGE TYPE CODES FOR ALL NAMES IN SYMTAB.                     C
C                                                                 C
C  CONTINUE THE SEARCH IF THE SYMTAB ENTRY IS EMPTY OR THE TYPE CODE  C
C  IS NEUTRAL,ELSE CHECK THE TYPE AND CONTINUE                    C
C*****************************************************************C
100   DO 200 I=1,LSYM
      IF(SYMTAB(1,I).EQ.0) GO TO 200
      TEMP=SYMTAB(3,I)
      IF(HIFECH(TEMP).EQ.5) GO TO 200
      NAM=SYMTAB(1,I)
```

```
      KU = 29
      GO TO 3
      END
*DECK FIMP
      SUBROUTINE FIMP
      IMPLICIT INTEGER (A-Z)
C*****************************************************************C
C  THIS ROUTINE PROCESSES IMPLICIT TYPE STATEMENTS.  AN ARRAY (LTRS) OF
C  26 WORDS (ONE WORD FOR EACH LETTER OF THE ALPHABET) IS USED TO STORE
C  THE TYPE SPECIFICATIONS INTRODUCED BY THE IMPLICIT STATEMENT.  EACH C
C  WORD OF THE ARRAY HAS FIVE 1-BIT FIELDS, ONE FIELD TO DESIGNATE EACH
C  POSSIBLE TYPE SPECIFICATION.                                        C
C          BIT SET                                                     C
C            1       INTEGER                                           C
C            2       REAL                                              C
C            3       DOUBLE PRECISION                                  C
C            4       COMPLEX                                           C
C            5       LOGICAL                                           C
C  UPON ENTRY, POINTERS JISS, JTST(ASSOCIATED WITH ARRAYS ISS AND      C
C  TSTAB RESP.) POINTS TO THE FIRST SYMBOL IN THE STATEMENT, I.E. THE  C
C  TSTAB ENTRY POINTED TO BY JTST CONTAINS THE STRING IMPLICIT.        C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES        C
C        EXTRACTED FROM FORTRAN STATEMENTS                             C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS      C
C                                                                      C
C  FIMP DEPENDS ON THE INTERNAL CHARACTER REPRESENTATION OF THE MACHINEC
C*****************************************************************C
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TYPES/LTRS(26)
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTR(2,100),LSTAL,INDST
      DATA INTE/4HINTE/,DOUB/4HDOUB/,COMP/4HCOMP/,LOGI/4HLOGI/,
     1 REAL/4HREAL/,CO/1H,/,RP/1H)/,HY/1H-/
      JTST=JTST+1
      JISS=JISS+1
C*****************************************************************C
C  CHECK TYPE SPECIFICATION
C*****************************************************************C
5     IF(TSTAB(1,JTST).NE.INTE) GO TO 10
      N=1
      GO TO 50
10    IF(TSTAB(1,JTST).NE.REAL) GO TO 20
      N=2
      GO TO 50
20    IF(TSTAB(1,JTST).NE.DOUB) GO TO 30
      N=3
      JTST=JTST+1
      JISS=JISS+1
      GO TO 50
30    IF(TSTAB(1,JTST).NE.COMP) GO TO 40
      N=4
      GO TO 50
40    IF(TSTAB(1,JTST).NE.LOGI) GO TO 45
```

```
     2 HB,HS,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
       PARAM=0
       JTST=JTST+1
       JISS=JISS+1
       IF(ISS(JISS).EQ.HLP)GO TO 1
C*****************************************************************************C
C  NO LEFT PARENTHESIS IN IF STATEMENT                                        C
C*****************************************************************************C
       CALL ERROR(12)
       RETURN
C*****************************************************************************C
C  CALL SEXP TO PROCESS CONDITIONAL EXPRESSION                                C
C      MODE = 20 TO INDICATE THAT VARIABLES PROCESSED ARE CONDITIONAL         C
C                BRANCH DECISION VARIABLES.                                   C
C*****************************************************************************C
     1 MODE=20
       JISS=JISS+1
       CALL SEXP(MODE)
       JISS=JISS+1
       IF(ISS(JISS).EQ.HI)GO TO 2
C*****************************************************************************C
C  IF NEXT LEXICAL ENTITY NOT AN INTEGER  (STATEMENT LABEL), THEN THE         C
C  IF STATEMENT IS OF THE TYPE : IF (EXP.) STATEMENT.                         C
C  SET PARAM TO ONE AND RECORD TRANSITION BY CALLING TRANS.                   C
C*****************************************************************************C
       PARAM=1
       MOD=4
       CALL TRANS(LOC,MOD)
       RETURN
C*****************************************************************************C
C  STORE TRANSFER LABELS OR SWITCH VARIABLES                                  C
C      CLASS = UNDEFINED LABEL                                                C
C      TYPE  = NEUTRAL                                                        C
C      USE   = LABEL REFERENCE                                                C
C*****************************************************************************C
     2 KC=5
       KT=5
       KU=10
     3 NAME(1)=TSTAB(1,JTST)
       NAME(2)=TSTAB(2,JTST)
       CALL TABS(KC,KT,KU)
       IF(JISS.EQ.INS)RETURN
       JTST=JTST+1
       JISS=JISS+1
       IF(ISS(JISS).EQ.HCC)JISS=JISS+1
C*****************************************************************************C
C  IF   ISS(JISS) = I     STATEMENT LABEL                                     C
C  IF   ISS(JISS) = V     SWITCH VARIABLE                                     C
C  ELSE STORE ERROR MESSAGE (TRANSFER LABEL OR REFERENCE NOT RECOGNIZED       C
C       IN AN IF STATEMENT                                                    C
C*****************************************************************************C
       IF(ISS(JISS).EQ.HI)GO TO 2
       IF(ISS(JISS).EQ.HV)GO TO 4
       CALL ERROR(11)
       RETURN
     4 KC=6
```

```
      GO TO 7
    9 IF(ISS(JISS).EQ.HCO)GO TO 7
C*****************************************************************C
C                                                                 C
C  CHECK FOR END OF LIST                                          C
C*****************************************************************C
      IF(ISS(JISS).EQ.HRP)GO TO 10
      CALL ERROR(9 )
      RETURN
C*****************************************************************C
C                                                                 C
C  CHECK FOR END OF STATEMENT                                     C
C*****************************************************************C
   10 IF(JISS.EQ.INS)RETURN
      JISS=JISS+1
      IF(ISS(JISS).EQ.HCO)GO TO 11
      CALL ERROR(10)
      RETURN
C*****************************************************************C
C     STORE INDEX VARIABLE OF COMPUTED GO TO                      C
C*****************************************************************C
   11 JISS=JISS+1
      IF(ISS(JISS).NE.HV)GO TO 12
      KC=6
      KT=5
      KU=24
      NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      CALL TABS(KC,KT,KU)
      RETURN
   12 CALL ERROR(26)
      RETURN
      END
#DECK FIF
      SUBROUTINE FIF(PARAM)
      IMPLICIT INTEGER(A-Z)
C*****************************************************************C
C  SUBROUTINE FIF IS A PARSING ROUTINE WHICH PROCESSES IF STATEMENTS.  C
C  UPON ENTRY, THE POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS      C
C  ISS,TSTAB RESP.) POINT TO THE FIRST SYMBOL OF THE STATEMENT, I.E.   C
C  THE TSTAB ENTRY CONTAINS THE STRINGS  IF  .                     C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES     C
C        EXTRACTED FROM FORTRAN STATEMENTS                         C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS   C
C  PARAM - PARAMETER PASSED BACK TO PARSER INDICATING TYPE OF IF STMNT.C
C          PARAM = 0  - IF STATEMENT OF TYPE :  IF (EXP.) STATEMENT C
C          PARAM = 1  - IF STATEMENT OF TYPE :  IF (EXP.) N1, N2, N3  C
C  KC,KU,KT - PARAMETERS TO SUBROUTINE TABS INDICATING THE CLASS,TYPE, C
C  AND USE CODES FOR THE NAME IN QUESTION                          C
C  MODE - PARAMETER TO SUBROUTINE SEXP INDICATING THE USE CODE FOR  C
C  THE HIGHEST LEVEL ENTITIES IN THE CONDITIONAL EXPRESSION TO BE   C
C  PROCESSED                                                       C
C*****************************************************************C
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,H$B,H$D,H$S,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
```

```
C      TYPE   = NEUTRAL                                                  C
C      USE    = VARIABLE REFERENCED IN ASSIGNED GOTO                    C
    1 KU=23
      KC=6
      KT=5
      CALL TABS(KC,KT,KU)
      RETURN
    2 IF(NAME(1).EQ.HBL)GO TO 5
C*************************************************************************C
C      STORE SWITCH VARIABLE OF ASSIGNED GO TO                           C
C  STATEMENT OF FORM :  GO TO M, (1, N2....,NK)                          C
C                  WHERE : M = SWITCH VARIABLE                           C
C                          N1, N2,...,NK = STATEMENT NUMBERS.            C
C*************************************************************************C
      IF(ISS(JISS).NE.HV)GO TO 3
      NAME(2)=TSTAB(1,JTST)
      JTST=JTST+1
      JISS=JISS+1
    3 KC=6
      KT=5
      KU=23
      CALL TABS(KC,KT,KU)
      IF(ISS(JISS).EQ.HCO)GO TO 4
      CALL ERROR(6 )
      RETURN
    4 JISS=JISS+1
      TEMP=23
      IF(ISS(JISS).EQ.HLP)GO TO 7
      CALL ERROR(7 )
      RETURN
    5 IF(ISS(JISS).EQ.HLP)GO TO 6
      CALL ERROR(8 )
      RETURN
    6 TEMP=24
C*************************************************************************C
C  COMPUTED GOTO STATEMENT                                                C
C      STORE LIST OF POSSIBLE TRANSFER LABELS                             C
C*************************************************************************C
    7 JISS=JISS+1
      IF(ISS(JISS).NE.HI)GO TO 8
      KC=5
      KT=5
      KU=10
      NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      CALL TABS(KC,KT,KU)
      JTST=JTST+1
      GO TO 7
    8 IF(ISS(JISS).NE.HV)GO TO 9
      KC=6
      KT=5
      KU=TEMP
      NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      CALL TABS(KC,KT,KU)
      JTST=JTST+1
```

```
C*****************************************************************C
C  STORE ENTRY POINT NAME                                         C
C*****************************************************************C
      NAME(1)=X(1)
      NAME(2)=X(2)
      KC=9
      KT=5
      KU=0
      CALL TABS(KC,KT,KU)
      GO TO 2
      END
*DECK FGOTO
      SUBROUTINE FGOTO
      IMPLICIT INTEGER(A-Z)
C*****************************************************************C
C  SUBROUTINE FGOTO IS A PARSING ROUTINE WHICH PROCESSES GO TO STMNTS. C
C  UPON ENTRY, THE POINTERS JISS,JTST, (ASSOCIATED WITH ARRAYS    C
C  ISS,TSTAB, RESP.) POINT TO THE FIRST SYMBOL OF THE STATEMENT, I.E. C
C  THE  TSTAB ENTRY CONTAINS THE STRING  GOTO N  OR  GOTO ---- ,  C
C  WHERE N IS A STATEMENT NUMBER OR SWITCH VARIABLE..             C
C                                                                 C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES   C
C        EXTRACTED FROM FORTRAN STATEMENTS                        C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS C
C  KC,KT,KU - PARAMETERS TO SUBROUTINE TABS INDICATING THE CLASS,TYPE, C
C  AND USE CODES FOR THE NAME IN QUESTION                         C
C*****************************************************************C
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HCU,HSL,HPE,HFORM,HAT,HG,HF,HI
      LOGICAL ALPHA
      JISS=JISS+1
      NAME(1)=TSTAB(2,JTST)
      NAME(2)=HBL
      JTST=JTST+1
      IF(JISS.LT.INS)GO TO 2
C*****************************************************************C
C  STATEMENT OF FORM :  GO TO STATEMENT LABEL OR SWITCH VARIABLE.  C
C      STORE TRANSFER LABEL                                       C
C      CLASS = UNDEFINED LABEL.                                   C
C      TYPE  = NEUTRAL                                            C
C      USE   = LABEL REFERENCE                                    C
C*****************************************************************C
      IF(JISS.EQ.INS)NAME(2)=TSTAB(1,JTST)
      IF(ALPHA(NAME(1)))GO TO 1
      KC=5
      KT=5
      KU=10
      CALL TABS(KC,KT,KU)
      RETURN
C*****************************************************************C
C  SWITCH VARIABLE                                                C
C      CLASS = VARIABLE                                           C
```

```
      IF(JISS.GT.INS)GO TO 3
      IF(ISS(JISS).EQ.HLP)GOTO 1
      Z(2)=TSTAB(1,JTST)
      J=7
      JTST=JTST+1
      JISS=JISS+1
      IF(JISS.GT.INS)GO TO 3
      IF(ISS(JISS).EQ.HLP)GO TO 1
      CALL ERROR(38)
      GO TO 3
    1 CALL SHIFTY(X,Z,I,J)
C*****************************************************************C
C  STORE ENTRY POINT NAME                                         C
C     CLASS = ENTRY POINT NAME                                    C
C     TYPE  = NEUTRAL                                             C
C     USE   = DECLARATION                                         C
C*****************************************************************C
      NAME(1)=X(1)
      NAME(2)=X(2)
      KC=9
      KT=5
      KU=0
      CALL TABS(KC,KT,KU)
C*****************************************************************C
C  PROCESS PARAMETER LIST                                         C
C     CLASS = VARIABLE                                            C
C     TYPE = NEUTRAL                                              C
C     USE = SUBROUTINE OR FUNCTION DUMMY PARAMETERS               C
C*****************************************************************C
      KC = 6
      KT = 5
      KU = 16
      IF ( (200.LT.MN) .AND. (MN.LT.300) ) KU = 17
      ARGNO = 0
  200 JISS = JISS + 1
      IF (JISS .GE. INS) GO TO 500
      IF ( ISS(JISS).NE.HV) GO TO 200
      ARGNO = ARGNO + 1
      NAME(1) = TSTAB (1, JTST)
      NAME (2) = TSTAB (2,JTST)
      CALL TABS (KC, KT, KU)
      JTST = JTST +1
      GO TO 200
C*****************************************************************C
C  STORE END OF PARAMETER LIST CODE IN PARTAB                     C
C*****************************************************************C
  500 MPOP = 1
      CALL TABS (KC, KT, KU)
C*****************************************************************C
C  CREATE TRANSITION TO ENTRY POINT                               C
C*****************************************************************C
    2 LCC=0
      MOD=7
      CALL TRANS(LCC,MOD)
      RETURN
    3 CALL SHIFTY(X,Z,I,J)
```

```
C*************************************************************************C
C  EXTRACT FIRST VARIABLE OF LIST                                         C
C*************************************************************************C
      JISS=JISS+1
      JTST=JTST+1
      I=2
      J=7
      Z(1)=TSTAB(1,JTST)
      Z(2)=TSTAB(2,JTST)
      Z(3)=HBL
      CALL SHIFTY(X,Z,I,J)
      KT=5
      KU=13
      JISS=JISS+1
      JTST=JTST+1
C*************************************************************************C
C  PROCESS VARIABLE LIST                                                  C
C*************************************************************************C
      CALL LISTP(X,KT,KU)
      RETURN
      END
*DECK FENTRY
      SUBROUTINE FENTRY
      IMPLICIT INTEGER (A-Z)
C*************************************************************************C
C  SUBROUTINE FENTRY IS A PARSING ROUTINE WHICH PROCESSES ENTRY STMNTS    C
C  ENTRY STATEMENTS HAVE THE FORM :                                       C
C      ENTRY NAME (P1, P2,....PN)                                         C
C        WHERE  NAME = SYMBOLIC NAME OF ENTRY POINT                       C
C               P1,P2....PN = DUMMY PARAMETERS                            C
C  UPON ENTRY, POINTERS JISS AND JTST (ASSOCIATED WITH ARRAYS ISS AND     C
C  TSTAB RESPECTIVELY) POINT TO THE FIRST SYMBOL OF THE STATEMENT,        C
C  I.E. THE STRING ENTRYNAME                                              C
C                                                                         C
C  KC, KT AND KU ARE THE CLASS, TYPE AND USE CODE FOR NAME PROCESSED.     C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES            C
C        EXTRACTED FROM FORTRAN STATEMENTS                                C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS         C
C*************************************************************************C
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAP/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HFP,HEQ,HAS,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      DIMENSION X(3),Z(3)
C*************************************************************************C
C  EXTRACT ENTRY POINT NAME                                               C
C*************************************************************************C
      Z(1)=TSTAB(2,JTST)
      Z(2)=HBL
      Z(3)=H9L
      I=2
      J=3
      JISS=JISS+1
      JTST=JTST+1
```

```
C*************************************************************************C
      TSTAB(1,JTST)=TSTAB(2,JTST)
      TSTAB(2,JTST)=H8L
      JISS=JISS-1
      GO TO 3
C*************************************************************************C
C  THE NAME OF THE FIRST DATA VARIABLE IS LONGER THAN 4 CHARS.,SHIFT    C
C  THE NAME LEFT TO REPLACE STRING DATA.                                C
C*************************************************************************C
    1 JTST=JTST+1
      TSTAB(2,JTST)=TSTAB(1,JTST)
      TSTAB(1,JTST)=TSTAB(2,JTST-1)
      GO TO 3
    2 JISS=JISS+1
      JTST=JTST+1
C*************************************************************************C
C  PROCESS DATA LIST                                                    C
C*************************************************************************C
    3 IOF=12
      CALL ICLIST(IOF)
      IF(ISS(JISS).EQ.HSL)GO TO 4
      CALL ERROR(30)
      RETURN
C*************************************************************************C
C  SCAN OVER LITERAL LIST                                               C
C*************************************************************************C
    4 JISS=JISS+1
      IF(ISS(JISS).EQ.HV.OR.ISS(JISS).EQ.HI)JTST=JTST+1
      IF(ISS(JISS).NE.HSL)GO TO 4
      IF(JISS.GE.INS)RETURN
      JISS=JISS+1
      IF(ISS(JISS).EQ.HCC)JISS=JISS+1
      GO TO 3
      END
*DECK FDIME
      SUBROUTINE FDIME
      IMPLICIT INTEGER(A-Z)
C*************************************************************************C
C  SUBROUTINE FDIME IS A PARSING ROUTINE WHICH PROCESSES DIMENSION STS.  C
C  FDIME EXTRACTS THE FIRST VARIABLE NAME AND CALL LISTP TO PROCESS      C
C  THE REMAINDER OF THE STATEMENT.                                       C
C  UPON ENTRY, POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS                C
C  ISS,TSTAB, RESP.) POINT TO THE FIRST SYMBOL OF THE STATEMENT, I.E.    C
C  THE STRING  DIMENSIO .                                                C
C  KT,KU - PARAMETERS TO SUBROUTINE LISTP INDICATING THE TYPE AND        C
C  USE CODES FOR THE NAMES IN THE LIST TO BE PROCESSED.                  C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES          C
C        EXTRACTED FROM FORTRAN STATEMENTS                               C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS        C
C*************************************************************************C
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDSL
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/H8L,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      DIMENSION X(3),Z(3)
```

```
      CALL TABS(KC,KT,KU)
      JTST=JTST+1
      JISS=JISS+1
      IF(ISS(JISS).EQ.HSL)GO TO 8
      CALL ERROR(17)
      RETURN
    8 JISS=JISS+1
      IF(ISS(JISS).EQ.HV)GO TO 9
      CALL ERROR(18)
      RETURN
C*****************************************************************C
C  PROCESS VARIABLE LIST                                          C
C*****************************************************************C
    9 X(1)=TSTAB(1,JTST)
      X(2)=TSTAB(2,JTST)
      JTST=JTST+1
      JISS=JISS+1
      KT=5
      KU=11
      CALL LISTP(X,KT,KU)
C*****************************************************************C
C  CHECK FOR END OF STATEMENT                                     C
C*****************************************************************C
   10 IF(JISS.GE.INS)RETURN
      GO TO 5
      END
*DECK FDATA
      SUBROUTINE FDATA
      IMPLICIT INTEGER (A-Z)
C*****************************************************************C
C  FDATA PREPROCESS THE VARIABLE LIST FOR SUBROUTINE IOLIST WHICH C
C  PERFORMS THE ACTUAL PARSING. THE STRING DATA STORED IN THE FIRST C
C  ROW OF TSTAB IS ELIMINATED. THE NAME OF THE FIRST DATA VARIABLE IS C
C  SHIFTED LEFT TO FILL THE EMPTY SPACE. WHEN IOLIST IS CALLED,JTST C
C  AND JISS (POINTERS ASSOCIATED WITH ARRAYS TSTAB AND ISS RESP.) C
C  POINTS TO THE FIRST DATA VARIABLE IN THE LIST.                 C
C                                                                 C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES   C
C        EXTRACTED FROM FORTRAN STATEMENTS                        C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS C
C*****************************************************************C
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOS,NFILE,ARGNO
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HC,HH,HI
C*****************************************************************C
C  GET FIRST ENTRY IN DATA LIST                                   C
C*****************************************************************C
      IF(ISS(2,JTST).EQ.HBL)GO TO 2
      JISS=JISS+1
      IF(ISS(JISS).EQ.HV)GO TO 1
C*****************************************************************C
C  THE NAME OF THE FIRST DATA VARIABLE IS LESS THAN 4 CHARACTERS LONG, C
C  SHIFT THE NAME TO REPLACE DATA.                                C
```

```
      GO TO 10
C*************************************************************************C
C  CHECK FOR VARIABLE NAME PRECEDING COMMON BLOCK NAME                    C
C*************************************************************************C
    4 Z(1)=TSTAB(2,JTST)
      Z(2)=HBL
      Z(3)=HBL
      JTST=JTST+1
      I=3
      J=2
      CALL SHIFTY(X,Z,I,J)
      IF(X(1).EQ.HBL)GO TO 5
C*************************************************************************C
C  VARIABLE NAME FOUND.  STORE BLANK FOR COMMON BLOCK NAME                C
C*************************************************************************C
      KC=7
      KT=5
      KU=0
      NAME(1)=HBL
      NAME(2)=HBL
      CALL TABS(KC,KT,KU)
C*************************************************************************C
C  STORE VARIABLE NAME                                                    C
C*************************************************************************C
      KC=6
      KT=5
      KU=11
      NAME(1)=X(1)
      NAME(2)=HBL
      CALL TABS(KC,KT,KU)
C*************************************************************************C
C  CHECK FOR BLANK COMMON BLOCK NAME                                      C
C*************************************************************************C
    5 JISS=JISS+1
      IF(ISS(JISS).EQ.HV)GO TO 7
      IF(ISS(JISS).EQ.HSL)GO TO 6
      CALL ERROR(16)
      RETURN
C*************************************************************************C
C  NO BLOCK NAME.  STORE BLANK FOR COMMON BLOCK NAME                      C
C*************************************************************************C
    6 KC=7
      KT=5
      KU=0
      NAME(1)=HBL
      NAME(2)=HBL
      CALL TABS(KC,KT,KU)
      GO TO 8
C*************************************************************************C
C  STORE COMMON BLOCK NAME                                                C
C*************************************************************************C
    7 KC=7
      KT=5
      KU=0
      NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
```

```
      CALL TABS(KC,KT,KU)
      RETURN
C*****************************************************************C
C  CHECK FOR COMMON BLOCK NAME                                    C
C*****************************************************************C
    1 JISS=JISS+1
      IF(ISS(JISS).EQ.HSL)GO TO 4
C*****************************************************************C
C  NO BLOCK NAME.  STORE BLANK FOR COMMON BLOCK NAME              C
C*****************************************************************C
      KC=7
      KT=5
      KU=0
      NAME(1)=HBL
      NAME(2)=HBL
      CALL TABS(KC,KT,KU)
      IF(ISS(JISS).NE.HV)GO TO 2
C*****************************************************************C
C  EXTRACT FIRST VARIABLE OF LIST                                 C
C*****************************************************************C
      Z(1)=TSTAB(2,JTST)
      JTST=JTST+1
      Z(2)=TSTAB(1,JTST)
      Z(3)=TSTAB(2,JTST)
      JTST=JTST+1
      JISS = JISS + 1
      I=3
      J=6
      CALL SHIFTY(X,Z,I,J)
C*****************************************************************C
C  PROCESS VARIABLE LIST                                          C
C*****************************************************************C
      KT=5
      KU=11
      CALL LISTP(X,KT,KU)
      GO TO 10
    2 IF(ISS(JISS).EQ.HLP)GO TO 3
      IF(ISS(JISS).EQ.HCO)GO TO 3
      CALL ERROR(15)
      RETURN
C*****************************************************************C
C  EXTRACT FIRST VARIABLE OF LIST                                 C
C*****************************************************************C
    3 Z(1)=TSTAB(2,JTST)
      Z(2)=HBL
      Z(3)=HBL
      JTST=JTST+1
      I=3
      J=2
      CALL SHIFTY(X,Z,I,J)
      KT=5
      KU=11
C*****************************************************************C
C  PROCESS VARIABLE LIST                                          C
C*****************************************************************C
      CALL LISTP(X,KT,KU)
```

```
C*****************************************************************C
      MODE=19
      CALL SEXP(MODE)
      RETURN
C*****************************************************************C
C  NO LEFT PARENTHESIS IN CALL STATEMENT                          C
C*****************************************************************C
    3 CALL ERROR(13)
      RETURN
      END
*DECK FCOMON
      SUBROUTINE FCOMON
      IMPLICIT INTEGER(A-Z)
C*****************************************************************C
C  SUBROUTINE FCOMON IS A PARSING ROUTINE WHICH PROCESSES COMMON STMNTS.
C  COMMON BLOCKS CAN BE BLANK COMMON OR LABELLED COMMON BLOCK.    C
C  UPON ENTRY, THE POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS     C
C  ISS,TSTAB, RESP.) POINT TO THE FIRST SYMBOL OF THE STATEMENT.  C
C  KC,KT,KU - PARAMETERS TO SUBROUTINE TABS INDICATING THE        C
C  CLASS, TYPE, USE CODES OF THE NAME IN QUESTION.                C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES   C
C        EXTRACTED FROM FORTRAN STATEMENTS                        C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS C
C*****************************************************************C
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      DIMENSION X(3),Z(3)
      IF(JISS.NE.INS)GO TO 1
C*****************************************************************C
C  STATEMENT FORM:  COMMON A                                      C
C  STORE BLANK FOR COMMON BLOCK NAME                              C
C*****************************************************************C
      KC=7
      KT=5
      KU=0
      NAME(1)=HBL
      NAME(2)=HBL
      CALL TABS(KC,KT,KU)
C*****************************************************************C
C  EXTRACT AND STORE COMMON VARIABLE                              C
C*****************************************************************C
      Z(1)=TSTAB(2,JTST)
      Z(2)=HBL
      Z(3)=HBL
      I=3
      J=2
      CALL SHIFTY(X,Z,I,J)
      KC=6
      KT=5
      KU=11
      NAME(1)=X(1)
      NAME(2)=X(2)
```

```
C        EXTRACTED FROM FORTRAN STATEMENTS                              C
C  TSTAB - TEMPORARY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS       C
C*************************************************************************C
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,H$P,H=Q,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      DATA EXIT/4HEXIT/
C*************************************************************************C
C  EXTERNAL PROGRAM REFERENCE. MAKE TRANSITION ENTRY                      C
C*************************************************************************C
      LOC = 0
      MOD=5
      CALL TRANS(LOC,MOD)
C*************************************************************************C
C  EXTRACT AND STORE SUBROUTINE NAME                                      C
C*************************************************************************C
      NAME(1)=TSTAB(2,JTST)
C*************************************************************************C
C  TERMINAL STATEMENT. MAKE TRANSITION ENTRY AND RETURN.                  C
C*************************************************************************C
      IF(NAME(1).NE.EXIT.OR.JISS.NE.INS) GO TO 4
      LOC=0
      MOD=3
      CALL TRANS(LOC,MOD)
      RETURN
4     NAME(2)=HBL
      JTST=JTST+1
      JISS=JISS+1
      IF(JISS.LT.INS)GO TO 1
      IF(JISS.EQ.INS)NAME(2)=TSTAB(1,JTST)
C*************************************************************************C
C  STORE NAME IN TABLES                                                   C
C     CLASS = SUBROUTINE NAME                                             C
C     TYPE  = NEUTRAL                                                     C
C     USE   = EXTERNAL PROCEDURE REFERENCE                                C
C*************************************************************************C
      KC=1
      KT=5
      KU=21
      CALL TABS(KC,KT,KU)
      RETURN
    1 IF(ISS(JISS).NE.HV)GO TO 2
      NAME(2)=TSTAB(1,JTST)
      JISS=JISS+1
      JTST=JTST+1
    2 KC=1
      KT=5
      KU=21
      CALL TABS(KC,KT,KU)
      IF(ISS(JISS).NE.HLP)GO TO 3
      JISS=JISS+1
C*************************************************************************C
C  PROCESS PARAMETER LIST                                                 C
```

```
      JTST=JTST+1
      Z(3)=TSTAB(1,JTST)
      GO TO 5
    3 IF(I.NE.3)GO TO 4
      NAME(1)=Z(2)
      NAME(2)=HBL
      IF(ISS(JISS).NE.HV)GO TO 6
      JISS=JISS+1
      JTST=JTST+1
      NAME(2)=TSTAB(1,JTST)
      GO TO 6
    4 I=I-4
      Z(1)=Z(2)
      Z(2)=HBL
      IF(ISS(JISS).NE.HV)GO TO 5
      JISS=JISS+1
      JTST=JTST+1
      Z(2)=TSTAB(1,JTST)
      Z(3)=TSTAB(2,JTST)
    5 J=13-I
      CALL SHIFTY(X,Z,I,J)
C*****************************************************************C
C     STORE INDEX VARIABLE                                        C
C     CLASS = VARIABLE                                            C
C     TYPE  = NEUTRAL                                             C
C     USE   = DO-LOOP INDEX VARIABLE.                             C
C*****************************************************************C
      NAME(1)=X(1)
      NAME(2)=X(2)
    6 KC=6
      KT=5
      KU=5
      CALL TABS(KC,KT,KU)
      JTST=JTST+1
C*****************************************************************C
C  SCAN DO-LOOP RANGE INDICATORS                                  C
C*****************************************************************C
      CALL DORANG
   11 RETURN
      END
*CECK FCALL
      SUBROUTINE FCALL
      IMPLICIT INTEGER(A-Z)
C*****************************************************************C
C  SUBROUTINE FCALL IS A PARSING ROUTINE WHICH PROCESSES CALL STATEMENTS
C  UPON ENTRY, THE POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS     C
C  ISS,TSTAB, RESP.) POINT TO THE FIRST SYMBOL OF THE STATEMENT, I.E.  C
C  THE SYMBOL STRING IN TSTAB IS  CALL NAME , WHERE  NAME  REPRESENTS  C
C  ALL OR PART OF THE CHARACTERS IN THE SUBROUTINE NAME.          C
C  KC, KT, KU ARE PARMETERS TO SUBROUTINE TABS WHICH STORES THE   C
C  SUBROUTINE NAME IN SYSTEM TABLES. KC, KT, AND KU ARE THE CLASS,     C
C  TYPE AND USE CODE FOR THE NAME IN QUESTION.                    C
C  MODE - PARAMETER TO SUBROUTINE SEXP INDICATING THE USE CODE FOR     C
C  NAMES FOUND AT THE HIGHEST LEVEL OF THE PARAMETER LIST         C
C  INS - POINTER TO THE LAST ENTRY IN ISS FOR CURRENT STATEMENT   C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES   C
```

```
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      LOGICAL ALPHA
      DIMENSION X(3),Z(3)
      Z(1)=TSTAB(1,JTST)
      Z(2)=TSTAB(2,JTST)
      Z(3)=HBL
C*****************************************************************C
C  DO N I IS STORED (BY SCAN) IN TSTAB AS ONE CONTINUOUS CHARACTER  C
C  STRING.                                                          C
C  DETERMINE STARTING POSITION OF THE INDEX VARIABLE IN TSTAB STRING C
C*****************************************************************C
      J=1
      DO 1 I=4,8
      CALL SHIFTY(X,Z,I,J)
      IF(ALPHA(X(1)))GO TO 2
    1 CONTINUE
      CALL ERROR(4 )
      GO TO 11
C*****************************************************************C
C     EXTRACT AND STORE LABEL                                      C
C     CLASS = UNDEFINED LABEL                                      C
C     TYPE  = NEUTRAL                                              C
C     USE   = LABEL REFERENCED IN A DO STATEMENT.                  C
C*****************************************************************C
    2 J=I-3
      I=3
      CALL SHIFTY(X,Z,I,J)
      KC=5
      KT=5
      KU=25
      NAME(1)=X(1)
      NAME(2)=X(2)
      CALL TABS(KC,KT,KU)
C*****************************************************************C
C     ENTER ASSIGNED NUMBER OF DO STATEMENT                        C
C     AND ADDRESS OF LABEL IN DO LOOP TABLE                        C
C  NADDR CONTAINS THE ADDRESS IN WHICH THE LABEL  IS STORED IN SYMTAB C
C*****************************************************************C
      INDDO=INDDO+1
      IF(INDDO.LE.LDO) GO TO 202
      PRINT 201
201   FORMAT(* FATAL ERROR:  ATTEMPT TO OVERSTORE DOTAB.*)
      STOP
202   DOTAB(1,INDDO)=NADDR
      DOTAB(1,INDDO)=HISTOR(DOTAB(1,INDDO),STNO)
C*****************************************************************C
C     EXTRACT INDEX VARIABLE                                       C
C*****************************************************************C
      I=J+3
      JISS=JISS+1
      IF(I.NE.4)GO TO 3
      IF(ISS(JISS).NE.HV)GO TO 5
      JISS=JISS+1
```

```
      DO 1 I=1,4
      CALL SHIFTY(X,Z,I,J)
      IF(X(1).EQ.HO)GO TO 3
    1 CONTINUE
      Z(1)=TSTAB(2,JTST)
      I=1
      CALL SHIFTY(X,Z,I,J)
      IF(X(1).EQ.HO)GO TO 2
      CALL ERROR(3 )
      GO TO 4
    2 I=5
    3 J=12-I
C*****************************************************************C
C  EXTRACT AND STORE INDEX VARIABLE NAME                          C
C      CLASS = VARIABLE                                           C
C      TYPE  = NEUTRAL                                            C
C      USE   = VARIABLE REFERENCED IN AN ASSIGN STATEMENT         C
C*****************************************************************C
      Z(1)=TSTAB(1,JTST)
      Z(2)=TSTAB(2,JTST)
      I=I+1
      JISS=JISS+1
      IF(JISS.LT.INS)Z(3)=TSTAB(1,JTST+1)
      CALL SHIFTY(X,Z,I,J)
      KC=6
      KT=5
      KU=22
      NAME(1)=X(1)
      NAME(2)=X(2)
      CALL TABS(KC,KT,KU)
    4 RETURN
      END
*DECK FDO
      SUBROUTINE FDO
      IMPLICIT INTEGER(A-Z)
C*****************************************************************C
C  SUBROUTINE FDO IS A PARSING ROUTINE WHICH PROCESSES DO STATEMENTS.  C
C  I.E.  DO N I=M1, M2, M3                                        C
C           WHERE  N  = DO STATEMENT LABEL                        C
C                  I  = INDEX VARIABLE                            C
C                  M1 = INITIAL VALUE                             C
C                  M2 = FINAL   VALUE                             C
C                  M3 = INCREMENT VALUE (DEFAULT IS 1)            C
C  UPON ENTRY, THE POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS     C
C  ISS,TSTAB, RESP.)  POINT TO THE FIRST SYMBOL OF THE STATEMENT, I.E. C
C  THE TSTAB ENTRY CONTAINS THE STRING  DO LABEL INDEX .          C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES    C
C        EXTRACTED FROM FORTRAN STATEMENTS                        C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS  C
C*****************************************************************C
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INOUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPDP,NFILE,ARGNO
```

```
      MPOP=1
      MPNC=MN
      CALL TABS(KC,KT,KU)
5     JISS=JISS+1
C*****************************************************************************C
C  CHECK FOR EQUALS SIGN                                                      C
C*****************************************************************************C
    2 IF(ISS(JISS).EQ.HEQ)GO TO 3
      CALL ERROR(2 )
      GO TO 4
C*****************************************************************************C
C  CALL SEXP TO PROCESS THE EXPRESSION TO THE RIGHT OF EQUAL SIGN.            C
C*****************************************************************************C
    3 JISS=JISS+1
      IF(MODE.NE.16) MODE=2
      CALL SEXP(MODE)
    4 RETURN
      END
*DECK PASS
      SUBROUTINE PASS
      IMPLICIT INTEGER(A-Z)
C*****************************************************************************C
C  SUBROUTINE PASS IS A PARSING ROUTINE WHICH PROCESSES ASSIGN STATEMENTS.    C
C  I.E. ASSIGN N TO M                                                         C
C          WHERE N IS A TRANSFER LABEL                                        C
C                M IS THE NAME OF A SWITCH VARIABLE.                          C
C  UPON ENTRY, THE POINTERS JISS,JTST (ASSOCIATED WITH ARRAYS                 C
C  ISS,TSTAB, RESP.) POINT TO THE FIRST SYMBOL OF THE STATEMENT.              C
C  SINCE THE ASSIGN STATEMENT CONSISTS ENTIRELY OF ALPHANUMERIC               C
C  CHARACTERS (ASSIGN N TO I), SUBROUTINE SHIFTY IS USED TO                   C
C  EXTRACT THE PERTINENT ENTITIES FROM THE TSTAB STRING.                      C
C  KC,KT,KU - PARAMETERS TO SUBROUTINE TABS INDICATING THE                    C
C  CLASS, TYPE, USE CODES OF THE NAME IN QUESTION.                            C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES               C
C        EXTRACTED FROM FORTRAN STATEMENTS                                    C
C  TSTAB - TEMPORARY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS             C
C*****************************************************************************C
      COMMON/SCRPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HRP,HEQ,HAS,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HFORM,HAT,HG,HH,HI
      DIMENSION X(3),Z(3)
C*****************************************************************************C
C  SKIP THE TSTAB WORD CONTAINING ASSIGN                                      C
C*****************************************************************************C
      JTST=JTST+1
      Z(1)=TSTAB(1,JTST)
      Z(2)=HBL
      Z(3)=HBL
      J=1
C*****************************************************************************C
C  DETERMINE THE STARTING POSITION OF THE INDEX VARIABLE IN TSTAB STRING      C
C  LOOK FOR O IN TC                                                           C
C*****************************************************************************C
```

```
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS      C
C*************************************************************************C
      COMMON/TSPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/PUTPS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HER,HEQ,HAS,HLP,HCC,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HQU,HSL,HPE,HEOM,HAT,HG,HH,HI
      MODE=2
      JISS=JISS+1
C*************************************************************************C
C  CHECK FOR SUBSCRIPTED VARIABLE.                                      C
C*************************************************************************C
      IF(ISS(JISS).EQ.HLP)GO TO 1
C*************************************************************************C
C  STORE SIMPLE VARIABLE NAME                                           C
C     CLASS = VARIABLE                                                  C
C     TYPE  = NEUTRAL                                                   C
C     USE   = OUTPUT VARIABLE.                                          C
C*************************************************************************C
      KC=6
      KT=5
      KU=1
      NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      CALL TABS(KC,KT,KU)
      JTST=JTST+1
      GO TO 2
C*************************************************************************C
C  STORE SUBSCRIPTED VARIABLE NAME                                      C
C  CLASS, TYPE AND USE CODE PASSED TO TABS IS USED TO INDICATE THAT     C
C  THE NAME MAY BE A SUBSCRIPTED VARIABLE OR FUNCTION NAME.             C
C  TABS WILL RESOLVE THIS AMBIGUITITY. UPON RETURN :                    C
C     KC = 4   IF FUNCTION NAME                                         C
C     KC = 3   IF SUBSCRIPTED VARIABLE.                                 C
C*************************************************************************C
    1 KC=4
      KT=5
      KU=1
      NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      CALL TABS(KC,KT,KU)
      JTST=JTST+1
C*************************************************************************C
C  CALL SEXP TO PROCESS SUBSCRIPT EXPRESSION OR PARAMETER LIST.         C
C     MODE = 15  SUBSCRIPTS.                                            C
C     MODE = 16  FUNCTION DUMMY PARAMETERS                              C
C*************************************************************************C
      MODE=15
      IF(KC.NE.3) MODE=16
      JISS=JISS+1
      CALL SEXP(MODE)
      IF(MODE.NE.16) GO TO 5
C*************************************************************************C
C  CALL TABS TO PUT AN END OF PARAMETER LIST CODE IN PARTAB             C
C*************************************************************************C
```

```
      END
*DECK EOFRIT
      SUBROUTINE EOFRIT
C*************************************************************************C
C     THIS ROUTINE CLOSES FILSET RANDOM AND PRESERVE IT FOR USE IN      C
C     A SUBSEQUENT JOB OR JOB STEP                                      C
C                                                                       C
C     THIS SUBROUTINE CALLS MACHINE DEPENDENT ROUTINES                  C
C*************************************************************************C
      CALL CLDISK
      RETURN
      END
*DECK ERROR
      SUBROUTINE ERROR(L)
      IMPLICIT INTEGER(A-Z)
C*************************************************************************C
C  SUBROUTINE ERROR IS CALLED BY ONE OF THE PARSING ROUTINES            C
C  TO STORE AN ERROR MESSAGE INDICATOR FOR THE STATEMENT BEING PARSED.  C
C  PARAMETER L IS AN INDEX TO A TABLE OF ERROR MESSAGES.                C
C  ERROR IS ALSO CALLED WITH L=999 AFTER PROCESSING A PROGRAM MODULE    C
C  TO PRINT THE ACCUMULATED ERROR MESSAGES FOR THAT MODULE.             C
C*************************************************************************C
      DIMENSION NERROR(2,50)
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      DATA NE/0/
      IF(L.EQ.999)GO TO 1
      NE=NE+1
      IF(NE.GT.50) RETURN
      NERROR(1,NE)=L
      NERROR(2,NE)=STNO
      RETURN
    1 IF(NE.EQ.0)RETURN
      PRINT 9000
      PRINT 9001,((NERROR(I,J),I=1,2),J=1,NE)
 9000 FORMAT(/////,10X,*ERROR SUMMARY*,//,12X,*MESG. NO.*,3X,*ST. NO.*)
 9001 FORMAT(/,17X,I3,9X,I3)
      NE=0
      RETURN
      END
*DECK FARITH
      SUBROUTINE FARITH
      IMPLICIT INTEGER(A-Z)
C*************************************************************************C
C  SUBROUTINE FARITH IS A PARSING ROUTINE WHICH PROCESSES ARITHMETIC    C
C  (ASSIGNMENT) STATEMENTS.   UPON ENTRY, THE POINTERS JISS,JTST        C
C  (ASSOCIATED WITH ARRAYS ISS,TSTAB ,RESP.)  POINT TO THE FIRST SYMBOL
C  OF THE STATEMENT,  I.E. THE OUTPUT VARIABLE NAME.                    C
C                                                                       C
C  KC, KT, KU PASSED TO SUBROUTINE TABS ARE THE CLASS, TYPE, AND USE    C
C  CODE OF THE SYMBOL (VARIABLE OR LABEL) PROCESSED. TABS IS THE MAIN   C
C  ROUTINE FOR MAKING ENTRIES IN SYSTEM TABLES.                         C
C  MODE - PARAMETER TO SUBROUTINE SEXP INDICATING THE USE CODE FOR      C
C  NAMES FOUND AT THE HIGHEST LEVEL OF THE EXPRESSION TO BE PROCESSED.  C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES         C
C        EXTRACTED FROM FORTRAN STATEMENTS                              C
```

```
C******************************************************************************C
    1 JISS=JISS+1
      TEMP=ISS(JISS)
      KU=KU+1
      KB1=0
      KB2=0
    2 IF(TEMP.EQ.HI)GO TO 3
      IF(TEMP.EQ.-V)GO TO 4
      IF(TEMP.EQ.HMI)GO TO 6
      IF(TEMP.EQ.HPL)GO TO 7
      CALL ERROR(5)
      RETURN
C******************************************************************************C
C  INDEX IS A CONSTANT. CONVERT CONSTANT FROM H FORMAT TO INTEGER              C
C  FORMAT.                                                                     C
C******************************************************************************C
    3 VALUE=CONST(TSTAB(1,JTST),TSTAB(2,JTST))
      IF(VALUE.GE.2**14)CALL ERROR(24)
      GO TO 5
C******************************************************************************C
C  INDEX IS A VARIABLE. STORE VARIABLE NAME IN TABLES BY TABS                  C
C        VALUE=POINTER TO SYMTAB                                               C
C             =NADDER                                                          C
C******************************************************************************C
    4 KB1=1
      NAME(1)=TSTAB(1,JTST)
      NAME(2)=TSTAB(2,JTST)
      CALL TABS(KC,KT,KU)
      VALUE=NADDR
C******************************************************************************C
C  MAKE ENTRIES IN DO-TABLE                                                    C
C******************************************************************************C
    5 IF(KU.EQ.6)DOTAB(3,INDDO)=BITSET(VALUE,KB1,KB2)
      IF(KU.EQ.7)DOTAB(3,INDDO)=HISTOR(DOTAB(3,INDDO),
     1   BITSET(VALUE,KB1,KB2))
      IF(KU.EQ.8)DOTAB(2,INDDO)=HISTOR(DOTAB(2,INDDO),
     1   BITSET(VALUE,KB1,KB2))
      GO TO 8
C******************************************************************************C
C  MINUS SIGN ENCOUNTERED.  KB2 = 1.                                           C
C******************************************************************************C
    6 KB2=1
    7 JISS=JISS+1
      TEMP=ISS(JISS)
      GO TO 2
C******************************************************************************C
C  CHECK FOR END OF STATEMENT, END OF LIST                                     C
C******************************************************************************C
    8 IF(JISS.GE.INS)RETURN
      JISS=JISS+1
      JTST=JTST+1
      TEMP=ISS(JISS)
      IF(TEMP.EQ.HCO)GO TO 1
      IF(TEMP.EQ.HPP)RETURN
    9 CALL ERROR(25)
      RETURN
```

```
C*********************************************************************C
      INTEGER CHAR,FLD
      ITEMP=0
      ITEMP=FLD(0,6,CHAR)
      IF((ITEMP.LT.338).OR.(ITEMP.GT.448)) GO TO 5
      DIGIT=.TRUE.
      RETURN
5     DIGIT=.FALSE.
      RETURN
      END
*DECK DORANG
      SUBROUTINE DORANG
      IMPLICIT INTEGER (A-Z)
C*********************************************************************C
C  SUBROUTINE DORANG IS A PARSING ROUTINE WHICH PROCESSES THE RANGE    C
C  INDICATORS OF A DO STATEMENT. IT IS CALLED BY SUBROUTINES FDO AND   C
C  IOLIST. THE INITIAL VALUE, TERMINAL VALUE AND THE INCREMENTAL VALUE C
C  OF THE DO LOOP IS STORED IN DOTAB (DO LOOP TABLE) WITH SPECIAL CODESC
C  (KB1, KB2) INDICATING WHETHER EACH ENTRY IS A POINTER TO A VARIABLE C
C  NAME IN SYMTAB OR AN ACTUAL VALUE. ADDITIONALLY, EACH ITEM IS TAGGEDC
C  WITH A CODE INDICATING THE SIGN (+ OR -) GIVEN IN THE SOURCE        C
C  LANGUAGE STATEMENT.                                                 C
C                                                                      C
C        KB1 = 0   VALUE IS AN INTEGER CONSTANT                        C
C            = 1   VALUE IS A POINTER TO SYMTAB.                       C
C        KB2 = 0   IF VALUE IS A POSITIVE CONSTANT                     C
C            = 1   IF VALUE IS A NEGATIVE CONSTANT                     C
C                                                                      C
C  ISS - CONTAINS CHARACTER CODES TO REPRESENT LEXICAL ENTITIES        C
C        EXTRACTED FROM FORTRAN STATEMENTS                             C
C  TSTAB - TEMPERORY SYMBOL TABLE CONTAINING ALPHANUMERIC STRINGS      C
C*********************************************************************C
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,MODTAB(3,1000),LMOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
      COMMON/SCRPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/TABAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/PNTRS/JISS,JTST,NE,NFUN,NSUB
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HPP,HEQ,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HDU,HSL,HPE,HFORM,HAT,HG,HH,HI
      DATA HBL,HMI/1H+,1H-/
      KC=6
      KT=5
      KU=5
C*********************************************************************C
C  ENTER SYMTAB ADDRESS OF INDEX VARIABLE AND INCREMENT DEFAULT VALUE  C
C  (1) IN DOTAB (THE UPPER HALF WORD OF DOTAB (2, INDDO) )             C
C*********************************************************************C
      DOTAB(2,INDDO)=NADDR
      VALUE=1
      DOTAB(2,INDDO)=HISTOR(DOTAB(2,INDDO),VALUE)
      IF(ISS(JISS).NE.HEQ)GO TO 9
C*********************************************************************C
C  EXTRACT AND STORE INDEX LIMITS AND INCREMENT VALUE                  C
```

```
C     X IS A 3 WORD OUTPUT ARRAY WHICH, UPON RETURN, CONTAINS          C
C     THE DESIRED CHARACTER STRING, 4 CHARACTERS PER WORD,             C
C     LEFT JUSTIFIED AND BLANK FILLED                                  C
C*****************************************************************C
      DIMENSION X(3),Y(3),Z(3)
      DATA HBL/1H /
      X(1)=HBL
      X(2)=HBL
      X(3)=HBL
      Z(1)=Y(1)
      Z(2)=Y(2)
      Z(3)=Y(3)
      I=IDUMMY
      J=JDUMMY
      N=0
      IF(I+J.GT.13)GO TO 1
      IF(I.LT.5) GO TO 40
      IF(I.LT.9) GO TO 3
      IF(I.LT.13) GO TO 2
    1 CALL ERROR(26)
      RETURN
    2 Z(1)=Z(3)
      I=I-8
      GO TO 4
    3 Z(1)=Z(2)
      Z(2)=Z(3)
      I=I-4
   40 IF(I.GT.1) GO TO 4
      X(1)=Z(1)
      X(2)=Z(2)
      X(3)=Z(3)
      RETURN
    4 N=N+1
      TEMP=Z(1)
      L=I-1
      K=MIN(5-I,J)
      IPOS=6*L
      NBITS=6*K
      ITEMP=FLD(IPOS,NBITS,TEMP)
      MDUM=2**NBITS-1
      MASK=.NOT.MDUM
      NS=60-NBITS
      NMASK=LSHIFT(MASK,NS)
      NTEMP=LSHIFT(ITEMP,NS)
      X(N)=X(N).AND.NMASK.OR.NTEMP
      J=J-K
      IF(J.LE.0) RETURN
      IF(L.NE.0) GO TO 5
      CALL ERROR(27)
      RETURN
    5 TEMP=Z(2)
      L=MIN(J,4-K)
      IPOS=0
      NBITS=6*L
      ITEMP=FLD(IPOS,NBITS,TEMP)
      MDUM=2**NBITS-1
```

```
      MASK=.NOT.MDUM
      NS=60-NBITS-6*K
      NMASK=LSHIFT(MASK,NS)
      NTEMP=LSHIFT(ITEMP,NS)
      X(N)=X(N).AND.NMASK.OR.NTEMP
      J=J-L
      IF(J.LE.0) RETURN
      IF(N.GT.2) GO TO 6
      Z(1)=Z(2)
      Z(2)=Z(3)
      I=L+1
      GO TO 4
    6 CALL ERROR(28)
      RETURN
      END
*DECK SORT
      SUBROUTINE SORT(ICOL)
      IMPLICIT INTEGER (A-Z)
C*********************************************************************C
C  THIS ROUTINE SORTS THE TRIP TABLE IN INCREASING ORDER ACCORDING TO  C
C  PREDECESSORS (COL 1) OR SUCCESSORS (COL 2).                         C
C  PARAMETER                                                           C
C  ICOL--INPUT PARAMETER INDICATING COLUMN BY WHICH TRIP TABLE IS TO   C
C  BE SORTED.                                                          C
C*********************************************************************C
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LOC,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
      IF (ICOL.EQ.1)NCOL=2
      IF(ICOL.EQ.2)NCOL=1
      NLE=ITR-1
      DO 20 J=1,NLE
      IF(TRIP(J,ICOL).LE.TRIP(J+1,ICOL))GO TO 20
      ICOPY=TRIP(J,ICOL)
      ICOPY2=TRIP(J,NCOL)
      TRIP(J,ICOL)=TRIP(J+1,ICOL)
      TRIP(J,NCOL)=TRIP(J+1,NCOL)
      TRIP(J+1,ICOL)=ICOPY
      TRIP(J+1,NCOL)=ICOPY2
      JM1=J-1
      IF(JM1.EQ.0)GO TO 20
      DO 10 KK=1,JM1
      K=JM1-KK+1
      IF(TRIP(K,ICOL).LE.TRIP(K+1,ICOL))GO TO 20
      ICOPY=TRIP(K,ICOL)
      ICOPY2=TRIP(K,NCOL)
      TRIP(K,ICOL)=TRIP(K+1,ICOL)
      TRIP(K,NCOL)=TRIP(K+1,NCOL)
      TRIP(K+1,ICOL)=ICOPY
   10 TRIP(K+1,NCOL)=ICOPY2
   20 CONTINUE
      RETURN
      END
*DECK STATL
      SUBROUTINE STATL
```

```
      IMPLICIT INTEGER (A-Z)
      COMMON SYMTAB (3,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     C ALTAB (56), LAL, INDAL, DOTAB (3,50), LDO, INDDO, IPRIME,
     C NODTAB (3,1000), LNOD, SUCTAB (1500), ISUC, PRETAB (1500), IPRE,
     C LPS, TRIP (1000,2), LTRIP, ITR, STALTB (2,100), LSTAL, INDST
      COMMON /TAPAR/ NAME (2), STNO, DIM (3), MN, MPOP, NFILE, ARGNO
      COUNT = 0
      DO 10 I = 1, LSYM
   10 IF (SYMTAB (1,I) .NE. 0) COUNT = COUNT + 1
      PRINT 20, COUNT, STNO, ISUC, IPRE, INDUT, INDAL, INDDO, INDST, ITR
   20 FORMAT (//, 1H , *NUMBER OF ACTIVE ENTRIES IN LOCAL TABLES*,/,
     1 1H , *SYMTAB = *, I5, 5X, *NODTAB = *, I5, 5X, *SUCTAB = *, I5,/,
     2 1H , *PRETAB = *, I5, 5X, *USETAB = *, I5, 5X, *ALTAB  = *, I5,/,
     3 1H , *DOTAB  = *, I5, 5X, *STALTB = *, I5, 5X, *TRIP   = *, I5)
      RETURN
      END
*DECK STATG
      SUBROUTINE STATG
      IMPLICIT INTEGER (A-Z)
      COMMON /GLOBAL/ DIREC(3, 100), LDIR, INDIR, CNTAB(4, 300),
     C LCN, INDCN, PARTAB (2000), LPAR, IPAR, JPRIME
      COUNT = 0
      DO 10 I = 1, LDIR
   10 IF (DIREC (1,I) .NE. 0)  COUNT = COUNT + 1
      PRINT 20, COUNT, INDCN, IPAR
   20 FORMAT (1H1, *LENGTH OF GLOBAL TABLES*,/, 1H , *DIREC  = *,I5, 5X,
     1 *CNTAB  = * , I5, 5X, *PARTAB = *, I5)
      RETURN
      END
*DECK STOW
      SUBROUTINE STOW(START,N)
      IMPLICIT INTEGER(A-Z)
C*********************************************************************C
C  SUBROUTINE STOW STORES AN ALPHANUMERIC STRING OF MAX. 8 CHARACTERS IN TSTAB,C
C  4 CHARACTERS PER WORD.  UPON ENTRY, ITS POINTS TO THE ROW OF TSTAB THAT    C
C  THE STRING IS TO BE STORED IN. IF THE BUFFER LBUF IS NOT BLANK, STOW WILL  C
C  FETCH THE FIRST CHARACTERS OF THE STRING FROM THIS BUFFER AND THE REMAINING C
C  CHARACTERS FROM STORE.                                                     C
C  PARAMETERS                                                                 C
C  START-(INPUT)--POINTER TO STARTING LOCATION OF STRING IN STORE TO BE STORED.C
C  N-(INPUT)--NO. OF CHARACTERS TO BE STORED.                                 C
C  COMMON VARIABLES                                                           C
C  SEE SUBROUTINE SCAN.                                                       C
C  LOCAL VARIABLES                                                            C
C  KOUNT--COUNTER FOR NUMBER OF CHARACTERS BEING STORED IN TSTAB.             C
C  INDX--INDEX TO LBUF                                                        C
C  ICOL--INDEX TO COLUMN IN TSTAB                                             C
C  I--STARTING BIT OF CHARACTER AS STORED IN TSTAB                            C
C  J--POINTER TO POSITION OF CHARACTER IN STORE                               C
C*********************************************************************C
      COMMON/INSTUF/STORE(2,80),INP,L,ISAV,NO,LFLAG,NLFLAG,HFLAG
     1,LBUF(10)
      COMMON/SCAPA/TSTAB(2,300),ISS(600),ITS,INS,JFLAG,EFLAG,ENDFL
      COMMON/H/HBL,HA,HB,HC,HD,HE,HF,HJ,HK,HL,HM,HN,HO,HP,HQ,HR,HS,HT,
     1 HU,HV,HW,HX,HY,HZ,HBP,HEQ,HAS,HLP,HCO,H0,H1,H2,H3,H4,H5,H6,H7,
     2 H8,H9,HCU,HSL,HBF,HFORM,HAT,HG,HH,HI
```

```
C*************************************************************************C
C  BLANK OUT CURRENT ROW OF TSTAB.                                        C
C*************************************************************************C
      TSTAB(1,ITS)=HBL
      TSTAB(2,ITS)=HBL
      KOUNT=0
      IF(LBUF(1).NE.HBL)GO TO 16
      IF(NLFLAG.EQ.0)GO TO 40
      IF(L.EQ.1)GO TO 10
      L=1
      GO TO 40
   10 L=2
      GO TO 40
   16 INDX=1
      ICOL=0
20    I=0
      ICOL=ICOL+1
      DO 30 K=1,4
      IPOS=0
      NBITS=6
      ITEMP=FLD(IPOS,NBITS,LBUF(INDX))
      MASK=.NOT.77B
      NS=54-I
      NMASK=LSHIFT(MASK,NS)
      NTEMP=LSHIFT(ITEMP,NS)
      TSTAB(ICOL,ITS)=TSTAB(ICOL,ITS).AND.NMASK.OR.NTEMP
      KOUNT=KOUNT+1
      IF(KOUNT.EQ.N) GO TO 65
25    INDX=INDX+1
      IF(LBUF(INDX).EQ.HBL) GO TO 35
30    I=I+6
      GO TO 20
C*************************************************************************C
C  GET REMAINING CHARACTERS FROM STORE.                                   C
C*************************************************************************C
35    J=7
36    IF(STORE(L,J).NE.HBL) GO TO 38
      J=J+1
      GO TO 36
38    IF(I.EQ.18) GO TO 45
      I=I+6
      GO TO 50
40    J=START
      ICOL=0
45    I=0
      ICOL=ICOL+1
50    DO 60 M=1,4
      IPOS=0
      NBITS=6
      ITEMP=FLD(IPOS,NBITS,STORE(L,J))
      MASK=.NOT.77B
      NS=54-I
      NMASK=LSHIFT(MASK,NS)
      NTEMP=LSHIFT(ITEMP,NS)
      TSTAB(ICOL,ITS)=TSTAB(ICOL,ITS).AND.NMASK.OR.NTEMP
      KOUNT=KOUNT+1
```

```
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,NFILE,ARGNO
      COMMON/TABSUB/ISTP,NSAV,LAN,ISUBAD,IPFLAG,ICTYPE
      COMMON/DOLAB/LABFLG
      IF(MPOP.NE.1) GO TO 2
C*****************************************************************C
C  CALL PARTAB UPDATING ROUTINE TO STORE END OF PARAMETER LIST CODE.  C
C  PARAMETERS UC, IAN, AND MODNO ARE NOT USED IN THIS CALL.           C
C*****************************************************************C
      MCOD=1
      CALL UPPAR(MCOD,MODNC,UC,IAN)
      MPOP=0
      RETURN
2     IF(MPOP.EQ.0) GO TO 3
C*****************************************************************C
C  CALL PARTAB UPDATING ROUTINE TO STORE CONSTANT PARAMETER.          C
C  PARAMETER UC IS NOT USED IN THIS CALL.                             C
C*****************************************************************C
      MODNC=MN
      MCOD=2
      IAN=ARGNO
      ICTYPE=TYPE
      CALL UPPAR(MCOD,MODNC,UC,IAN)
      MPOP=0
      RETURN
    3 NAM1=NAME(1)
      NAM2=NAME(2)
      MODNC=MN
      IF(NSAV.EQ.STNO) GO TO 5
      NSAV=STNO
C*****************************************************************C
C  STORE USETAB POINTER IN NODTAB.                                    C
C*****************************************************************C
      ITEM=NODTAB(1,STNO)
      IUP=INDUT
      NODTAB(1,STNO)=LOSTOR(ITEM,IUP)
5     CALL HASH(IFLAG,NAM1,NAM2)
      IF(IFLAG.NE.0) GO TO 10
C*****************************************************************C
C  STORE NAME IN SYMTAB.                                              C
C*****************************************************************C
      SYMTAB(1,NADDR)=NAME(1)
      SYMTAB(2,NADDR)=NAME(2)
      IF(CLASS.EQ.0.OR.CLASS.EQ.1.OR.CLASS.EQ.5.OR.CLASS.EQ.7.OR.
     1 CLASS.EQ.8.OR.CLASS.EQ.11) GO TO 25
      IF(USE.EQ.14.OR.CLASS.EQ.4.AND.TYPE.NE.5)GOTO 25
C*****************************************************************C
C  CHECK VARIABLE TYPE.                                               C
C*****************************************************************C
      IF(CLASS.NE.10) GO TO 8
      TYPE=0
      GO TO 25
8     CALL TYPCHK(NAM1,ITYP)
      TYPE=ITYP
      GO TO 25
10    IF(USE.EQ.14) GO TO 12
      IT=SYMTAB(3,NADDR)
```

```
      IF(KOUNT.EQ.N) GO TO 65
55    J=J+1
      IF(STORE(L,J).EQ.HBL) GO TO 55
      IF(I.EQ.18) GO TO 45
60    I=I+6
      GO TO 45
C*****************************************************************************C
C  BLANK OUT LBUF.                                                            C
C*****************************************************************************C
65    DO 70 KK=1,10
70    LBUF(KK)=HBL
75    ITS=ITS+1
      IF(NLFLAG.EQ.0) GO TO 85
C*****************************************************************************C
C  SWITCH TO NEXT CARD IF SWITCHED BACK ORIGINALLY.                           C
C*****************************************************************************C
      IF(L.EQ.1) GO TO 80
      L=1
      RETURN
80    L=2
85    RETURN
      END
*DECK TABS
      SUBROUTINE TABS(CLASS,TYPE,USE)
      IMPLICIT INTEGER (A-Z)
C*****************************************************************************C
C  THIS IS THE MAIN ROUTINE FOR MANAGING TABLE STORAGE.  TABS IS CALLED       C
C  ANYTIME A TABLE ENTRY IS TO BE MADE (EXCEPT FOR DOTAB) AND TABS            C
C  DECIDES BY ANALYZING CODES WHICH TABLES ARE TO BE UPDATED.                 C
C  VARIABLES                                                                  C
C  CLASS--CLASS CODE FOR NAME BEING PROCESSED                                 C
C  TYPE--TYPE CODE FOR NAME BEING PROCESSED                                   C
C  USE -- USE CODE FOR NAME BEING PROCESSED                                   C
C  MPOP--SPECIAL COMMUNICATION FLAG BETWEEN PARSER AND TABS.  IN              C
C  NORMAL CASES MPOP=0.  IF MPOP=1, A SPECIAL CALL IS BEING MADE TO           C
C  TABS TO INDICATE A RIGHT PARENTHESIS FOUND IN PARAMETER LIST.  WHEN        C
C  MPOP=2, A SPECIAL CALL IS BEING MADE TO INDICATE A CONSTANT IN A           C
C  PARAMETER LIST.                                                            C
C  NAME(2)--STORES NAME BEING PROCESSED, 4 CHAR. S PER WORD.                  C
C  STNO--STATEMENT NO. BEING PROCESSED.                                       C
C  DIM(3)--CONTAINS DIMENSIONS OF AN ARRAY                                    C
C  MN--MODULE NO. BEING PROCESSED                                             C
C  ARGNO--CONTAINS ARGUMENT NO. OF VARIABLE OR CONSTANT IN A PARAMETER        C
C  LIST.                                                                      C
C  NSAV--CONTAINS STATEMENT NO. IN WHICH LAST NAME PROCESSED BY TABS          C
C  OCCURRED.                                                                  C
C  LAN--CONTAINS SYMTAB ADDRESS OF LAST ARRAY NAME PROCESSED.                 C
C  IPFLAG--FLAG INDICATING PARAMETER LIST IS BEING PROCESSED WHEN VALUE       C
C  IS 1.  OTHERWISE, VALUE IS 0.                                              C
C  ICTYPE--CONTAINS TYPE CODE (AS USED IN SCANNING ROUTINE) OF CONSTANT       C
C  IN A PARAMETER LIST.                                                       C
C*****************************************************************************C
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTR(2,100),LSTAL,INDST
```

```
      TYPE =HIFECH(IT)
      GO TO 13
C*****************************************************************C
C  STORE TYPE CODE IN SYMTAB.                                     C
C*****************************************************************C
12    NTYP=TYPE
      IT=SYMTAB(3,NADDR)
      SYMTAB(3,NADDR)=HISTOR(IT,NTYP)
   13 IF(USE.NE.9) GO TO 14
      CLASS=STNO+10000
      GO TO 35
14    NCLASS=LOFECH(IT)
C*****************************************************************C
C  CHECK FOR ARRAY OR STATEMENT FUNCTION.                         C
C*****************************************************************C
      IF(NCLASS.EQ.2) GO TO 15
      IF(NCLASS.EQ.3) GO TO 141
      IF(CLASS.EQ.4.AND.USE.EQ.1) GO TO 30
      IF(CLASS.EQ.3) GO TO 35
      IF(CLASS.EQ.4) GO TO 35
      GOT O 40
  141 LAN=NADDR
15    CLASS=NCLASS
      GO TO 40
25    IF(USE.NE.9)GOTO 26
      CLASS=10000+STNO
   26 NTYP=TYPE
      IT=SYMTAB(3,NADDR)
      SYMTAB(3,NADDR)=HISTOR(IT,NTYP)
      IF(CLASS.NE.4.OR.USE.NE.1) GO TO 35
C*****************************************************************C
C  STORE CLASS CODE FOR STATEMENT FUNCTION.                       C
C*****************************************************************C
30    CLASS=2
      USE=0
35    NCLAS=CLASS
      IT=SYMTAB(3,NADDR)
      SYMTAB(3,NADDR)=LOSTOR(IT,NCLAS)
   40 IF ( (CLASS.NE.0) .AND. (CLASS.NE.1) .AND. (CLASS.NE.4)
     1 .AND.(CLASS.NE.9) ) GO TO 42
      NF=NFILE
      IF(USE.NE.0) NF=0
C*****************************************************************C
C  STORE MODULE NAME IN DIRECTORY.                                C
C*****************************************************************C
      NMOD = MODNO
      IF (USE .NE. 0) NMOD = 0
      CALL UPDIR (NAM1, NAM2, NF, NMOD)
42    IT=SYMTAB(5,NADDR)
      IALP=LOFECH(IT)
      IF(CLASS.NE.3.OR.IALP.NE.0) GO TO 50
C*****************************************************************C
C  STORE ARRAY DIMENSIONS IN ALTAB.                               C
C*****************************************************************C
      ID1=DIM(1)
      ID2=DIM(2)
```

```
      ID3=DIM(3)
      CALL UPALT(ID1,ID2,ID3)
      ISAP=HIFECH(IT)
      IF(ISAP.EQ.0) GO TO 50
C*****************************************************************C
C  CHANGE DIMENSIONS OF COMMON VARIABLE IN STALTB.                C
C*****************************************************************C
      ICHANG=ISAP
      IF(ID2.EQ.0) ID2=1
      IF(ID3.EQ.0) ID3=1
      IDIM=ID1*ID2*ID3
      CALL UPSTAL(    IDIM,ICHANG)
   50 IF(USE.EQ.15) USE=10000+LAN
      IUSE=USE
      ISTNO=STNO
      CALL UPUSE(    IUSE,ISTNO)
C*****************************************************************C
C  CHECK FOR PARAMETER NAME.                                      C
C*****************************************************************C
      IF(USE.LT.16.OR.USE.GT.19) GO TO 53
      MODNO=MN
      MCOD=0
      IUC=USE
      IAN=ARGNO
      CALL UPPAR(MCOD,MODNO,IUC,IAN)
C*****************************************************************C
C  CHECK FOR FUNCTION OR SUB. NAME.                               C
C*****************************************************************C
   53 IF ( (CLASS.NE.1) .AND. (CLASS.NE.2) .AND. (CLASS.NE.4).AND.
     1 (CLASS.NE.9) ) GO TO 55
      IPFLAG=0
      ISUBAD=NADDR
      RETURN
C*****************************************************************C
C  CHECK FOR COMMON BLOCK NAME.                                   C
C*****************************************************************C
   55 IF(CLASS.NE.7) GO TO 60
      MODNO=MN
      CALL UPCNT(NAM1,NAM2,MODNO)
      RETURN
C*****************************************************************C
C  CHECK FOR COMMON BLOCK ENTRY.                                  C
C*****************************************************************C
60    IF(USE.NE.11) GO TO 90
      IT=SYMTAB(5,NADDR)
      IALP=LOFECH(IT)
      IF(IALP.NE.0) GO TO 65
      IDIM=1
      GO TO 80
65    IT=ALTAB(IALP)
      ND1=LOFECH(IT)
      ND2=HIFECH(IT)
      IF(ND2.EQ.0) ND2=1
      IT=ALTAB(IALP+1)
      ICD=HIFECH(IT)
      IF(ICD.EQ.0) GO TO 70
```

```
      ND3=WMFECH(IT)
      GO TO 75
   70 ND3=1
   75 IDIM=ND1*ND2*ND3
   80 ICHANG=0
      CALL UPSTAL(    IDIM,ICHANG)
      RETURN
   90 IF(USE.NE.10) GO TO 95
C*****************************************************************C
C  CALL TRANS ROUTINE FOR LABEL REFERENCE.                      C
C*****************************************************************C
      LOC=NADDR
      MOD=1
      CALL TRANS (LOC,MOD)
      GO TO 96
   95 IF(USE.NE.25) RETURN
C*****************************************************************C
C  CALL TRANS ROUTINE FOR LABEL REFERENCED IN DO STATEMENT.     C
C*****************************************************************C
      LOC=NADDR
      MOD=2
      LABFLG=0
      IF(IFLAC.NE.0) LABFLG=1
      CALL TRANS (LOC,MOD)
   96 CONTINUE
      RETURN
      END
*DECK TRANS
      SUBROUTINE TRANS(LOC,MOD)
      IMPLICIT INTEGER (A-Z)
C*****************************************************************C
C  THIS ROUTINE BUILDS THE TRANSITION PAIRS TABLE (TRIP). IT IS CALLED C
C  FOR EVERY NON-NORMAL TRANSFER POINT IN THE PROGRAM.          C
C  VARIABLES                                                    C
C  MOD--INPUT CODE INDICATING TYPE OF TRANSFER BEING MADE.      C
C       = 1  EXPLICIT TRANSFER TO A LABEL                       C
C       = 2  LABEL REFERENCED IN DO STATEMENT                   C
C       = 3  TERMINAL STATEMENT (STOP, RETURN)                  C
C       = 4  ONE WAY LOGICAL IF STATEMENT                       C
C       = 5  EXTERNAL SUBROUTINE OR FUNCTION REFERENCE.         C
C       = 6  END STATEMENT                                      C
C  LOC - POINTER TO A LABEL NAME IN SYMTAB IF MODE = 1 OR 2     C
C        NOT USED OTHERWISE                                     C
C  LABFLG - =1  INDICATES LABEL HAS ALREADY APPEARED IN A DO STATEMENT C
C           =0  OTHERWISE                                       C
C  ITR - POINTS TO THE NEXT UNUSED ENTRY IN TRIP.               C
C  CODE - A THREE BIT CODE FIELD USED TO IDENTIFY THE TYPE OF RECORD C
C         MADE                                                  C
C         =0 SECOND WORD CONTAINS POINTER TO A LABEL NAME IN SYMTAB C
C         =1 FIRST WORD CONTAINS POINTER TO A LABEL NAME IN SYMTAB C
C         =2 THIS RECORD IS RESERVED TO STORE THE EXIT TRANSITION OF C
C            A DO LOOP                                          C
C         =3 THE FIRST AND SECOND WORDS OF RECORD ARE ST. NUMBERS. C
C         =4 EXTERNAL FUNCTION OR SUBROUTINE REFERENCE.         C
C                                                               C
C*****************************************************************C
```

```
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
      COMMON/TAPAR/NAME(2),STNO,DIM(3),MN,MPOP,MFILE,ASGNO
      COMMON/DOLAB/LABFLG
      ITR=ITR+1
      IF(ITR.LE.LTRIP) GO TO 90
      PRINT 80
   80 FORMAT(* FATAL ERROR:  ATTEMPT TO OVERSTORE TRIP TABLE.*)
      STOP
   90 TRIP(ITR,1)=STNO
      GO TO (100,120,130,140,150,160,170),MOD
C*************************************************************************C
C     MOD=1                                                               C
C     LABEL REFERENCE                                                     C
C*************************************************************************C
  100 TRIP(ITR,2)=LOC
      GO TO 999
C*************************************************************************C
C     MOD=2                                                               C
C     LABEL REFERENCED IN DO STATEMENT                                    C
C*************************************************************************C
  120 TRIP(ITR,1)=LOC
      TRIP(ITR,2)=OR(HIGHS(1),STNO)
      IF(LABFLG.NE.0) GO TO 999
      ITR=ITR+1
      IF(ITR.LE.LTRIP) GO TO 125
      PRINT 80
      STOP
  125 TRIP(ITR,2)=HIGHS(2)
      GO TO 999
C*************************************************************************C
C     MOD=3                                                               C
C     TERMINAL STATEMENT                                                  C
C*************************************************************************C
  130 TRIP(ITR,2)=OR(HIGHS(3),20000)
      GO TO 999
C*************************************************************************C
C     MOD=4                                                               C
C     LOGICAL IF STATEMENT                                                C
C*************************************************************************C
  140 STDM=STNO+1
      TRIP(ITR,2)=OR(HIGHS(3),STDM)
      STDM=STDM+1
      ITR=ITR+1
      IF(ITR.LE.LTRIP) GO TO 145
      PRINT 80
      STOP
  145 TRIP(ITR,1)=STNO
      TRIP(ITR,2)=OR(HIGHS(3),STDM)
      GO TO 999
C*************************************************************************C
C     MOD=5                                                               C
C     EXTERNAL REFERENCE                                                  C
C*************************************************************************C
```

```
  150 TRIP(ITR,2)=OR(HIGHS(4),LOC)
      ITR=ITR+1
      IF(ITR.LE.LTRIP) GO TO 155
      PRINT 80
      STOP
155   TRIP(ITR,1)=STNO
      STOM=STNO+1
      TRIP(ITR,2)=OR(HIGHS(3),STOM)
      GO TO 999
C*************************************************************************C
C     MOD=6                                                               C
C     END STATEMENT                                                       C
C*************************************************************************C
  160 TRIP(ITR,2)=OR(HIGHS(3),30000)
      GO TO 999
C*************************************************************************C
C     MOD=7                                                               C
C     ENTRY SYATEMENT                                                     C
C*************************************************************************C
  170 TRIP(ITR,2)=OR (HIGHS(3),STNO)
      TRIP(ITR,1)=40000
  999 RETURN
      END
*DECK TYPCHK
      SUBROUTINE TYPCHK(NAM,ITYP)
      IMPLICIT INTEGER (A-Z)
C*************************************************************************C
C  THIS ROUTINE DETERMINES THE TYPE OF A VARIABLE BY CHECKING THE WORD    C
C  OF THE LTRS ARRAY THAT CORRESPONDS TO THE FIRST LETTER OF THE VAR-     C
C  IABLE.  IF THE WORD IS ZERO, THEN THE FORTRAN DEFINED IMPLICIT TYPING  C
C  IS USED.  OTHERWISE, THE TYPE IS DETERMINED ACCORDING TO WHICH BIT     C
C  IS SET IN THE WORD IN LTRS.                                            C
C  VARIABLES                                                              C
C  NAM--(INPUT PARAMETER) FIRST FOUR CHARACTERS OF VARIABLE WHOSE TYPE    C
C  IS TO BE DETERMINED.                                                   C
C  ITYP--(OUTPUT PARAMETER) TYPE OF NAM                                   C
C       = 0  INTEGER                                                      C
C       = 1  FLOATING POINT                                               C
C       = 2  DOUBLE PRECISION                                             C
C       = 3  COMPLEX                                                      C
C       = 4  LOGICAL                                                      C
C  LTRS(26)--EACH WORD CORRESPONDS TO ONE LETTER OF THE ALPHABET AND      C
C  CONTAINS 5 1-BIT FIELDS CORRESPONDING TO THE POSSIBLE TYPES OF         C
C  VARIABLES.  THE APPROPRIATE BIT IS SET TO ONE IF AN IMPLICIT TYPE      C
C  STATEMENT HAS SO INDICATED.                                            C
C     FIRST  BIT SET - INTEGER                                            C
C     SECOND BIT SET - FLOATING POINT                                     C
C     THIRD  BIT SET - DOUBLE PRECISION                                   C
C     FOURTH BIT SET - COMPLEX                                            C
C     FIFTH  BIT SET - LOGICAL                                            C
C*************************************************************************C
      COMMON/TYPES/LTRS(26)
      NCHAR=FLD(0,6,NAM)
      ITEST=LTRS(NCHAR)
      IF(ITEST.NE.0) GO TO 20
C*************************************************************************C
```

```
C  USE FORTRAN--DEFINED IMPLICIT TYPE RULES TO DETERMINE TYPE OF NAM1
C*****************************************************************C
      IF(NCHAR.GE.9.AND.NCHAR.LE.14) GO TO 10
5     ITYP=1
      RETURN
10    ITYP=0
      RETURN
C*****************************************************************C
C  CHECK BIT FIELDS FOR IMPLICIT TYPING.                          C
C*****************************************************************C
20    IF(GETONE(1,ITEST).NE.0) GO TO 10
      IF(GETONE(2,ITEST).NE.0) GO TO 5
      IF(GETONE(3,ITEST).EQ.0) GO TO 30
      ITYP=2
      RETURN
30    IF(GETONE(4,ITEST).EQ.0) GO TO 40
      ITYP=3
      RETURN
40    ITYP=4
      RETURN
      END
*DECK UPALT
      SUBROUTINE UPALT(ID1,ID2,ID3)
      IMPLICIT INTEGER (A-Z)
C*****************************************************************C
C  THIS ROUTINE UPDATES THE ARRAY LENGTH TABLE (ALTAB) EACH TIME AN  C
C  ARRAY IS DIMENSIONED.                                          C
C  VARIABLES                                                      C
C  ID1,ID2,ID3--FIRST, SECOND, AND THIRD DIMENSIONS OF ARRAY, RESP.  C
C    IF NO SECOND OR THIRD DIM. EXISTS, THE PARAMETER=1.          C
C  INDAL - POINTER TO THE NEXT UNUSED ENTRY IN ALTAB (ARRAY LENGTH TAB)C
C*****************************************************************C
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
C*****************************************************************C
C  STORE PTR. TO ALTAB IN SYMTAB.                                 C
C*****************************************************************C
      ITEM=INDAL
      IT=SYMTAB(5,NADDR)
      SYMTAB(5,NADDR)=LOSTOR(IT,ITEM)
C*****************************************************************C
C  STORE FIRST DIMENSION IN THE LOWER HALF WORD OF ALTAB          C
C*****************************************************************C
      IDTEM=ID1
      IT=ALTAB(INDAL)
      ALTAB(INDAL)=LOSTOR(IT,IDTEM)
C*****************************************************************C
C  STORE SECOND DIMENSION IN THE UPPER HALF WORD OF ALTAB         C
C*****************************************************************C
      IF(ID2.EQ.0) GO TO 25
      IDTEM=ID2
      IT=ALTAB(INDAL)
      ALTAB(INDAL)=HISTOR(IT,IDTEM)
C*****************************************************************C
```

```
C  STORE THIRD DIMENSION WITH 1-CODE.                                    C
C***********************************************************************C
      IF(ID3.EQ.0) GO TO 25
      INDAL=INDAL+1
      IF(INDAL.LE.LAL) GO TO 20
      PRINT 15
15    FORMAT(* FATAL ERROR:  ATTEMPT TO OVERSTORE ALTAB.*)
      STOP
20    ALTAB(INDAL)=ID3
      ICD=1
      TT=ALTAB(INDAL)
      ALTAB(INDAL)=HTSTOR(IT,ICD)
25    INDAL=INDAL+1
      IF(INDAL.LE.LAL) RETURN
      PRINT 15
      STOP
      END
*DECK UPCNT
      SUBROUTINE UPCNT(NAM1,NAM2,MODNO)
      IMPLICIT INTEGER (A-Z)
C***********************************************************************C
C  THIS ROUTINE UPDATES THE COMMON NAME TABLE.  IT IS CALLED EACH TIME C
C  A COMMON BLOCK IS FOUND.                                             C
C  VARIABLES                                                            C
C  NAM1,NAM2--COMMON BLOCK NAME, 4 CHAR. S/WD.                          C
C  MODNO--MODULE NO. IN WHICH COMMON BLOCK IS FOUND.                    C
C  INDCN - POINTER TO THE LAST ENTRY IN CNTAB.                          C
C  INDST - POINTER TO THE NEXT UNUSED ENTRY IN STALTB.                  C
C  ISTP--POINTER TO STALTB ENTRY OF LAST VARIABLE IN COMMON BLOCK.      C
C***********************************************************************C
      COMMON/GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAR, IPAR, JPRIME
      COMMON/TABSUB/ISTP,NSAV,LAN,ISUBAD,IFFLAG,ICTYPE
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
C***********************************************************************C
C  INCREMENT POINTER TO CNTAB.                                          C
C***********************************************************************C
      INDCN=INDCN+1
      IF(INDCN.LE.LCN) GO TO 5
      PRINT 2
2     FORMAT(* FATAL ERROR:  ATTEMPT TO OVERSTORE CNTAB.*)
      STOP
C***********************************************************************C
C  STORE NAME OF COMMON BLOCK IN CNTAB.                                 C
C***********************************************************************C
5     CNTAB(1,INDCN)=NAM1
      CNTAB(2,INDCN)=NAM2
C***********************************************************************C
C  STORE MODULE NO. IN CNTAB.                                           C
C***********************************************************************C
      CNTAB(4,INDCN)=MODNO
C***********************************************************************C
C  STORE STALTB TOP POINTER IN CNTAB.                                   C
```

```
C*************************************************************************C
      IT=0
      IPTR=INDST
      CNTAB(3,INDCN)=HISTOR(IT,IPTR)
      ISTP=INDST
      RETURN
      END
*DECK UPDIR
      SUBROUTINE UPDIR(NAM1,NAM2,NF,MODNO)
      IMPLICIT INTEGER (A-Z)
C*************************************************************************C
C  THIS ROUTINE UPDATES THE GLOBAL DIRECTORY EACH TIME A MODULE NAME     C
C  IS ENCOUNTERED (DEFINITION OR CALL).  IF THE NAME IS NOT ALREADY      C
C  IN THE DIRECTORY, IT IS ENTERED THERE (BY HASH CODE).                 C
C  VARIABLES                                                             C
C  NAM1,NAM2--MODULE NAME, 4 CHAR. S/WD.
C  NF--FILE NO., INCREMENTED EACH TIME A MODULE IS DEFINED.              C
C  MODNO--MODULE NO.                                                     C
C  INDIR - CONTAINS THE HASH CODED ADDRESS FOR THE NAME BEING PROCESSED  C
C*************************************************************************C
      COMMON/GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAR, IPAR, JPRIME
C*************************************************************************C
C  CALCULATE HASH CODE.                                                  C
C*************************************************************************C
      KOUNT=0
      NUM1=FLD(0,24,NAM1)
      NUM2=FLD(0,24,NAM2)
      NUM=(NUM1+NUM2)/2
      INDIR=MOD(NUM,JPRIME)+1
C*************************************************************************C
C  CHECK TO SEE IF NAME IS IN DIRECTORY.                                 C
C*************************************************************************C
    2 IF(DIREC(1,INDIR).EQ.0) GO TO 10
      IF(DIREC(1,INDIR).NE.NAM1) GO TO 5
      IF(DIREC(2,INDIR).NE.NAM2) GO TO 5
C*************************************************************************C
C  NAME IS ALREADY IN DIRECTORY.                                         C
C  IF NF ≠ 0, THEN THIS IS A MODULE DECLARATION, STORE MODULE NUMBER     C
C  AND FILE NUMBER IN DIREC, ELSE RETURN                                 C
C*************************************************************************C
      IF(NF.EQ.0) RETURN
C*************************************************************************C
C  STORE MODNO. AND FILE NO. IN DIREC.                                   C
C*************************************************************************C
    4 DIREC(3,INDIR)=MODNO
      ITEM1=NF
      ITEM2=DIREC(3,INDIR)
      DIREC(3,INDIR)=HISTOR(ITEM2,ITEM1)
      RETURN
C*************************************************************************C
C  TRY NEXT ADDRESS IN DIREC.
C*************************************************************************C
    5 INDIR=INDIR+1
      IF(INDIR.GT.LDIR)INDIR=1
      KOUNT=KOUNT+1
```

```
      IF(KOUNT.LE.LDIR) GO TO 2
      PRINT 7
7     FORMAT(* LIMITS OF DIREC EXCEEDED.*)
C*************************************************************************C
C  STORE NAME IN DIRECTORY.                                               C
C*************************************************************************C
   10 DIREC(1,INDIR)=NAM1
      DIREC(2,INDIR)=NAM2
      GO TO 4
      END
*DECK UPPAR
      SUBROUTINE UPPAR(MCOD,MODNO,UC,IAN)
      IMPLICIT INTEGER (A-Z)
C*************************************************************************C
C  THIS ROUTINE UPDATES THE PARAMETER TABLE WHEN EVER A VARIABLE OR       C
C  CONSTANT IN A PARAMETER LIST IS FOUND OR WHEN A RIGHT PARENTHESIS      C
C  IN THE LIST IS PARSED.                                                 C
C  VARIABLES                                                              C
C  MCOD--INDICATES TYPE OF ENTRY TO BE MADE IN PARTAB.  WHEN MCOD=0       C
C  ACTUAL OR FORMAL PARAMETER (NOT A CONSTANT) IS ENTERED.  WHEN MCOD=1   C
C  END-OF-LIST POINTER IS STORED.  WHEN MCOD=2, CONSTANT PARAMETER IS     C
C  STORED.                                                                C
C  MODNO--MODULE NO. IN WHICH PARAMETER LIST IS FOUND.                    C
C  UC--USE CODE OF PARAMETER                                              C
C  IAN--ARGUMENT NO. OF PARAMETER                                         C
C  IPAR - POINTER TO THE LAST ENTRY IN PARTAB                             C
C  ISUBAD - THE SYMTAB ADDRESS OF THE SUBROUTINE OR FUNCTION NAME         C
C           BEING PROCESSED                                               C
C  IPFLAG - = 1  IF A PARAMETER LIST IS BEING PROCESSED                   C
C           = 0  OTHERWISE                                                C
C  ICODE - PARAMETER CODE                                                 C
C          = 0 POINTER TO DUMMY PARAMETER IN SYMTAB                       C
C          = 1 POINTER TO ACTUAL PARAMETER IN SYMTAB                      C
C          = 2 POINTER TO PARTAB FOR THE NEXT CALL OF SAME ROUTINE        C
C          = 3 POINTER TO SUB. OR FUNCTION NAME IN SYMTAB                 C
C          = 4 THIS WORD MARKS THE END OF CURRENT PARAMETER LIST          C
C          = 5 THIS IS A CONSTANT PARAMETER. POINTER FIELD CONTAINS       C
C              CONSTANT TYPE.                                             C
C*************************************************************************C
      COMMON SYMTAB (5,1800), LSYM, MADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,MODTAB(3,1000),LMOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITP,STALTR(2,100),LSTAL,INDST
      COMMON/GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAR, IPAR, JPRIME
      COMMON/TABSUB/ISTP,NSAV,LAN,ISUBAD,IPFLAG,ICTYPE
      IF(MCOD.NE.1) GO TO 5
C*************************************************************************C
C  STORE END-OF-LIST CODE IN PARTAB.                                      C
C  MCOD = 1                                                               C
C*************************************************************************C
      ICO=4
      ITEM=PARTAB(IPAR)
      PARTAB(IPAR)=HTSTOR(ITEM,ICO)
      GO TO 50
    5 IF(IPFLAG.NE.0) GO TO 40
```

```
C*************************************************************************C
C  IS THIS THE FIRST PARAM. LIST FOR THIS SUB. OR FUNCTION                C
C  IPADD POINTS TO PARTAB WHICH IS A LINKED LIST CONTAINING CALL          C
C  PARAMETERS TO THIS FUNCTION OR SUBROUTINE                              C
C*************************************************************************C
      LADDR=ISUBAD
      ITEM1=SYMTAB(5,LADDR)
      IPADD=HIFECH(ITEM1)
      IF(IPADD.EQ.0) GO TO 30
C*************************************************************************C
C  SEARCH PARTAB TO END OF PARAM. LIST.                                   C
C*************************************************************************C
      ISTAK=1
    9 IPADD=IPADD+1
      IF(IPADD.LE.IPAR) GO TO 9
      PRINT 800
800   FORMAT(* ATTEMPT TO SEARCH PAST LIMITS OF PARTAB*)
      STOP
C*************************************************************************C
C  ICODE IS THE PARAMETER CODE                                            C
C*************************************************************************C
    9 ITEM=PARTAB(IPADD)
      ICODE=HTFECH(ITEM)
      IF(ICODE.NE.3) GO TO 10
C*************************************************************************C
C  INCREMENT ISTAK IF POINTER POINTS TO A SUBROUTINE OR FUNCTION NAME     C
C  IN SYMTAB.                                                             C
      ISTAK=ISTAK+1
      GO TO 8
   10 IF(ICODE.NE.2) GO TO 12
      IF(ISTAK.GT.1) GO TO 13
      IPADD=LPFECH(ITEM)
      GO TO 8
   12 IF(ICODE.NE.4) GO TO 8
13    ISTAK=ISTAK-1
      IF(ISTAK.NE.0) GO TO 8
C*************************************************************************C
C  END OF LIST REACHED, SET UP POINTER FIELD TO POINT TO THE NEXT CALL    C
C  OF THE SAME FUNCTION OR SUBROUTINE.(THE ONE BEING PROCESSED)           C
C*************************************************************************C
      ITEM=IPAR
      ICODE=2
      PARTAB(IPADD)=HTSTOR(ITEM,ICODE)
      GO TO 35
C*************************************************************************C
C  STORE PARTAB PTR. IN SYMTAB                                            C
C*************************************************************************C
30    IADD=IPAR
      SYMTAB(5,LADDR)=HTSTOR(ITEM1,IADD)
C*************************************************************************C
C  STORE MODULE NUMBER, ARGUMENT NUMBER AND SYMTAB ADDRESS OF THE         C
C  FUNCTION OR SUBROUTINE NAME IN PARTAB.                                 C
C*************************************************************************C
35    ITEM=LADDR
      ICODE=3
      ITEM=HTSTOR(ITEM,ICODE)
```

```
      MONO=MODNO
      ITEM=HFSTOR(ITEM,MONO)
      IARNUM=IAN
      PARTAB(IPAR)=MFSTOR(ITEM,IARNUM)
      IPAR=IPAR+1
      IF(IPAR.LE.LPAR) GO TO 40
      PRINT 37
37    FORMAT(* FATAL ERROR:  ATTEMPT TO OVERSTORE PARTAB.*)
      STOP
C*************************************************************************C
C  STORE POINTER TO PARAM. NAME OR TYPE IF CONSTANT.                      C
C*************************************************************************C
40    IF(MCOD.EQ.2) GO TO 48
      IF(UC.EQ.16.OR.UC.EQ.17) GO TO 45
C*************************************************************************C
C  ACTUAL PARAMETERS                                                      C
C*************************************************************************C
      ICODE=1
      GO TO 47
C*************************************************************************C
C  DUMMY PARAMETERS                                                       C
C*************************************************************************C
   45 ICODE=0
   47 ITEM=NADDR
      GO TO 49
C*************************************************************************C
C  CONSTANT PARAMETERS                                                    C
C*************************************************************************C
48    ICODE=5
      ITEM=ICTYPE
C*************************************************************************C
C  STORE PARAMETER CODE, MODULE NUMBER, ARGUMENT NUMBER AND POINTER TO    C
C  SYMTAB OR CONSTANT TYPE IN PARTAB.                                     C
C  SET IPFLAG TO 1 TO INDICATE THAT A PARAMETER LIST IS BEING PROCESSED   C
C*************************************************************************C
49    ITEM=HTSTOR(ITEM,ICODE)
      MONO=MODNO
      ITEM=HFSTOR(ITEM,MONO)
      IARNUM=IAN
      PARTAB(IPAR)=MFSTOR(ITEM,IARNUM)
      IPFLAG=1
   50 IPAR=IPAR+1
      IF(IPAR.LE.LPAR) RETURN
      PRINT 37
      STOP
      END
*DECK UPSTAL
      SUBROUTINE UPSTAL(IDIM,ICHANG)
      IMPLICIT INTEGER(A-Z)
C*************************************************************************C
C  THIS ROUTINE UPDATES THE STORAGE ALLOCATION TABLE (STALTR) WHENEVER    C
C  A COMMON VARIABLE IS DECLARED.                                         C
C  VARIABLES                                                              C
C  IDIM--TOTAL DIMENSIONS (STORAGE SPACE) OF COMMON VARIABLE.             C
C  ICHANG--FLAG INDICATING TYPE OF CALL BEING MADE TO UPSTAL.  IN A       C
C  NORMAL CASE (MAKING AN ENTRY FOR A COMMON VAR.), ICHANG=0.  WHEN A     C
```

```
C  CALL IS MADE TO UPSTAL TO CHANGE THE DIMENSIONS OF A PREVIOUSLY
C  STORED COMMON VARIABLE, ICHANG EQUALS THE VALUE OF THE POINTER
C  TO STALTB WHERE THE COMMON VARIABLE IS TO BE CHANGED.
C  INDST - POINTER TO THE NEXT UNUSED ENTRY IN STALTB.
C  INDCN - POINTER TO THE LAST USED ENTRY IN CNTAB.
C  ISBP  - MARKER TO INDICATE THE STARTING LOCATION OF STALTB IN
C          WHICH VARIABLES DEFINED WITHIN THE CURRENTLY PROCESSED
C          COMMON BLOCK IS STORED.
C  NSL - STARTING LOCATION OF THE VARIABLE RELATIVE TO THE FIRST
C        VARIABLE IN THE COMMON BLOCK.
C*************************************************************************C
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUBTAB(1500),ISUB,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
      COMMON/GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAR, IPAR, JPRIME
      COMMON/TABSUB/ISTR,NSAV,LAN,ISUBAD,IRFLAG,ICTYPE
      IF(ICHANG.NE.0) GO TO 40
C*************************************************************************C
C  STORE POINTER TO SYMTAB IN STALTB.                                     C
C*************************************************************************C
      STALTB(1,INDST)=NADDR
C*************************************************************************C
C  STORE POINTER TO CNTAB IN STALTB.                                      C
C*************************************************************************C
      IT=STALTB(1,INDST)
      IPTR=INDCN
      STALTB(1,INDST)=HISTOR(IT,IPTR)
C*************************************************************************C
C  STORE STARTING LOCATION AND NO. OF WORDS IN STALTB.                    C
C*************************************************************************C
      IF(INDST.NE.ISTR)GO TO 10
C*************************************************************************C
C  THIS IS THE FIRST VARIABLE DEFINED IN THE COMMON BLOCK, INITIALIZE     C
C  STARTING LOCATION TO 1.                                                C
      NSL=1
      GO TO 20
C*************************************************************************C
C  EXTRACT THE STARTING LOCATION AND LENGTH OF THE LAST VARIABLE.         C
C  COMPUTE THE NEW STARTING LOCATION.                                     C
C*************************************************************************C
10    IT=STALTB(2,INDST-1)
      LSLOC=HIFECH(IT)
      NW=LOFECH(IT)
      NSL=LSLOC+NW
20    STALTB(2,INDST)=IDIV
      IT=STALTB(2,INDST)
      STALTB(2,INDST)=HISTOR(IT,NSL)
C*************************************************************************C
C  STORE STALTB BOTTOM POINTER IN CNTAB.                                  C
C*************************************************************************C
      IT=CNTAB(3,INDCN)
      ISBP=INDST
      CNTAB(3,INDCN)=LOSTOR(IT,ISBP)
C*************************************************************************C
```

```
C  STORE STALTB POINTER IN SYMTAB.                                        C
C*************************************************************************C
      IT=SYMTAB(5,NADDR)
      SYMTAB(5,NADDR)=HISTOR(IT,ISBP)
C*************************************************************************C
C  INCREMENT INDEX TO STALTB.                                             C
C*************************************************************************C
      INDST=INDST+1
      IF(INDST.LE.LSTAL) RETURN
      PRINT 30
30    FORMAT(* FATAL ERROR:  ATTEMPT TO OVERSTORE STALTB*)
      STOP
C*************************************************************************C
C  CHANGE THE DIMENSION OF A PREVIOUSLY DEFINED COMMON VARABLE.           C
C*************************************************************************C
40    IT=STALTB(2,ICHANG)
      ND=IDIM
      STALTB(2,ICHANG)=LOSTOR(IT,ND)
C*************************************************************************C
C  CHANGE STARTING LOCATION OF OTHER VARIABLES IN COMMON BLOCK.           C
C*************************************************************************C
45    ICHANG=ICHANG+1
      IT=STALTB(2,ICHANG)
      NSL=HIFECH(IT)
      IF(NSL.EQ.1.OR.NSL.EQ.0) RETURN
      NSL=NSL+IDIM-1
      STALTB(2,ICHANG)=HISTOR(IT,NSL)
      GO TO 45
      END
*DECK UPUSE
      SUBROUTINE UPUSE(IUSE,ISTNO)
      IMPLICIT INTEGER (A-Z)
C*************************************************************************C
C  THIS ROUTINE UPDATES THE USAGE TABLE (USETAB) EACH TIME A NAME IS      C
C  FOUND IN THE PROGRAM BEING ANALYZED.  AN ENTRY IS MADE FOR EACH        C
C  APPEARANCE OF A NAME AND IS LINKED TO THE PREVIOUS OCCURRENCE.         C
C  VARIABLES                                                              C
C  IUSE--USE CODE OF CURRENT ENTRY                                        C
C  ISTNO--STATEMENT NO. OF CURRENT ENTRY                                  C
C  INOUT - POINTER TO THE NEXT AVAILABLE SPACE IN USETAB.                 C
C*************************************************************************C
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INOUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
      ITEMP=SYMTAB(4,NADDR)
      IF(HIFECH(ITEMP).NE.0) GO TO 20
C*************************************************************************C
C  THIS IS THE FIRST APPEARANCE OF THE NAME.                              C
C  STORE TOP POINTER FROM SYMTAB TO USETAB.                               C
C*************************************************************************C
      IUT=INOUT
      SYMTAB(4,NADDR)=HISTOR(ITEMP,IUT)
C*************************************************************************C
C  STORE POINTER FROM USETAB TO SYMTAB.                                   C
C*************************************************************************C
```

```
      NADTEM=NADDR
      ITEMP5=USETAB(2,INDUT)
      II=1
      USETAB(2,INDUT)=HCSTOR(HMSTOR(ITEMP5,NADTEM),II)
      GO TO 50
C*****************************************************************C
C  STORE BACK POINTER IN USETAB TO LINK TO ITS PREVIOUS REFERENCE  C
C*****************************************************************C
20    ITEMP2=LOFECH(ITEMP)
      ITEMP3=USETAB(2,INDUT)
      IO=0
      USETAB(2,INDUT)=HCSTOR(HMSTOR(ITEMP3,ITEMP2),IO)
C*****************************************************************C
C  STORE FORWARD POINTER IN USETAB PREVIOUS REFERENCE TO POINT TO THE  C
C  CURRENT REF.                                                 C
C*****************************************************************C
      ITEMP4=USETAB(2,ITEMP2)
      USETAB(2,ITEMP2)=LOSTOR(ITEMP4,INDUT)
C*****************************************************************C
C  UPDATE THE USETAB BOTTOM POINTER IN SYMTAB.                  C
C*****************************************************************C
50    ITEMP=SYMTAB(4,NADDR)
      SYMTAB(4,NADDR)=LOSTOR(ITEMP,INDUT)
C*****************************************************************C
C  STORE STATEMENT NO. AND USE CODE IN USETAB.                  C
C*****************************************************************C
      ISN=ISTNO
      IUC=IUSE
      ITEMP=USETAB(1,INDUT)
      USETAB(1,INDUT)=HISTOR(LOSTOR(ITEMP,IUC),ISN)
      INDUT=INDUT+1
      IF(INDUT.LE.LUSE) RETURN
      PRINT 60
60    FORMAT(* FATAL ERROR:  ATTEMPT TO OVERSTORE USETAB.*)
      STOP
      END
*DECK ZTABS
      SUBROUTINE ZTABS
      IMPLICIT INTEGER(A-Z)
C*****************************************************************C
C     THIS ROUTINE INITIALIZES ALL LOCAL TABLES AND POINTERS TO  C
C     THESE TABLES. THESE POINTERS POINT TO EITHER THE LAST ENTRY MADE C
C     OR THE NEXT UNUSED ENTRY IN THE TABLE.                     C
C*****************************************************************C
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 TRIP(1000,2),LTRIP,ITR,STALTB(2,100),LSTAL,INDST
      COMMON/TYPES/LTRS(26)
      DO 20 J=1,LSYM
      DO 10 I=1,5
   10 SYMTAB(I,J)=0
   20 CONTINUE
      DO 40 J=1,LUSE
      DO 30 I=1,2
   30 USETAB(I,J)=0
```

```
   40 CONTINUE
      DO 50 I=1,LAL
   50 ALTAB(I)=0
      DO 70 J=1,LDO
      DO 60 I=1,3
   60 DOTAB(I,J)=0
   70 CONTINUE
      DO 90  J=1,LTRIP
      DO 80  I=1,2
   80 TRIP(J,I)=0
   90 CONTINUE
      DO 100 I=1,LSTAL
      STALTB(1,I)=0
  100 STALTB(2,I)=0
      DO 110 I=1,LNOD
      NODTAB(1,I)=0
      NODTAB(2,I)=0
  110 NODTAB(3,I)=0
      DO 120 I=1,26
  120 LTRS(I)=0
      DO 130 I = 1, LPS
      SUCTAB(I) = 0
  130 PRETAB(I) = 0
      INDST=2
      NSAV=0
      IPFLAG=0
      ITR=1
      INDDO=1
      INDUT=2
      INDAL=2
      RETURN
      END
*DECK WMFECH
      INTEGER FUNCTION WMFECH(IWORD)
      INTEGER FLD
      WMFECH=FLD(3,57,IWORD)
      RETURN
      END
*DECK HIGHS
      INTEGER FUNCTION HIGHS(INT)
      MASK1=77000000000000000008
      HIGHS=LSHIFT(INT,54).AND.MASK1
      RETURN
      END
*DECK OR
      INTEGER FUNCTION OR(I,J)
      OR=I.OR.J
      RETURN
      END
*DECK MFSTOR
      FUNCTION MFSTOR(ITWORD,ISWORD)
      MASK1=00000077777000000008
      MASK2=.NOT.MASK1
      NS=27
      ITEMP=LSHIFT(ISWORD,NS).AND.MASK1
      ITWORD=ITWORD.AND.MASK2.OR.ITEMP
```

```
      MESTOR=ITWORD
      RETURN
      END
*DECK MESTOR
      INTEGER FUNCTION MESTOR(ITWORD,ISWORD)
      MASK1=07777700000300000008
      MASK2=.NOT.MASK1
      NS=42
      ITEMP=LSHIFT(ISWORD,NS).AND.MASK1
      ITWORD=ITWORD.AND.MASK2.OR.ITEMP
      MESTOR=ITWORD
      RETURN
      END
*DECK LOF
      FUNCTION LOF(IWRD)
      INTEGER FLD
      LOF=FLD(6,54,IWRD)
      RETURN
      END
*DECK HIGHF
      INTEGER FUNCTION HIGHF(WRD)
      INTEGER WRD,FLD
      HIGHF=FLD(0,6,WRD)
      RETURN
      END
*DECK GETONE
      INTEGER FUNCTION GETONE(N,IWORD)
      INTEGER FLD
      I=60-N
      GETONE=FLD(I,1,IWORD)
      RETURN
      END
*DECK SETONE
      INTEGER FUNCTION SETONE(N,IWORD)
      ITEMP=2**(N-1)
      SETONE=IWORD.OR.ITEMP
      RETURN
      END
*DECK MOVBIT
      INTEGER FUNCTION MOVBIT(IWORD)
      MOVBIT=LSHIFT(IWORD,6).AND.77B
      RETURN
      END
*DECK LOSTOR
      INTEGER FUNCTION LOSTOR(ITWORD,ISWORD)
      ITWORD=ITWORD.AND.777777777700000000B
      ISWORD=ISWORD.AND.77777777778
      ITWORD=ITWORD.OR.ISWORD
      LOSTOR=ITWORD
      RETURN
      END
*DECK HMSTOR
      INTEGER FUNCTION HMSTOR(ITWORD,ISWORD)
      MASK1=67777777730000000008
      MASK2=.NOT.MASK1
      ITEMP=LSHIFT(ISWORD,30).AND.MASK1
```

```
      ITWORD=ITWORD.AND.MASK2.OR.ITEMP
      HMSTOR=ITWORD
      RETURN
      END
*DECK HCSTOR
      INTEGER FUNCTION HCSTOR(ITWORD,ISWORD)
      IBIT=ISWORD.AND.1
      IBIT=LSHIFT(IBIT,59)
      ITWORD=ITWORD.OR.IBIT
      HCSTOR=ITWORD
      RETURN
      END
*DECK HCFECH
      INTEGER FUNCTION HCFECH(IWORD)
      INTEGER FLD
      HCFECH=FLD(0,1,IWORD)
      RETURN
      END
*DECK HMFECH
      INTEGER FUNCTION HMFECH(IWORD)
      INTEGER FLD
      HMFECH=FLD(1,29,IWORD)
      RETURN
      END
*DECK LPFECH
      FUNCTION LPFECH(IWORD)
      INTEGER FLD
      LPFECH=FLD(33,27,IWORD)
      RETURN
      END
*DECK MFFECH
      FUNCTION MFFECH(IWORD)
      INTEGER FLD
      MFFECH=FLD(18,15,IWORD)
      RETURN
      END
*DECK HFFECH
      INTEGER FUNCTION HFFECH(IWORD)
      INTEGER FLD
      HFFECH=FLD(3,15,IWORD)
      RETURN
      END
*DECK LOFECH
      INTEGER FUNCTION LOFECH(IWORD)
      INTEGER FLD
      LOFECH=FLD(30,30,IWORD)
      RETURN
      END
*DECK HIFECH
      INTEGER FUNCTION HIFECH(IWORD)
      INTEGER FLD
      HIFECH=FLD(0,30,IWORD)
      RETURN
      END
*DECK FLD
      INTEGER FUNCTION FLD(IPOS,N,IWORD)
```

```
      DIMENSION MASKAR(59)
      DATA MASKAR/1,3,7,17B,37B,77B,177B,377B,777B,1777B,3777B,7777B,
     1 17777B,37777B,77777B,177777B,377777B,777777B,
     1 1777777B,3777777B,7777777B,17777777B,37777777B,77777777B,
     1 177777777B,377777777B,777777777B,
     1 1777777777B,3777777777B,7777777777B,
     1 17777777777B,37777777777B,77777777777B,177777777777B,
     1377777777777B,777777777777B,1777777777777B,3777777777777B,
     17777777777777B,17777777777777B,37777777777777B,77777777777777B,
     1177777777777777B,377777777777777B,777777777777777B,
     1177777777777777B,377777777777777B,777777777777777B,
     1177777777777777B,377777777777777B,777777777777777B,
     1177777777777777B,377777777777777B,777777777777777B,
     1177777777777777B,377777777777777B,777777777777777B,
     1177777777777777777B,377777777777777777B/
      NS=N+IPOS
      IF(N.GE.60.OR.N.LE.0) GO TO 30
      MASK=MASKAR(N)
      FLD=LSHIFT(IWORD,NS).AND.MASK
      RETURN
30    FLD=0
      RETURN
      END
*DECK HISTOR
      INTEGER FUNCTION HISTOR(ITWORD,ISWORD)
      ITWORD=ITWORD.AND.77777777775
      ITEMP=ISWORD.AND.7777777777B
      ITWORD=LSHIFT(ISWORD,30).OR.ITWORD
      HISTOR=ITWORD
      RETURN
      END
*DECK HTFECH
      INTEGER FUNCTION HTFECH(IWORD)
      INTEGER FLD
      HTFECH=FLD(0,3,IWORD)
      RETURN
      END
*DECK HTSTOR
      INTEGER FUNCTION HTSTOR(ITWORD,ISWORD)
      IBITS=ISWORD.AND.7B
      IBITS=LSHIFT(IBITS,57)
      ITWORD=ITWORD.AND.777777777777777777B.OR.IBITS
      HTSTOR=ITWORD
      RETURN
      END
```

DIAGNOSTIC ROUTINES SOURCE LISTING

```
*DECK BACKUP
      SUBROUTINE BACKUP(J,J1,J2)
      IMPLICIT INTEGER (A-Z)
C*****************************************************************************C
C  BACKUP FINDS THE NAME IN SYMTAB OF A PARTICULAR VARIABLE WHOSE USE IS      C
C  DESCRIBED BY THE PARAMETERIZED ENTRY IN USETAB.  EACH ENTRY IN USETAB HAS A C
C  POINTER TO THE NEXT PREVIOUS USE OF THE VARIABLE.  THE FIRST ENTRY OF A    C
C  VARIABLE IN USETAB HAS A POINTER TO THE LOCATION IN SYMTAB OF THE VARIABLE. C
C  BY LOOPING BACK UP THROUGH USETAB IN THIS MANNER IS IT POSSIBLE TO FIND THE C
C  VARIABLE NAME.                                                             C
C                                                                             C
C  PARAMETERS                                                                 C
C  J--INDEX OF THE VARIABLES USAGE IN USETAB                                  C
C  J1,J2--THE VARIABLE NAME AS FOUND BY BACKUP IN SYMTAB                      C
C                                                                             C
C  COMMON VARIABLES ARE DESCRIBED IN BWVT.                                    C
C*****************************************************************************C
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     2   ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME,NODTAB(3,1000)
     3  ,LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,STALTB(2,100),
     4   LSTAL,INDST
      I=J
C*****************************************************************************C
C  FIND INDEX OF PREVIOUS USE IN USETAB                                       C
C*****************************************************************************C
   20 IBACK=HMFECH(USETAB(2,I))
C*****************************************************************************C
C  IS ENTRY A POINTER TO SYMTAB                                               C
C*****************************************************************************C
      IF(HCFECH(USETAB(2,I)).NE.0) GO TO 25
      I=IBACK
      GOTO 20
C*****************************************************************************C
C  SYMTAB POINTER IS FOUND                                                    C
C*****************************************************************************C
25    J1=SYMTAB(1,IBACK)
      J2=SYMTAB(2,IBACK)
      RETURN
      END
*DECK BLKDATA
      BLOCK DATA
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 STALTB(2,100),LSTAL,INDST
      COMMON/GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAR, IPAR, JPRIME
      DATA LDIR/100/, LCN/300/, LPAR/2000/, JPRIME/97/
      COMMON/LTAB2/LUNS
      DATA LUNS/30/
      END
*DECK BWVT
      SUBROUTINE BWVT(IV1,IV2,ND1)
      IMPLICIT INTEGER (A-Z)
C*****************************************************************************C
C  BWVT (BACKWARD VARIABLE TRACE) DETERMINES WHICH VARIABLES MAY HAVE         C
```

```
C  AFFECTED THE VALUE OF THE SUBJECT VARIABLE AT HE SPECIFIED NODE.  THIS IS
C  ACCOMPLISHED BY FOLLOWING EACH ENTRY PATH UNTIL THE VALUE OF THE SUBJECT
C  VARIABLE IS CHANGED--EITHER BY AN ASSIGNMENT STATEMENT OR A DO STATEMENT.
C  EACH VARIABLE WHICH MAY HAVE AFFECTED THIS VALUE IS RECORDED, ALONG WITH
C  THE NODE, IN VARIB.  IF THE SUBJECT VARIABLE IS INVOLVED IN ITS OWN
C  REVALUATION, THE TRACE ALONG THAT PATH CONTINUES.  WHEN ALL PATHS FOR THE
C  NODE AND VARIABLE HAVE BEEN CONCLUDED, PROCESSING RESUMES WITH THE ENTRIES
C  IN VARIB AND CONTINUES UNTIL THEY HAVE ALL BEEN PROCESSED.  THE OUTPUT
C  CONSISTS OF EACH VARIABLE WHICH MAY AFFECT THE SUBJECT VARIABLE AND THE
C  NODE AT WHICH THIS OCCURS.
C
C  PARAMETERS
C  IV1,IV2--THE NAME OF THE SUBJECT VARIABLE, 4 CHARACTERS PER WORD, LEFT-
C        JUSTIFIED AND BLANK-FILLED
C  ND1--THE STARTING NODE
C
C  COMMON VARIABLES REFERENCED BY BWVT
C  SYMTAB(3,LSYM)--A LOCAL TABLE PRODUCED BY FACES.  IT CONTAINS A ENTRY FOR
C               EACH SYMBOL IN THE ROUTINE.  THIS AND ALL OTHER FACES TABLES
C               ARE DESCRIBED IN FULL IN THE EXTERNAL DOCUMENTATION.
C  USETAB(2,LUSE)--TABLE OF USAGES OF EACH SYMBOL IN SYMTAB
C  NODTAB(3,LNOD)--TABLE OF NODES APPEARING IN THE ROUTINE
C  SUCTAB(LPS)--TABLE OF SUCCESSOR NODES FOR EACH NODE IN NODTAB
C  PRETAB(LPS)--TABLE OF PREDECESSOR NODES FOR EACH NODE IN NODTAB
C  NODSTK(1000)--A STACK OF NODES WHOSE PREDECESSORS ARE TO BE CHECKED IN
C            THE BACKWARD TRACE
C  UNSTCK(LUNS)--A TABLE USED TO FLAG THOSE NODES WHICH HAVE ALREADY BEEN
C            PLACED ON NODSTK
C  VARIB(200,3)--A TABLE OF THOSE VARIABLES AND ASSOCIATED NODES WHICH WILL
C            BE INVOLVED IN UPCOMING BACKWARD VARIABLE TRACES
C  OUT(200,3)--A TABLE OF THOSE VARIABLES AND ASSOCIATED NODES WHICH AFFECTED
C           THE VALUE OF THE SUBJECT VARIABLE
C
C  OTHER IMPORTANT VARIABLES
C  LAST--INDEX OF LAST ENTRY IN VARIB
C  LOUT--INDEX OF LAST ENTRY IN OUT
C  LSTK--INDEX OF LAST ENTRY IN NODSTK
C  NODE--THE NODE NUMBER UNDER CURRENT CONSIDERATION
C  IVAR1,IVAR2--THE VARIABLE (STORED IN THE SAME MANNER AS IV1,IV2) UNDER
C           CURRENT CONSIDERATION
C  JCODE--THE USE CODE FOR CURRENT ENTRY IN USETAB
C  JTYPE--THE STATEMENT TYPE OF CURRENT NODE
C  IPNT--POINTER TO FIRST ENTRY IN USETAB OF THE NODE
C  NPNT--INDEX OF USETAB ENTRY BEING CONSIDERED
C  JTEMP1,JTEMP2--VARIABLE REFERRED TO BY CURRENT USETAB ENTRY
C  IPRED--INDEX IN PRETAB OF FIRST PREDECESSOR OF CURRENT NODE
C  LPRED--INDEX IN PRETAB OF LAST  PREDECESSOR OF CURRENT NODE
C  JJ--TEMPORARY STORAGE FOR PREDECESSOR NODE BEING CONSIDERED
C
C  REQUIRED ROUTINES (DESCRIBED ELSEWHERE IN DETAIL)--
C  HIFECH,LOFECH,ENTER,NCHECK,INCLUD,BACKUP,SEARCH,AND SRTOUT
C*****************************************************************************
      COMMON /STACKS/NODSTK(1000),UNSTCK(30),LIST(30),NFLAG
     2               ,VARIB(200,3),OUT(200,3)
      COMMON /LTAB2/LUSE
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INOUT,
```

```
     2   ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME,NODTAB(3,1000)
     3  ,LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,STALTB(2,100),
     4   LSTAL,INDST
C*****************************************************************************C
C  INITIALIZATION                                                             C
C*****************************************************************************C
      LAST=0
      LOUT=0
      LSTK=0
      OUT(1,1)=1H
      OUT(1,2)=1H
      OUT(1,3)=0
      NODE=ND1
      IVAR1=IV1
      IVAR2=IV2
      PRINT 701, IVAR1,IVAR2,NODE
701   FORMAT(15H THE VALUE OF * ,2A4,10H* AT NODE  ,I4,40H CAN BE AFFECT
     2ED BY THE FOLLOWING                    )
C*****************************************************************************C
C  INITIALIZE UNSTCK FOR TRACE OF NEW VARIABLE                                C
C*****************************************************************************C
 8    DO 9 I=1,LUNS
 9    UNSTCK(I)=0
C*****************************************************************************C
C  FLAG NODE IN UNSTCK                                                        C
C*****************************************************************************C
 10   CALL ENTER(NODE,UNSTCK)
 12   JTYPE=HIFECH(NODTAB(1,NODE))
      IF(JTYPE.NE.52)GOTO 50
C*****************************************************************************C
C  STATEMENT IS AN ASSIGNMENT STATEMENT                                       C
C*****************************************************************************C
      IPNT=LOFECH(NODTAB(1,NODE))
      JCODE=LOFECH(USETAB(1,IPNT))
      IF(JCODE.NE.1)IPNT=IPNT+1
C*****************************************************************************C
C  FIND NAME THAT IS ASSIGNED A VALUE                                         C
C*****************************************************************************C
      NPNT=IPNT
      CALL BACKUP(NPNT,JTEMP1,JTEMP2)
      IF(JTEMP1.NE.IVAR1.OR.JTEMP2.NE.IVAR2)GOTO 100
C*****************************************************************************C
C  TARGET SYMBOL IS ASSIGNED A VALUE AT THIS NODE                             C
C*****************************************************************************C
      NPNT=IPNT+1
C*****************************************************************************C
C  DETERMINE WHICH VARIABLES AFFECT THIS VALUE                                C
C*****************************************************************************C
 30   JNODE=HIFECH(USETAB(1,NPNT))
      IF(JNODE.NE.NODE)GOTO 90
      CALL BACKUP(NPNT,JTEMP1,JTEMP2)
      IF(JTEMP1.EQ.IVAR1.AND.JTEMP2.EQ.IVAR2)GOTO 35
      K=SEARCH(JTEMP1,JTEMP2,LOUT,OUT,NODE)
      IF(K.NE.0)GOTO 35
      K=NODE
      CALL INCLUD(VARIB,LAST,JTEMP1,JTEMP2,NODE,K)
```

```
   35 NPNT=NPNT+1
      GOTO 30
   50 IF(JTYPE.NE.53)GOTO 100
C*************************************************************************C
C  STATEMENT IS A DO STATEMENT                                            C
C*************************************************************************C
      IPNT=LOFECH(NODTAB(1,NODE))+1
      JCODE=LOFECH(USETAB(1,IPNT))
      IF(JCODE.NE.5)IPNT=IPNT+1
      NPNT=IPNT
      CALL BACKUP(NPNT,JTEMP1,JTEMP2)
      IF(JTEMP1.NE.IVAR1.OR.JTEMP2.NE.IVAR2)GOTO 100
C*************************************************************************C
C  THE INDEX VARIABLE IS TARGET SYMBOL                                    C
C*************************************************************************C
      NPNT=IPNT+1
C*************************************************************************C
C  DETERMINE WHICH VARIABLES AFFECT THIS INDEX                            C
C*************************************************************************C
   60 JNODE=HIFECH(USETAB(1,NPNT))
      IF(JNODE.NE.NODE)GOTO 110
      CALL BACKUP(NPNT,JTEMP1,JTEMP2)
      K=SEARCH(JTEMP1,JTEMP2,LOUT,OUT,NODE)
      IF(K.NE.0)GOTO 65
      K=NODE
      CALL INCLUD(VARIB,LAST,JTEMP1,JTEMP2,NODE,K)
   65 NPNT=NPNT+1
      GOTO 60
   90 JNODE=LOFECH(USETAB(2,IPNT))
      JNODE=HIFECH(USETAB(1,JNODE))
C*************************************************************************C
C  IF SUBJECT VARIABLE DOES NOT AFFECT ITS OWN VALUE, TRACE ALONG THAT PATH    C
C    HALTS                                                                C
C*************************************************************************C
      IF(JNODE.NE.NODE)GOTO 110
C*************************************************************************C
C  FIND PREDECESSORS FOR THAT NODE                                        C
C*************************************************************************3
  100 IPRED=HIFECH(NODTAB(3,NODE))
      LPRED=LOFECH(NODTAB(3,NODE))+IPRED-1
      DO 105  I=IPRED,LPRED
      JJ=PRETAB(I)
      IF(JJ.GT.LNOD)GOTO 105
      L=NCHECK(JJ,UNSTCK)
      IF(L.EQ.1)GOTO 105
      LSTK=LSTK+1
      NODSTK(LSTK)=JJ
      CALL ENTER(JJ,UNSTCK)
  105 CONTINUE
  110 IF(LSTK.EQ.0)GOTO 200
C*************************************************************************C
C  CONTINUE PROCESSING WITH LAST NODE ON NODSTK                           C
C*************************************************************************C
      NODE=NODSTK(LSTK)
      LSTK=LSTK-1
      GOTO 12
```

```
  200  IF(LAST.EQ.0)GOTO 300
C*****************************************************************************
C  FIND NEXT VARIABLE AND NODE TO BE TRACED                                    *
C*****************************************************************************
      K=SEARCH(IVAR1,IVAR2,LAST,VARIB,0)
      IF(K.NE.0)GOTO 230
      IVAR1=VARIB(1,1)
      IVAR2=VARIB(1,2)
      K=1
  230 NODE=VARIB(K,3)
      LAST=LAST-1
      DO 250 I=K,LAST
      J=I+1
      DO 250 JX=1,3
      VARIB(I,JX)=VARIB(J,JX)
  250 CONTINUE
      CALL INCLUD(OUT,LOUT,IVAR1,IVAR2,NODE,0)
      GOTO 8
C*****************************************************************************
C  END OF RUN - PRINT RESULTS                                                  *
C*****************************************************************************
  300 PRINT 710
710   FORMAT(/,35H    VARIABLE      NODE OF USE        )
      CALL SRTOUT(LOUT)
      PRINT 711,((OUT(I,J),J=1,3),I=1,LOUT)
711   FORMAT(/,(4X,2A4,9X,I5))
  350 CONTINUE
      RETURN
      END
*DECK COMBAL
      SUBROUTINE COMBAL(NMCK)
      IMPLICIT INTEGER (A-Z)
C*****************************************************************************
C  SUBROUTINE COMBAL IS A DIAGNOSTIC ROUTINE WHICH, WHEN USED IN CONJUNCTION  C
C  WITH FACES, WILL SPECIFY THE ACCURACY OF COMMON BLOCK ALIGNMENT FOR THE    C
C  PROCESSES MODULES.  IT WILL DETERMINE IF EACH COMMON BLOCK OF THE SAME NAME C
C  HAS AN IDENTICAL SIZE (INCLUDING NUMBER OF VARIABLE NAMES AND THE          C
C  DIMENSIONS OF EACH VARIABLE), IDENTICAL TYPING FOR EACH ELEMENT, AND (IF   C
C  DESIRED) IDENTICAL NAMES FOR CORRESPONDING ELEMENTS.  FOR THOSE MODULES    C
C  WHICH FAIL ON ANY OF THESE SPECIFICATIONS, DIAGNOSTIC MESSAGES WILL BE     C
C  PRODUCED.  FOR A PARTICULAR COMMON BLOCK ONLY ONE TYPE OF ERROR WILL BE    C
C  INDICATED PER MODULE. THE COMMON BLOCK USED AS THE STANDARD WILL BE THE    C
C  FIRST CONSIDERED WHICH MATCHES THE BLOCK IN ANOTHER MODULE. IF MISALIGNMENT C
C  IS INDICATED BETWEEN EACH OF THE FIRST THREE MODULES FURTHER PROCESSING    C
C  ON THAT COMMON BLOCK IS HALTED.  IF IT IS NOT DESIREABLE TO REQUIRE THAT   C
C  ELEMENTS OF A COMMON BLOCK HAVE IDENTICAL NAMES IN EACH MODULE, THIS       C
C  REQUIREMENT MAY BE DELETED IN THE CALL TO COMBAL THROUGH THE PARAMETER     C
C  NMCK.  ONLY IF NMCK=1 WILL NAMING IRREGULARITIES BE CONSIDERED.            C
C                                                                             C
C  ALIGNMENT TESTING WILL BE PERFORMED ON EACH AND EVERY COMMON BLOCK         C
C  INCLUDED IN THE GLOBAL TABLES PRODUCED BY FACES3 AND EACH OCCURRENCE OF    C
C  A COMMON BLOCK WILL BE PRODUCED IN THE OUTPUT--WITH A DIAGNOSTIC MESSAGE   C
C  IF ONE IS INDICATED.  THESE MESSAGES WILL TAKE ONE OF THE FOLLOWING FORMS- C
C     SIZE - THE BLOCK HAS ONE OF TWO FORMS OF IRREGULARITIES,                C
C            (1) THE WRONG TOTAL OF ALLOTTED SPACE, OR                        C
C            (2) A VARIABLE IN THE BLOCK IS DIMENSIONED IN AN IRREGULAR WAY.  C
```

```
C     TYPE - THE BLOCK HAS AT LEAST ONE VARIABLE WHICH HAS AN IRREGULAR TYPE
C     NAME - THE BLOCK HAS AT LEAST ONE VARIABLE WHICH HAS AN IRREGULAR NAME
C
C  COMMON VARIABLES USED BY COMBAL
C  CNTAB(4,LCNTAB)--COMMON BLOCK NAME TABLE PRODUCED BY FACES3. EACH COMMON
C    BLOCK IN EACH MODULE HAS ONE ENTRY FOUR WORDS LONG. (SEE EXTERNAL
C    DOCUMENTATION)
C  SYMTAB(5,LSYM)--SYMBOL TABLE PRODUCED BY FACES3 FOR THE MODULE UNDER
C    CONSIDERATION. (SEE EXTERNAL DOCUMENTATION)
C  ALTAB(LALT)--ARRAY LENGTH TABLE PRODUCED BY FACES3 FOR THE MODULE UNDER
C    CONSIDERATION. (SEE EXTERNAL DOCUMENTATION)
C  STALTB(2,LSTAL)--STORAGE ALLOCATION TABLE PRODUCED BY FACES3 FOR THE
C    MODULE UNDER CONSIDERATION. (SEE EXTERNAL DOCUMENTATION)
C  CBCODE(LCODE,4,3)--COMMON BLOCK CODE TABLE, CONTAINS THE CODED
C    REPRESENTATION OF UP TO THREE COMMON BLOCKS, EACH VARIABLE OF A BLOCK
C    REPRESENTED BY FOUR WORDS.  (SEE EXTERNAL DOCUMENTATION)
C  ERROR(LERR)--IDENTIFICATION AND DIAGNOSTIC INDICATION TABLE, CONTAINS THE
C    MODULE NUMBER AND ALIGNMENT STATE FOR EACH MODULE CONTAINING THE
C    COMMON BLOCK BEING CONSIDERED.  (SEE EXTERNAL DOCUMENTATION)
C  N1--A CONSTANT EQUAL TO 2**33
C  N2--A CONSTANT EQUAL TO 2**34
C  KFLAG--A FLAG WHOSE VALUE IS 1 IF THERE HAS BEEN FOUND TWO MODULES WITH
C    THE TWO CORRESPONDING COMMON BLOCKS PROPERLY ALIGNED, AND 0 OTHERWISE
C  NPNTR--THE POINTER TO THE CURRENT LAST ENTRY TO ERROR
C  LALT--THE LENGTH OF ARRAY ALTAB
C  LSTAL--THE LENGTH OF THE ARRAY STALTB
C  LCNTAB--THE LENGTH OF ARRAY CNTAB
C  LSYM--THE LENGTH OF ARRAY SYMTAB
C
C  OTHER IMPORTANT VARIABLES
C  IEND--INDEX OF THE LAST ENTRY IN CNTAB
C  JJ--INDEX OF THE FIRST ENTRY IN CNTAB OF COMMON BLOCK BEING PROCESSED
C  JK--INDEX IN CNTAB OF COMMON BLOCK CURRENTLY BEING USED AS STANDARD
C  K--INDEX IN CNTAB OF COMMON BLOCK IN SECOND SEGMENT OF CBCODE
C  I--INDEX IN CNTAB OF COMMON BLOCK IN THIRD SEGMENT OF CBCODE (IF USED)
C  M--INDEX IN CNTAB OF LATEST ENTRY LOCATED OF A PARTICULAR COMMON BLOCK
C  KERR(3)--TABLE CONTAINING -KIND OF ERROR- DIAGNOSTIC MESSAGES
C  KTYPE--TEMPORARY STORAGE FOR THE KIND OF ERROR MESSAGE
C  JCLC--TEMPORARY STORAGE FOR THE MODULE NUMBER
C
C  ROUTINES REQUIRED (FULLER DESCRIPTIONS APPEAR IN THE ROUTINES THEMSELVES)
C  INPUT--LOADS THE GLOBAL TABLE CNTAB FROM EXTERNAL STORAGE
C  TRISET(I,J)--A MASTER ROUTINE WHOSE PURPOSE IS TO SEQUENCE ROUTINES
C    LOAD AND CODER
C  LOAD(J)--LOADS THE LOCAL TABLES FOR THE MODULE INDICATED BY ENTRY J
C    IN CNTAB
C  CODER(I,NOP,NBOT)--PRODUCES IN CODED FORM THE COMMON BLOCK INDICATED IN
C    STALTB, FROM ENTRY NTOP TO ENTRY NBOT.  I INDICATES INTO WHICH OF THE
C    THREE SEGMENTS OF CBCODE THAT THE CODE WILL BE PLACED
C  COMP(NMCK,I,J)--COMPARES WORD BY WORD THE ITH AND JTH SEGMENTS OF CBCODE,
C    AND SPECIFIES DIAGNOSTICS FOR IMPROPER ALIGNMENT.
C  RESTOR--INSERTS THE CODE FROM SEGMENT 3 OF CBCODE INTO SEGMENT 1
C  ROUTINES HIFECH, HISTOR, HIFECH, LOFECH, LOSTOR (USED FOR BIT MANIPULATION)
C    ARE DESCRIBED IN THE EXTERNAL DOCUMENTATION.
C*****************************************************************************C
      COMMON /GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
```

```
      C PARTAB (2000), LPAR, IPAR, JPRIME
       COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INPUT,
      2   ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME,NODTAB(3,1000)
      3  ,LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,STALTB(2,100),
      4   LSTAL,INDST
       COMMON /CODE/CBCODE(50,4,3),ERROR(20),N1,N2,KFLAG,NPNTR
       DIMENSION KERR(3)
       DATA KERR/4HSIZE,4HTYPE,4HNAME/
C*****************************************************************************C
C  CALL TABLNTH TO INITIALIZE TABLE LENGTH                                     C
C  CALL TO ROUTINE INPUT WHICH LOADS THE GLOBAL TABLE CNTAB                    C
C*****************************************************************************C
       CALL TABLNTH
       CALL INPUT
       IEND = CNTAB(1,1)
C*****************************************************************************C
C  LOOP THROUGH CNTAB IEND TIMES PROCESSING EACH OF ITS ELEMENTS               C
C*****************************************************************************C
       DO 500  JJ=2,IEND
       IF(CNTAB(1,JJ).EQ.0)GOTO 500
       NPNTR=0
       KFLAG=0
       PRINT 699,CNTAB(1,JJ),CNTAB(2,JJ)
699    FORMAT(44H1 RESULTS OF ALIGNMENT TEST FOR COMMON BLOCK ,
      2 10H LABELED *,2A4,1H*)
C*****************************************************************************C
C  CALL TO ROUTINE TRISET WHICH LOADS THE REQUIRED LOCAL TABLES AND CODES      C
C    THE COMMON BLOCK INTO SIMPLIFIED FORM                                     C
C*****************************************************************************C
       CALL TRISET(1,JJ)
       JK=JJ
       M=JJ+1
C*****************************************************************************C
C  SEARCH FOR A COMMON BLOCK OF THE SAME NAME                                  C
C*****************************************************************************C
   10  DO 15 K=M,IEND
       IF(CNTAB(1,JK).EQ.CNTAB(1,K).AND.CNTAB(2,JK).EQ.CNTAB(2,K))
      2    GOTO 20
   15  CONTINUE
       CNTAB(1,JK)=0
       GOTO 100
C*****************************************************************************C
C  A COMMON BLOCK OF THE SAME NAME IS FOUND.  IT IS SIMILARLY PROCESSED        C
C    BY THE CALL TO TRISET                                                     C
C*****************************************************************************C
   20  M=K+1
       CALL TRISET(2,K)
C*****************************************************************************C
C  THE CODED FORMS OF THE TWO COMMON BLOCKS ARE COMPARED, DIAGNOSTICS ARE      C
C    INDICATED, AND THE VALUE OF KFLAG IS SET                                  C
C*****************************************************************************C
       CALL COMP(NMCK,1,2)
       CNTAB(1,K)=0
       IF(KFLAG.EQ.0)GOTO 30
       GOTO 10
C*****************************************************************************C
```

```
C  FIRST TWO MODULES ARE NOT ALIGNED.  PROCESSING CONTINUES TO DETERMINE
C     IF EITHER IS ALIGNED WITH A THIRD.
C*****************************************************************************C
   30  DO 35 I=M,IEND
       IF(CNTAB(1,JK).EQ.CNTAB(1,I).AND.CNTAB(2,JK).EQ.CNTAB(2,I))
      2    GOTO 40
   35  CONTINUE
       GOTO 90
   40  M=I+1
C*****************************************************************************C
C  A THIRD COMMON BLOCK IS FOUND, ITS LOCAL TABLES LOADED, AND IT IS          C
C     CODED INTO THE SIMPLIFIED FORM                                          C
C*****************************************************************************C
       CALL TRISET(3,I)
C*****************************************************************************C
C  COMPARE CODE BLOCKS IN SEGMENTS ONE AND THREE OF OBCODE                    C
C*****************************************************************************C
       CALL COMP(NMCK,1,3)
       IF(KFLAG.EQ.0)GOTO 50
       ERROR(NPNTR-2)=CNTAB(4,JJ)
       CNTAB(1,I)=0
       GOTO 10
C*****************************************************************************C
C  COMPARE CODE BLOCKS IN SEGMENTS TWO AND THREE OF OBCODE                    C
C*****************************************************************************C
   50  CALL COMP(NMCK,2,3)
       IF(KFLAG.EQ.0)GOTO 60
       ERROR(NPNTR-1)=CNTAB(4,K)
       ERROR(NPNTR)=CNTAB(4,I)
       JK=J
C*****************************************************************************C
C  TRANSFER CONTENTS OF SEGMENT THREE TO SEGMENT ONE                          C
C*****************************************************************************C
       CALL RESTOR
       CNTAB(4,K)=0
       GOTO 10
C*****************************************************************************C
C  FIRST THREE MODULES DO NOT MATCH.  OTHER MODULES WITH THAT BLOCK ARE       C
C     LOCATED BUT NOT PROCESSED.                                              C
C*****************************************************************************C
   60  CNTAB(1,I)=0
       DO 70 L=M,IEND
       IF(CNTAB(1,JK).NE.CNTAB(1,L).OR.CNTAB(2,JK).NE.CNTAB(2,L))
      2    GOTO 70
       NPNTR=NPNTR+1
       ERROR(NPNTR)=CNTAB(4,L)
       CNTAB(1,L)=0
C*****************************************************************************C
C  ALIGNMENT CHECK IF FOUND TO BE UNFEASABLE AS IT IS NOT POSSIBLE TO         C
C     DETERMINE A STANDARD COMMON BLOCK                                       C
C*****************************************************************************C
   70  CONTINUE
       PRINT 700
  700  FORMAT(44H     FIRST THREE MODULES ARE NOT IN ALIGNMENT
      2  42H, OTHER MODULES ARE LISTED WITHOUT COMMENT  )
       GOTO 101
```

```
   90  PRINT 701
  701  FORMAT(44H    COMMON BLOCK IS IN ONLY TWO MODULES -
     2 ,16HNOT IN ALIGNMENT   )
       GOTO 101
  100  PRINT 702
  702  FORMAT(44H    FIRST MODULE LISTED WITHOUT ERROR CONTAI
     2 29HNS THE STANDARD COMMON BLOCK  )
  101  PRINT 703
  703  FORMAT(32H0   MODULE NUMBER          ERROR   )
C*************************************************************************C
C  PRINT DIAGNOSTIC MESSAGES APPROPRIATE FOR EACH MODULE WITH THE BLOCK    C
C*************************************************************************C
       DO 120 MM=1,NPNTR
       KTYPE=1H
       KIND=HTFECH(ERROR(MM))
       IF(KIND.NE.0)KTYPE=KERP(KIND)
       JOLD=LOFECH(ERROR(MM))
       PRINT 704,JOLD,KTYPE
  704  FORMAT(6X,I6,14X,A6)
  120 CONTINUE
  500 CONTINUE
       RETURN
       END
*DECK CODER
       SUBROUTINE CODER(LL,JTOP,JBTM)
       IMPLICIT INTEGER (A-Z)
C*************************************************************************C
C  CODER DEVELOPES THE VARIABLES OF A COMMON BLOCK INTO CODED REPRESENTATION C
C  FOR DISPLAY IN CBCODE.  THIS CODE CONSISTS OF THE NAME OF THE VARIABLE,  C
C  ITS DIMENSION(S), AND ITS TYPE.                                          C
C                                                                           C
C  PARAMETERS                                                               C
C  LL--THE SEGMENT OF CBCODE IN WHICH THE CODE IS STORED                    C
C  JTOP--THE FIRST ENTRY IN STALTB OF THE COMMON BLOCK                      C
C  JBTM--THE LAST ENTRY IN  STALTB OF THE COMMON BLOCK                      C
C                                                                           C
C  IMPORTANT VARIABLES                                                      C
C  N--THE CURRENT INDEX USED FOR CBCODE                                     C
C  IPNTR--THE INDEX IN SYMTAB OF THE VARIABLE BEING CONSIDERED              C
C  LNGTH--THE LENGTH OF THE STORAGE ALLOTTED TO THIS VARIABLE               C
C  KPNTR--THE INDEX IN ALTAB FOR DATA ON THIS VARIABLE                      C
C  JFLAG--FLAG TO INDICATE IF THE VARIABLE HAS A THIRD DIMENSION            C
C  COMMON VARIABLES ARE DESCRIBED IN COMBAL                                 C
C  THE BIT MANIPULATIVE ROUTINES LOFECH,HTFECH,HIFECH, AND HISTOR ARE       C
C    DESCRIBED IN THE EXTERNAL DOCUMENTATION.                               C
C*************************************************************************C
       COMMON /CODE/CBCODE(50,4,3),ERROR(20),N1,N2,KFLAG,NPNTR
       COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
      2   ALTAB(56),LAL,INDAL,DCTAB(3,50),LDC,INDDC,IPRIME,NODTAB(3,1000)
      3  ,LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,STALTB(2,100),
      4   LSTAL,INDST
       N=1
       N=1
       DO 100  M=JTOP,JBTM
       N=N+1
       IPNTR=LOFECH(STALTB(1,M))
```

```
C*****************************************************************************C
C  COPY THE VARIABLE NAME INTO THE FIRST TWO WORDS OF ITS SECTOR IN CBCODE     C
C*****************************************************************************C
      CBCODE(N,1,LL)=SYMTAB(1,IPNTR)
      CBCODE(N,2,LL)=SYMTAB(2,IPNTR)
      LNGTH=LOFECH(STALTB(2,M))
      CBCODE(N,4,LL)=0
      IF(LNGTH.GT.3)GOTO 20
C*****************************************************************************C
C  IF THE LENGTH IS LESS THAN FOUR, STORE IN THE THIRD WORD OF THE SECTOR     C
C*****************************************************************************C
      CBCODE(N,3,LL)=LNGTH
      GOTO 40
C*****************************************************************************C
C  IF THE LENGTH IS GREATER THAN THREE, COPY THE DIMENSIONS, FROM ALTAB,       C
C    INTO THE THIRD WORD OF THE SECTOR                                        C
C*****************************************************************************C
   20 KPNTR=LOFECH(SYMTAB(5,IPNTR))
      CBCODE(N,3,LL)=ALTAB(KPNTR)
      JFLAG=HIFECH(ALTAB(KPNTR+1))
C*****************************************************************************C
C  STORE THE THIRD DIMENSION INTO THE LOWER HALF OF THE FOURTH WORD OF        C
C    THE SECTOR IN CBCODE                                                     C
C*****************************************************************************C
      IF(JFLAG.NE.0)CBCODE(N,4,LL)=LOFECH(ALTAB(KPNTR+1))
   40 JTEMP=HIFECH(SYMTAB(3,IPNTR))
C*****************************************************************************C
C  STORE THE TYPE OF THE VARIABLE IN THE UPPER HALF OF THE FOURTH WORD OF     C
C    ITS SECTOR IN CBCODE                                                     C
C*****************************************************************************C
      CBCODE(N,4,LL)=HISTOR(CBCODE(N,4,LL),JTEMP)
  100 CONTINUE
C*****************************************************************************C
C  STORE THE NUMBER OF SECTORS OF CBCODE THAT ARE USED INTO ITS FIRST WORD    C
C*****************************************************************************C
      CBCODE(1,1,LL)=N
      CBCODE(N+1,3,LL)=0
      RETURN
      END
*DECK COMP
      SUBROUTINE COMP(NMCK,J1,J2)
C*****************************************************************************C
C  COMP COMPARES THE CODED REPRESENTATION OF TWO MODULES VERSIONS OF A        C
C  COMMON BLOCK AND INSERTS INTO ERROR ANY DIAGNOSTIC INDICATIONS.  IF ONE    C
C  VERSION HAS BEEN CERTIFIED, THROUGH THE COMMON VARIABLE KFLAG, AS THE      C
C  STANDARD, THIS VERSION DOES NOT RECEIVE THE DIAGNOSTIC.                    C
C                                                                             C
C  PARAMETERS                                                                 C
C  NMCK--A VALUE SET TO 1 IF NAMING IRREGULARITIES ARE TO BE CONSIDERED       C
C  J1--THE INDEX OF THE FIRST CODE SEGMENT IN THE COMPARISON                  C
C  J2--THE INDEX OF THE SECOND CODE SEGMENT IN THE COMPARISON                 C
C                                                                             C
C  IMPORTANT VARIABLES                                                        C
C  IEND--INDEX OF THE FINAL ENTRY IN CBCODE SEGMENT J1                        C
C  ITYPE--CONTAINS THE TYPE OF ERROR INDICATION                               C
C  COMMON VARIABLES ARE DESCRIBED IN COMBAL                                   C
```

```
C  THE BIT MANIPULATIVE ROUTINE LOFECH IS DESCRIBED IN THE EXTERNAL          C
C     DOCUMENTATION.                                                        C
C*************************************************************************C
      COMMON /CODE/CBCODE(50,4,3),ERROR(20),N1,N2,KFLAG,NPNTR
      INTEGER CBCODE,ERROR,HTSTOR
C*************************************************************************C
C  LOOP TO COMPARE EACH ENTRY OF THE TWO SEGMENTS IN CBCODE                 C
C*************************************************************************C
      IEND=CBCODE(1,1,J1)
      DO 100 K=2,IEND
C*************************************************************************C
C  DIMENSION AND LENGTH CHECK                                               C
C*************************************************************************C
      IF(CBCODE(K,3,J1).EQ.CBCODE(K,3,J2))GOTO 20
    5 ITYPE= 1
      GOTO 110
C*************************************************************************C
C  TYPE CHECK                                                               C
C*************************************************************************C
   20 IF(CBCODE(K,4,J1).EQ.CBCODE(K,4,J2))GOTO 40
      ITEMP=LOFECH(CBCODE(K,4,J1))
      JTEMP=LOFECH(CBCODE(K,4,J2))
      IF(ITEMP.NE.JTEMP)GOTO 5
      ITYPE= 2
      GOTO 110
C*************************************************************************C
C  NAME CHECK                                                               C
C*************************************************************************C
   40 IF(NMCK.NE.1)GOTO 100
      IF(CBCODE(K,1,J1).EQ.CBCODE(K,1,J2).AND.CBCODE(K,2,J1)
     2 .EQ.CBCODE(K,2,J2))GOTO 100
      ITYPE=3
      GOTO 110
  100 CONTINUE
C*************************************************************************C
C  COMPARE THE NUMBER OF ENTRIES IN THE TWO SEGMENTS                        C
C*************************************************************************C
      IF(CBCODE(1,1,J2).NE.IEND)GOTO 105
C*************************************************************************C
C  COMMON BLOCKS ARE PROPERLY ALIGNED                                       C
C*************************************************************************C
      KFLAG=1
      RETURN
C*************************************************************************C
C  DIAGNOSTIC MESSAGE(S) IS (ARE) INSERTED                                  C
C*************************************************************************C
  105 ITYPE=1
  110 IF(KFLAG.EQ.1.OR.J2.EQ.3)GOTO 115
      ERROR(NPNTR-1)=HTSTOR(ERROR(NPNTR-1),ITYPE)
  115 IF(J1.EQ.1)ERROR(NPNTR)=HTSTOR(ERROR(NPNTR),ITYPE)
      RETURN
      END
*DECK ENTER
      SUBROUTINE ENTER(NCODE,IRAY)
C*************************************************************************
C  THE PURPOSE OF ENTER IS TO STORE A 1 IN A SPECIFIC  BIT LOCATION.  THIS
```

```
C     LOCATION IS INDICATED BY THE TWO PARAMETERS, THE FIRST GIVING THE BIT     C
C     INDEX, THE SECOND THE NAME OF THE ARRAY TO WHICH THE INDEX APPLIES.       C
C     THE INDEX MAY BE NO LARGER THAN 1080, AND REFERS TO THE CONSECUTIVE BIT   C
C     LOCATIONS (36 PER WORD) IN THE ARRAY. THE FIRST BIT IS BIT 0              C
C                                                                               C
C   PARAMETERS                                                                  C
C   NODE--THE INDEX OF THE BIT TO BE ASSIGNED THE VALUE 1                       C
C   IRAY--THE NAME OF THE ARRAY TO BE MANIPULATED                               C
C                                                                               C
C   IMPORTANT VARIABLES                                                         C
C   IWORD--THE INDEX OF THE WORD IN IRAY IN WHICH OCCURS THE BIT TO BE ASSIGNED C
C     THE VALUE 1                                                               C
C   IBIT--THE INDEX OF THE BIT IN IWORD WHICH TO BE ASSIGNED THE VALUE          C
C                                                                               C
C   ROUTINES REQUIRED                                                           C
C   IABS--A FUNCTION WHICH RETURNS THE ABSOLUTE VALUE OF ITS LONE PARAMETER     C
C   FLD--A MACHINE-DEPENDENT FUNCTION DESCRIBED IN EXTERNAL DOCUMENTATION       C
C*******************************************************************************
      DIMENSION IRAY(30)
C*******************************************************************************
C  DERIVE THE CORRECT WORD AND BIT LOCATION IN IRAY                             C
C*******************************************************************************
      IWORD=NODE/36+1
      IBIT=NODE-(IWORD-1)*36
      IF(IBIT.EQ.0)GOTO 20
      IBIT=IBIT-1
      IBIT=IABS(IBIT)
      GOTO 30
   20 IBIT=35
      IWORD=IWORD-1
   30 CONTINUE
C*******************************************************************************
C  ASSIGN ONE TO THE PROPER BIT LOCATION                                        C
C*******************************************************************************
      IDATA=1
      NS=59-IBIT
      NDATA=LSHIFT(IDATA,NS)
      IRAY(IWORD)=IRAY(IWORD).OR.NDATA
      RETURN
      END
*DECK FWVT
      SUBROUTINE FWVT(IV1,IV2,ND1)
      IMPLICIT INTEGER (A-Z)
C*******************************************************************************C
C  FWVT (FORWARD VARIABLE TRACE) DETERMINES WHICH VALUES MAY BE AFFECTED BY     C
C  THE VALUE OF THE SUBJECT VARIABLE ENTERING THE SPECIFIED NODE.  THIS IS      C
C  ACCOMPLISHED BY FOLLOWING ALL PATHS FORWARD FROM AND INCLUSIVE OF THE        C
C  NAMED NODE.  AT EACH NODE AT WHICH THE SUBJECT VARIABLE IS USED AS AN INPUT  C
C  VARIABLE, THE AFFECTED VARIABLE AND NODE ARE RECORDED IN VARIB.  IF THE      C
C  SUBJECT VARIABLE APPEARS AS AN OUTPUT VARIABLE, THE TRACE ALONG THAT PATH    C
C  HALTS.  WHEN ALL PATHS FOR A VARIABLE HAVE BEEN TRAVERSED, PROCESSING        C
C  RESUMES WITH THE LAST ENTRY IN VARIB USED AS THE SUBJECT VARIABLE.  OUTPUT   C
C  CONSISTS OF ALL VARIABLES WHICH MIGHT BE AFFECTED AND THE NODE AT WHICH      C
C  THIS OCCURS.                                                                 C
C                                                                               C
C  PARAMETERS                                                                   C
```

```
C  IV1,IV2--THE NAME OF THE SUBJECT VARIABLE, 4 CHARACTERS PER WORD, LEFT-
C        JUSTIFIED AND BLANK-FILLED
C  NO1--THE STARTING NODE
C
C  COMMON VARIABLES REFERENCED BY FWVT
C  SYMTAB(5,LSYM)--A LOCAL TABLE PRODUCED BY FACES.  IT CONTAINS A ENTRY FOR
C              EACH SYMBOL IN THE ROUTINE.  THIS AND ALL OTHER FACES TABLES
C              ARE DESCRIBED IN FULL IN THE EXTERNAL DOCUMENTATION.
C  USETAB(2,LUSE)--TABLE OF USAGES OF EACH SYMBOL IN SYMTAB
C  NODTAB(3,LNOD)--TABLE OF NODES APPEARING IN THE ROUTINE
C  SUCTAB(LPS)--TABLE OF SUCCESSOR NODES FOR EACH NODE IN NODTAB
C  PRETAB(LPS)--TABLE OF PREDECESSOR NODES FOR EACH NODE IN NODTAB
C  NODSTK(1000)--A STACK OF NODES WHOSE SUCCESSORS ARE TO BE CHECKED IN
C              THE FORWARD TRACE
C  UNSTCK(LUNS)--A TABLE USED TO FLAG THOSE NODES WHICH HAVE ALREADY BEEN
C              PLACED ON NODSTK
C  VARIB(200,3)--A TABLE OF THOSE VARIABLES AND ASSOCIATED NODES WHICH WILL
C              BE INVOLVED IN UPCOMING FORWARD VARIABLE TRACES
C  OUT(200,3)--A TABLE OF THOSE VARIABLES AND ASSOCIATED NODES WHICH WERE
C             AFFECTED BY THE VALUE OF THE SUBJECT VARIABLE
C
C  OTHER IMPORTANT VARIABLES
C  LAST--INDEX OF LAST ENTRY IN VARIB
C  LOUT--INDEX OF LAST ENTRY IN OUT
C  LSTK--INDEX OF LAST ENTRY IN NODSTK
C  NODE--THE NODE NUMBER UNDER CURRENT CONSIDERATION
C  IVAR1,IVAR2--THE VARIABLE (STORED IN THE SAME MANNER AS IV1,IV2) UNDER
C           CURRENT CONSIDERATION
C  JCODE--THE USE CODE FOR CURRENT ENTRY IN USETAB
C  JTYPE--THE STATEMENT TYPE OF CURRENT NODE
C  IPNT--POINTER TI FIRST VARIABLE USE IN USETAB AT CURRENT NODE
C  NEXT--INDEX IN USETAB OF NEXT USE OF CURRENT VARIABLE
C  NPNT--INDEX OF USETAB ENTRY BEING CONSIDERED
C  JTEMP1,JTEMP2--VARIABLE REFERRED TO BY CURRENT USETAB ENTRY
C  JT1,JT2--NAME OF OUTPUT VARIABLE AT CURRENT NODE
C  ISUK--INDEX IN SUCTAB OF FIRST SUCCESSOR NODE OF CURRENT NODE
C  LSUK--INDEX IN SUCTAB OF LAST  SUCCESSOR NODE OF CURRENT NODE
C  JJ--TEMPORARY STORAGE FOR SUCCESSOR NODE BEING CONSIDERED

C  REQUIRED ROUTINES (DESCRIBED ELSEWHERE IN DETAIL)--
C  HIFECH,LOFECH,ENTER,NCHECK,INCLUD,BACKUP,SEARCH,AND SRTOUT
C*****************************************************************
      COMMON /STACKS/NODSTK(1000),UNSTCK(30),LIST(30),NFLAG
     2               ,VARIB(200,3),OUT(200,3)
      COMMON/LTAB2/LUNS
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INOUT,
     2   ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME,NODTAB(3,1000)
     3  ,LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,STALTB(2,100),
     4   LSTAL,INDST
C*****************************************************************
C     INITIALIZATION
C*****************************************************************
      LAST=0
      LOUT=0
      LSTK=0
      OUT(1,1)=1H
```

```
      OUT(1,2)=1H
      OUT(1,3)=0
      DO 5 I=1,LUNS
    5 UNSTCK(I)=0
      NODE=ND1
      IVAR1=IV1
      IVAR2=IV2
  701 FORMAT(11H FROM NODE ,I5,15H THE VARIABLE = ,2A4,37H= CAN AFFECT T
     2HE FOLLOWING VARIABLES  )
      PRINT 701,NODE,IVAR1,IVAR2
C*****************************************************************************C
C  FLAG NODE IN UNSTCK                                                        C
C*****************************************************************************C
   10 CALL ENTER(NODE,UNSTCK)
   12 JTYPE=HIFECH(NODTAB(1,NODE))
      IF(JTYPE.NE.52)GOTO 50
C*****************************************************************************C
C  STATEMENT IS AN ASSIGNMENT STATEMENT                                       C
C*****************************************************************************C
      IPNT=LOFECH(NODTAB(1,NODE))
      JCODE=LOFECH(USETAB(1,IPNT))
      IF(JCODE.NE.1)IPNT=IPNT+1
C*****************************************************************************C
C  FIND NAME THAT IS ASSIGNED A VALUE                                         C
C*****************************************************************************C
      CALL BACKUP(IPNT,JT1,JT2)
      IF(JT1.NE.IVAR1.OR.JT2.NE.IVAR2)GOTO 29
C*****************************************************************************C
C  IT IS THE SUBJECT VARIABLE                                                 C
C*****************************************************************************C
      NEXT=LOFECH(USETAB(2,IPNT))
      JNODE=HIFECH(USETAB(1,NEXT))
C*****************************************************************************C
C  IF IT IS NOT INVOLVED IN ITS OWN REVALUATION,  TRACE ALONG THIS PATH ENDS  C
C*****************************************************************************C
      IF(JNODE.NE.NODE)GOTO 110
      GOTO 100
   29 NPNT=IPNT+1
C*****************************************************************************C
C  DETERMINE IF SUBJECT VARIABLE IS USED IN THE ASSIGNMENT                    C
C*****************************************************************************C
   30 JNODE=HIFECH(USETAB(1,NPNT))
      IF(JNODE.NE.NODE)GOTO 100
      CALL BACKUP(NPNT,JTEMP1,JTEMP2)
      IF(JTEMP1.EQ.IVAR1.AND.JTEMP2.EQ.IVAR2)GOTO 40
      NPNT=NPNT+1
      GOTO 30
C*****************************************************************************C
C  DETERMINE IF INPUT VARIABLE APPEARS IN ARRAY OUT                           C
C*****************************************************************************C
   40  K=SEARCH(JT1,JT2,LOUT,OUT,NODE)
      IF(K.NE.0)GOTO 100
      K=NODE
      CALL INCLUD (VARIB,LAST,JT1,JT2,NODE,K)
      GOTO 100
   50 IF(JTYPE.NE.53)GOTO 100
```

```
C*************************************************************************C
C  STATEMENT IS A DO STATEMENT                                            C
C*************************************************************************C
      IPNT=LOFECH(NODTAB(1,NODE))+1
      JCODE=LOFECH(USETAB(1,IPNT))
      IF(JCODE.NE.5)IPNT=IPNT+1
C*************************************************************************C
C  FIND NAME OF INDEX VARIABLE                                            C
C*************************************************************************C
      CALL BACKUP(IPNT,JT1,JT2)
      IF(JT1.EQ.IVAR1.AND.JT2.EQ.IVAR2)GOTO 110
      NPNT=IPNT+1
      GO TO 61
C*************************************************************************C
C  DETERMINE IF SUBJECT VARIABLE AFFECTS THE VALUE OF INDEX VARIABLE      C
C*************************************************************************C
   60 CALL BACKUP(NPNT,JTEMP1,JTEMP2)
      IF(JTEMP1.EQ.IVAR1.AND.JTEMP2.EQ.IVAR2)GOTO 40
      NPNT=NPNT+1
   61 JNODE = HIFECH (USETAB (1,NPNT) )
      IF(JNODE.NE.NODE)GOTO 100
      GOTO 60
C*************************************************************************C
C  LOCATE THE SUCCESSOR NODES TO THE CURRENT NODE                         C
C*************************************************************************C
  100 ISUK=HIFECH(NODTAB(2,NODE))
      LSUK=LOFECH(NODTAB(2,NODE))+ISUK-1
      DO 105  I=ISUK,LSUK
      JJ=SUCTAB(I)
      IF(JJ.GT.LNOD)GOTO 105
      L=NCHECK(JJ,UNSTCK)
      IF(L.EQ.1)GOTO 105
      LSTK=LSTK+1
      NODSTK(LSTK)=JJ
      CALL ENTER(JJ,UNSTCK)
  105 CONTINUE
  110 IF(LSTK.EQ.0)GOTO 200
C*************************************************************************C
C  CONTINUE PROCESSING WITH LAST NODE ON NODSTK                           C
C*************************************************************************C
      NODE=NODSTK(LSTK)
      LSTK=LSTK-1
      GOTO 12
  200 IF(LAST.EQ.0)GOTO 300
C*************************************************************************C
C  FIND NEXT VARIABLE AND NODE TO BE TRACED                               C
C*************************************************************************C
      K=SEARCH(IVAR1,IVAR2,LAST,VARIB,0)
      IF(K.NE.0)GOTO 230
      IVAR1=VARIB(1,1)
      IVAR2=VARIB(1,2)
C*************************************************************************C
C  INITIALIZE UNSTCK FOR TRACE OF NEW VARIABLE                            C
C*************************************************************************C
      DO 210 I=1,LUNS
  210 UNSTCK(I)=0
```

```
      K=1
  230 NODE=VARIB(K,3)
      LAST=LAST-1
      DO 250 I=K,LAST
      J=I+1
      DO 250 JX=1,3
      VARIB(I,JX)=VARIB(J,JX)
  250 CONTINUE
      CALL INCLUD(OUT,LOUT,IVAR1,IVAR2,NODE,0)
      CALL ENTER(NODE,UNSTCK)
      GOTO 100
C*****************************************************************************C
C  END OF RUN - PRINT RESULTS                                                   C
C*****************************************************************************C
  300 PRINT 710
      CALL SRTOUT(LOUT)
710   FORMAT(/,35H    VARIABLE      NODE OF USE         )
      PRINT 711,((OUT(I,J),J=1,3),I=1,LOUT)
711   FORMAT(/,(4X,2A4,9X,I5))
  350 CONTINUE
      RETURN
      END
*DECK HASHMN
      SUBROUTINE HASHMN(NAM1,NAM2,MNADD)
      IMPLICIT INTEGER (A-Z)
C*****************************************************************************C
C  THIS SUBROUTINE CALCULATES THE HASH CODE FOR THE MODULE NAMED BY     C
C  PARAMETERS NAM1 AND NAM2 AND RETURNS THE ADDRESS OF THE NAME IN      C
C  SYMTAB THROUGH THE PARAMETER MNADD.                                  C
C  VARIABLE DESCRIPTIONS
C  NAM1--FIRST FOUR CHARACTERS OF MODULE NAME                           C
C  NAM2--LAST THREE CHARACTERS OF MODULE NAME                           C
C  MNADD--ADDRESS OF MODULE NAME AS STORED IN SYMTAB
C  SYMTAB(5,2000)--SYMBOL TABLE                                         C
C*****************************************************************************C
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(5E),LAL,INDAL,DDTAB(3,50),LDD,INDDD,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 STALTB(2,100),LSTAL,INDST
      ICHARS=FLD(0,24,NAM1)
      NCHARS=FLD(0,24,NAM2)
      NUM=(NCHARS+ICHARS)/2
      MNADD=MOD(NUM,IPRIME)+1
2     IF(SYMTAB(1,MNADD).EQ.0) GO TO 10
      IF(SYMTAB(1,MNADD).NE.NAM1) GO TO 5
      IF(SYMTAB(2,MNADD).NE.NAM2) GO TO 5
      RETURN
5     MNADD=MNADD+1
      IF(MNADD.GT.LSYM)MNADD=1
      GO TO 2
10    PRINT 15,NAM1,NAM2
15    FORMAT(* MODULE NAME *,2A4,* NOT FOUND IN SYMBOL TABLE*)
      RETURN
      END
*DECK INCLUD
      SUBROUTINE INCLUD(IRAY,IEND,J1,J2,NODE,K)
```

```
      IMPLICIT INTEGER (A-Z)
C*****************************************************************************C
C  ROUTINE CALLED TO ENTER A VARIABLE AND ITS NODE INTO EITHER OUT OR VARIB.   C
C  ENTRY IS MADE WHENEVER THE ENTRY DOES NOT ALREADY APPEAR IN THE PROPER      3
C  ARRAY.  THE ENTRY IS MADE AT THE END OF THE ARRAY, EXCEPT IN SOME CASES FOR C
C  OUT, WHICH RESULT IN INSERTING THE VARIABLE AND NODE AFTER OTHER ENTRIES    C
C  OF THE VARIABLE.  IT IS FIRST DETERMINED BY CALL TO SEARCH IF THE ENTRY     C
C  IS ALREADY IN THE ARRAY AND IF SO AT WHICH LOCATION OF THE ARRAY.           C
C                                                                             C
C  PARAMETERS                                                                 C
C  IRAY--THE NAME OF THE ARRAY TO BE CONSIDERED                               C
C  IEND--THE FINAL ENTRY IN IRAY                                              C
C  J1,J2--THE VARIABLE NAME TO BE INCLUDED IN IRAY, 4 CHARACTERS PER WORD,    C
C       LEFT-JUSTIFIED AND BLANK-FILLED                                       C
C  NODE--THE NODE OF THE VARIABLES USE                                        C
C  K--FOR ENTRY TO OUT K=0, FOR ENTRY TO VARIB K=NODE                         C
C*****************************************************************************C
      DIMENSION IRAY(200,3)
      L=SEARCH(J1,J2,IEND,IRAY,K)
      IF(L.EQ.0)GOTO 40
      IF(K.NE.0)RETURN
      JDIF=IEND-L
C*****************************************************************************C
C  PATH FOR MAKING ENTRY TO OUT                                               C
C*****************************************************************************C
      DO 20 I=1,JDIF
      J=IEND-I+1
      DO 20 JX=1,3
   20 IRAY(J+1,JX)=IRAY(J,JX)
      L1=L+1
      IRAY(L1,1)=J1
      IRAY(L1,2)=J2
      IRAY(L1,3)=NODE
      IEND=IEND+1
      RETURN
C*****************************************************************************C
C  PATH FOR MAKING ENTRY TO VARIB                                             3
C*****************************************************************************C
   40 IEND=IEND+1
      IRAY(IEND,1)=J1
      IRAY(IEND,2)=J2
      IRAY(IEND,3)=NODE
      RETURN
      END
*DECK INPUT
      SUBROUTINE INPUT
      IMPLICIT INTEGER (A-Z)
C*****************************************************************************C
C  SUBROUTINE INPUT IS DESIGNED TO LOAD TO MAIN STORAGE FROM EXTERNAL STORAGE  C
C   THE GLOBAL TABLE CNTAB PRODUCED BY FACES.                                 C
C*****************************************************************************C
      COMMON /GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAR, IPAR, JPRIME
      CALL READGTS(2)
      RETURN
      END
```

```
*DECK INPUT2
      SUBROUTINE INPUT2(MDNAM1,MDNAM2)
      IMPLICIT INTEGER (A-Z)
C*********************************************************************
C  THE PURPOSE OF THIS ROUTINE IS TO LOAD THE LOCAL TABLES SYMTAB,NODTAB,      C
C  PRETAB, AND USETAB, AS PRODUCED BY FACES.  THE GLOBAL TABLE DIREC MUST      C
C  FIRST BE LOADED IN ORDER TO DETERMINE THE MODULE NUMBER OF THE ROUTINE      C
C  WHICH WILL HAVE ITS LOCAL TABLES INPUT.  THE MACHINE-DEPENDENT ROUTINE      C
C  NTRAN IS USED TO LOAD THE TABLES FROM FILE.                                 C
C                                                                              C
C  PARAMETERS                                                                  C
C  MDNAM1 AND MDNAM2 CONTAIN THE NAME OF THE ROUTINE, FOUR CHARACTERS PER WORD C
C  LEFT-JUSTIFIED AND BLANK-FILLED.                                            C
C                                                                              C
C  IMPORTANT VARIABLES (COMMON VARIABLES ARE DESCRIBED IN TRACE)               C
C  ISZ--THE SIZE OF THE ARRAY TO BE READ IN, IN TOTAL WORDS                    C
C  ISTAT--A STATUS WORD AS REQUIRED BY NTRAN                                   C
C  NUM--THE AVERAGE OF THE TWO NUMBERS IN MDNAM1 AND MDNAM2, USED TO REFERENCE C
C       THE HASH CODED TABLE DIREC                                             C
C  IN--THE HASH CODE PRODUCED                                                  C
C  MDNUM--THE MODULE NUMBER AS FOUND IN DIREC                                  C
C*********************************************************************
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     2   ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME,NODTAB(3,1000)
     3  ,LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,STALTB(2,100),
     4   LSTAL,INDST
      COMMON /GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAR, IPAR, JPRIME
C*********************************************************************
C  LOAD THE GLOBAL TABLE DIREC                                                 C
C*********************************************************************
      CALL READGTS(1)
C*********************************************************************
C  DETERMINE THE LOCATION IN DIREC OF THE NAME OF THE ROUTINE                  C
C*********************************************************************
   2  NUM1=FLD(0,24,MDNAM1)
      NUM2=FLD(0,24,MDNAM2)
      NUM=(NUM1+NUM2)/2
      IN=MOD(NUM,97)+1
   3  IF(IN.GT.LDIR)IN=1
      IF(DIREC(1,IN).EQ.0)GOTO 50
      IF(DIREC(1,IN).NE.MDNAM1)GOTO 4
      IF(DIREC(2,IN).NE.MDNAM2)GOTO 4
      MDNUM=LOFECH(DIREC(3,IN))
      GOTO 5
   4  IN=IN+1
      GOTO 3
C*********************************************************************C
C  LOAD SYMTAB                                                         C
C*********************************************************************C
5     CALL READLTS(MDNUM,1)
C*********************************************************************
C  LOAD NODTAB                                                         C
C*********************************************************************
      CALL READLTS(MDNUM,2)
C*********************************************************************C
```

```
C  LOAD SUCTAB
C*****************************************************************************C
      CALL READLTS(MDNUM,3)
C*****************************************************************************C
C  LOAD PRETAB
C*****************************************************************************C
      CALL READLTS(MDNUM,4)
C*****************************************************************************C
C  LOAD USETAB
C*****************************************************************************C
      CALL READLTS(MDNUM,5)
      RETURN
   50 PRINT 702
  702 FORMAT(20H1    MODULE NOT FOUND      )
      MDNAM1=0
      RETURN
      END
*DECK LOAD
      SUBROUTINE LOAD(M)
      IMPLICIT INTEGER (A-Z)
C*****************************************************************************C
C  LOAD TRANSFERS FROM EXTERNAL STORAGE TO MAIN STORAGE THE LOCAL TABLES (AS   C
C  PRODUCED BY FACES3) SYMTAB,ALTAB,AND STALTB FOR THE MODULE IDENTIFIED       C
C  IN CNTAB(4,M).  IT REQUIRES THE USE OF THE MACHINE-DEPENDENT ROUTINE        C
C  NTRAN.  ONLY ONE SET OF TABLES ARE IN CENTRAL MEMORY AT ANY ONE TIME.       C
C                                                                              C
C  PARAMETER                                                                   C
C  M--THE INDEX IN CNTAB OF THE COMMON BLOCK BEING CONSIDERED                  C
C*****************************************************************************C
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     2   ALTAB(56),LAL,INDAL,DCTAB(3,50),LDC,INDDO,IPRIME,NODTAB(3,1000)
     3  ,LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,STALTB(2,100),
     4   LSTAL,INDST
      COMMON /GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAP, IPAP, JPRIME
C*****************************************************************************C
C  READ SYMBOL TABLE
C*****************************************************************************C
      K=CNTAB(4,M)
      CALL READLTS(K,1)
C*****************************************************************************C
C  READ ARRAY LENGTH TABLE
C*****************************************************************************C
      CALL READLTS(K,6)
C*****************************************************************************C
C  READ STORAGE ALLOCATION TABLE.
C*****************************************************************************C
      CALL READLTS(K,8)
      RETURN
      END
*DECK NCHECK
      INTEGER FUNCTION NCHECK(NODE,IRAY)
C*****************************************************************************
C  NCHECK IS A FUNCTION WHICH RETURNS THE VALUE OF THE SPECIFIED BIT LOCATION. C
C     ITS PARAMETERS, VARIABLES, AND REQUIRED ROUTINES ARE THE SAME AS THOSE   C
C     DESCRIBED FOR SUBROUTINE ENTER                                           C
C                                                                              C
```

```
C*********************************************************************
      INTEGER FLD
      DIMENSION IRAY(30)
C*********************************************************************
C  DERIVE THE CORRECT WORD AND BIT LOCATION IN IRAY
C*********************************************************************
      IWORD=NODE/36+1
      IBIT=NODE-(IWORD-1)*36
      IF(IBIT.EQ.0)GOTO 20
      IBIT=IBIT-1
      IBIT=IABS(IBIT)
      GOTO 30
   20 IBIT=35
      IWORD=IWORD-1
C*********************************************************************
C  ASSIGN TO NCHECK THE VALUE OF THE PROSCRIBED BIT
C*********************************************************************
   30 NCHECK=FLD(IBIT,1,IRAY(IWORD))
      RETURN
      END
*CECK PARAL
      SUBROUTINE PARAL(IATOPT)
C*********************************************************************C
C  THE PURPOSE OF THIS ROUTINE IS TO CHECK THE ALIGNMENT OF PARAMETERS C
C  IN CALLS TO ALL SUBROUTINES AND FUNCTIONS ANALYZED BY THE FACES     C
C  PACKAGE.  EACH ACTUAL PARAMETER LIST IS CHECKED AGAINST THE CORRES- C
C  PONDING FORMAL PARAMETER LIST TO INSURE THAT CORRESPONDING PARAME-  C
C  TERS ARE OF THE SAME TYPE AND THAT THE PARAMETER LISTS ARE OF THE    C
C  SAME LENGTH.  IF THE OPTION TO THIS ROUTINE SO DESIGNATES (SPECIFIED C
C  BY ARGUMENT IATOPT), THE DIMENSIONS OF CORRESPONDING PARAMETERS WILLC
C  BE COMPARED IF EITHER OF THE ACTUAL OR FORMAL PARAMETERS IS AN       C
C  ARRAY.  THE FOLLOWING TABLES, AS PRODUCED BY THE FORTRAN FRONT-END, C
C  ARE READ FROM A FILE AND USED IN THE ANALYSIS   PARTAB--GLOBAL      C
C  PARAMETER TABLE, DIREC--GLOBAL DIRECTORY, SYMTAB--LOCAL SYMBOL      C
C  TABLES, ALTAB--LOCAL ARRAY LENGTH TABLES.                           C
C  DESCRIPTION OF VARIABLES                                            C
C  IATOPT--OPTION TO CHECK ARRAY DIMENSIONS.  WHEN VALUE IF 0, DIMEN-  C
C  SIONS ARE NOT CHECKED.  OTHERWISE, THEY ARE CHECKED.                C
C  DIREC(3,100)--ARRAY INTO WHICH GLOBAL DIRECTORY IS STORED.  CONTAINSC
C  NAMES AND MODULE NO. S OF ALL ROUTINES IN THE SYSTEM BEING ANALYSED.C
C  PARTAB(300)--ARRAY INTO WHICH PARAMETER TABLE IS STORED.  CONTAINS  C
C  POINTERS TO ALL PARAMETERS IN SYSTEM AND SPECIAL CODES DESCRIBING   C
C  THEM.                                                               C
C  SYMTAB(5,2000)--ARRAY INTO WHICH SYMBOL TABLES ARE STORED WHEN READ C
C  FROM FILE.  CONTAINS NAMES OF ALL VARIABLES, LABELS, ETC. IN A      C
C  SPECIFIC MODULE ALONG WITH INFORMATION DESCRIBING THE NAME.         C
C  ALTAB(50)--ARRAY INTO WHICH ARRAY LENGTH TABLES ARE STORED WHEN READC
C  FROM FILE.  CONTAINS LENGTHS OF ALL ARRAYS IN A SPECIFIC MODULE.    C
C  ITYPE(20)--ARRAY IN WHICH FORMAT PARAMETER TYPE CODES ARE STORED    C
C  AND POINTERS TO IDP FOR ARRAYS.                                     C
C  IDP(30)--CONTAINS DIMENSIONS OF FORMAL PARAMETERS AS STORED IN ALTAB.
C  MN--MODULE NO. OF ROUTINE WHOSE PARAMETERS ARE CURRENTLY BEING      C
C  CHECKED.                                                            C
C  IPAR--INDEX TO PARAMETER TABLE                                      C
C  INIDP--INDEX TO IDP ARRAY.                                          C
C  NP--NO. OF FORMAL PARAMETERS IN CURRENT LIST.                       C
```

```
C  ISYM--INDEX TO SYMTAB
C  IAL--INDEX TO ALTAB                                                    C
C  MODNO--MODULE NO. IN WHICH A CALL IS MADE TO ROUTINE WHOSE PARA-       C
C  METERS ARE BEING CHECKED.                                              C
C  ICODE--PARAMETER CODE AS STORED IN PARTAB.                             C
C  ISTAKP--STACK VARIABLE (SEE EXTERNAL DOCUMENTATION)                    C
C  K--ARGUMENT NO. AS STORED IN PARTAB.                                   C
C**********************************************************************C
      IMPLICIT INTEGER(A-Z)
      DIMENSION ITYPE(20),IDP(30)
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB(56),LAL,INDAL,DCTAB(3,50),LDC,INDDC,IPRIME
     1,NODTAB(3,1000),LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,
     1 STALTR(2,100),LSTAL,INDST
      COMMON/GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAR, IPAR, JPRIME
C**********************************************************************C
C  READ DIRECTORY AND PARAMETER TABLE AND ARRAY LENGTH TABLE IF OPTION IS SET.
C**********************************************************************C
      PRINT 2
2     FORMAT(1H1,* PARAMETER ALIGNMENT MESSAGES*/)
      CALL TABLNTH
      CALL READGTS(1)
      CALL READGTS(3)
20    LENPAR=PARTAB(1)
      DO 1000 J=1,LDIR
C**********************************************************************C
C  FIND NEXT MODULE IN DIREC.
C**********************************************************************C
      IF(DIREC(1,J).EQ.0) GO TO 1000
C**********************************************************************C
C  IS MODULE DEFINED.
C**********************************************************************C
      IF(HIFECH(DIREC(3,J)).EQ.0) GO TO 1000
C**********************************************************************C
C  IS MODULE A SUBROUTINE OR FUNCTION.
C**********************************************************************C
      MN=LOFECH(DIREC(3,J))
      IF(MN.LT.200) GO TO 1000
C**********************************************************************C
C  READ SYMBOL TABLE FOR THIS MODULE.
C**********************************************************************C
      CALL READLTS(MN,1)
300   IF(IATOPT.EQ.0) GO TO 30
C**********************************************************************C
C  READ ARRAY LENGTH TABLE.
C**********************************************************************C
      CALL READLTS(MN,6)
C**********************************************************************C
C  LOCK UP ROUTINE NAME AND SEE IF IT HAS PARAMETERS.
C**********************************************************************C
30    CALL HASHMN(DIREC(1,J),DIREC(2,J),MNADD)
      ITEMP=SYMTAB(5,MNADD)
      IPAR=HIFECH(ITEMP)
      IF(IPAR.EQ.0) GO TO 1000
C**********************************************************************C
```

```
C  REFERENCE PARTAB STORING NO. OF PARAMS, THEIR TYPES, AND DIMENSIONS.
C*************************************************************************C
      INIDP=0
      NP=0
35    IPAR=IPAR+1
      IF(HTFECH(PARTAB(IPAR)).NE.0) GO TO 50
      NP=NP+1
      ISYM=LPFECH(PARTAB(IPAR))
      ITEMP=SYMTAB(3,ISYM)
      ITYPE(NP)=HIFECH(ITEMP)
      IF(IATOPT.EQ.0) GO TO 35
      ITEMP=SYMTAB(5,ISYM)
      IAL=LOFECH(ITEMP)
      IF(IAL.EQ.0) GO TO 35
      INIDP=INIDP+1
      ITYPE(NP)=HISTOR(ITYPE(NP),INIDP)
      IDP(INIDP)=ALTAB(IAL)
      IF(HTFECH(ALTAB(IAL+1)).EQ.0) GO TO 35
      INIDP=INIDP+1
      IDP(INIDP)=ALTAB(IAL+1)
      GO TO 35
C*************************************************************************C
C  SEARCH PARTAB FOR CALLS TO THIS ROUTINE FROM OTHER ROUTINES AND CHECK
C    PARAMETER ALIGNMENT.
C*************************************************************************C
50    I=0
55    I=I+1
      IF(I.GT.LENPAR) GO TO 1000
      IF(HTFECH(PARTAB(I)).NE.3) GO TO 55
      MODNO=HFFECH(PARTAB(I))
      IF(MODNO.EQ.MN) GO TO 55
      IF(MODNO.EQ.MSAV) GO TO 70
C*************************************************************************C
C  READ SYMBOL TABLE FOR THIS MODULE.
C*************************************************************************C
      MSAV=MODNO
      CALL READLTS(MODNO,1)
70    ISYM=LPFECH(PARTAB(I))
      IF(SYMTAB(1,ISYM).NE.DIREC(1,J)) GO TO 55
      IF(SYMTAB(2,ISYM).NE.DIREC(2,J)) GO TO 55
C*************************************************************************C
C  READ ALTAB IF OPTION IS ON.
C*************************************************************************C
      IF(IATOPT.EQ.0) GO TO 80
      CALL READLTS(MODNO,6)
C*************************************************************************C
C  BEGIN PARAMETER COMPARISON.
C*************************************************************************C
80    MFLAG=0
      ISTAKP=1
85    I=I+1
      IF(I.LE.LPAR) GO TO 87
      PRINT 86
86    FORMAT(* ATTEMPT TO SEARCH PAST LENGTH OF PARTAB.*)
      RETURN
```

```
 87     ICODE=HTFECH(PARTAB(I))
        IF(ICODE.NE.3) GO TO 90
        ISTAKP=ISTAKP+1
        GO TO 85
 90     IF(ICODE.NE.4) GO TO 100
        ISTAKP=ISTAKP-1
        IF(ISTAKP.NE.0) GO TO 85
        IF(K.EQ.NP) GO TO 55
        PRINT 95,DIREC(1,J),DIREC(2,J),MODNO
 95     FORMAT(*0PARAMETER LIST IN CALL TO *,2A4,*FROM MODULE*, I1,
       1 * IS TOO SHORT.*)
        GO TO 55
 100    IF(ICODE.NE.2) GO TO 110
        IF(ISTAKP.EQ.1) GO TO 105
        ISTAKP=ISTAKP-1
        GO TO 85
 105    I=LPFECH(PARTAB(I))
        IF(K.EQ.NP) GO TO 80
        PRINT 95,DIREC(1,J),DIREC(2,J),MODNO
        GO TO 80
 110    IF(ISTAKP.NE.1) GO TO 85
C*******************************************************************************C
C  CHECK NO. OF PARAMETERS.
C*******************************************************************************C
        K=MFFECH(PARTAB(I))
        IF(K.LE.NP) GO TO 120
        PRINT 115,DIREC(1,J),DIREC(2,J),MODNO
 115    FORMAT(*0TOO MANY PARAMETERS IN CALL TO *,2A4,* FROM MODULE*,I4)
        GO TO 55
C*******************************************************************************C
C  CHECK PARAMETER TYPE.
C*******************************************************************************C
 120    IF(ICODE.EQ.5) GO TO 122
        ISYM=LPFECH(PARTAB(I))
        ITEMP=SYMTAB(3,ISYM)
        NTYP=HIFECH(ITEMP)
        GO TO 124
 122    NTYP=LPFECH(PARTAB(I))
 124    IF(NTYP.EQ.LOFECH(ITYPE(K)))GO TO 130
        PRINT 125,K,DIREC(1,J),DIREC(2,J),MODNO
 125    FORMAT(*0PARAMETER NO.*,I3,* IN CALL TO *,2A4,*FROM MODULE*,I4,
       1 * IS OF WRONG TYPE.*)
 130    IF(IATOPT.EQ.0) GO TO 85
C*******************************************************************************C
C  CHECK ARRAY LENGTH
C*******************************************************************************C
        IF(ICODE.EQ.5) GO TO 85
        ITEMP=SYMTAB(5,ISYM)
        IAL=LOFECH(ITEMP)
        IF(IAL.EQ.0) GO TO 85
        JAL=HIFECH(ITYPE(K))
        IF(JAL.EQ.0) GO TO 85
        IF(IDP(JAL).EQ.ALTAB(IAL)) GO TO 140
 135    PRINT 137,SYMTAB(1,ISYM),SYMTAB(2,ISYM),DIREC(1,J),DIREC(2,J),
       1 MODNO
 137    FORMAT(*0ARRAY *,2A4,*IN CALL TO *,2A4,*FROM MODULE*,I4,
```

```
     1* DIFFERS IN DIMENSIONS FROM THE CORRESPONDING FORMAL PARAMETER.*)
      GO TO 85
140   IF(HTFECH(ALTAB(IAL+1)).EQ.0) GO TO 145
      IF(ALTAB(IAL+1).EQ.IDP(JAL+1) ) GO TO 85
      GO TO 135
145   IF(HCFECH(IDP(JAL+1)).EQ.0) GO TO 85
      GO TO 135
1000  CONTINUE
      PRINT 1001
1001  FORMAT(1HO,*END OF PARAMETER ALIGNMENT CHECK*)
      RETURN
      END
*DECK PATHS
      SUBROUTINE PATHS(I,MD1,MD2,V1,V2,NODE)
C*****************************************************************************C
C  PATHS IS A PART OF THE DIAGNOSTIC PACKAGE ASSOCIATED WITH THE FACES SYSTEM.  C
C  IT IS A MASTER ROUTINE WHOSE FUNCTION IS TO REFERENCE OTHER ROUTINES. THE    C
C  ROUTINES IT CALLS PERFORM VARIABLE TRACES, EITHER FORWARD BY CALL TO FWVT,   C
C  OR BACKWARD BY CALL TO BWVT. (A DESCRIPTION OF THESE TRACES APPEAR IN THE    C
C  LISTING OF THE TRACE ROUTINES THEMSELVES.)  PATHS FIRST REFERENCES INPUT2    C
C  TO LOAD ALL NEEDED TABLES.  THEN IF I=1 IT CALLS FWVT.  OTHERWISE BWVT IS    C
C  REFERENCED.                                                                  C
C                                                                               C
C  PARAMETERS                                                                   C
C  I--FLAG AS DESCRIBED ABOVE                                                   C
C  MD1,MD2--NAME OF MODULE UNDER CONSIDERATION, 4 CHARACTERS PER WORD,LEFT-     C
C        JUSTIFIED AND BLANK-FILLED                                             C
C  V1,V2--VARIABLE NAME TO BE TRACED, STORED LIKE THE MODULE NAME               C
C  NODE--NODE FROM WHICH THE TRACE ORIGINATES                                   C
C*****************************************************************************C
      IMPLICIT INTEGER (A-Z)
      PRINT 101,MD1,MD2
101   FORMAT(27H1TARGET MODULE FOR PATHS -  ,2A4)
      CALL TABLNTH
      CALL INPUT2(MD1,MD2)
      IF(MD1.EQ.0)RETURN
      IF(I.EQ.1)GOTO 20
      CALL BWVT(V1,V2,NODE)
      RETURN
   20 CALL FWVT(V1,V2,NODE)
      RETURN
      END
*DECK READGTS
      SUBROUTINE READGTS(NTABLE)
C*****************************************************************************C
C     ROUTINE READGTS READS IN GLOBAL TABLES FROM FILE                        C
C     NTABLE = 1    DIREC                                                     C
C            = 2    CNTAB                                                     C
C            = 3    PARTAB                                                    C
C                                                                             C
C     NGTBS = NUMBER OF GLOBAL TABLES                                         C
C                                                                             C
C     SUBROUTINE RDDISK (IR, ARRAY, KR) : A CAL SYSTEM ROUTINE FOR            C
C     READING RANDOM FILE RECORDS.                                            C
C     IR = THE IDENTIFICATION NUMBER OF THE RECORD TO BE READ                 C
C     ARRAY = FIRST WORD OF THE BLOCK INTO WHICH THE DATA IN THE              C
```

```
C              INDICATED RECORD ARE TO BE READ.                          C
C      KP = THE NUMBER OF WORDS TO BE READ FROM THE INDICATED RECORD.    C
C                                                                        C
C      RDDISK IS A MACHINE DEPENDENT ROUTINE.                            C
C*************************************************************************C
       IMPLICIT INTEGER (A-Z)
       COMMON /GLOBAL/ DIREC (3,100), LDIR, INDIR, CNTAB (4,300), LCN,
      1 INDCN, PARTAB (2000), LPAR, IPAR, JPRIME
       DATA NGTBS /3/
       IF ( (NTABLE.GE.1) .AND. (NTABLE.LE.NGTBS) ) GO TO 20
       PRINT 10, NTABLE
    10 FORMAT (////, 1H , *SUBROUTINE READGTS*,/, 1H ,
      1 *COMPUTED GO TO STATEMENT INDEX OUT OF RANGE*,/,
      2 1H , *NTABLE = *, I4)
       STOP
    20 GO TO (1000, 2000, 3000), NTABLE
C*************************************************************************C
C      READ GLOBAL TABLE.                                                C
C*************************************************************************C
 1000 KOUNT = LDIR * 3
       CALL RDDISK (NTABLE, DIREC, KOUNT)
       RETURN
C*************************************************************************C
C      READ CNTAB.                                                       C
C      CNTAB (1,1) CONTAINS CNTAB CURRENT INDEX.                         C
C*************************************************************************C
 2000 KOUNT = LCN * 4
       CALL RDDISK (NTABLE, CNTAB, KOUNT)
       INDCN = CNTAB (1,1)
       RETURN
C*************************************************************************C
C      READ PARTAB                                                       C
C      PARTAB (1) CONTAINS PARTAB CURRENT POINTER.                       C
C*************************************************************************C
 3000 KOUNT = LPAR
       CALL RDDISK (NTABLE, PARTAB, KOUNT)
       IPAR = PARTAB (1)
       RETURN
       END
*DECK ERPRINT
       SUBROUTINE ERPRINT(NWL,NTAB)
       PRINT 99,NWL,NTAB
99     FORMAT(1H1,I5,* WORDS NOT READ IN TABLE *,A7)
       RETURN
       END
*DECK READLTS
       SUBROUTINE READLTS(MODNC,NTABLE)
C*************************************************************************C
C      ROUTINE READLTS READS IN LOCAL TABLES FROM FILE                   C
C      MODNO = MODULE NUMBER                                             C
C      NTABLE =      LOCAL TABLE NUMBER                                  C
C             = 1    SYMTAB                                              C
C             = 2    MODTAB                                              C
C             = 3    SUCTAB                                              C
C             = 4    PRETAB                                              C
C             = 5    USETAB                                              C
```

```
C             = 6   ALTAB                                              C
C             = 7   DDTAB                                              C
C             = 8   STALTB                                             C
C                                                                      C
C     NLTBS = NUMBER OF LOCAL TABLES IN A MODULE                       C
C     LGTBS = NUMBER OF GLOBAL TABLES                                  C
C                                                                      C
C     SUBROUTINE RDDISK (IR, ARRAY, KR) : A CAL SYSTEM ROUTINE FOR     C
C     READING RANDOM FILE RECORDS.                                     C
C     IR = THE IDENTIFICATION NUMBER OF THE RECORD TO BE READ          C
C     ARRAY = FIRST WORD OF THE BLOCK INTO WHICH THE DATA IN THE       C
C             INDICATED RECORD ARE TO BE READ.                         C
C     KR = THE NUMBER OF WORDS TO BE READ FROM THE INDICATED RECORD.   C
C                                                                      C
C     RDDISK IS A MACHINE DEPENDENT ROUTINE.                           C
C*********************************************************************C
      IMPLICIT INTEGER (A-Z)
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     1 ALTAB (56), LAL, INDAL, DDTAB (3,50), LDD, INDDD, IPRIME,
     2 NDDTAB (3,1000), LNOD, SUCTAB (1500), ISUC, PRETAB (1500), IPRE,
     3 LPS, STALTB (2,100), LSTAL, INDST
      DATA NLTBS /8/, NGTBS /3/
C*********************************************************************C
C     EVALUATE THE STARTING RECORD NUMBER OF MODULE.                   C
C*********************************************************************C
      M = MODNO
      IBSN = ( MOD(M,100) - 1 ) * NLTBS + NGTBS
      IF ( (NTABLE.GE.1) .AND. (NTABLE.LE.NLTBS) ) GO TO 20
      PRINT 10, NTABLE
   10 FORMAT (////, 1H , *SUBROUTINE READLTS*,/, 1H ,
     1 *COMPUTED GO TO STATEMENT INDEX OUT OF RANGE*,/,
     2 1H , *NTABLE = *, I4)
      STOP
   20 IBSN = IBSN + NTABLE
      GO TO (1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000), NTABLE
C*********************************************************************C
C     READ SYMTAB.                                                     C
C*********************************************************************C
 1000 KOUNT = LSYM * 5
      CALL RDDISK (IBSN, SYMTAB, KOUNT)
      RETURN
C*********************************************************************C
C     READ  NDDTAB.                                                    C
C*********************************************************************C
 2000 KOUNT = LNOD * 3
      CALL RDDISK (IBSN, NDDTAB, KOUNT)
      RETURN
C*********************************************************************C
C     READ SUCTAB.                                                     C
C     SUCTAB (1) CONTAINS SUCTAB LAST ENTRY POINTER.                   C
C*********************************************************************C
 3000 KOUNT = LPS
      CALL RDDISK (IBSN, SUCTAB, KOUNT)
      ISUC = SUCTAB (1)
      RETURN
C*********************************************************************C
```

```
C     READ PRETAB.
C     PRETAB (1) CONTAINS POINTER TO THE LAST ENTRY OF PRETAB.
C*****************************************************************
 4000 KOUNT = LPS
      CALL RDDISK (IBSN, PRETAB, KOUNT)
      IPRE = PPETAB (1)
      RETURN
C*****************************************************************
C     READ USETAB.
C     USETAB (1,1) CONTAINS USETAB LAST ENTRY POINTER.
C*****************************************************************
 5000 KOUNT = LUSE * 2
      CALL RDDISK (IBSN, USETAB, KOUNT)
      INDUT = USETAB (1,1)
      RETURN
C*****************************************************************
C     READ ALTAB
C     ALTAB (1) CONTAINS ALTAB LAST ENTRY POINTER.
C*****************************************************************
 6000 KOUNT = LAL
      CALL RDDISK (IBSN, ALTAB, KOUNT)
      INDAL = ALTAB (1)
      RETURN
C*****************************************************************
C     READ DOTAB.
C     DOTAB (1,1) CONTAINS POINTER TO LAST ENTRY OF DOTAB.
C*****************************************************************
 7000 KOUNT = LDO * 3
      CALL RDDISK (IBSN, DOTAB, KOUNT)
      INDDO = DOTAB (1,1)
      RETURN
C*****************************************************************
C     READ STALTB.
C     STALTB (1,1) CONTAINS POINTER TO THE LAST ENTRY OF STALTB.
C*****************************************************************
 8000 KOUNT = LSTAL * 2
      CALL RDDISK (IBSN, STALTB, KOUNT)
      INDST = STALTB (1,1)
      RETURN
      END
*DECK RESTOR
      SUBROUTINE RESTOR
C*****************************************************************
C  THIS ROUTINE TRANSFERS THE ENTRIES OF SEGMENT THREE OF CBCODE INTO
C  SEGMENT ONE OF CBCODE.   SEGMENT ONE IS CONSIDERED TO ALWAYS CONTAIN CODE
C  FOR THE STANDARD COMMON BLOCK.  IF THE ORIGINAL ENTRY INTO SEGMENT ONE IS
C  NOT IN ALIGNMENT WITH THE ENTRIES IN SEGMENTS TWO AND THREE, THEN IT MUST
C  BE REPLACED.  KK REPRESENTS THE LENGTH OF THE ENTRY IN SEGMENT THREE.
C*****************************************************************
      COMMON /CODE/CBCODE(50,4,3),ERROR(20),N1,N2,KFLAG,NPNTR
      INTEGER CBCODE
C*****************************************************************
C  DETERMINE NUMBER OF ENTRIES IN SEGMENT THREE OF CBCODE
C*****************************************************************
      KK=CBCODE(1,1,3)
      CBCODE(1,1,1)=KK
```

```
C*****************************************************************************C
C  TRANSFER CODE FROM SEGMENT THREE TO SEGMENT ONE                             C
C*****************************************************************************C
      DO 15  I=2,KK
      DO 10 J=1,4
   10 CBCODE(I,J,1)=CBCODE(I,J,3)
   15 CONTINUE
      RETURN
      END
*DECK SEARCH
      INTEGER FUNCTION SEARCH(J1,J2,LAST,IRAY,K)
C*****************************************************************************C
C  SEARCH DETERMINES IF A VARIABLE IS ENTERED IN THE ARRAY IRAY.  IF K 0, THEN C
C  THE SEARCH INCLUDES DETERMINATION IF THE NODE ASSOCIATED WITH THE VARIABLE  C
C  IS EQUAL TO K.  IF SEARCH IS SUCCESSFUL ITS VALUE IS THE INDEX IN IRAY OF   C
C  THE MATCHING ENTRY.  OTHERWISE SEARCH=0.                                    C
C                                                                              C
C  PARAMETERS                                                                  C
C  J1,J2--THE VARIABLE SEARCHED FOR, FOUR CHARACTERS PER WORD, LEFT-JUSTIFIED  C
C  LAST--THE FINAL ENTRY OF IRAY                                               C
C  IRAY--THE ARRAY TO BE SEARCHED                                              C
C  K--A FLAG AS DESCRIBED ABOVE                                                C
C*****************************************************************************C
      DIMENSION IRAY(200,3)
      DO 10 I=1,LAST
      IF(J1.NE.IRAY(I,1))GOTO 10
      IF(J2.NE.IRAY(I,2))GOTO 10
      IF(K.EQ.0)GOTO 20
      IF(K.EQ.IRAY(I,3))GOTO 20
   10 CONTINUE
      SEARCH=0
      RETURN
   20 SEARCH=I
      RETURN
      END
*DECK SRTOUT
      SUBROUTINE SRTOUT(LOUT)
      IMPLICIT INTEGER (A-Z)
C*****************************************************************************C
C  THIS ROUTINE IS DESIGNED TO SORT THE CONTENTS OF THE ARRAY OUT SO THAT THE  C
C  VARIABLES APPEAR IN ALPHABETICAL ORDER AND, FOR MORE THAN ONE ENTRY OF A    C
C  SINGLE VARIABLE, THE NODES ARE LISTED IN THEIR ORDER OF APPEARANCE.  THE    C
C  WORDS OUT(I,1),OUT(I,2),AND OUT(I,3), FOR EACH I, ARE TREATED AS IF THEY    C
C  WERE A SINGLE STRING OF CHARACTERS.                                         C
C                                                                              C
C  PARAMETER--LOUT--THE NUMBER OF ENTRIES TO OUT                               C
C*****************************************************************************C
      COMMON /STACKS/NODSTK(1000),UNSTCK(30),LIST(30),NFLAG
     2               ,VARIB(200,3),OUT(200,3)
      I=0
   5  I=I+1
      IF(I.GE.LOUT)GOTO 50
      J=I+1
      DO 10 KK=1,3
      IF(OUT(I,KK).GT.OUT(J,KK))GOTO 15
      IF(OUT(I,KK).LT.OUT(J,KK))GOTO  5
```

```
   10  CONTINUE
       RETURN
   15  DO 22 KK=1,3
       L=OUT(I,KK)
       OUT(I,KK)=OUT(J,KK)
   22  OUT(J,KK)=L
       II=I-1
   25  IF(II.LT.1)GOTO 5
       JJ=II+1
       DO 30  KK=1,3
       IF(OUT(II,KK).GT.OUT(JJ,KK))GOTO 35
       IF(OUT(II,KK).LT.OUT(JJ,KK))GOTO  5
   30  CONTINUE
       RETURN
   35  DO 37 KK=1,3
       L=OUT(II,KK)
       OUT(II,KK)=OUT(JJ,KK)
   37  OUT(JJ,KK)=L
       II=II-1
       GOTO 25
   50  CONTINUE
       RETURN
       END
*DECK TABLNTH
       SUBROUTINE TABLNTH
C**************************************************************************C
C  INITIALIZE LENGTH OF LOCAL TABLES                                       C
C**************************************************************************C
       COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INOUT,
      C ALTAB(56), LAL, INDAL, DOTAB(3,50), LDO, INODO, IPRIME,
      C NODTAB(3,1000), LNOD, SUCTAB(1500), ISUC, PRETAB(1500), IPRE,
      C LPS, STALTB(2,100), LSTAL, INDST
       LSYM = 1800
       LUSE = 2500
       LAL = 56
       LDO = 50
       IPRIME = 1799
       LNOD = 1000
       LPS = 1500
       LSTAL = 100
       RETURN
       END
*DECK TRACE
       SUBROUTINE TRACE(MDNAM1,MDNAM2)
       IMPLICIT INTEGER (A-Z)
C**************************************************************************
C  TRACE IS PART OF THE DIAGNOSTIC PACKAGE ASSOCIATED WITH THE FACES SYSTEM.  C
C  IT PROVIDES A MEANS OF VERIFYING THAT EACH VARIABLE IN A ROUTINE IS IN     C
C  SOME WAY INITIALIZED BEFORE BEING USED IN ANY OF 3 WAYS  TO AFFECT THE VALUE
C  OF OTHER VARIABLES, TO REGULATE THE FLOW OF CONTROL, OR TO AFFECT OUTPUT.  C
C  A VARIABLE OR AN ARRAY AND IS NOT HELD IN COMMON OR REFERENCED IN A DATA   C
C  STATEMENT, AN INITIALIZATION TRACE IS BEGUN. FOR EACH INPUT USAGE FOR SUCH C
C  A SYMBOL, ALL POSSIBLE ENTRY PATHS ARE TRACED BACK.  EACH PATH MUST AT     C
C  SOME NODE ALONG ITS LENGTH INITIALIZE THE VARIABLE.  THIS IS ASSURED BY    C
C  THE FOLLOWING STEPS                                                        C
```

```
C     (1) LOCATE AN INPUT USAGE.  IF NONE EXISTS, PROCESSING TERMINATES FOR    C
C         THAT VARIABLE.                                                       C
C     (2) FLAG THAT NODE AND PLACE ALL OF ITS PREDECESSOR NODES, THAT ARE NOT  C
C         FLAGGED, ON A STACK.  IF ANY OF THESE NODES ARE OUTSIDE THE RANGE    C
C         OF THE ROUTINE, GOTO STEP 7.                                         C
C     (3) IF THE STACK IS EMPTY GOTO STEP 1                                    C
C     (4) DOES THE TOP NODE ON THE STACK USE THE VARIABLE.  IF NOT REMOVE IT   C
C         AND GO TO STEP 2.                                                    C
C     (5) IS THE USE OF THE VARIABLE AN INITIALIZATION.  IF NOT REMOVE THE     C
C         NODE FROM THE STACK AND GO TO STEP 2.                                C
C     (6) REMOVE THE NODE FROM THE STACK AND GO TO STEP 3.                     C
C     (7) PRINT A DIAGNOSTIC MESSAGE FOR THAT SYMBOL.                          C
C   THE  NEXT SYMBOL IS THEN FOUND AND PROCESSING CONTINUES WITH IT.  FOR      C
C   SITUATIONS WHICH MAY RESULT IN INACCURATE DIAGNOSTICS SEE THE EXTERNAL     C
C   DOCUMENTATION.                                                           C
C                                                                            C
C   PARAMETERS                                                               C
C   MDNAM1--CONTAINS THE FIRST FOUR (OR FEWER) CHARACTERS, LEFT-JUSTIFIED, OF C
C           THE ROUTINE TO BE ANALYZED                                       C
C   MDNAM2--CONTAINS THE FIFTH THROUGH EIGHTH CHARACTERS(IF THEY EXIST) OF THE C
C           ROUTINE TO BE ANALYZED.                                          C
C                                                                            C
C   COMMON VARIABLES REFERENCED BY TRACE OR ITS SUBROUTINES                  C
C   SYMTAB(5,LSYM)--A LOCAL TABLE PRODUCED BY THE FACES SYSTEM  IT CONTAINS AN C
C                ENTRY FOR EACH SYMBOL IN THE ROUTINE.  THIS AND ALL OTHER FACES C
C                TABLES ARE DESCRIBED IN FULL IN THE EXTERNAL DOCUMENTATION.  C
C   NODTAB(3,LNOD)--THE LOCAL NODE TABLE                                     C
C   PRETAB(IPRE)--TABLE OF PREDECESSOR NODES FOR EACH NODE IN NODTAB         C
C   USETAB(2,LUSE)--TABLE OF USAGES OF EACH SYMBOL IN SYMTAB                 C
C   DIREC(3,100)--GLOBAL TABLE OF THE NAMES OF EACH ANALYZED ROUTINE         C
C   NODSTK(1000)--A STACK OF EACH NODE WHOSE PREDECESSORS ARE TO CHECKED IN THE C
C                INITIALIZATION VERIFICATION                                 C
C   UNSTCK(LUNS)--A TABLE USED TO FLAG THOSE NODES WHICH SHOULD NOT BE PLACED C
C                ON NODSTK                                                   C
C   LIST--A TABLE USED TO FLAG THOSE NODES IN WHICH THE VARIABLE BEING       C
C                PROCESSED OCCURS                                            C
C   NFLAG--A FLAG SET TO ONE IF THE PATH BEING CONSIDERED INVOLVES UNCERTAIN  C
C                INITIALIZATION                                              C
C                                                                            C
C   OTHER IMPORTANT VARIABLES                                                C
C   ITH--THE INDEX IN SYMTAB OF SYMBOL BEING SCRUTINIZED                     C
C   K--THE USE CODE OF THE SYMBOL                                            C
C   ITOP--POINTER TO THE VARIABLES FIRST ENTRY IN USETAB                     C
C   JTH--POINTER TO ENTRY IN USETAB BEING  CONSIDERED                        C
C   JCODE--THE USE CODE OF THE ENTRY IN USETAB                               C
C   NNODE--THE NODE AT WHICH  THE USE APPLIES                                C
C   LL--A FLAG SET TO ONE WHEN AN INITIALIZATION USE IF FOUND                C
C   JTYPE--THE CODE FOR THE TYPE OF STATEMENT                                C
C   NC--A FLAG SET TO THE TERMINAL NODE OF THE DO STATEMENT (IF NEEDED)       C
C   JJ--A PREDECESSOR NODE OF THE CURRENT NODE                               C
C   IBACK--THE INDEX IN USETAB OF THE PREVIOUS USE OF THE SYMBOL             C
C   ICODE USE CODE OF THE VARIABLE AT THE CURRENT NODE                       C
C                                                                            C
C   REQUIRED ROUTINES (DESCRIBED ELSEWHERE IN DETAIL)                        C
C     INCLUDED IN THE TRACE PACKAGE ARE THE ROUTINES ZERO, INPUT2, ENTER, AND C
C   NCHECK.  THE ROUTINES HIFECH AND LOFECH ARE BIT MANIPULATIVE ROUTINES     C
```

```
C  AN OUTPUT USAGE OF THE SYMBOL IS FOUND. LOOP THROUGH USETAB FOR EACH INPUT  C
C*************************************************************************C
      JTH=ITOP
   32 JCODE=LOFECH(USETAB(1,JTH))
      IF(JCODE.EQ.2.OR.JCODE.EQ.3.OR.JCODE.EQ.6.OR.JCODE.EQ.7.OR.JCODE.
     2   EQ.8.OR.JCODE.EQ.20.OR.JCODE.EQ.23.OR.JCODE.EQ.24)GOTO 36
      IF(JCODE.GT.30)GOTO 100
      IF(JCODE.EQ.18.OR.JCODE.EQ.19)GOTO 35
   34 JTH=LOFECH(USETAB(2,JTH))
      IF(JTH.EQ.0)GOTO 450
      GOTO 32
C*************************************************************************C
C  INPUT USAGE IS FOUND                                                   C
C*************************************************************************C
   35 NFLAG=HIFECH(USETAB(1,JTH))
      NFLAG=0
   36 NNODE=HIFECH(USETAB(1,JTH))
      CALL ENTER(NNODE,UNSTCK)
C*************************************************************************C
C  ENTER PREDESSORS OF NNODE ENTRY                                        C
C*************************************************************************C
   40 I=NODTAB(3,NNODE)
      JTYPE=HIFECH(NODTAB(1,NNODE))
      IPN=HIFECH(I)
      INUM=LOFECH(I)+IPN-1
      NO=0
C*************************************************************************C
C  IF NODE IS A DO STATEMENT, FIND ITS NODE OF TERMINATION AND OMIT THAT PATH  C
C*************************************************************************C
      IF(JTYPE.NE.53)GOTO 45
      K=LOFECH(NODTAB(1,NNODE))
      JCODE=LOFECH(USETAB(1,K))
      IF(JCODE.EQ.9)K=K+1
   42 K=LOFECH(USETAB(2,K))
      JCODE=LOFECH(USETAB(1,K))
      IF(JCODE.NE.9)GOTO 42
      NO=HIFECH(USETAB(1,K))
   45 DO 50 I=IPN,INUM
      JJ=PRETAB(I)
      IF(JJ.GT.LNOD)GOTO 410
      IF(JJ.EQ.NO)GOTO 50
      L=NCHECK(JJ,UNSTCK)
      IF(L.EQ.1)GOTO 50
      KK=KK+1
      NODSTK(KK)=JJ
   50 CONTINUE
C*************************************************************************C
C  CHECK THE PREDECESSORS                                                 C
C*************************************************************************C
   52 IF(KK.EQ.0)GOTO 34
      NNODE=NODSTK(KK)
      KK=KK-1
      L=NCHECK(NNODE,UNSTCK)
      IF(L.EQ.1)GOTO 52
      CALL ENTER(NNODE,UNSTCK)
C*************************************************************************C
```

```
C  THAT ARE A PART OF THE FACES PACKAGE.
C*****************************************************************************C
      COMMON /STACKS/NODSTK(1000),UNSTCK(30),LIST(30),NFLAG
     2                ,VARIB(200,3),OUT(200,3)
      COMMON SYMTAB (5,1800), LSYM, NADDR, USETAB(2,2500), LUSE, INDUT,
     2   ALTAB(56),LAL,INDAL,DOTAB(3,50),LDO,INDDO,IPRIME,NODTAB(3,1000)
     3  ,LNOD,SUCTAB(1500),ISUC,PRETAB(1500),IPRE,LPS,STALTB(2,100),
     4   LSTAL,INDST
      COMMON /LTAB2/LUNS
      COMMON /GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAR, IPAR, JPRIME
      PRINT 807, MDNAM1,MDNAM2
807   FORMAT(45H1 RESULTS OF INITIALIZATION CHECK FOR MODULE ,2A4/)
      CALL TABLNTH
      CALL INPUT2(MDNAM1,MDNAM2)
      IF(MDNAM1.EQ.0)RETURN
C*****************************************************************************C
C  LOOP THROUGH THE SYMBOL TABLE TO TRACE EACH VARIABLE                       C
C*****************************************************************************C
      DO 500 ITH=1,LSYM
      IF(SYMTAB(1,ITH).EQ.0)GOTO 500
      K=LOFECH(SYMTAB(3,ITH))
      IF(K.NE.3.AND.K.NE.6)GOTO 500
C*****************************************************************************C
C  SYMBOL FOUND IS AN ARRAY OR VARIABLE NAME                                  C
C*****************************************************************************C
      J=HIFECH(SYMTAB(5,ITH))
      IF(J.NE.0)GOTO 500
C*****************************************************************************C
C  SYMBOL IS NOT IN COMMON.  FIND POINTERS TO USETAB                          C
C*****************************************************************************C
      ITOP=HIFECH(SYMTAB(4,ITH))
      JTH=ITOP
      JCODE=LOFECH(USETAB(1,JTH))
      IF(JCODE.EQ.12)GOTO 500
C*****************************************************************************C
C  SYMBOL IS NOT IN A DATA STATEMENT                                          C
C*****************************************************************************C
      CALL ZERO
      LL=0
      KK=0
C*****************************************************************************C
C  SEARCH FOR AN INITIALIZING USAGE AND RECORD IN LIST EACH USAGE             C
C*****************************************************************************C
   20 NNODE=HIFECH(USETAB(1,JTH))
      CALL ENTER(NNODE,LIST)
      IF(LL.EQ.1)GOTO 24
      IF(JCODE.EQ.1.OR.JCODE.EQ.4.OR.JCODE.EQ.5.OR.JCODE.EQ.16.OR.JCODE.
     2  EQ.17.OR.JCODE.EQ.18.OR.JCODE.EQ.19.OR.JCODE.EQ.22)LL=1
   24 JTH=LOFECH(USETAB(2,JTH))
      IF(JTH.EQ.0)GOTO 28
      JCODE=LOFECH(USETAB(1,JTH))
      IF(JCODE.EQ.12)GOTO 500
      GOTO 20
   28 IF(LL.EQ.0)GOTO405
C*****************************************************************************C
```

```
C  DETERMINE IF SYMBOL IS USED AT THIS NODE                              C
C*************************************************************************
      L=NCHECK(NNODE,LIST)
      IF(L.EQ.0)GOTO 40
C*************************************************************************
C  THE VARIABLE DOES APPEAR IN THE STATEMENT                              C
C*************************************************************************
   60 IPNT=LOFECH(NODTAB(1,NNODE))
      ICODE=LOFECH(USETAB(1,IPNT))
      IF(ICODE.EQ.10)IPNT=IPNT+1
C*************************************************************************
C  FIND THE ENTRY IN USETAB OF THE VARIABLE                               C
C*************************************************************************
   64 I=IPNT
   65 IBACK=HMFECH(USETAB(2,I))
      IF(HOFECH(USETAB(2,I)).NE.0) GO TO 70
      I=IBACK
      GOTO 65
C*************************************************************************
C  IS THE POINTER IBACK TO THE RIGHT VARIABLE IN SYMTAB                   C
C*************************************************************************
70    IF(ITH.EQ.IBACK)GOTO 80
      IPNT=IPNT+1
      GOTO 64
C*************************************************************************
C  THE VARIABLE IS FOUND -- DETERMINE ITS TYPE OF USAGE                   C
C*************************************************************************
   80 ICODE=LOFECH(USETAB(1,IPNT))
      IF(ICODE.EQ.1.OR.ICODE.EQ.4.OR.ICODE.EQ.5.OR.ICODE.EQ.16.OR.ICODE.
     2EQ.17.OR.ICODE.EQ.22)GOTO 52
C*************************************************************************
C  THE USE IS NOT A DEFINITE OUTPUT                                       C
C*************************************************************************
      IF(ICODE.NE.18.AND.ICODE.NE.19)GOTO 40
C*************************************************************************
C  FLAG THE NODE AS BEING  A POSSIBLE OUTPUT                              C
C*************************************************************************
      IF(KK.GE.0)GOTO 40
      NFLAG=NNODE
      GO TO 40
  100 NNODE=HIFECH(USETAB(1,JTH))
      JTYPE=HIFECH(NODTAB(1,NNODE))
      IF(JTYPE.NE.4.AND.JTYPE.NE.15)GOTO 36
      LTH=JTH
  110 LTH=LOFECH(USETAB(2,LTH))
      N2=HIFECH(USETAB(1,LTH))
      IF(N2.NE.NNODE)GOTO 36
      LCODE=LOFECH(USETAB(1,LTH))
      IF(LCODE.NE.5)GOTO 110
      GOTO 34
  405 PRINT 796, SYMTAB(1,ITH),SYMTAB(2,ITH)
      GOTO 500
  410 IF(NFLAG.EQ.0)GOTO 420
      PRINT 797, SYMTAB(1,ITH),SYMTAB(2,ITH),NFLAG
      GOTO 500
  420 PRINT 798, SYMTAB(1,ITH),SYMTAB(2,ITH)
```

```
      GOTO 500
  796 FORMAT(17H  THE VARIABLE *  ,2A4,24H * IS NEVER INITIALIZED  )
  797 FORMAT(17H  THE VARIABLE *  ,2A4,54H * IS POSSIBLY REFERENCED WITH
     2OUT BEING INITIALIZED.  ,47H IS IT INITIALIZED BY THE ROUTINE CALL
     3 AT NODE   ,I5)
  798 FORMAT(17H  THE VARIABLE * ,2A4, 48H * MAY BE REFERENCED BEFORE BE
     2ING INITIALIZED.            )
  450 CONTINUE
  500 CONTINUE
      PRINT 808
  808 FORMAT(3H     )
      PRINT 799
  799 FORMAT(54H  INITIALIZATION CHECK HAS BEEN MADE ON EACH VARIABLE  )
      RETURN
      END
*DECK TRISET
      SUBROUTINE TRISET(I,JJ)
      IMPLICIT INTEGER (A-Z)
C*****************************************************************************C
C  TRISET IS A MASTER ROUTINE USED TO REFERENCE THE ROUTINES LOAD AND CODER.   C
C                                                                              C
C  PARAMETERS                                                                  C
C  I--THE SEGMENT OF CBCODE TO BE USED IN CALL TO CODER                        C
C  JJ--INDEX IN CNTAB OF COMMON BLOCK BEING CONSIDERED                         C
C                                                                              C
C  IMPORTANT VARIABLES                                                         C
C  NPNTR--A COMMON VARIABLE USED TO INDEX ENTRIES TO ERROR                     C
C  NTOP--INDEX OF THE COMMON BLOCKS FIRST ENTRY IN STALTB                      C
C  NBOT--INDEX OF THE COMMON BLOCKS LAST ENTRY IN STALTB                       C
C  SEE COMBAL FOR DESCRIPTIONS OF THE COMMON VARIABLES AND THE REQUIRED        C
C  ROUTINES.                                                                   C
C*****************************************************************************C
      COMMON /CODE/CBCODE(50,4,3),ERROR(20),N1,N2,KFLAG,NPNTR
      COMMON /GLOBAL/DIREC(3,100),LDIR,INDIR,CNTAB(4,300),LCN,INDCN,
     C PARTAB (2000), LPAR, IPAR, JPRIME
C*****************************************************************************C
C  ENTER MODULE NUMBER INTO ERROR                                              C
C*****************************************************************************C
      NPNTR=NPNTR+1
      ERROR(NPNTR)=CNTAB(4,JJ)
C*****************************************************************************C
C  LOAD THE LOCAL TABLES OF THAT MODULE                                        C
C*****************************************************************************C
      CALL LOAD(JJ)
C*****************************************************************************C
C  DETERMINE INDICES OF FIRST AND LAST ENTRIES OF THE COMMON BLOCK IN STALTB    C
C*****************************************************************************C
      N=CNTAB(3,JJ)
      NTOP=HIFECH(N)
      NBOT=LOFECH(N)
      CALL CODER(I,NTOP,NBOT)
      RETURN
      END
      SUBROUTINE ZERO
*DECK ZERO
C*****************************************************************************C
```

```
C  THE PURPOSE OF THIS ROUTINE IS TO STORE ZERO IN NFLAG AND IN ALL WORDS OF   C
C  OF THE ARRAYS UNSTCK AND LIST.                                              C
C*******************************************************************************C
      COMMON /STACKS/NODSTK(1000),UNSTCK(30),LIST(30),NFLAG
     2               ,VARIB(200,3),OUT(200,3)
      COMMON /LTAB2/LUNS
      INTEGER UNSTCK
      NFLAG=0
      DO 10  I=1,LUNS
      LIST(I)=0
   10 UNSTCK(I)=0
      RETURN
      END
*DECK HCFECH
      INTEGER FUNCTION HCFECH(IWORD)
      INTEGER FLD
      HCFECH=FLD(0,1,IWORD)
      RETURN
      END
*DECK HMFECH
      INTEGER FUNCTION HMFECH(IWORD)
      INTEGER FLD
      HMFECH=FLD(1,29,IWORD)
      RETURN
      END
*DECK LPFECH
      FUNCTION LPFECH(IWORD)
      INTEGER FLD
      LPFECH=FLD(33,27,IWORD)
      RETURN
      END
*DECK MFFECH
      FUNCTION MFFECH(IWORD)
      INTEGER FLD
      MFFECH=FLD(18,15,IWORD)
      RETURN
      END
*DECK HFFECH
      INTEGER FUNCTION HFFECH(IWORD)
      INTEGER FLD
      HFFECH=FLD(3,15,IWORD)
      RETURN
      END
*DECK LOFECH
      INTEGER FUNCTION LOFECH(IWORD)
      INTEGER FLD
      LOFECH=FLD(30,30,IWORD)
      RETURN
      END
*DECK HIFECH
      INTEGER FUNCTION HIFECH(IWORD)
      INTEGER FLD
      HIFECH=FLD(0,30,IWORD)
      RETURN
      END
*DECK FLD
```

```
      INTEGER FUNCTION FLD(IPOS,N,IWORD)
      DIMENSION MASKAR(59)
      DATA MASKAR/1,3,7,17B,37B,77B,177B,377B,777B,1777B,3777B,7777B,
     1 17777B,37777B,77777B,177777B,377777B,777777B,
     1 1777777B,3777777B,7777777B,17777777B,37777777B,77777777B,
     1 177777777B,377777777B,777777777B,
     1 1777777777B,3777777777B,7777777777B,
     1 17777777777B,37777777777B,77777777777B,177777777777B,
     137777777777B,777777777777B,1777777777777B,3777777777777B,
     17777777777777B,17777777777777B,37777777777777B,77777777777777B,
     117777777777777B,277777777777777B,777777777777777B,
     117777777777777B,377777777777777B,777777777777777B,
     117777777777777B,377777777777777B,777777777777777B,
     117777777777777B,377777777777777B,777777777777777B,
     117777777777777B,377777777777777B,777777777777777B,
     117777777777777777B,3777777777777777B/
      NS=N+IPOS
      IF(N.GE.60.OR.N.LE.0) GO TO 30
      MASK=MASKAR(N)
      FLD=LSHIFT(IWORD,NS).AND.MASK
      RETURN
30    FLD=0
      RETURN
      END
*DECK HISTOR
      INTEGER FUNCTION HISTOR(ITWORD,ISWORD)
      ITWORD=ITWORD.AND.77777777778
      ITEMP=ISWORD.AND.77777777778
      ITWORD=LSHIFT(ISWORD,30).OR.ITWORD
      HISTOR=ITWORD
      RETURN
      END
      INTEGER FUNCTION HTFECH(IWORD)
*DECK HTFECH
      INTEGER FLD
      HTFECH=FLD(0,3,IWORD)
      RETURN
      END
*DECK HTSTOR
      INTEGER FUNCTION HTSTOR(ITWORD,ISWORD)
      IBITS=ISWORD.AND.7B
      IBITS=LSHIFT(IBITS,57)
      ITWORD=ITWORD.AND.7777777777777777B.OR.IBITS
      HTSTOR=ITWORD
      RETURN
      END
```

FORTRAN FRONT END SAMPLE RUN OUTPUT.

```
 1            SUBROUTINE TEST1
 2            DIMENSION KWORDS (10), LWORDS (12)
 3            COMMON /BLOCK1/ ALPHA, BETA, GAMMA, DELTA, IOTA
 4            COMMON /BLOCK2/ IARRAY (20), VAR1, VAR2
 5            IF (IOTA.EQ. 0) GO TO 25
 7            DO 20 K= 1,20
 8         20 IARRAY (K) = 0
 9         25 BETA = VAR1 * VAR2 + FUNCT1 (GAMMA, DELTA)
10            CALL SUB2 (BETA, KWORDS)
11            IF (BETA.GT. 0) GO TO 30
13            ALPHA = VAR1/ZETA
14            CALL SUB2 (ALPHA, KWORDS, I)
15            RETURN
16         30 CALL SUB2 (ALPHA, LWORDS)
17            RETURN
18            END


NUMBER OF ACTIVE ENTRIES IN LOCAL TABLES
SYMTAB =    21     NODTAB =    18     SUCTAB =    25
PRETAB =    20     USETAB =    45     ALTAB  =     5
DOTAB  =     2     STALTB =    10     TRIP   =    18

LOCAL TABLES FOR MODULE  201 HAVE BEEN STORED ON FILE
```

LOCAL TABLES GENERATED FOR MODULE 201

SYMBOL TABLE

THIS TABLE CONTAINS ALL NAMES AND LABELS THAT APPEAR IN EACH MODULE.
NAMES ARE HASH-CODED AND APPEAR WITH FOLLOWING CODES AND POINTERS

TYPE CODES
0--INTEGER
1--FLOATING POINT
2--DOUBLE PRECISION
3--COMPLEX
4--LOGICAL
5--NEUTRAL

CLASS CODES
0--PROGRAM NAME
1--SUBROUTINE NAME
2--STATEMENT FUNCTION NAME
3--ARRAY NAME
4--FUNCTION NAME
5--LABEL
6--VARIABLE
7--COMMON BLOCK NAME

USETAB TOP POINTER--POINTS TO FIRST ENTRY OF NAME IN USAGE TABLE
USETAB BOTTOM POINTER--POINTS TO FIRST ENTRY OF NAME IN USAGE TABLE
STALTB PTR.--POINTS TO LOCATION OF NAME IN STORAGE ALLOCATION TABLE IF NAME IS A COMMON VARIABLE.
OR TO ENTRY IN PARTAB IF SUB. OR FUNCTION NAME
ALTAB PTR.--POINTS TO LOCATION OF NAME IN ARRAY LENGTH TABLE IF NAME IS AN ARRAY.

INDEX	NAME	TYPE	CLASS	USETAB TOP PTR.	USETAB BOT. PTR.	STALTB PTR.	ALTAB PTR.
20	VAR1	1	6	13	35	8	0
21	VAR2	1	6	14	25	9	0
47	ZETA	1	6	36	36	0	0
185	20	5	10008	17	19	0	0
262	K	0	6	18	21	0	0
288	FUNCT1	1	4	26	26	2	0
439	DELTA	1	6	9	28	5	0
492	TEST1	5	1	2	2	0	0
506	GAMMA	1	6	3	27	4	0
575	IARRAY	0	3	12	20	7	4
770	BETA	1	6	7	32	3	0
772	I	0	6	40	40	0	0
1134	LWORDS	0	3	4	44	0	3
1270	ALPHA	1	6	5	43	2	0
1389	KWORDS	0	3	3	39	0	2
1394	BLOCK1	5	7	5	5	0	0
1430	25	5	10009	16	22	0	0
1475	IOTA	0	6	10	15	6	0
1643	BLOCK2	5	7	11	11	0	0
1656	SUB2	5	1	29	42	6	0
1729	30	5	10016	33	41	0	0

USAGE TABLE

THIS TABLE CONTAINS ONE ENTRY FOR EACH APPEARANCE OF A NAME IN A MODULE. EACH ENTRY CONTAINS THE STATEMENT NO. IN WHICH THE NAME APPEARS IN THE SOURCE LISTING AND A USE CODE INDICATING HOW THE NAME IS USED IN THAT STATEMENT. EACH ENTRY IS LINKED TO THE PREVIOUS AND NEXT ENTRIES OF THE SAME NAME BY BACKWARD AND FORWARD POINTERS. THE FIRST ENTRY OF A NAME CONTAINS A POINTER TO THE NAME IN THE SYMBOL TABLE.

BACK POINTER--POINTS TO PREVIOUS ENTRY OF NAME IN USAGE TABLE OR TO NAME IN SYMBOL TABLE
FORWARD POINTER--POINTS TO NEXT ENTRY OF NAME IN USAGE TABLE. FOR LAST ENTRY, FORWARD POINTER IS 0.
BP-CODE--1 WHEN BACK PTR. POINTS TO SYMBOL TABLE
0 WHEN BACK POINTER POINTS TO USAGE TABLE

INDEX	STATEMT NO.	USE CODE	BP-CODE	BACK PTR.	FORWARD PTR.
1	0	44	0	0	0
2	1	0	1	492	0
3	2	13	1	1389	31
4	2	13	1	1134	44
5	3	0	1	1394	0
6	3	11	1	1270	34
7	3	11	1	770	23
8	3	11	1	506	27
9	3	11	1	439	28
10	3	11	1	1475	15
11	4	0	1	1643	0
12	4	11	1	575	20
13	4	11	1	20	24
14	4	11	1	21	25
15	5	20	0	10	0
16	6	10	1	1430	22
17	7	25	1	185	19
18	7	5	1	262	21
19	8	9	0	17	0
20	8	1	0	12	0
21	8	10575	0	18	0
22	9	9	0	16	0
23	9	1	0	7	30
24	9	2	0	13	35
25	9	2	0	14	0
26	9	2	1	288	0
27	9	18	0	8	0
28	9	18	0	9	0
29	10	21	1	1656	37
30	10	19	0	23	32
31	10	19	0	3	39
32	11	20	0	30	0
33	12	10	1	1729	41
34	13	1	0	6	38
35	13	2	0	24	0
36	13	2	1	47	0
37	14	21	0	29	42
38	14	19	0	34	43
39	14	19	0	31	0
40	14	19	1	772	0

41	16	9	0	33	0
42	16	21	0	37	0
43	16	19	0	39	0
44	16	19	0	4	0

ARRAY LENGTH TABLE

THIS TABLE CONTAINS DIMENSIONS FOR ALL ARRAYS THAT APPEAR IN A MODULE. EACH ENTRY IN THIS TABLE IS POINTED TO BY THE ALTAB POINTER OF THE ARRAY NAME IN THE SYMBOL TABLE.

INDEX	DIM. 1	DIM. 2	DIM. 3
1	4	1	1
2	10	1	1
3	12	1	1
4	20	1	1

TRANSITION PAIRS TABLE

50000	1
5	6
5	7
8	7
6	9
8	9
10	11
11	12
11	13
14	15
12	16
16	17
10	10000
14	10000
16	10000
15	20000
17	20000
18	30000

TARGET MODULE FOR PATHS - FUNCT1
FROM NODE 6 THE VARIABLE *GAMMA * CAN AFFECT THE FOLLOWING VARIABLES

VARIABLE	NODE OF USE
FTEMP	11
FUNCT1	12
TEMP	10
Y	6

TARGET MODULE FOR PATHS - FUNCT1
THE VALUE OF *TEMP * AT NODE 10 CAN BE AFFECTED BY THE FOLLOWING

VARIABLE	NODE OF USE
ALPHA	6
ALPHA	8
BETA	6
BETA	8
DELTA	8
GAMMA	6
IARRAY	10
I	10
VAR1	6
VAR1	8
X1	10
X2	10
Y	10

TARGET MODULE FOR PATHS - SUB2
THE VALUE OF *PARAM1 * AT NODE 11 CAN BE AFFECTED BY THE FOLLOWING

VARIABLE	NODE OF USE
ARRAY	8
DELTA	7
FUNCT1	9
FUNCT1	10
GAMMA	7
VAR2	9
X	9
X	10
Y	9
Z1	11
Z2	11

RESULTS OF INITIALIZATION CHECK FOR MODULE FUNCT1

THE VARIABLE * FTEMP * MAY BE REFERENCED BEFORE BEING INITIALIZED.

INITIALIZATION CHECK HAS BEEN MADE ON EACH VARIABLE

```
RESULTS OF INITIALIZATION CHECK FOR MODULE TEST1

THE VARIABLE * ZETA     * IS NEVER INITIALIZED
THE VARIABLE * I        * MAY BE REFERENCED BEFORE BEING INITIALIZED.
THE VARIABLE * LWORDS   * MAY BE REFERENCED BEFORE BEING INITIALIZED.
THE VARIABLE * KWORDS   * MAY BE REFERENCED BEFORE BEING INITIALIZED.

INITIALIZATION CHECK HAS BEEN MADE ON EACH VARIABLE
```

PARAMETER ALIGNMENT MESSAGES

TOO MANY PARAMETERS IN CALL TO SUB2 FROM MODULE 201

ARRAY LWORDS IN CALL TO SUB2 FROM MODULE 201 DIFFERS IN DIMENSIONS FROM THE CORRESPONDING FORMAL PARAMETER.

PARAMETER NO. 2 IN CALL TO FUNCT1 FROM MODULE 202 IS OF WRONG TYPE.

END OF PARAMETER ALIGNMENT CHECK

```
RESULTS OF ALIGNMENT TEST FOR COMMON BLOCK LABELED *BLOCK2  *
  FIRST MODULE LISTED WITHOUT ERROR CONTAINS THE STANDARD COMMON BLOCK

  MODULE NUMBER          ERROR
       201
       202               SIZE
       303
```

DIAGNOSTIC ROUTINES SAMPLE OUTPUT.

```
RESULTS OF ALIGNMENT TEST FOR COMMON BLOCK LABELED *BLOCK1  *
  FIRST MODULE LISTED WITHOUT ERROR CONTAINS THE STANDARD COMMON BLOCK

  MODULE NUMBER          ERROR
       201
       202
       303               TYPE
```

PARAMETER TABLE

THIS TABLE CONTAINS ENTRIES FOR ALL PARAMETERS IN SYSTEM. THE CODE IN EACH WORD DISTINGUISHES THE TYPE OF POINTER STORED IN THAT WORD. THE MOD. NO. IS THE MODULE IN WHICH THE PARAMETER APPEARS

CODES

0--PTR. TO DUMMY PARAMETER IN SYMTAB
1--PTR TO ACTUAL PARAMETER IN SYMTAB
2--PTR. TO PARAMETER LIST FOR NEXT CALL OF SAME ROUTINE
3--PTR. TO SUB. OR FUNCTION NAME IN SYMTAB
4--THIS WORD MARKS END OF CURRENT PARAMETER LIST
5--THIS IS A CONSTANT PARAMETER. POINTER FIELD CONTAINS CONSTANT TYPE

INDEX	CODE	MODULE NO.	ARGUMT NO.	POINTER
1	0	0	0	34
2	3	201	1	288
3	1	201	1	506
4	1	201	2	439
5	4	0	0	0
6	3	201	1	1656
7	1	201	1	770
8	1	201	2	1389
9	2	0	0	10
10	3	201	1	1656
11	1	201	1	1270
12	1	201	2	1389
13	1	201	3	772
14	2	0	0	15
15	3	201	1	1656
16	1	201	1	1270
17	1	201	2	1134
18	4	0	0	0
19	3	202	1	1656
20	0	202	1	67
21	0	202	2	320
22	4	0	0	0
23	3	202	1	288
24	1	202	1	545
25	1	202	2	290
26	2	0	0	27
27	3	202	1	288
28	1	202	1	545
29	5	202	2	0
30	4	0	0	0
31	3	303	1	288
32	0	303	1	1709
33	0	303	2	159
34	4	0	0	0
35	0	0	0	0

COMMON NAME TABLE

THIS TABLE CONTAINS NAMES OF ALL COMMON BLOCKS IN THE SYSTEM. ONE ENTRY IS MADE FOR EACH DECLARATION OF A COMMON BLOCK. POINTERS TO THE TOP AND BOTTOM OF THE LIST OF VARIABLES IN THE COMMON BLOCK ARE STORED WITH EACH ENTRY.

STALTB TOP PTR.--POINTER TO FIRST VARIABLE OF COMMON BLOCK IN STORAGE ALLOCATION TABLE
STALTB BOTTOM PTR.--POINTER TO LAST VARIABLE OF COMMON BLOCK IN STORAGE ALLOC. TABLE
MODULE NO.--NO. OF MODULE IN WHICH COMMON BLOCK DECLARATION APPEARS.

INDEX	BLOCK NAME	STALTB TOP PTR.	STALTB BOT. PTR.	MODULE NO.
1		0	0	0
2	BLOCK1	2	6	201
3	BLOCK2	7	9	201
4	BLOCK1	2	6	202
5	BLOCK2	7	8	202
6	BLOCK1	2	6	303
7	BLOCK2	7	9	303

GLOBAL TABLES

DIRECTORY

THIS TABLE LISTS NAMES OF ALL MODULES IN SYSTEM ALONG WITH AN ASSIGNED MODULE NO. AND CODE.

INDEX	MODULE NAME	FILE	NUMBER
48	TEST1	1	201
85	SUB2	2	202
95	FUNCT1	3	303

GLOBAL TABLE HAS BEEN STORED ON FILE

```
LENGTH OF GLOBAL TABLES
DIREC  =     3     CNTAB  =     7     PARTAB =    35
```

STORAGE ALLOCATION TABLE

THIS TABLE CONTAINS LISTS OF ALL COMMON VARIABLES IN ALL MODULES AND THEIR ASSOCIATED WORD LENGTHS. ONE ENTRY IS MADE EACH TIME A COMMON VAR. IS DECLARED IN A COMMON BLOCK.

CNTAB PTR.--POINTS TO NAME OF COMMON BLOCK IN COMMON NAME TABLE
SYMTAB PTR.--POINTS TO VARIABLE NAME IN SYMBOL TABLE
STARTING LOCATION--INDICATES STORAGE STARTING LOCATION RELATIVE TO CURRENT COMMON BLOCK
NO. OF WORDS--TOTAL DIMENSION OF VARIABLE

INDEX	CNTAB PTR.	SYMTAB PTR.	START LOC.	NO. OF WDS.
1	0	9	0	0
2	6	1270	1	1
3	6	770	2	1
4	6	506	3	1
5	6	439	4	1
6	6	870	5	1
7	7	575	1	20
8	7	20	21	1
9	7	21	22	1

DO LOOP TABLE

ST. NO.	LABEL PTR.	INDEX PTR.	VP-CODE	INIT. VALUE	VP-CODE	TERM. VALUE	VP-CODE	INCREMENT
0	2	0	0	+ 0	0	+ 0	0	+ 0
9	709	772	0	+ 1	0	+ 20	0	+ 1

INDEX	SUCCESSOR TABLE	PREDECESSOR TABLE
1	17	16
2	2	50000
3	3	1
4	4	2
5	5	3
6	6	4
7	8	4
8	7	6
9	9	5
10	9	7
11	10	11
12	11	8
13	9	9
14	12	10
15	13	11
16	20000	12
17	30000	0

NODE TABLE

STATEMT NO.	STATEMT TYPE	USETAB PTR.	SUCCESS. PTR.	SUCCESS. NO.	PRED. PTR.	PRED. NO.
1	27	2	2	1	2	1
2	36	6	3	1	3	1
3	36	12	4	1	4	1
4	10	16	5	2	5	1
5	45	17	7	1	6	1
6	52	18	8	1	7	1
7	45	23	9	1	8	1
8	52	24	10	1	9	1
9	53	30	11	1	10	3
10	52	33	12	1	13	1
11	52	39	13	2	14	1
12	52	43	15	1	15	1
13	51	0	16	1	16	1
14	14	0	17	1	0	0

TRANSITION PAIRS TABLE

50000	1
4	5
4	6
5	8
7	9
11	9
11	12
13	20000
14	30000

ARRAY LENGTH TABLE

THIS TABLE CONTAINS DIMENSIONS FOR ALL ARRAYS THAT APPEAR IN A MODULE. EACH ENTRY IN THIS TABLE IS POINTED TO BY THE ALTAB POINTER OF THE ARRAY NAME IN THE SYMBOL TABLE.

INDEX	DIM. 1	DIM. 2	DIM. 3
1	2	1	1
2	20	1	1

41	11	2	0	40	44
42	11	2	0	33	0
43	12	1	0	3	0
44	12	2	0	41	0

USAGE TABLE

THIS TABLE CONTAINS ONE ENTRY FOR EACH APPEARANCE OF A NAME IN A MODULE. EACH ENTRY CONTAINS THE STATEMENT NO. IN WHICH THE NAME APPEARS IN THE SOURCE LISTING AND A USE CODE INDICATING HOW THE NAME IS USED IN THAT STATEMENT. EACH ENTRY IS LINKED TO THE PREVIOUS AND NEXT ENTRIES OF THE SAME NAME BY BACKWARD AND FORWARD POINTERS. THE FIRST ENTRY OF A NAME CONTAINS A POINTER TO THE NAME IN THE SYMBOL TABLE.

BACK POINTER--POINTS TO PREVIOUS ENTRY OF NAME IN USAGE TABLE OR TO NAME IN SYMBOL TABLE
FORWARD POINTER--POINTS TO NEXT ENTRY OF NAME IN USAGE TABLE. FOR LAST ENTRY, FORWARD POINTER IS 0.
BP-CODE--1 WHEN BACK PTR. POINTS TO SYMBOL TABLE
0 WHEN BACK POINTER POINTS TO USAGE TABLE

INDEX	STATEMT NO.	USE CODE	BP-CODE	BACK PTR.	FORWARD PTR.
1	0	44	0	0	0
2	1	0	1	288	3
3	1	14	0	2	43
4	1	16	1	1709	36
5	1	16	1	159	37
6	2	0	1	1394	0
7	2	11	1	1270	19
8	2	11	1	770	20
9	2	11	1	506	21
10	2	11	1	439	28
11	2	11	1	870	16
12	3	0	1	1643	0
13	3	11	1	575	34
14	3	11	1	20	22
15	3	11	1	21	0
16	4	20	0	11	0
17	5	10	1	1219	24
18	6	1	1	290	25
19	6	2	0	7	26
20	6	2	0	8	27
21	6	2	0	9	0
22	6	2	0	14	29
23	7	10	1	964	30
24	8	9	0	17	0
25	8	1	0	18	38
26	8	2	0	19	0
27	8	2	0	20	0
28	8	2	0	10	0
29	8	2	0	22	0
30	9	9	0	23	0
31	9	25	1	709	39
32	9	5	1	772	35
33	10	1	1	1360	42
34	10	2	0	13	0
35	10	10575	0	32	0
36	10	2	0	4	0
37	10	2	0	5	0
38	10	2	0	25	0
39	11	9	0	31	0
40	11	1	1	1411	41

LOCAL TABLES GENERATED FOR MODULE 303

SYMBOL TABLE

THIS TABLE CONTAINS ALL NAMES AND LABELS THAT APPEAR IN EACH MODULE.
NAMES ARE HASH-CODED AND APPEAR WITH FOLLOWING CODES AND POINTERS

TYPE CODES	CLASS CODES
0--INTEGER	0--PROGRAM NAME
1--FLOATING POINT	1--SUBROUTINE NAME
2--DOUBLE PRECISION	2--STATEMENT FUNCTION NAME
3--COMPLEX	3--ARRAY NAME
4--LOGICAL	4--FUNCTION NAME
5--NEUTRAL	5--LABEL
	6--VARIABLE
	7--COMMON BLOCK NAME

USETAB TOP POINTER--POINTS TO FIRST ENTRY OF NAME IN USAGE TABLE
USETAB BOTTOM POINTER--POINTS TO FIRST ENTRY OF NAME IN USAGE TABLE
STALTB PTR.--POINTS TO LOCATION OF NAME IN STORAGE ALLOCATION TABLE IF NAME IS A COMMON VARIABLE.
OR TO ENTRY IN PARTAB IF SUB. OR FUNCTION NAME
ALTAB PTR.--POINTS TO LOCATION OF NAME IN ARRAY LENGTH TABLE IF NAME IS AN ARRAY.

INDEX	NAME	TYPE	CLASS	USETAB TOP PTR.	USETAB BOT. PTR.	STALTB PTR.	ALTAB PTR.
20	VAR1	1	6	14	29	8	0
21	VAR2	1	6	15	15	9	0
159	X2	1	6	5	37	0	0
288	FUNCT1	1	4	2	43	31	0
290	Y	1	6	18	38	0	0
439	DELTA	1	6	10	28	5	0
505	GAMMA	1	6	9	21	4	0
575	IARRAY	0	3	13	34	7	2
709	70	5	10011	31	39	0	0
770	BETA	1	6	8	27	3	0
772	I	0	6	32	35	0	0
870	EPSILON	1	6	11	16	6	0
964	60	5	10009	23	30	0	0
1219	50	5	10008	17	24	0	0
1270	ALPHA	1	6	7	26	2	0
1360	TEMP	1	6	33	42	0	0
1394	BLOCK1	5	7	6	6	0	0
1411	FTEMP	1	6	40	44	0	0
1643	BLOCK2	5	7	12	12	0	0
1709	X1	1	6	4	36	0	0

```
  1            FUNCTION FUNCT1 (X1, X2)
  2            COMMON /BLOCK1/ ALPHA, BETA, GAMMA, DELTA, EPSILON
  3            COMMON /BLOCK2/ IARRAY(20), VAR1, VAR2
  4            IF (EPSILON .LT. 0) GO TO 50
  6            Y = ALPHA + BETA + GAMMA * VAR1
  7            GO TO 60
  8         50 Y = ALPHA + BETA + DELTA * VAR1
  9         60 DO 70 I = 1,20
 10            TEMP = (IARRAY(I) + X1) /X2 + Y
 11         70 FTEMP = FTEMP + TEMP
 12            FUNCT1 = FTEMP
 13            RETURN
 14            END

NUMBER OF ACTIVE ENTRIES IN LOCAL TABLES
SYMTAB =    20     NODTAB =    14     SUCTAB =    17
PRETAB =    16     USETAB =    45     ALTAB  =     3
DOTAB  =     2     STALTB =    10     TRIP   =     9

LOCAL TABLES FOR MODULE  303 HAVE BEEN STORED ON FILE
```

STORAGE ALLOCATION TABLE /

THIS TABLE CONTAINS LISTS OF ALL COMMON VARIABLES IN ALL MODULES AND THEIR ASSOCIATED WORD LENGTHS. ONE ENTRY IS MADE EACH TIME A COMMON VAR. IS DECLARED IN A COMMON BLOCK.

CNTAB PTR.--POINTS TO NAME OF COMMON BLOCK IN COMMON NAME TABLE
SYMTAB PTR.--POINTS TO VARIABLE NAME IN SYMBOL TABLE
STARTING LOCATION--INDICATES STORAGE STARTING LOCATION RELATIVE TO CURRENT COMMON BLOCK
NO. OF WORDS--TOTAL DIMENSION OF VARIABLE

INDEX	CNTAB PTR.	SYMTAB PTR.	START LOC.	NO. OF WDS.
1	0	8	0	0
2	4	1270	1	1
3	4	770	2	1
4	4	506	3	1
5	4	439	4	1
6	4	1475	5	1
7	5	575	1	20
8	5	20	21	1

DC LOOP TABLE

ST. NO.	LABEL PTR.	INDEX PTR.	VP-CODE	INIT. VALUE	VP-CODE	TERM. VALUE	VP-CODE	INCREMENT
0	2	0	0	+ 0	0	+ 0	0	+ 0
5	1729	772	0	+ 1	0	+ 10	0	+ 1

INDEX	SUCCESSOR TABLE	PREDECESSOR TABLE
1	15	14
2	2	50000
3	3	1
4	4	2
5	5	3
6	6	6
7	5	4
8	7	5
9	8	6
10	9	7
11	10	8
12	11	9
13	12	10
14	20000	11
15	30000	0

NODE TABLE

STATEMT NO.	STATEMT TYPE	USETAB PTR.	SUCCESS. PTR.	SUCCESS. NO.	PRED. PTR.	PRED. NO.
1	30	2	2	1	2	1
2	36	5	3	1	3	1
3	36	11	4	1	4	1
4	28	14	5	1	5	1
5	53	16	6	1	6	2
6	52	18	7	2	8	1
7	52	23	9	1	9	1
8	52	26	10	1	10	1
9	52	28	11	1	11	1
10	52	33	12	1	12	1
11	52	36	13	1	13	1
12	51	0	14	1	14	1
13	14	0	15	1	0	0

TRANSITION PAIRS TABLE

50000	1
6	5
6	7
12	20000
13	30000

ARRAY LENGTH TABLE

THIS TABLE CONTAINS DIMENSIONS FOR ALL ARRAYS THAT APPEAR IN A MODULE. EACH ENTRY IN THIS TABLE IS POINTED TO BY THE ALTAB POINTER OF THE ARRAY NAME IN THE SYMBOL TABLE.

INDEX	DIM. 1	DIM. 2	DIM. 3
1	4	1	1
2	20	1	1
3	10	1	1
4	2	3	1

USAGE TABLE

THIS TABLE CONTAINS ONE ENTRY FOR EACH APPEARANCE OF A NAME IN A MODULE. EACH ENTRY CONTAINS THE STATEMENT NO. IN WHICH THE NAME APPEARS IN THE SOURCE LISTING AND A USE CODE INDICATING HOW THE NAME IS USED IN THAT STATEMENT. EACH ENTRY IS LINKED TO THE PREVIOUS AND NEXT ENTRIES OF THE SAME NAME BY BACKWARD AND FORWARD POINTERS. THE FIRST ENTRY OF A NAME CONTAINS A POINTER TO THE NAME IN THE SYMBOL TABLE.

BACK POINTER--POINTS TO PREVIOUS ENTRY OF NAME IN USAGE TABLE OR TO NAME IN SYMBOL TABLE
FORWARD POINTER--POINTS TO NEXT ENTRY OF NAME IN USAGE TABLE. FOR LAST ENTRY, FORWARD POINTER IS 0.
BP-CODE--1 WHEN BACK PTR. POINTS TO SYMBOL TABLE
0 WHEN BACK POINTER POINTS TO USAGE TABLE

INDEX	STATEMT NO.	USE CODE	BP-CODE	BACK PTR.	FORWARD PTR.
1	0	38	0	0	0
2	1	0	1	1656	0
3	1	17	1	67	21
4	1	17	1	320	14
5	2	0	1	1394	0
6	2	11	1	1270	0
7	2	11	1	770	0
8	2	11	1	506	24
9	2	11	1	439	25
10	2	11	1	1475	0
11	3	0	1	1643	0
12	3	11	1	575	0
13	3	11	1	20	0
14	4	13	0	4	19
15	4	13	1	303	27
16	5	25	1	1729	18
17	5	5	1	772	20
18	6	9	0	16	0
19	6	1	0	14	0
20	6	10320	0	17	22
21	6	2	0	3	36
22	6	2	0	20	0
23	7	1	1	545	30
24	7	2	0	8	0
25	7	2	0	9	0
26	8	1	1	290	31
27	8	2	0	15	0
28	9	1	1	1199	37
29	9	2	1	288	34
30	9	18	0	23	35
31	9	18	0	26	0
32	9	2	1	21	0
33	10	1	1	1448	38
34	10	2	0	29	0
35	10	18	0	30	0
36	11	1	0	21	0
37	11	2	0	28	0
38	11	2	0	33	0

LOCAL TABLES GENERATED FOR MODULE 202

SYMBOL TABLE

THIS TABLE CONTAINS ALL NAMES AND LABELS THAT APPEAR IN EACH MODULE.
NAMES ARE HASH-CODED AND APPEAR WITH FOLLOWING CODES AND POINTERS

TYPE CODES
0--INTEGER
1--FLOATING POINT
2--DOUBLE PRECISION
3--COMPLEX
4--LOGICAL
5--NEUTRAL

CLASS CODES
0--PROGRAM NAME
1--SUBROUTINE NAME
2--STATEMENT FUNCTION NAME
3--ARRAY NAME
4--FUNCTION NAME
5--LABEL
6--VARIABLE
7--COMMON BLOCK NAME

USETAB TOP POINTER--POINTS TO FIRST ENTRY OF NAME IN USAGE TABLE
USETAB BOTTOM POINTER--POINTS TO FIRST ENTRY OF NAME IN USAGE TABLE
STALTB PTR.--POINTS TO LOCATION OF NAME IN STORAGE ALLOCATION TABLE IF NAME IS A COMMON VARIABLE.
OR TO ENTRY IN PARTAB IF SUB. OR FUNCTION NAME
ALTAB PTR.--POINTS TO LOCATION OF NAME IN ARRAY LENGTH TABLE IF NAME IS AN ARRAY.

INDEX	NAME	TYPE	CLASS	USETAB TOP PTR.	USETAB BOT. PTR.	STALTB PTR.	ALTAB PTR.
20	VAR1	1	6	13	13	8	0
21	VAR2	1	6	32	32	0	0
67	PARAM1	1	6	3	36	0	0
288	FUNCT1	1	4	29	34	23	0
290	Y	1	6	26	31	0	0
303	ARRAY	1	3	15	27	0	4
320	JARRAY	0	3	4	19	0	3
439	DELTA	1	6	9	25	5	0
506	GAMMA	1	6	8	24	4	0
545	X	1	6	23	35	0	0
575	IARRAY	0	3	12	12	7	2
770	BETA	1	6	7	7	3	0
772	I	0	6	17	22	0	0
1199	Z1	1	6	28	37	0	0
1270	ALPHA	1	6	6	6	2	0
1394	BLOCK1	5	7	5	5	0	0
1448	Z2	1	6	33	38	0	0
1475	IOTA	0	6	10	10	6	0
1643	BLOCK2	5	7	11	11	0	0
1656	SUB2	5	1	2	2	19	0
1729	30	5	10006	16	18	0	0

```
  1            SUBROUTINE SUB2 (PARAM1, JARRAY)
  2            COMMON /BLOCK1/ ALPHA, BETA, GAMMA, DELTA, IOTA
  3            COMMON /BLOCK2/ IARRAY(20), VAR1
  4            DIMENSION JARRAY (10), ARRAY (2,3)
  5            DO 30 I= 1,10
  6         30 JARRAY (I) = PARAM1 *I
  7            X = GAMMA * DELTA
  8            Y = ARRAY (1,1)
  9            Z1 = FUNCT1 (X,Y) + VAR2
 10            Z2 = FUNCT1 (X,0)
 11            PARAM1 = Z1 *Z2
 12            RETURN
 13            END

NUMBER OF ACTIVE ENTRIES IN LOCAL TABLES
SYMTAB =    21     NODTAB =    13     SUCTAB =    15
PRETAB =    14     USETAB =    39     ALTAB  =     5
OCTAB  =     2     STALTB =     9     TRIP   =     5

LOCAL TABLES FOR MODULE  202 HAVE BEEN STORED ON FILE
```

STORAGE ALLOCATION TABLE

THIS TABLE CONTAINS LISTS OF ALL COMMON VARIABLES IN ALL MODULES AND THEIR ASSOCIATED WORD LENGTHS. ONE ENTRY IS MADE EACH TIME A COMMON VAR. IS DECLARED IN A COMMON BLOCK.

CNTAB PTR.--POINTS TO NAME OF COMMON BLOCK IN COMMON NAME TABLE
SYMTAB PTR.--POINTS TO VARIABLE NAME IN SYMBOL TABLE
STARTING LOCATION--INDICATES STORAGE STARTING LOCATION RELATIVE TO CURRENT COMMON BLOCK
NO. OF WORDS--TOTAL DIMENSION OF VARIABLE

INDEX	CNTAB PTR.	SYMTAB PTR.	START LOC.	NO. OF WDS.
1	0	9	0	0
2	2	1270	1	1
3	2	770	2	1
4	2	506	3	1
5	2	439	4	1
6	2	1475	5	1
7	3	575	1	20
8	3	20	21	1
9	3	21	22	1

DO LOOP TABLE

ST. NO.	LABEL PTR.	INDEX PTR.	VP-CODE	INIT. VALUE	VP-CODE	TERM. VALUE	VP-CODE	INCREMENT
0	2	0	0	+ 0	0	+ 0	0	+ 0
7	185	262	0	+ 1	0	+ 20	0	+ 1

INDEX	SUCCESSOR TABLE	PREDECESSOR TABLE
1	25	20
2	2	50000
3	3	1
4	4	2
5	5	3
6	6	4
7	7	5
8	9	5
9	8	8
10	7	7
11	9	6
12	10	8
13	10000	9
14	11	10
15	12	11
16	13	11
17	16	13
18	14	14
19	10000	12
20	15	16
21	20000	0
22	10000	0
23	17	0
24	20000	0
25	30000	0

NODE TABLE

STATEMT NO.	STATEMT TYPE	USETAB PTR.	SUCCESS. PTR.	SUCCESS. NO.	PRED. PTR.	PRED. NO.
1	30	2	2	1	2	1
2	28	3	3	1	3	1
3	36	5	4	1	4	1
4	36	11	5	1	5	1
5	10	15	6	2	6	1
6	45	16	8	1	7	1
7	53	17	9	1	8	2
8	52	19	10	2	10	1
9	52	22	12	1	11	2
10	34	29	13	2	13	1
11	10	32	15	2	14	1
12	45	33	17	1	15	1
13	52	34	18	1	16	1
14	34	37	19	2	17	1
15	51	0	21	1	18	1
16	34	41	22	2	19	1
17	51	0	24	1	20	1
18	14	0	25	1	0	0