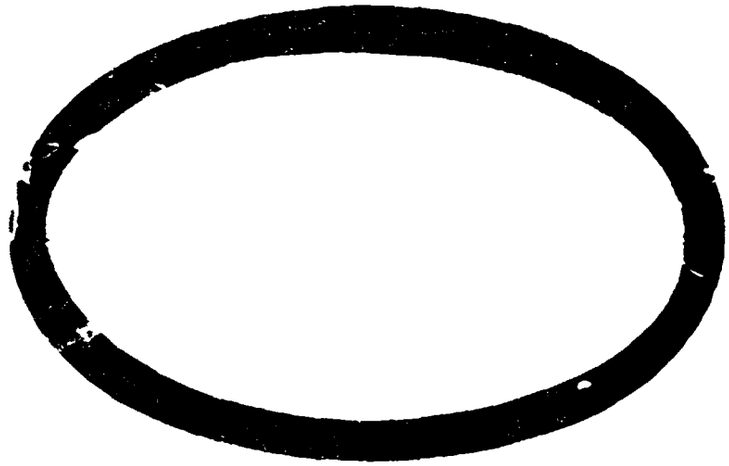
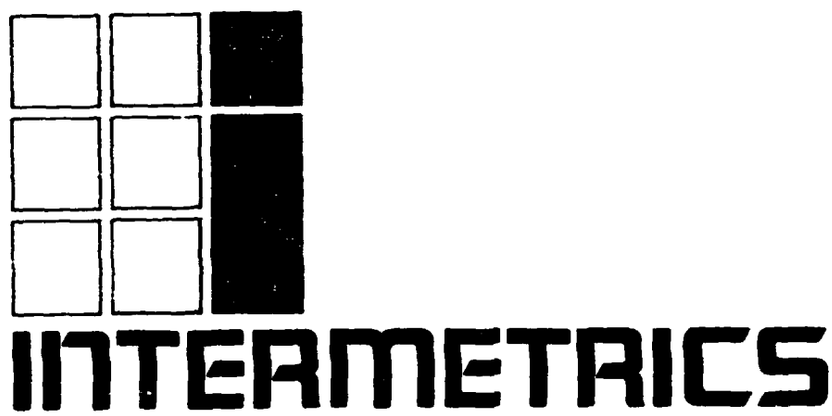


(NASA-CR-138159) THE GOAL-TO-HAL/S TRANSLATOR SPECIFICATION Final Report (Intermetrics, Inc.) 172 p HC $11.75 CSCL 09B G3/08

N74-21838

Unclas 37567



THE GOAL-TO-HAL/S TRANSLATOR SPECIFICATION

Item 14, Contract NAS 10-8385

December 15, 1973

INTERMETRICS INCORPORATED • 701 CONCORD AVENUE • CAMBRIDGE, MASSACHUSETTS 02138 • (617) 661-1840

STANDARD TITLE PAGE

1. Report No.	2. Government Access.on No.	3. Recipient's Catalog No.

4. Title and Subtitle	5. Report Date
THE GOAL-TO-HAL/S TRANSLATOR SPECIFICATION	December 15, 1973
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Saul F. Stanten and James H. Flanders	
9. Performing Organization Name and Address	10. Work Unit No.
INTERMETRICS, INC. 701 Concord Avenue Cambridge, Mass. 02138	11. Contract or Grant No. NAS 10-8385
	13. Type of Report and Period Covered Final Report DRL Item 14
12. Sponsoring Agency Name and Address National Aeronautics and Space Admin. John F. Kennedy Space Center Kennedy Space Center, Florida 32899	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This report comprises a Specification for a GOAL Language to HAL Language Translator. Ground Operations Aerospace Language, or GOAL, is a test-oriented higher order language developed by NASA's John F. Kennedy Space Center for use in the checkout and launch of the Space Shuttle. HAL is a structured higher order language developed by NASA's Johnson Space Center for manned space flight. HAL/S is the version of HAL selected by NASA for writing the flight software for the onboard Data Processing System. Since the onboard computers will extensively support ground checkout of the Space Shuttle and their operational system works exclusively with HAL/S, the translation of GOAL-to-HAL/S assumes significance.

The Specification sets forth a technical framework within which to deal with the transfer of specific GOAL features to HAL/S. Key technical features of the Translator are described which communicate with the data bank, handle repeat statements, and deal with software interrupts. GOAL programs, databank information, and GOAL system subroutines are integrated into one GOAL_MASTER_PROGRAM in HAL/S. This output is fully compatible HAL/S source code ready for insertion into the HAL/S compiler.

The Translator uses a PASS1 to establish all the "global" data needed for the HAL/S output program. Individual GOAL statements are translated in PASS2. The Specification document makes extensive use of flowcharts to specify exactly how each variation of each GOAL statement is to be translated. The Specification also deals with Definitions and Assumptions, Executive Support Structure and Implementation. An Appendix, entitled GOAL-to-HAL Mapping, provides examples of translated GOAL statements. This Translation Specification is based upon an earlier Final Report on GOAL-to-HAL Translation Study, NAS 9-12291, 6c.

17. Key Words	18. Distribution Statement
Test Oriented Language Space Shuttle GOAL HAL	

19. Security Classif.(of this report)	20. Security Classif.(of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified		

NOTICE

This report was prepared as an account of Government-sponsored work. Neither the United States, nor the National Aeronautics and Space Administration (NASA), nor any person acting on behalf of NASA:

(1) Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately-owned rights; or

(2) Assumes any liabilities with respect to the use of, or for damages resulting from the use of, any information, apparatus, method or process disclosed in this report.

As used above, "person acting on behalf of NASA" includes any employee or contractor of NASA, or employee of such contractor, to the extent that such employee or contractor of NASA or employee of such contractor prepares, disseminates, or provides access to any information pursuant to his employment or contract with NASA, or his employment with such contractor.

THIS PAGE INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

1.0 SCOPE

1.1 Introduction

GOAL is a higher order language developed at the NASA Kennedy Space Center to fulfill a need for a standard ground test language in manned space flight. This language will be used by NASA and contractor personnel to write software for ground maintenance, checkout, and launch of the Space Shuttle.

HAL/S is a higher order language which was developed by the NASA Johnson Space Center for the flight software which is to be resident in the Space Shuttle's onboard computers.

A need exists for a capability for running test programs written in GOAL in the onboard computers during ground maintenance and checkout for the Space Shuttle. These onboard computers will have a Flight Computer Operating System (FCOS) which is uniquely designed to be driven by application software written in HAL/S. To provide a different FCOS, which can be driven by GOAL, is not practicable (Ref. 1). Thus, the requirement for a GOAL to HAL/S translation capability has arisen. The feasibility and approach for such a translation was studied in an earlier document (Ref. 2). The favorable results of that study formed the foundation and justification for this document, The GOAL-to-HAL Translator Specification.

1.2 Genesis of GOAL

The development of GOAL was brought about by the need for a standard test language to be used for maintenance, refurbishment, checkout, and launch of the Space Shuttle. Apollo experience had already proven the value of computer-automated checkout programs, while at the same time highlighting the importance of early source language capabilities. ATOLL was such a language and was applied to Saturn V checkout and launch.

As the requirements of the Shuttle program unfolded, it was evident that a high degree of checkout computer automation would be required to meet schedule and cost objectives. Furthermore, the opportunity existed to develop a high order language early in the program so that, from the beginning, it was an integral part of the system. Requirements contracts were let by KSC in July of 1970. In May 1971, a language requirements document was published (KSC-TR-111). Currently, three documents have been published which define the language. These are the GOAL Overview Document, the Syntax Diagrams Handbook (KSC-TR-1213), and the GOAL Textbook (KSC-TR-1228). Also, a GOAL compiler is currently being developed.

1.3 Genesis of HAL/S

The development of HAL was stimulated by the same combination of Apollo experience and anticipated Shuttle requirements that stimulated GOAL, except that HAL is oriented towards the onboard mission software for manned spaceflight with its great emphasis on 1) the mathematical requirements of navigation, guidance, and control and 2) the need for highly reliable real time control programs. Apollo experience had shown that the resources needed to program mission software in assembly language in a multi-program environment were excessive.

Development of HAL began with a contract let by JSC early in 1970. This contract supported the generation of requirements, a survey of other languages, synthesis of a new language, and the building of a HAL compiler to run on the IBM 360/75 at the JSC Real Time Control Center. This effort was augmented a year and one-half later by a JSC contract to advance HAL to an operational status. As a result of this last contract, the HAL language was ready when the decision to specify the onboard software for Shuttle came up, and HAL/S, the Shuttle version, was chosen as the language in which the flight software will be written.

1.4 Purpose of the Specification Document

This document is intended to:

1) treat the GOAL to HAL/S translation process as a complete software system,

2) to define completely and definitively all aspects of the translation process where the facts are known (the GOAL Specification, the HAL/S Specification, etc.), and

3) to identify, segregate, and define in concept and scope all remaining aspects of the translation process where the facts are incomplete, in transition, or not known (the Databank, FCOS, etc.).

1.5 Scope of the Specification Document

It is the objective of this document that it will form the basis for an implementation of the Translator once a host computing system and a Translator language have been designated. Accordingly, it is necessary that this Specification go beyond the mere mapping of GOAL statements into HAL/S statements and arrive at a technical structure which can encompass all the powerful GOAL features of databank resource and control, concurrent program execution, and software interrupts and many other features.

1.6 Outline of the Specification Document

The main body of the Specification begins by presenting the technical approach in the critical areas of communication with the databank, software interrupts, repeats, etc. (See Section 3.0 DISCUSSION OF KEY TECHNICAL FEATURES). There then follows a section devoted to 1) spelling out assumptions that have been made for the translation process and 2) setting forth definitions and conventions applicable to the translator itself (See Section 4.0 ASSUMPTIONS AND DEFINITIONS). The heart of the Specification is presented next (Section 5.0 TRANSLATION REQUIREMENTS). This begins with a presentation of the overall translator structure, followed by definition of various translator subroutines which have been found to be useful. The translation process is then defined in terms of a first or global look at the GOAL source (PASS 1), followed by a statement by statement translation (PASS 2) of specific GOAL source statements. In keeping with the specific technical approach adopted in this Specification, it was decided that flow charts would be the best means for presenting the processing of individual GOAL statements by the translator. Most statements involve alternatives and a textual treatment was deemed very inadequate compared with the flow chart approach.

The interface to the FCOS that results from the proposed technical approach to translation is discussed next (Section 6.0 EXECUTIVE SUPPORT STRUCTURE). The last section deals with reliable implementation of the Translator (Section 7.0 IMPLEMENTATION, VERIFICATION, AND DOCUMENTATION REQUIREMENTS). Appendix A presents the GOAL to HAL MAPPING material which illustrates the translation process without getting into the intricate details of implementation.

THIS PAGE INTENTIONALLY LEFT BLANK.

2.0 APPLICABLE DOCUMENTS

2.1 GOAL Documents

a) Ground Operations Aerospace Language (GOAL) Syntax Diagrams Handbook, TR-1213, 16 April 1973, NASA John F. Kennedy Space Center.

b) Ground Operations Aerospace Language (GOAL) Textbook, TR-1228, 16 April 1973, NASA John F. Kennedy Space Center.

2.2 HAL/S Documents

a) HAL/S Language Specification, 15 September 1973, Intermetrics, Inc.

b) HAL/S Language Forms, Rev. 1, 8 May 1973, Intermetrics, Inc.

c) HAL/S-360 Compiler System Functional Specification, 13 July 1973, Intermetrics, Inc.

THIS PAGE INTENTIONALLY LEFT BLANK.

3.0 DISCUSSION OF KEY TECHNICAL FEATURES

3.1 The GOAL Master Program (GMP) Concept

There are several translation situations which can be handled together by a single unified strategy - the concept of a "GOAL Master Program" written in HAL/S, produced by the translator program, and responsible for coordinating all the HAL/S blocks produced by the translation process. The situations which lead to use of this strategy are several:

a) GOAL has a "TERMINATE SYSTEM" statement which is supposed to cause "complete GOAL application program system shutdown". Thus, a means must be provided to make its effects global to all translated GOAL modules at execution time.

b) GOAL allows a "CONCURRENTLY PERFORM PROGRAM" statement with no restrictions on the number of such statements referencing a single PROGRAM and how many such concurrent uses exist simultaneously off a single program module. The translator shall force the situation whereby a program can only be concurrently performed by one process at a time. i.e., if A and B are concurrent programs, then A and B cannot concurrently perform program C. The first statement to gain access will win the race. The second statement will have to wait for the first to finish.

c) The set of translated GOAL programs have their own internal system of software interrupts and other signals, which, for reliability, should be kept self-contained to prevent unwanted interaction with the other HAL/S applications software.

d) Data Bank information needs to be inserted into the translated program. This is contained within the GOAL Master Program.

e) GOAL System Subroutines can be employed by all GOAL translated programs. These are declared as HAL/S Procedures at the GMP level.

f) The software interrupt mechanism requires processing external to a GOAL program.

In order to treat these situations properly, a GOAL Master Program shall be provided, with the following characteristics: Figures 3-1 and 3-2 indicate the structure of the GMP.

a) When the GOAL system is to be initiated in the flight computer, it is the GOAL_MASTER_PROGRAM* which is scheduled and executed.

b) Each GOAL Program which is to be part of the running GOAL system will become a single HAL/S Procedure within the GOAL_MASTER_PROGRAM, nested at the program level. If the original GOAL Program is subject to multiple concurrent perform statements, then real time attributes of EXCLUSIVE will be applied in order to assure no conflicts.

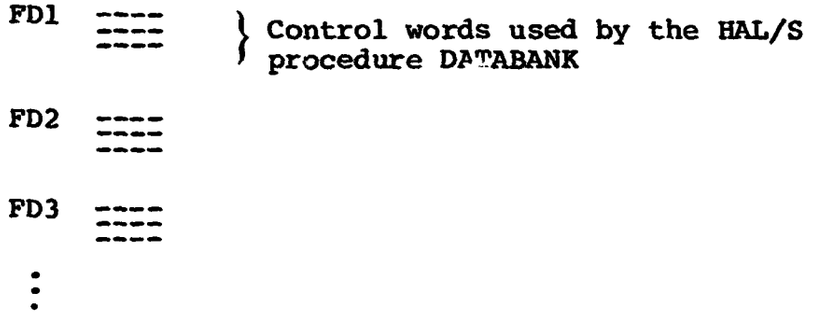
c) Each concurrent statement will generate a uniquely named HAL/S task block nested at the program level. The executable action CONCURRENTLY will consist of a SCHEDULE for the task so generated.

d) At the GOAL_MASTER_PROGRAM level, a Boolean array INT is maintained. This array carries the set of software interrupts available to all the GOAL system's blocks. The array INTNUM, used for interrupt communication, is also maintained.

e) The Boolean array TERMSYS indicates which Goal processes have executed a TERMINATE SYSTEM command.

f) The array of data bank buffers NUMBER, DIMENSION, STATE, TEXT, CONTROL are declared at the GMP level.

g) The GMP will contain all function designator information organized so that the HAL/S procedure DATABANK can access the appropriate data by using the function designator number FD as an index. For example,



* Written here in caps and with underscores to conform to HAL/S requirements.

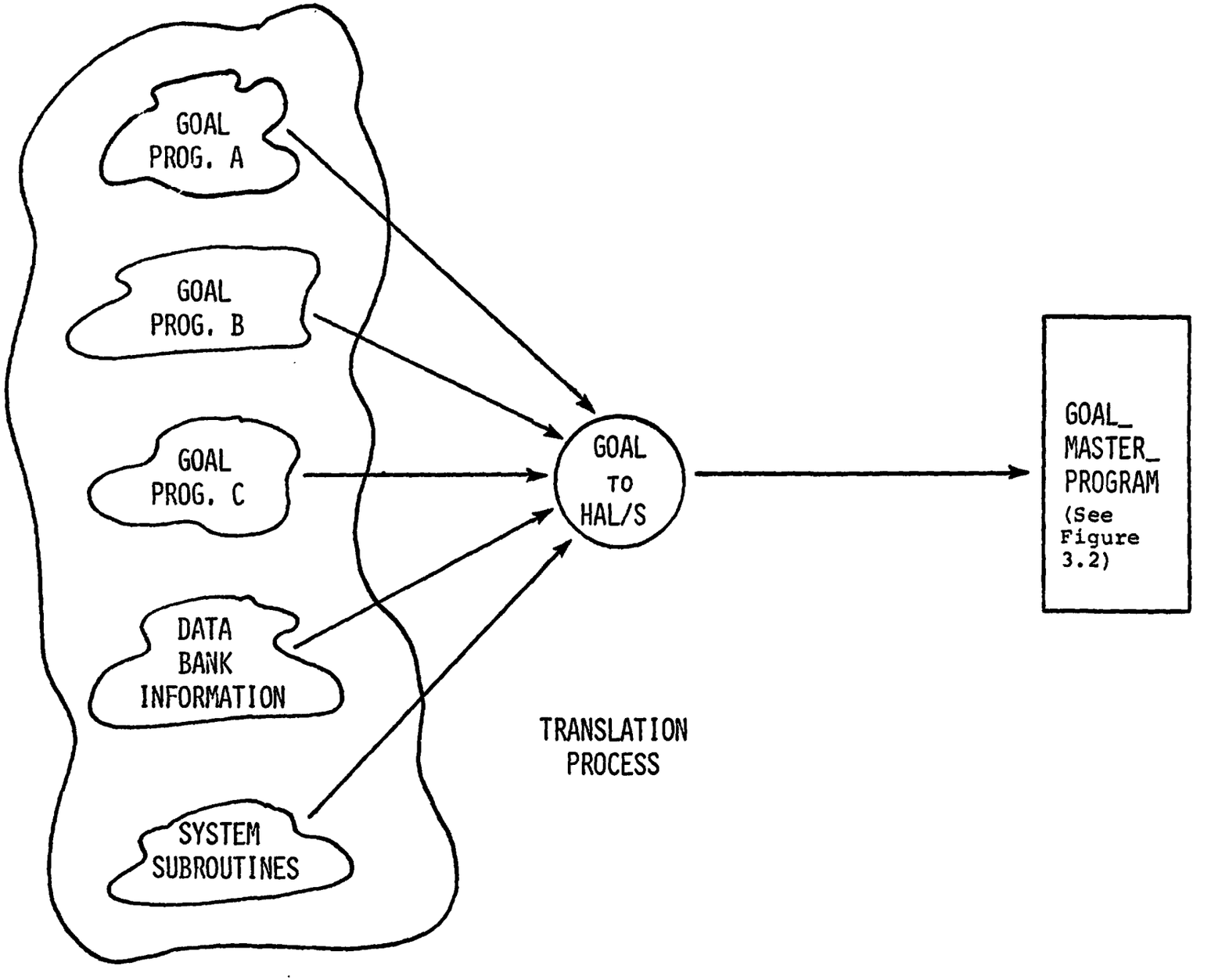

GOAL
PROG. A
GOAL
PROG. B
GOAL
PROG. C
DATA
BANK
INFORMATION
SYSTEM
SUBROUTINES
GOAL
TO
HAL/S
GOAL_
MASTER_
PROGRAM
(See
Figure
3.2)
TRANSLATION
PROCESS
GOAL INPUT
HAL/S OUTPUT

Figure 3-1

Notes for Figure 3-2:

PR, PN_{MAX} and P_{MAX} are defined in Figure 5-1, p. 36.

INT and INTNUM are arrays utilized in the processing of software interrupts (See Section 3.5, p. 21).

TERMSYS is used to implement the TERMINATE SYSTEM Statement. (See Flowchart 78, p. 94).

These are the 5 DATABANK Buffer Arrays discussed in Section 3.3-6, p.16.

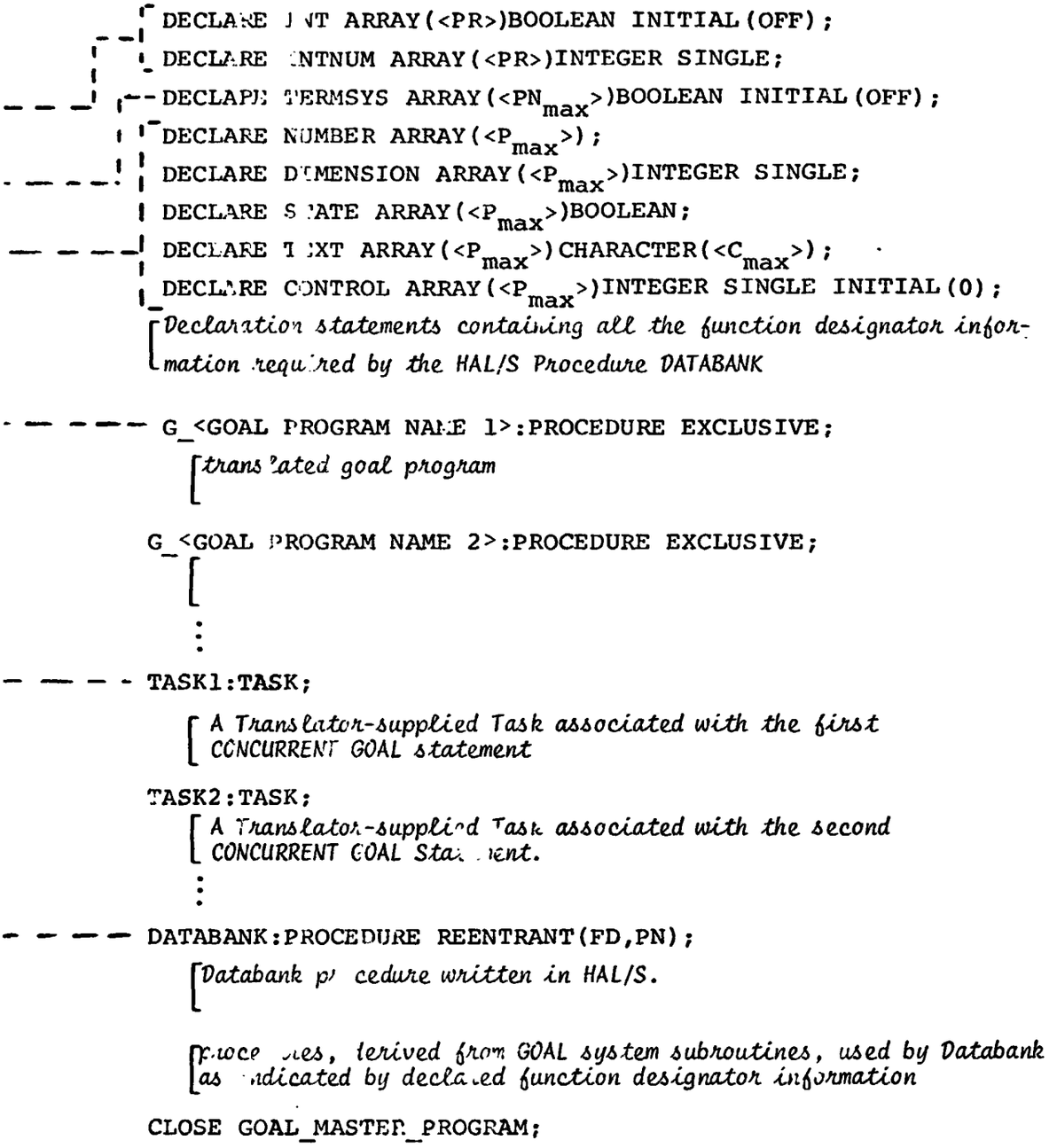
<GOAL PROGRAM NAME> is the name of the first Goal Program in the group being translated.

The implementation of concurrent processes within a HAL/S program is accomplished through the use of HAL/S TASK blocks. Therefore, the Translator will generate an appropriate HAL/S TASK for every CONCURRENTLY PERFORM, VERIFY, or RECORD statement which appears in a GOAL program. See Section 5.1.2, Item 3, and Concurrent Statement Flowchart (15).

FD and PN are the function designator number and the process number received as parameters by the procedure DATABANK. The attribute of REENTRANT allows more than one process to execute "simultaneously" the procedure DATABANK.

701 CONCORD AVENUE • CAMBRIDGE, MASSACHUSETTS 02138 • (617) 661-1840

Figure 3-2: Structure of the GOAL_MASTER_PROGRAM (GMP)



h) The GMP contains the HAL/S procedure DATABANK as well as any other procedures required to service function designator calls.

i) All Goal subroutines become HAL/S procedures nested with the translated Goal Program.

3.2 Structure of a Translated Goal Program

Each Goal program or subroutine is translated into a HAL/S procedure with the structure as indicated in Figure 3-3. The following features should be noted:

a) Declaration of interrupt related variables ENVIRON, ACTIVE.

b) Declaration of variables associated with processing Repeat Instructions; HEAD, TAIL, RPTCTR, RPTACT, SAVE, RPT, RS.

c) A place is reserved for all the declarations created by the Translator such as the loop indices I, J, K, ...

d) The functions FLUSH, RPTCONT, and HANDLER appear in each program or subroutine contingent, of course, on whether or not WHEN INTERRUPT or REPEAT statements are present.

e) The DOCASE label and RETURN_LABEL_CASE label contain the GO TO statements used in returning from interrupts and repeat statements respectively.

THIS PAGE INTENTIONALLY LEFT BLANK.

Notes for Figure 3-3:

A translated GOAL subroutine from a translated GOAL Program only in the first statement. The first statement for a subroutine should read:

G_<Goal NAME>: Procedure (C) Assign (<Parameters>);

where C is the critical bit.

ACTIVE and ENVIRON are associated with software interrupt handling at the program and subroutine levels (See Section 3.5-3, pp. 23-25).

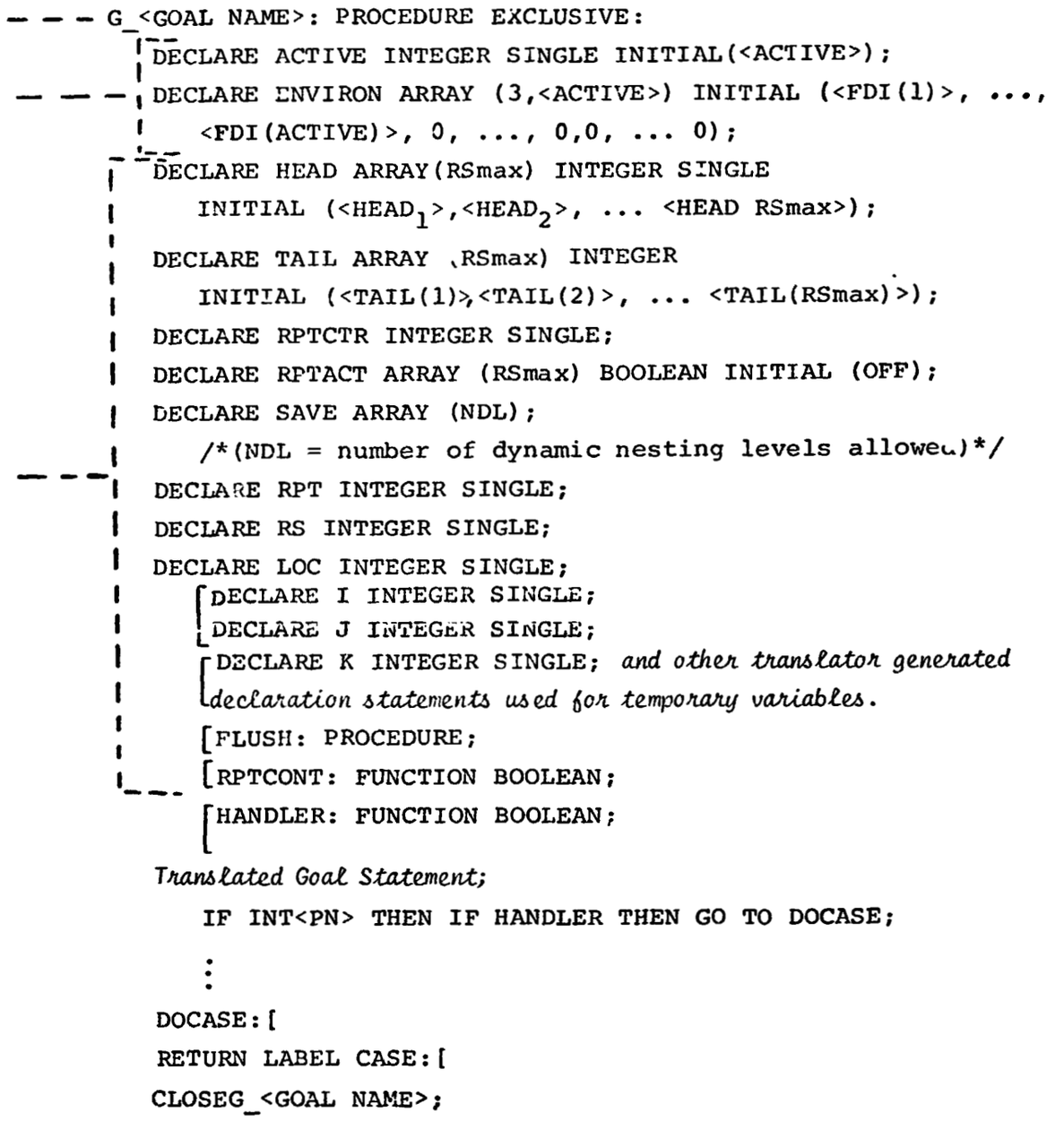
ACTIVE is the total number of interrupts specified within a GOAL program or subroutine. It is determined in Pass 1 and is a static value.

<FDI(K)> is the interrupt function designator number associated with the K^{th} interrupt.

These statements or functions are eliminated if no WHEN INTERRUPT statements appear in the program or subroutine.

The GOAL REPEAT Statements require these declaration statements. Section 3.4 describes the purpose of each of these variables and how they are used. The Procedure FLUSH is used to re-initialize nested REPEAT groups if a GO TO statement is encountered which causes execution outside of an activated REPEAT group. The declaration statements, procedures, and functions associated with REPEAT statements are encountered in translating the program.

Figure 3-3: Structure of a Translated Goal Program



3.3 Communication with the Databank

In order to provide the features of function designators, as described in the Goal Text, the following design features have been incorporated into the Translator. Figure 3-4 indicates the various major features.

1) All function designator names are resolved into unique numbers during Pass 1.

2) Function Designator number 0 is reserved for the "Stop and Restart" display.

3) Function Designator 1 is reserved for the interrupt service routines.

4) Function Designator 2 is reserved for the system device referred to in the RECORD and OUTPUT EXCEPTION statements.

5) Any function designator can be reached by the HAL/S statement:

```
CALL DATABANK (function designator number,
               process number);
```

DATABANK is a procedure at the GOAL_MASTER_PROGRAM level which is unspecified at this time with respect to internal functioning. It performs whatever function is specified by the function designator.

A process is defined, for the purposes of this Translator, as any Goal translated program or any task created by the translator as a result of a Goal Concurrent statement.

6) Communication with DATABANK occurs through 5 DATABANK buffer arrays declared at the GOAL_MASTER_PROGRAM level:

NUMBER	scalar
DIMENSION	integer
STATE	boolean
TEXT	character string
CONTROL	integer

These arrays act as a communication buffer for data transfer between Goal processes and the DATABANK procedure. The length of these arrays equals the maximum number of processes. These arrays are indexed

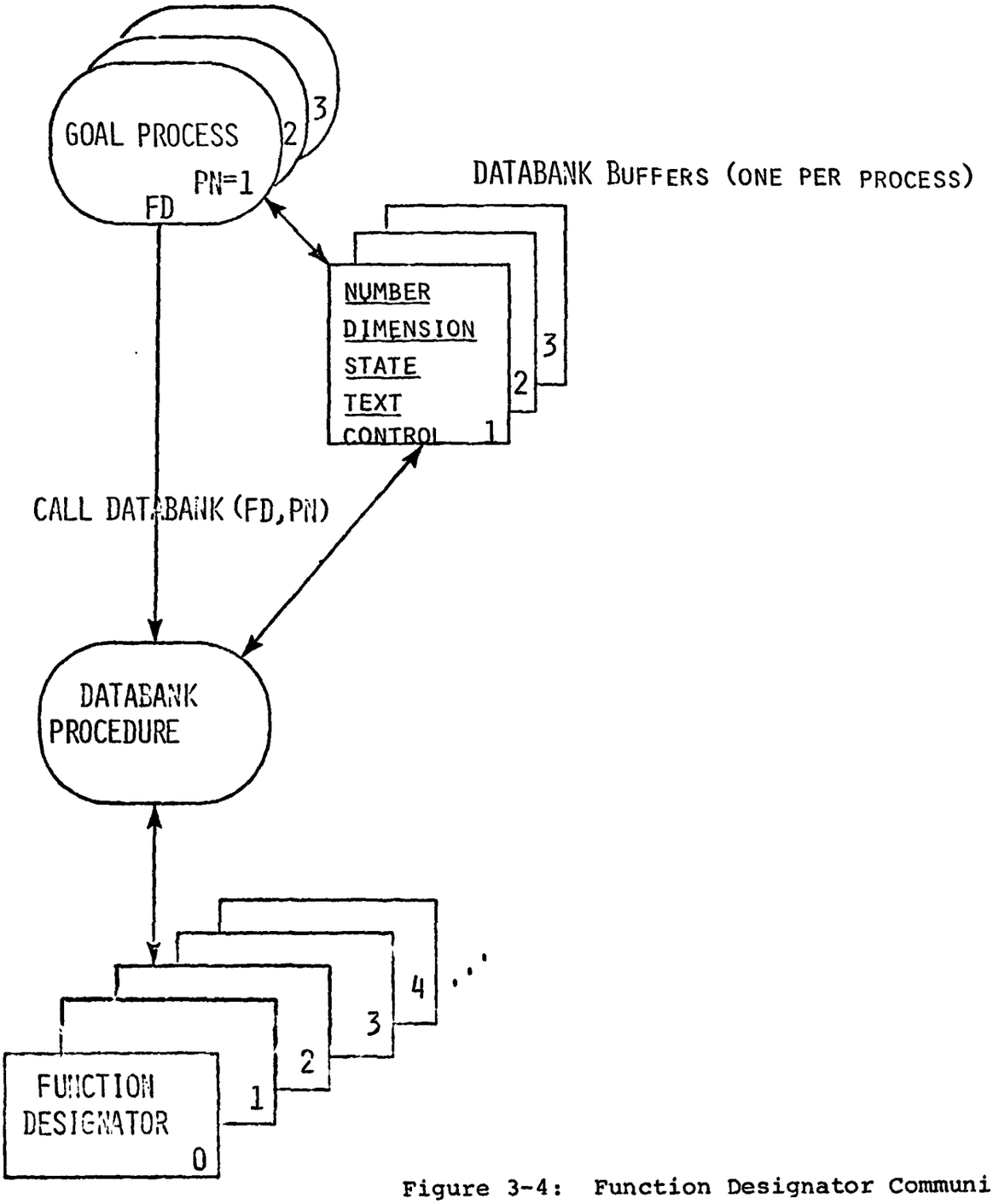


Figure 3-4: Function Designator Communication

by the process number (<PN>). For example, Number <PN> is the DATABANK buffer for numeric data dedicated to process number <PN>.

7) The function designator information used by the procedure DATABANK will be assembled by the Translator.

3.4 REPEAT Statements

Under the restriction of no overlapping repeat groups, the Translator shall implement the following features in order to execute Goal REPEAT statements as specified in the Goal Reference Manual.

1) The Repeat translation process is handled separately for each Goal Program or subroutine.

2) The following variables are used in the implementation.

   a) RPT is an integer which records the dynamic nesting level of repeat group execution.

   b) RS is an integer representing the Repeat statement number currently being executed. RSMAX is the total number of repeat statements in the Goal program or subroutine being translated.

   c) RPTACT is an array of booleans of length RSMAX. Each bit corresponds to a Repeat statement. It is set if the Repeat statement is in the process of being executed. Many bits in the array RPTACT may be set simultaneously due to the possibility of dynamic nesting of repeat groups. Normal flow through a group of statements may not be the result of a REPEAT command. Then RPTACT is not set.

   d) SAVE is an array used to save the current value of RS when nesting of repeat groups occur. This array is of length NDL which is the maximum level of dynamic nesting allowed.

   e) RPTCTR is an array of repeat counters. Each counter is associated with a Repeat statement.

   f) HEAD is an array of step numbers which indicate the step number of the first statement of the repeat group. HEAD is indexed by RS to find the starting location of the repeat group.

g) TAIL is an array of step numbers associated with the last statement of a repeat group. TAIL(RS) is the last GOAL statement of the Repeat statement RS.

h) RETURN_LABEL_<RSN>: is the HAL/S statement label to which control is to be transferred after execution of the repeat group initiated by Repeat statement RSN.

i) LOC is a translator declared variable used by the function FLUSH. The value of LOC is set equal t the target of the GO TO statement when this targ is outside of the repeat group. See flowchart GO TO (34).

3) Each Repeat statement is translated into the following in-line HAL/S code with RSN = repeat number of the repeat instruction being translated and N = number of iterations. (See Figure 5.2)

```
    RPT = RPT + 1;          increment repeat nesting depth

    RPTACT<RSN> = ON;       active repeat group

    SAVE_RPT = RS;          save old repeat statement no.

    RS = <RSN>;             establish new repeat statement
                            no.

    RPTCTR_RS = <N>;        establish no. of iterations

    GO TO HEAD_RS;          transfer control to repeat
                            group

RETURN_LABEL_<RSN>:         return to this label

    RS = SAVE_RPT;          restore old repeat statement no.

    RPT = RPT - 1;          decrement repeat nesting depth
```

4) At the TAIL of each repeat group there shall be inserted:

```
HEAD_RS:  ┌--------------
          |  REPEAT GROUP
TAIL_RS:  └--------------
IF RPTACT_RS = TRUE AND RPTCONT = TRUE THEN
                /* RPT CONT is a HAL/S function which
                controls the execution of the Repeat
                group.  (See Note 6), next page) */
GO TO HEAD_RS;
ELSE GO TO RETURN_LABEL_CASE;
```

5) Once per Goal program or subroutine translated into HAL/S there shall be inserted the following statements:

```
RETURN_LABEL_CASE: DO CASE RS;
   GO TO RETURN_LABEL_1;
   GO TO RETURN_LABEL_2;
     :
   GO TO RETURN_LABEL_<PSMAX>;
END;
```

6) The RPTCONT function call shall consist of the following HAL/S code:

```
RPTCONT: FUNCTION;
RPTCTR_RS = RPTCTR_RS - 1; /*Decrements repeat
   counter by 1*/
IF RPTCTR_RS = 0 THEN
    DO;
    RESET RPTACT_RS;
    RETURN FALSE;
    END;
ELSE RETURN TRUE;
CLOSE;
```

7) The FLUSH procedure is called whenever a GO TO statement is encountered within an activated repeat group and when the target of the GO TO statement is outside the repeat group. The FLUSH function shall consist of the following HAL/S statements:

```
FLUSH:  PROCEDURE;
BACK:  RPTCTR_RS = 0;
RESET RPTACT_RS;
RS = SAVE_RPT;
RPT = RPT - 1;
IF RPT = 0 THEN RETURN;
```

IF NOT ($HEAD_{RS} \leq$ LOC AND $TAIL_{RS} \geq$ LOC)

THEN GO TO BACK;

ELSE RETURN;

CLOSE;

3.5 Software Interrupts

GOAL software interrupts possess the following qualities:

1) An interrupt may only be serviced at the end of a Goal statement.

2) Every GOAL program and subroutine possesses its own interrupt environment created by WHEN INTERRUPT and DISABLE statements.

3) Subroutines which are performed in the critical mode ignore all interrupts.

4) If enabled, an interrupt may cause one or both of the following actions to occur:

   a) a subroutine may be performed

   b) control may be passed to a specified step number. (GO TO or RETURN TO Options)

In order to provide these features in the GOAL_MASTER_PROGRAM, each translated GOAL program, and each translated GOAL subroutine are impacted. The following structure is specified for handling interrupts. Figure 3-5 shows the overall information flow.

1) At the GOAL_MASTER_PROGRAM level two arrays are declared.

   a) INT is an array of booleans indicating that an interrupt is pending. Each Goal program possesses one bit.

   b) INTNUM is an array of integers which indicate the number of the function designator associated with the pending interrupt. There is one integer for each GOAL program.

2) Function Designator number one (1) is reserved to provide all the logic necessary to control interrupts.

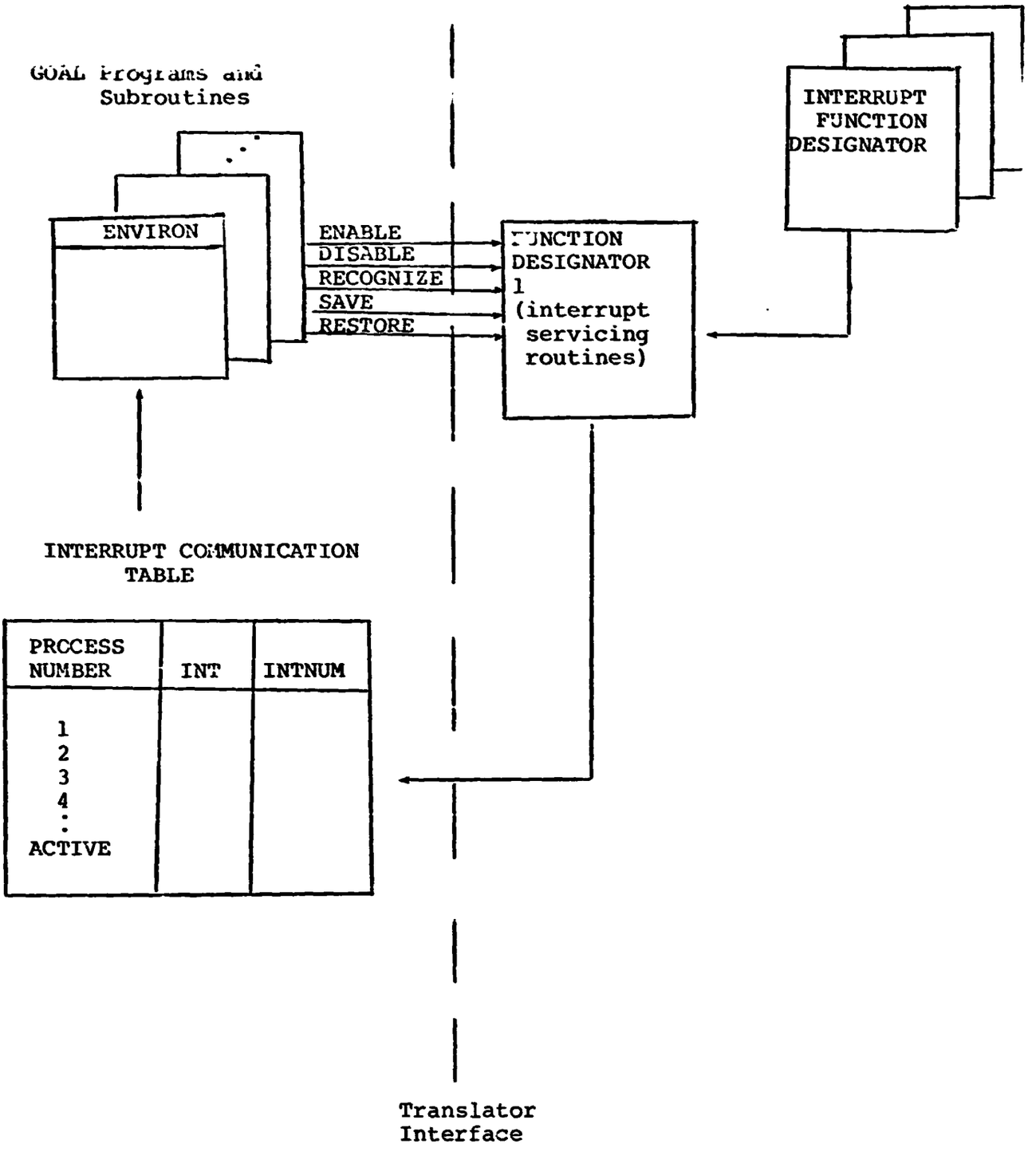


Figure 3-5: Interrupt Information Flow

Function designator 1 or FD(1) is, in actuality, a series of procedures which can be called via the HAL/S procedure DATABANK. The following services are provided:

a) FD(1) maintains the enabled or disabled status of all the interrupts associated with each GOAL Program.

b) It provides a stack mechanism so that the interrupt status may be PUSHED (saved and cleared) as well as POPPED (restored).

c) The interrupt condition associated with the various interrupt function designators are all communicated to FD(1). Depending upon the interrupt status, associated with the interrupt, the array INT and INTNUM are written into.

d) Any priority issue associated with interrupt conflicts is resolved by FD(1).

e) Because interrupt function designators are not completely specified in the GOAL Reference Manual, all the interrupt mechanisms requiring definition are placed within FD(1).

3) Each Goal program and subroutine maintains its own interrupt environment in a local two dimensional array called ENVIRON.

a) ENVIRON (1,K) contains the number of the K^{th} interrupt function designator used in the program or subroutine. The contents of ENVIRON (1,K) are established in Pass 1 of the Translator and do not change during the life of the translated program.

b) ENVIRON (2,K) contains the number of the subroutine to be performed when the K^{th} interrupt occurs. A 0 will indicate that no subroutine is to be performed. The WHEN INTERRUPT statement causes ENVIRON (2,K) to be written into.

c) ENVIRON (3,K) contains the case number of the GO TO statement to be executed during the servicing of interrupt K. A 0 indicates no GO TO option is present. The WHEN INTERRUPT statement causes ENVIRON (3,K) to be written into.

4) The translation of the WHEN INTERRUPT statement will cause the interrupt to be activated with the following HAL/S code, where <FD> is the interrupt function designator associated with the WHEN INTERRUPT statement:

```
IF NOT C DO; /*not a critical subroutine*/
   CONTROL<PN> = <enable interrupt>;
   NUMBER <PN>= <FD>;
   CALL DATABANK (1, <PN>);
END;
```

5) Disabling of a specified interrupt is accomplished by

```
CONTROL<PN> = <disable interrupt>;
NUMBER<PN> = <FD>;
CALL DATABANK (1, <PN>);
```

6) The saving and clearing of the interrupt status associated with a GOAL program is accomplished by

```
CONTROL<PN> =  <push interrupt environment>;
CALL DATABANK (1, <PN>);
```

7) Restoration of the interrupt status is accomplished by

```
CONTROL<PN> = <pop interrupt environment>;
CALL DATABANK (1, <PN>);
```

8) At the end of every goal statement the following is inserted

```
IF INT<PN> = TRUE THEN
   IF HANDLER = TRUE THEN GO TO DOCASE;
```

9) HANDLER is a translator defined function associated with each program or subroutine.

```
HANDLER:  FUNCTION BOOLEAN;
DO FOR I=1 TO ACTIVE;
```

```
IF ENVIRON (1,I), = INTNUM<PN> THEN EXIT:
   ELSE SIGNAL ERROR_X;

END;

CONTROL      = <INT BEING SERVICED>;
       <PN>
CALL DATABANK (1, <PN>);

DO CASE ENVIRON (2,I) ELSE;

CALL ...

   : Listing of all possible subroutine calls contingent
   * upon interrupts within the program or subroutine.

CALL ...

END;

IF ENVIRON (3,I) > 0 THEN RETURN TRUE; ELSE
   RETURN FALSE;

CLOSE;
```

10) At the end of every translated GOAL program or subroutine (excluding those with no interrupt statements) is the following:

```
DOCASE:  LOC=0; CALL FLUSH; /* Resets appropriate
                               repeat counters */

DOCASE ENVIRON (3,I) ELSE;

GO TO ...

   : Listing of all possible GO TO's contingent upon
   * interrupts

GO TO ...

END;
```

THIS PAGE INTENTIONALLY LEFT BLANK.

4.0 DEFINITIONS AND ASSUMPTIONS

4.1 Definitions and Notation

< > means substitute the contents of. For example:

$$A = \langle NAME \rangle D_{\langle LI \rangle}'$$

where

NAME = SAUL_S
LI = INDEX

IMPORTANT NOTE:
The Translator must substitute underscores for spaces in GOAL source identifiers.

then

$$\langle A \rangle = SAUL\_SD_{INDEX}$$

(K) means that K is a variable of an element.

NAME(K) Means that NAME is a function of K. For example,

$$Data = \langle TN \rangle D_{K,\langle CI \rangle}$$

where

TN = TABLE

CI = COLUMN

$$\langle DATA(K) \rangle = TABLED_{K,\ COLUMN}$$

$$\langle DATA(3) \rangle = TABLED_{3,COLUMN}$$

See subroutine section 5.2 for notations associated with CONVERT_NUMERIC, CONVERT_TIME, EVAL_INT_NAME, EVAL_ED, EVAL_NUM_FORM, LIM_FORM and REL_FORM.

In the Flowcharts, GOAL Source input is indicated by ' ', whereas HAL/S Source output is indicated by " ".

4.2 Assumptions

4.2.1 Overall Scope

a) Output of the Translator (Reference: GOAL-to-HAL Translator Final Report, Section 2.2.1a, Page 2-15)

The GOAL-to-HAL Translator will produce HAL/S code in full conformance with the appropriate (see Section 2.2) HAL/S documentation including:

HAL/S Language Specifications

HAL/S Language Forms

HAL/S Compiler System Functional Specifications

b) "Two-Pass" Structure

The Translator will operate in more than one pass. Conceptually, the first pass is required to take inventory of that GOAL program data which must be viewed from a "global" point of view. The first pass also processes the declare statements. The second pass processes the individual GOAL procedural and system statements. The HAL/S code generated in PASS 1 becomes the front end of HAL/S code generated in PASS 2.

c) Interface to the Flight Computer Operating System (FCOS) (Reference: Shuttle Avionics and the GOAL Language Final Report, Section 3.1.4 and Figure 3-4, Pages 3-9, 3-10)

GOAL programs translated into HAL/S will have the same interface specification as that provided for the mission application programs. The basic interfaces to the FCOS arise from HAL/S real time and error control statements. These interfaces will also apply to code derived from GOAL statements translated into HAL/S.

A corollary to this specification of the FCOS interface is that the Translator is not necessarily configured to supply HAL/S code for other computing systems in the overall Shuttle program, such as the Software Development Laboratory (SDL), Shuttle Avionics Integration Laboratory (SAIL), etc. GOAL-translated programs will run in these facilities to the extent that the operating systems are similar.

d) Use of the GOAL Master Program (Reference: GOAL-to-HAL Translation Final Report, Section 3.2, Pages 3-14 to 3-22)

The referenced text proposed a technical approach for dealing with GOAL capabilities in the areas of software interrupts, system terminations, and concurrent operations. This approach, designated the GOAL Master Program, is expanded

in Section 3.0 of this Specification document and will be the basis for implementing the translation process.

e) Non-GOAL (Reference: GOAL-to-HAL Translation Final Report, Section 3.1.5, Page 3-7)

The Translator will not have any capability for dealing with Non-GOAL source code.

f) Databank (Reference: GOAL-to-HAL Translation Final Report, Section 3.1.6, Pages 3-8 to 3-12)

The Translator plays no role in the creation and management of the Databank, which is unspecified at this time. However, the Translator, in addition to processing the GOAL source code into HAL/S, will provide a complete software system in the GOAL_MASTER_PROGRAM for communicating correctly and efficiently with the Databank. This is described in Section 3.3 of this Specification.

Creation and management of the Databank is accomplished by an unspecified Data Management System on the ground.

4.2.2 Features

a) Revision Numbers (Reference: GOAL-to-HAL Translation Final Report, Section 3.1.1, Page 3-1)

Revision labels are always part of a GOAL program name or a databank name. There are no run-time decisions involving revision number values.

b) Unrestricted STOP (Reference: GOAL-to-HAL Translation Final Report, Section 3.1.2, Page 3-2)

The Translator will provide the capability of stopping and restarting at selected points. The ability to restart at any point, the unrestricted STOP, will not be provided.

c) Overlapping REPEATS (Reference: GOAL-to-HAL Translation Final Report, Section 3.1.3, Pages 3-2 through 3-6)

As recommended by the referenced text, REPEAT groups are to be limited to non-overlapping groups with the grouping performed by the programmer.

d) Dimensional Labels in GOAL Quantities (Reference: GOAL-to-HAL Translation Final Report, Section 4.1, Pages 4-2, 4-4, 4-5, and 4-9)

GOAL quantities will be treated as in the referenced text by developing a label declaration, arrayed if necessary, to match the parameter declared as a Quantity in GOAL. The label is to be used for output labeling only and plays no role in dimension checking or automatic scaling.

e) Table Activation (Reference: GOAL-to-HAL Translation Final Report, Section 2.1.3.8, Page 2-13, and Section 4.13, Pages 4-7 through 4-12)

The activation of function designators and their associated rows of data will be handled by arrays of HAL/S BOOLEANs as described in the referenced text.

f) CONCURRENTLY (Reference: GOAL-to-HAL Translation Final Report, Section 3.2.2, Pages 3-21 and 3-22)

HAL/S does not permit use of a program module by more than one user process at a time. Accordingly, the Translator shall generated from each concurrent statement a uniquely named HAL/S task block, which will be executed via a HAL/S SCHEDULE statement.

g) Time Values

All time values are converted to, and maintained as, seconds (or other FCOS-defined unit of time) in floating point format. Conversion back to days, hours, minutes, and seconds can be performed by a function designator when required.

h) Comparisons

In a comparison test, if an array of elements is being tested, then all the tests must be true in order for the condition of the statement to be true. If tables are being tested, then tests are performed only on activated function designators.

i) Numerics

Number data is single precision floating point. Integer is a subset of floating point with a zero exponent. All number patterns may be converted to single precision floating point format without loss of precision. Function designators will perform the appropriate conversion for output.

j) SET For Time Value

The SET discrete statement will accomplish the FOR time-value option by calling the referenced function designator twice. The latter will have its own built-in toggle.

k) RECORD Statement Features

The system device default option in the RECORD tatement will be activated through the function designator <system device>. The ',' in the RECORD statement will be passed to the appropriate function designator for interpretation as a Line Feed or other formatting action consistent with the device being addressed.

l) Assumptions Concerning Present Value of (PVO)

* The number of elements in the first External Designator must equal the number of elements in the second External Designator.

* One External Designator may be a Table name, the other may be a list of Function Designators.

* In order for the PVO to transfer data from a sender to a receiver, the activate bits of both sender and receiver must be 1.

Illegal GOAL:

```
Send PVO <FD1> TO <FD2>, <FD3>;
Send PVO <FD1>, <FD2> TO <FD3>;
```

Legal GOAL:

```
SEND ALPHA FUNCTIONS TO <FD1>,
<FD2>, <FD3>;
(Table ALPHA must have 3 rows)
```

4.2.3 Ground-Based Operating System (GBOS)

a) GBOS Support Services (Reference: None)

During run-time, the GBOS may be called upon to support the flight computer with services requiring the downlink of data to CRT's and printers controlled on the ground. The details of accessing ground-based peripheral equipment from the flight computer is external to the GOAL-HAL/S Translator. The flight computer interface is handled through an appropriate function designator.

4.2.4 Undefined

a) FEEDBACK LOOPS (Reference: GOAL-to-HAL Translation Final Report, Section 2.1.4, Page 2-14)

The GOAL syntax loops presently have limitations designated by letter symbols. These symbols are summarized in the feedback letters in the GOAL Syntax Diagrams Handbook, NASA/KSC Document TR-1213, dated 16 April, 1973. Values have not yet been assigned to these letter symbols. Tentatively, the GOAL-to-HAL Translator could be designed to flag an error if the number of executions of the loop exceeds the letter. At a later time, the letter would have to be specified.

5.0 TRANSLATION REQUIREMENTS

5.1 Structure of the Translator

5.1.1 Introduction

The implementation of the translator may ultimately consists of one or many passes through the GOAL source code. The actual number of passes is a matter of detailed design of the Translator for efficiency.

In order to isolate all the functions involved in the Translator, a conceptual division into two passes will be made. This is done mainly for the purpose of achieving clarity in this specification. Much of the effort expended in one pass of the Translator could just as well be accomplished in another pass. Some freedom to choose should be available to the implementation stage.

Conceptually, the Translator consists of two sequential processing sections called PASS 1 and PASS 2 and a final Output Processor which generates the GOAL_MASTER_PROGRAM and organizes the translated statements in an appropriate format. This format is determined by the output device used by the machine in which HAL/S compilation is to take place.

The segregation of functions between PASS 2 and Output Processor is a bit arbitrary, but it was done in order to make the Translator more modular and place any machine dependent (the machine in which the compiler is to run) features in the Output Processor.

5.1.2 PASS 1

The following operations shall be performed in Pass 1:

1) Process GOAL Declarations

All GOAL Declaration statements shall be processed in order to place into the Translator's symbol table the following information:

a) For single element variables

<NAME>TYPE

where <NAME> is the name of the variable as it appears in the GOAL declaration statement. <NAME> TYPE can assume one of four values:

0 - Numeric data
1 - Quantity data
2 - State data
3 - Text data

b) For list declarations

<LISTNAME>TYPE = 0,1,2,3 as for single elements

<LISTNAME> = length of the list as described in the GOAL declaration statement.

c) For table declarations

<TABLENAME>TYPE = 0,1,2,3 as with single elements.

<NAME>TYPE

<TABLENAME>_C = This is an array of column names as declared in the GOAL declaration statement. This array is an optional feature of the GOAL declaration and consequently may not appear in the symbol table.

<TABLENAME>CS = This is the column size of the table as declared in the GOAL declaration statement.

<TABLENAME>RS = This is the row size of the table as declared in the GOAL declaration statement.

The actual generation of the appropriate HAL/S declaration statements, as presented in flow charts, can either be performed in Pass 1 or in Pass 2. This is implementation dependent and does not impact the final output of the Translator.

2) Step Number Resolution

Pass 1 will assign step numbers to all GOAL statements in sequential order. When Pass 2 processes the GOAL statements, all statements will possess step numbers. Pass 1 will substitute any GOAL source step numbers with the appropriate translator supplied step numbers.

3) Assign Process Number (PN)

A process is defined as any GOAL Program or HAL/S generated Task (created by a Concurrently Verify, Concurrently Record or Concurrently Perform statement).

Pass 1 will assign a process number (PN) to each process. The maximum number of processes (PNmax) equals the number of GOAL programs plus the total number of concurrent statements

which appear in all the Goal Programs being translated. A Process Array will be generated for use by Pass 2. The organization of this array is shown in Figure 5-1.

All GOAL programs are assigned process numbers first, followed by Concurrently Verify, Record and Perform statements. The Concurrently Perform statements are placed last in this array so that the numbers PR, PNmax, and Pmax can be differentiated. These numbers are later used in the creation of declaration statements in the GOAL_MASTER_PROGRAM.

The process array is utilized many places in Pass 2.

a) For convenience the task number associated with each concurrent statement will be made equivalent to the process number which appears in the process array.

b) Whenever <PN> is indicated in a Pass 2 flow chart, the process number associated with the GOAL process being translated (as indicated in the process array) will be used.

c) RELEASE statements require the association of a step number in a program with the task name. The task name is generated by concatenating the process number associated with the step number with the letters TASK, i.e., TASK <PN>.

d) In Pass 2, when tasks are generated by concurrent statements the task number (process number) is obtained from the process array.

4) Repeat Statement Analysis

In order to generate the appropriate HAL/S code associated with Repeat Statements the following information must be gathered in Pass 1 and placed in the Repeat Array (Figure 5-2). This is done for each GOAL program or subroutine:

a) Each repeat statement is assigned a number (RSN) according to the order in which they are processed. RSMAX equals the number of repeat statements.

b) The repeat statement step number (e.g., step 5) is used for identification purposes.

c) Associated with each Repeat statement is the step number of the first GOAL statement of the repeat group. This is placed in the column HEAD.

Figure 5-1: Process Array

Process Number	Identifiers
1 2 : : PR	Name of First Goal program Second Goal Program : : Last Goal Program
PR+1 PR + PV	Program name and step number associated with first Concurrently Verify statement Program number and step number associated with second Concurrently Verify statement : : with last Concurrently Verify statement
PR + PV + 1 Pmax = PR+PV+PREC	Program name and step number associated with first Concurrently Record statement Program name and step number associated with second Concurrently Record statement : : with last Concurrently Record statement
PR+PV+PREC+1 PNmax=PR+PV+PREC+PPER	Program name and step number associated with first Concurrently Perform statement Program name and step number associated with second Concurrently Perform statement : : Program name and step number associated with last Concurrently Perform statement

PR = number of goal programs
PV = number of concurrently verify statements
PREC = number of concurrently record statements
PPER = number of concurrently perform statements

Figure 5-2: REPEAT Array

RSN	Repeat Statement Number	HEAD	TAIL
1	Step_5	Step_7	Step_19
2	Step_12	Step_103	Step_103*
3			
⋮			
RSMAX			

RSMAX = number of repeat statements within the Goal program or subroutine being translated.

* In the case of a REPEAT group containing one statement HEAD and TAIL refer to the same statement number.

d) Associated with each Repeat statement is the step number of the last Goal statement in the repeat group. This is placed in column TAIL.

The following declarations are created and are to be incorporated into the translation of the GOAL program or GOAL subroutine.

```
DECLARE HEAD ARRAY (RSmax) INTEGER SINGLE
   INITIAL (<Head(1)>, <Head(2)>, ... <Head(RSmax)>);
DECLARE TAIL ARRAY (RSmax) INTEGER SINGLE
   INITIAL (<TAIL(1)>, <TAIL(2)>, ... <TAIL(RSmax)>);
DECLARE RPTCTR INTEGER SINGLE;
DECLARE RPTACT ARRAY (RSmax) BOOLEAN
   INITIAL (OFF);
DECLARE SAVE ARRAY (NDL);
   (NDL = number of dynamic nesting levels allowed)
DECLARE RPT INTEGER SINGLE;
DECLARE RS INTEGER SINGLE;
```

$HEAD_K$ and $TAIL_K$ are the K^{th} entries in the repeat array.

e) If no Repeat statements are encountered in the program or a subroutine being translated, a flag is set so that all Repeat Mechanisms will be eliminated in Pass 2.

5) Replace

All the Replace statements encountered in the GOAL source are executed in Pass 1 prior to code generation in Pass 2.

6) Macros

All macros are expanded in both Pass 1 and Pass 2. The contents of the expanded macros are processed identically to other Goal statements.

7) Interrupts

In order to generate the appropriate code and data to handle software interrupts the following action must be performed in Pass 1. These actions are performed for each GOAL program and subroutine.

a) An interrupt number (IN) is assigned to each interrupt function designator encountered in the programmer subroutine. The list of interrupt function designators is determined by analyzing all the WHEN INTERRUPT statements encountered. The interrupt numbers are assigned on a first come, first served basis. The variable ACTIVE will contain the total number of interrupts used in the program or subroutine being processed.

b) A list of all subroutines referenced in the WHEN INTERRUPT statement is assembled. A subroutine number, SN, is assigned to each subroutine on a first come, first served basis.

c) A list of all the GO TO or RETURN TO step numbers is assembled and a case number (CN) is assigned to each case.

The arrays IN, SN, CN will be used in translating WHEN INTERRUPT statements as well as in synthesizing the final structure of each GOAL program or subroutine.

d) The following declaration statements are created by Pass 1 and entered at the beginning of the translated GOAL program or subroutine.

```
DECLARE ENVIRON ARRAY (3,<active>) INTEGER

    SINGLE INITIAL(<FDI(1)>, <FDI(2)>, ...
      <FDI(Active)>,
                    0, 0,  ...  0,
                    0, 0,  ...  0);
```

<FDI(K)> is the number of the function designator associated with the K^{th} interrupt.

```
DECLARE ACTIVE INTEGER SINGLE INITIAL
    (<ACTIVE>);
```

e) A list must be generated which relates the step number of the WHEN INTERRUPT Statement to the interrupt function designator number. This list is used in translating the DISABLE statement in Pass 2.

f) If no WHEN INTERRUPT statements are encountered by Pass 1 in a program or subroutine, then a flag is set in the translator so that all interrupt mechanisms are eliminated in the code generation part of Pass 2.

8) Function Designators, External Designators and the Databank

In order to provide a proper communication to databank information the following actions are performed by Pass 1:

a) All the function designators referenced in the GOAL source being compiled are gathered. This includes all GOAL programs and subroutines in the GOAL Translation submittal. Each function designator is assigned a function designator number on a first-encountered basis.

b) Every time an external designator is encountered which consists of a grouping of function designators (rather than a table name) the array of function designator numbers (AFD) is declared

```
DECLARE AFD<N> ARRAY (<KMAX>) INTEGER SINGLE
   INITIAL (<FD(1)>, <FD(2)>, ... <FD(KMAX)>);
```

where <FD(K)> is the K^{th} function designator number encountered in the external designator.

<KMAX> is the number of function designators in the external designator.

<N> is the number of the external designator assigned on a first come basis.

c) The list of function designator numbers gathered in Pass 1 is sufficient information to extract from the ground based data bank all the information required for the flight machine. This process will be performed by the Translator.

9) GOAL Comments

All the GOAL comments appearing within a GOAL statement will be collected by Pass 1, and inserted into the GOAL Source after the semicolon (;) of the GOAL statement. The translated GOAL comments will then be handled according to the rules of the HAL/S output writer.

5.2 Translator Subroutines

5.2.1 CONVERT_NUMERIC

The CONVERT_NUMERIC subroutine shall take a GOAL number or a GOAL number pattern and convert it to single precision format. The size of the number pattern (BIN, OCT, HEX), shall be such as to fit into the fraction portion of the floating point format without any loss of information. This routine shall return the result as the Translator variable <VALUE>.

5.2.2 CONVERT_TIME

This is a procedure which accepts TIME VALUE and Returns :

TIM = number (e.g., 3)

or

name (e.g., ALPHA)

The literal time value returned is converted into a HAL/S scalar in units of seconds or other FCOS defined unit of time.

5.2.3 EVAL_INT_NAME

The Translator shall provide a subroutine called EVAL_INT_NAME which shall accept the GOAL internal name and return the parameters NAME, DATA, DIM, ACT, KMAX, and T as defined in the chart below.

Case / Parameter	Single Name	List (1)	TABLE(2,3,4,5) with a Row index	Table Without a Row Index
NAME	<NAME>	<LISTNAME>	<TABLENAME>	<TABLENAME>
DATA	<NAME>D	<LISTNAME>D$_{<LI>}$	<TABLENAME>D$_{<RI>,<CI>}$	<TABLENAME>D$_{K,<CI>}$
DIM	<NAME>DIM	<LISTNAME>DIM$_{<LI>}$	<TABLENAME>DIM$_{<RI>,<CI>}$	<TABLENAME>DIM$_{K,<CI>}$
ACT			<TABLENAME>A$_{<RI>}$	<TABLENAME>A$_{K}$
KMAX	1	1	1	<TABLENAME>RS
T	0	0	1	2

Notes:

1) LI is the list index. It can either be an integer (e.g., 3) or an alphanumeric name (e.g., K).

2) RI is the table row index. It can either be an integer or an alphanumeric name.

3) CI is the table column index. The column name is converted to a column number by searching the array <TABLENAME> C created in Pass 1.

4) A degenerate table of 1 row will be returned with T=1, DATA= $<TN>D_{1,<CI>}$, ACT=$<TN>A_1$, KMAX=1.

5) If TABLENAME is not specified then the tablename previously defined in the statement being translated is used.

6) KMAX = number of elements in internal name.

7) T = Tag, identifying the cases shown on the previous page.

5.2.4 EVAL_ED

The Translator shall provide a subroutine called EVAL_ED which accepts the GOAL external designator and returns the parameters T, AFD(K), KMAX, and TN as defined in the table below.

Parameter \ Case	Array of Function Designators	Table Name
T	0	1
AFD(K)	Array of function designators AFD(K) = AFD<N>$_K$	
KMAX	length of array	<TABLENAME>RS
TN		<TABLENAME>

Pass 1 will generate the following declaration

```
DECLARE AFD<N> ARRAY<KMAX> INTEGER;
```

Where N = a unique external designator identification number assigned in Pass 1.

5.2.5 EVAL_NUM_FORM

The Translator shall include a subroutine designated EVAL_NUM_FORM. This subroutine shall generate a HAL/S equivalent numeric formula. EVAL_NUM_FORM shall:

1) Return NF(K) = Numeric Formula

   The index K is used if the numeric formula contains table names as variables. In this case, K is used to index down the rows of table.

   For example, if A and B are table names in GOAL, the GOAL numeric formula:

   (A) + (B) + 1;

   would become the HAL/S numeric formula:

   $$NF(K) = AD_{K,<CI_1>} + BD_{K,<CI_2>} + 1$$

   HAL/S will perform an element by element addition if AA_K and BA_K are True.

2) Return TNA which is an array of table names used in the numeric formulae.

   Example:

   $TNA(K) = K^{th}$ table name

   $<TNA(K)>A_j = j^{th}$ activation bit of the K^{th} table na

3) Return LMAX = length of TNA array. All quantity data and number data is assumed to be single precision floating point.

4) Return T, which is a control flag.

   T = 0, if no table names are in the formula.

   T = 1, if there are table names in the formula.

5.2.6 LIM_FORM

The Translator shall contain a subroutine called LIM_FORM which accepts the GOAL limit formula, without the optional internal name. The internal name, if necessary, will be evaluated separately. The following items are returned by LIM_FORM to the caller.

1) Lower Limit Information

   If an internal name is indicated the EVAL_INT_NAME is called and the following parameters are returned.

   LLT = <T>
   LLNAME = <NAME>
   LLDATA = <DATA>
   LLACT = <ACT>

   A number will return:

   LLT = 0
   LLDATA = <NUMBER VALUE>

   A Quantity will return:

   LLT = 0
   LLDATA = <QUANTITY VALUE>

   All of the above parameters are character strings to be used by the Translator.

2) Upper Limit Information

   This is evaluated similar to the lower limit. The following parameters are returned:

   ULT = <T>
   ULNAME = <NAME>
   ULDATA = <DATA>
   ULACT = <ACT>

   A number will return:

   ULT = 0
   ULDATA = <number value>

   A Quantity will return:

   ULT = 0
   ULDATA = <Quantity value>

3) LF = 'OR' if the Not option is used

'AND' if the Not option is not used.

Lower and Upper limit information is reversed for the Not option.

4) EXDATA = 'NUMBER$_{<PN>}$'

This is used only when external designators are employed in conjunction with the limit formula.

5.2.7 REL_FORM

The Translator shall contain a subroutine called REL_FORM which accepts the GOAL relational formula without the optional internal name. The internal name, if necessary, will be evaluated separately. The following items are returned by REL_FORM.

1) The relation

RF which is either =, >, <, $\geq$, $\leq$

2) RT = 0 single name or type:

RNAME = number, converted number pattern or quantity value, single name, <LIST NAME>$_{<LI>}$' TEXT data, STATE data.

RT = 1: RNAME =<TABLENAME>$_{<RI><CI>}$

RACT(K) = <TABLENAME>A$_{<RI>}$

RT = 2: RNAME(K) = <TABLENAME>$_{K, <CI>}$

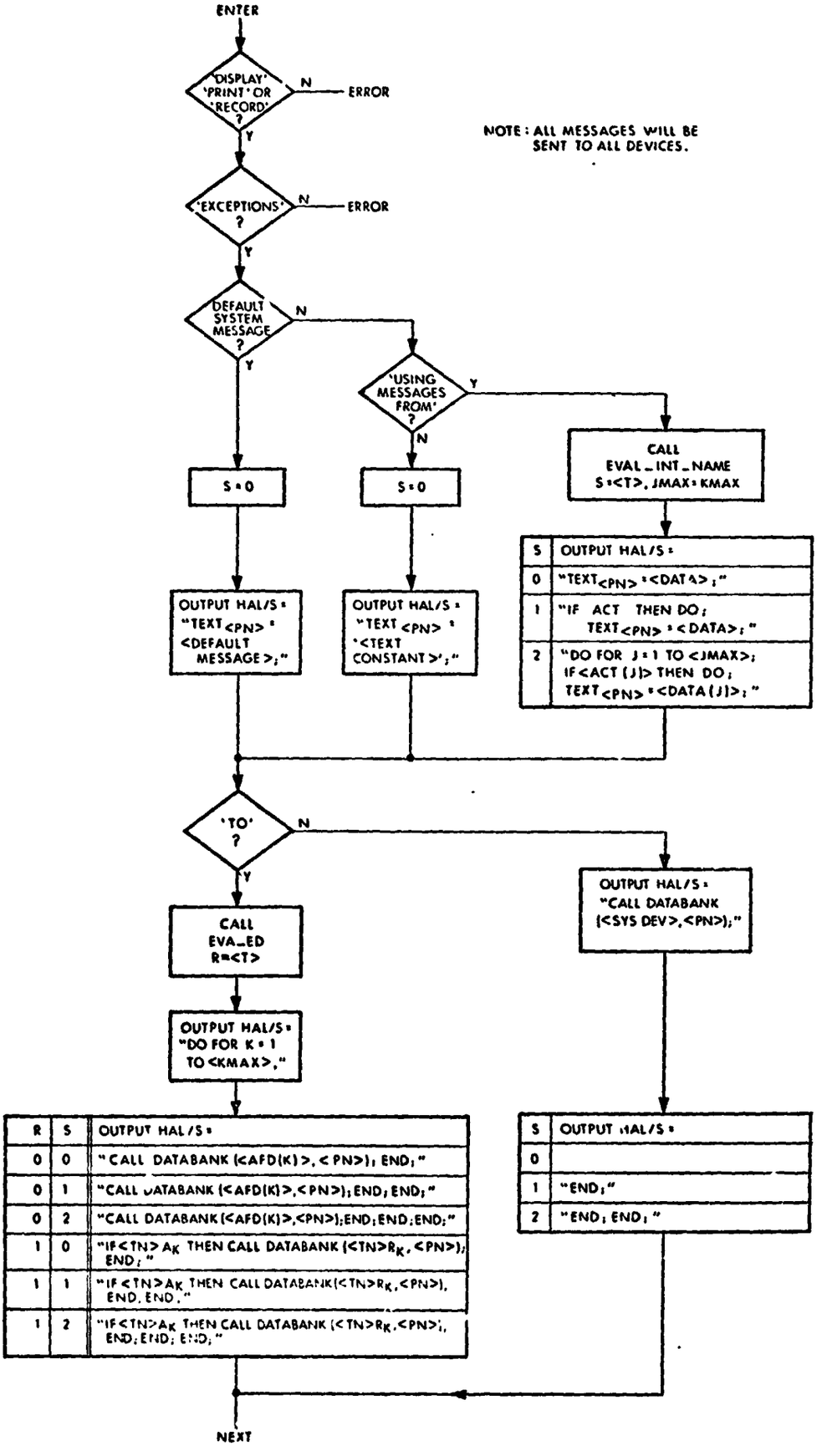
RACT(K) = <TABLENAME>A$_{K}$

KMAX = <TABLENAME>RS

3) EXDATA = 'NUMBER$_{<PN>}$', STATE$_{<PN>}$', 'TEXT$_{<PN>}$'

(used for external designators only.)

OUTPUT EXCEPTION SUBROUTINE (53)



NOTE: ALL MESSAGES WILL BE SENT TO ALL DEVICES.

CHART 1 OF 1

PRESENT VALUE OF (PVO) SUBROUTINE

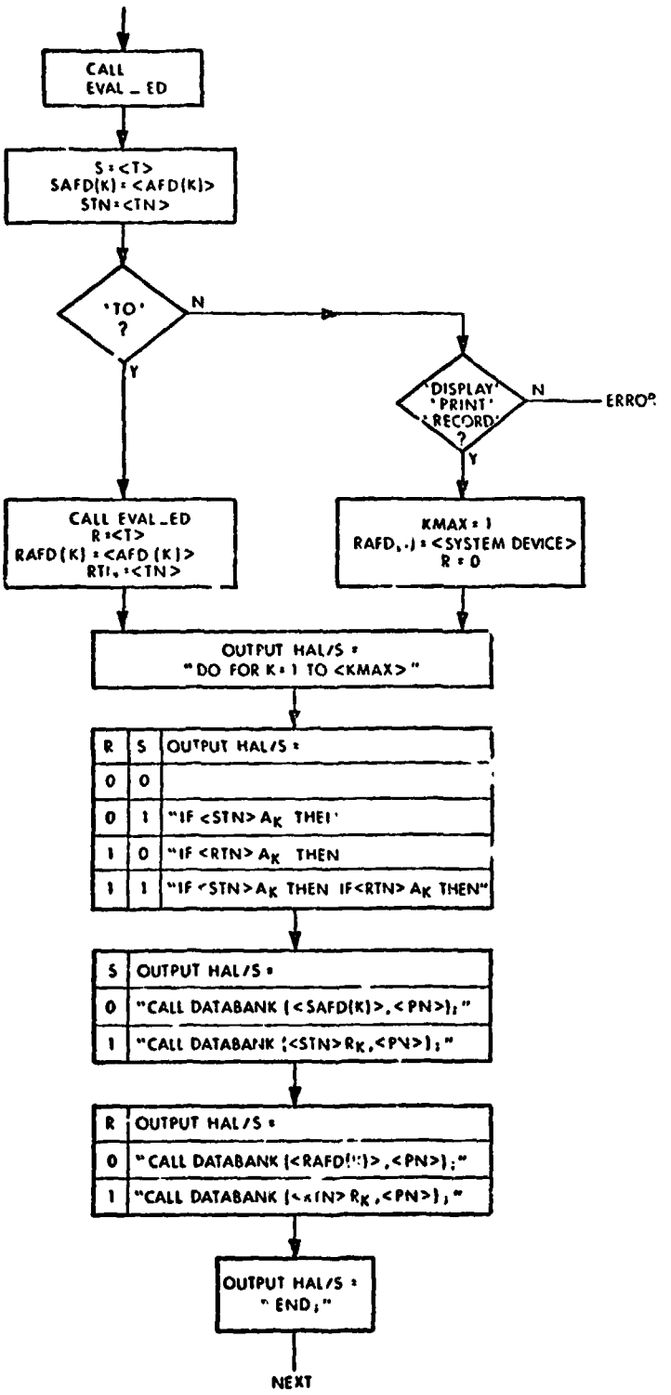


CHART 1 OF 1

5.3 Flowcharts

5.3.1 Declaration Statements

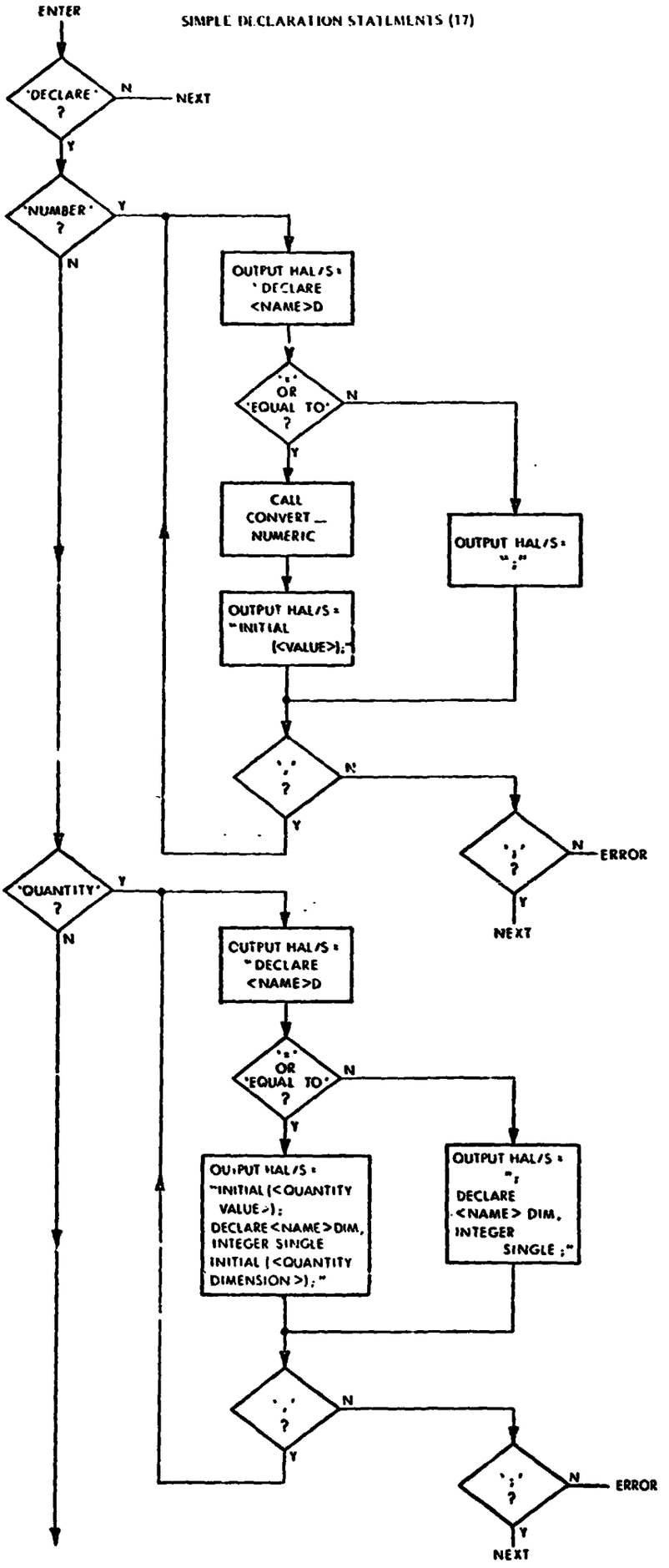

ENTER
SIMPLE DECLARATION STATEMENTS (17)
'DECLARE' ?
N
NEXT
Y
'NUMBER' ?
Y
N
OUTPUT HAL/S = ' DECLARE <NAME>D
':' OR 'EQUAL TO' ?
N
Y
CALL CONVERT_ NUMERIC
OUTPUT HAL/S = ";"
OUTPUT HAL/S = "INITIAL (<VALUE>);"
';' ?
N
Y
';' ?
N
ERROR
Y
NEXT
'QUANTITY' ?
Y
N
OUTPUT HAL/S = "DECLARE <NAME>D
':' OR 'EQUAL TO' ?
N
Y
OUTPUT HAL/S = "INITIAL (<QUANTITY VALUE>); DECLARE <NAME> DIM, INTEGER SINGLE INITIAL (<QUANTITY DIMENSION>);"
OUTPUT HAL/S = "; DECLARE <NAME> DIM, INTEGER SINGLE;"
';' ?
N
Y
';' ?
N
ERROR
Y
NEXT

CHART 1 OF 2

SIMPLE DECLARATION STATEMENTS (17), (CONT.)

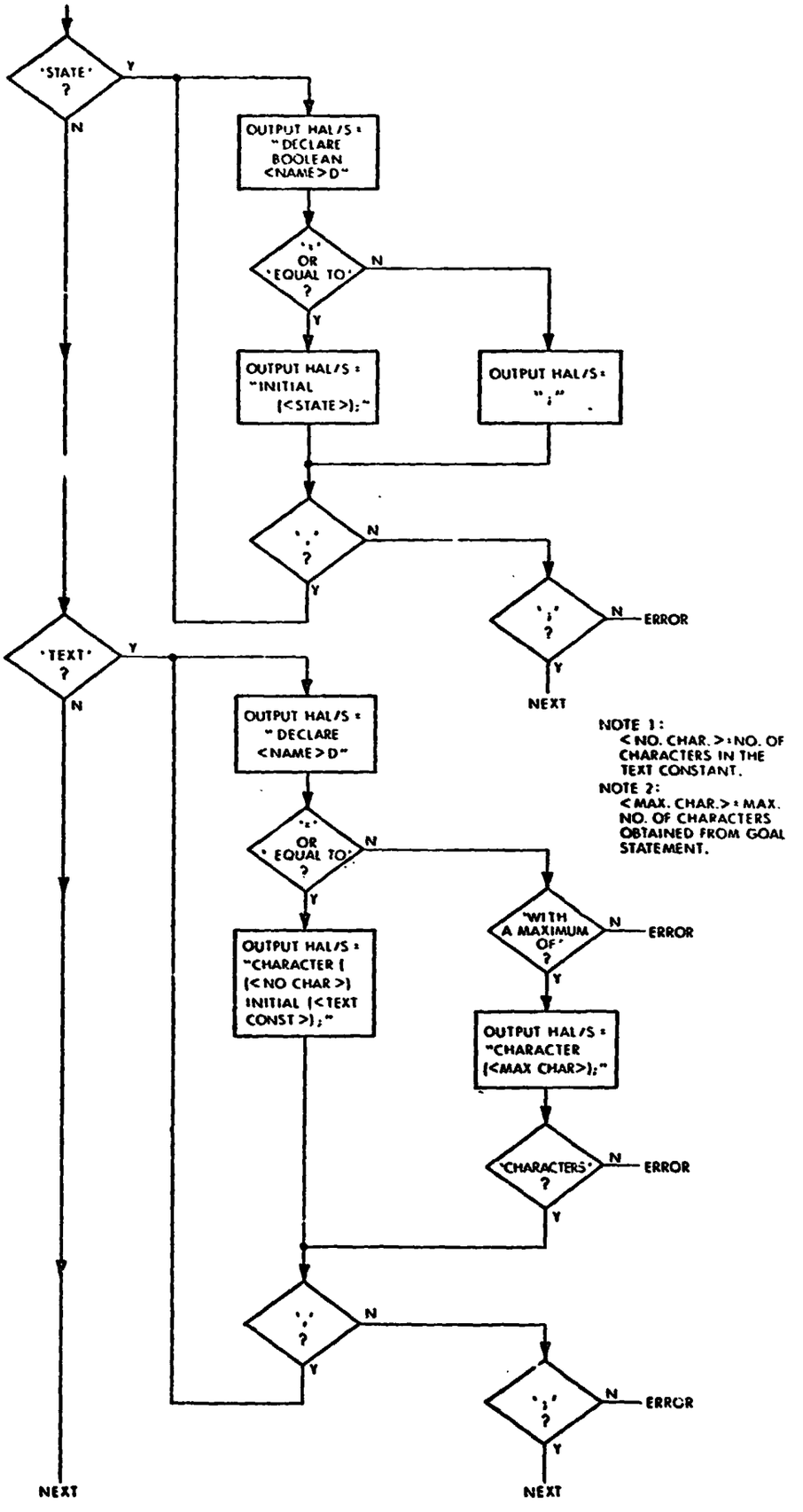


CHART 2 OF 2

DECLARE NUMERIC LIST STATEMENT (18)

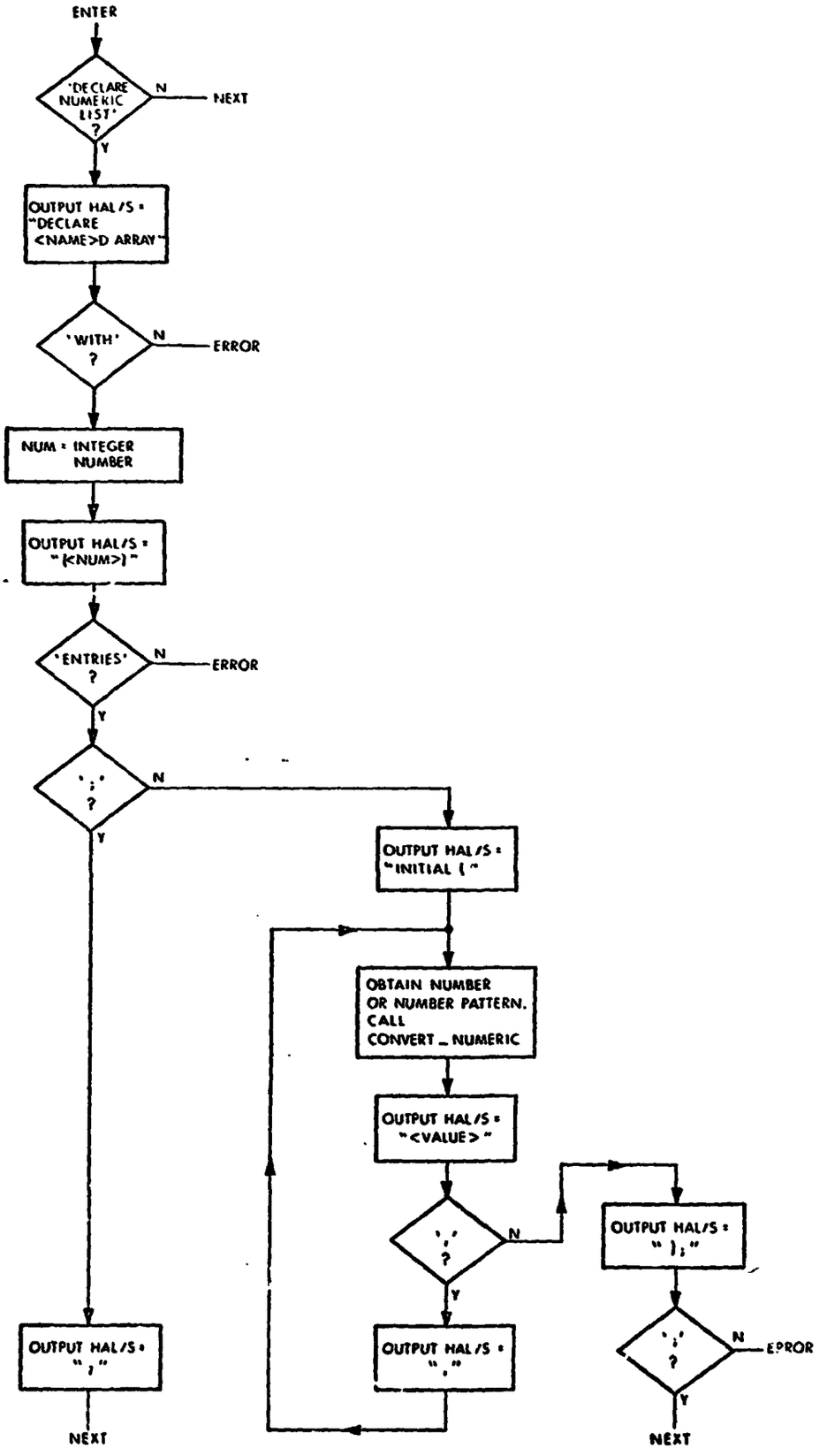


CHART 1 OF 1

DECLARE QUANTITY LIST STATEMENT (20)

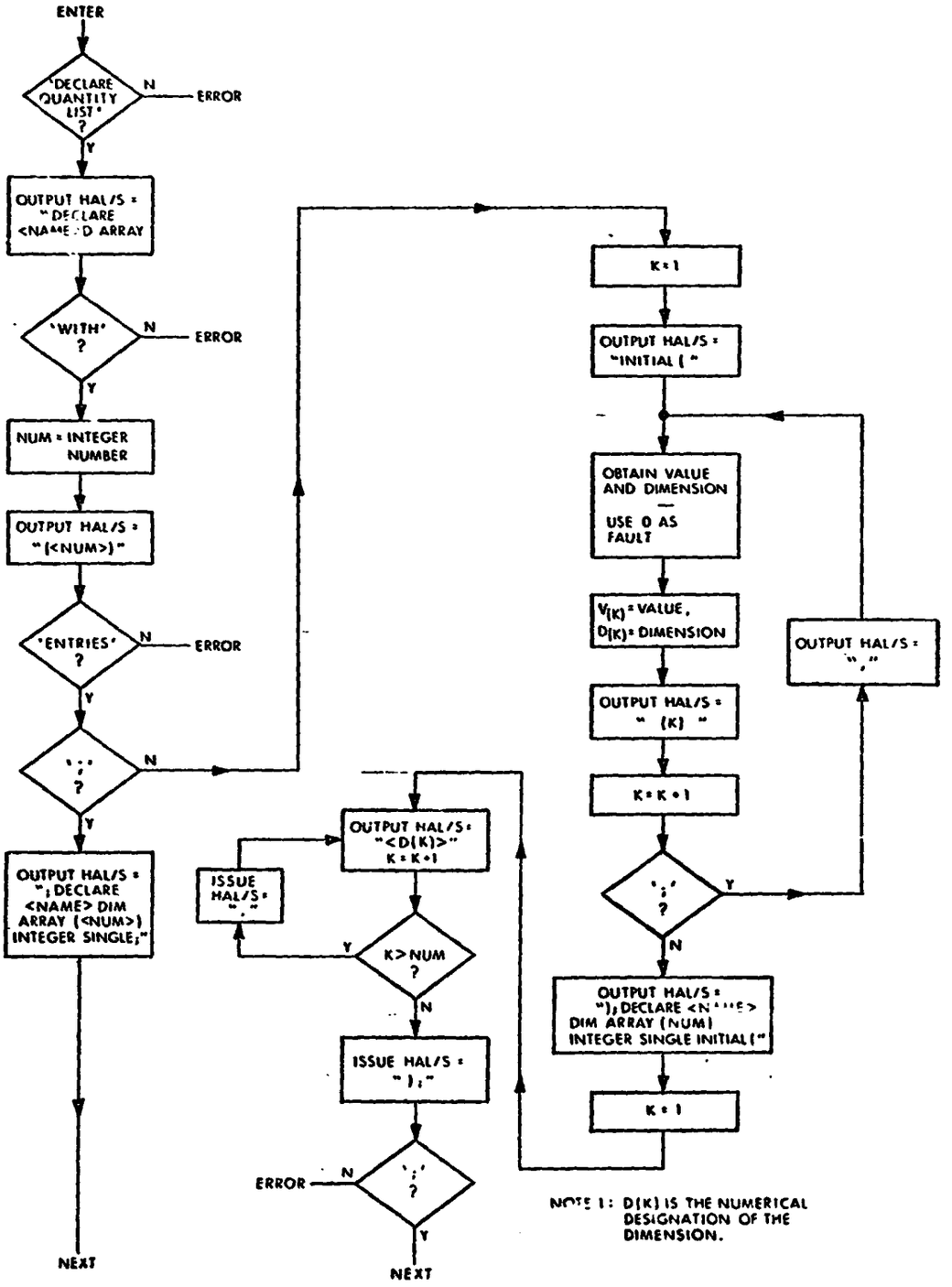


CHART 1 OF 1

DECLARE STATE LIST STATEMENT (22)

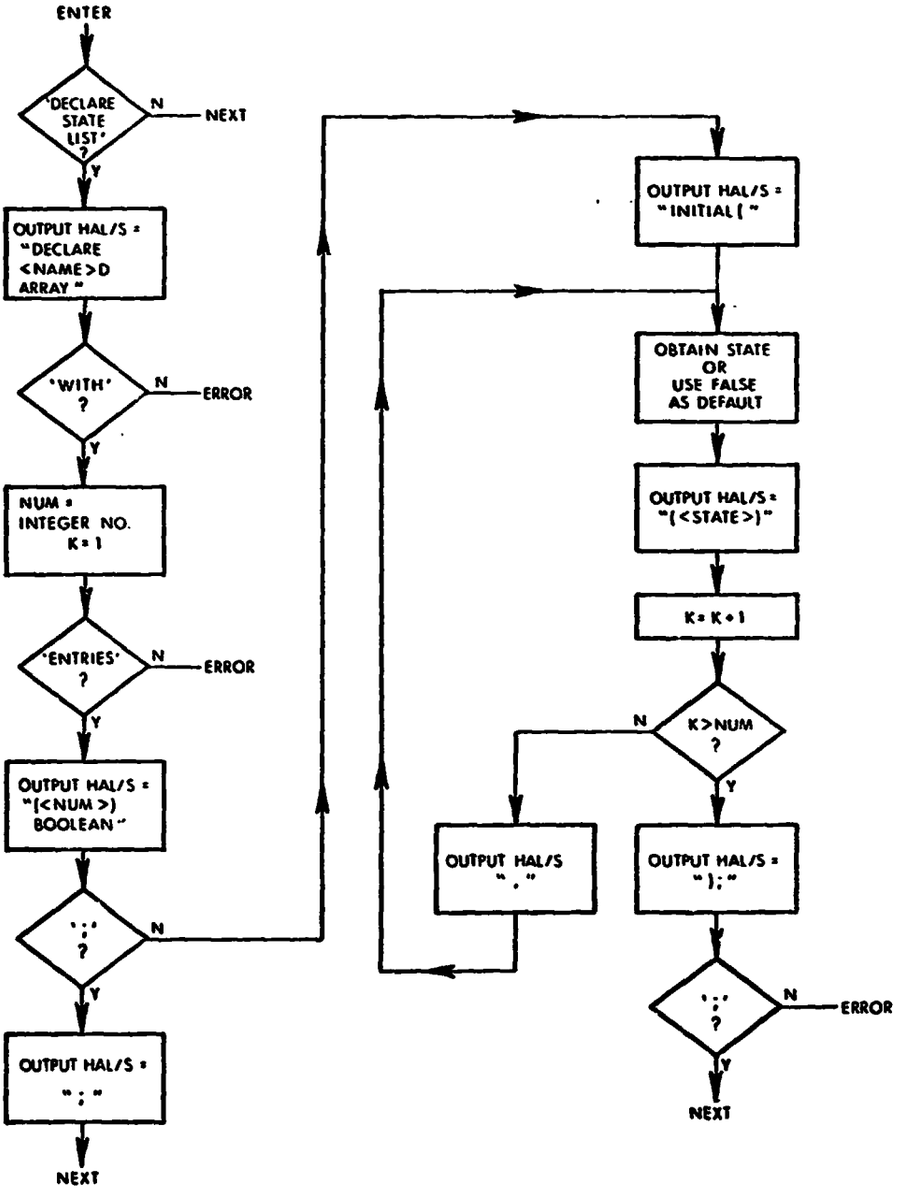


CHART 1 OF 1

THIS PAGE INTENTIONALLY LEFT BLANK.

DECLARE TEXT LIST STATEMENT (24)

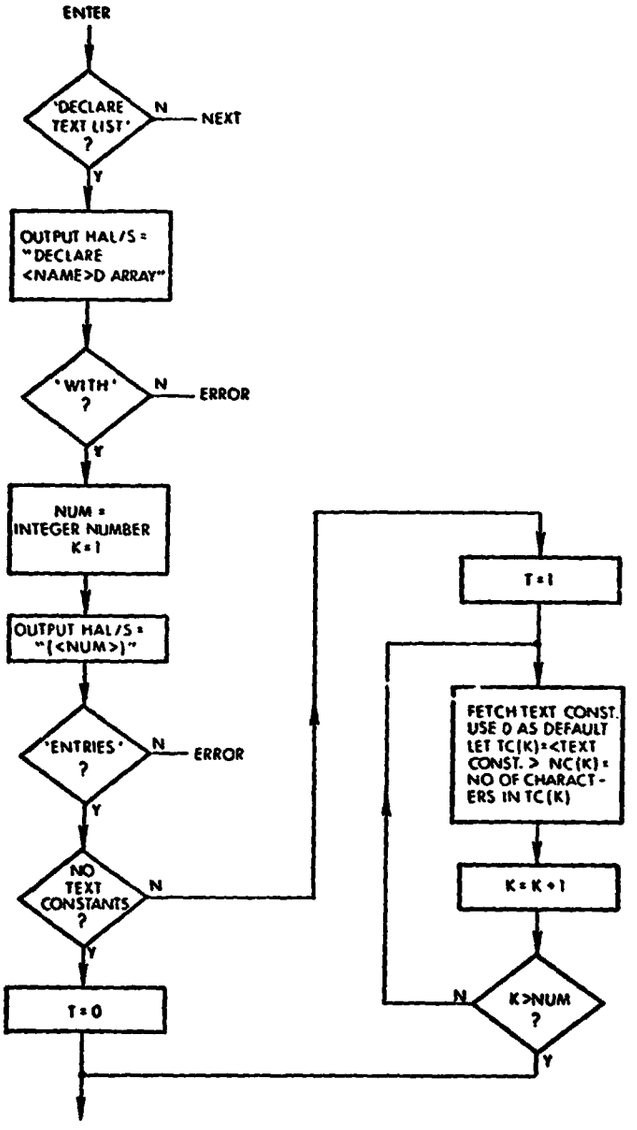


CHART 1 OF 2

DECLARE TEXT LIST STATEMENT (24) , (CONT.)

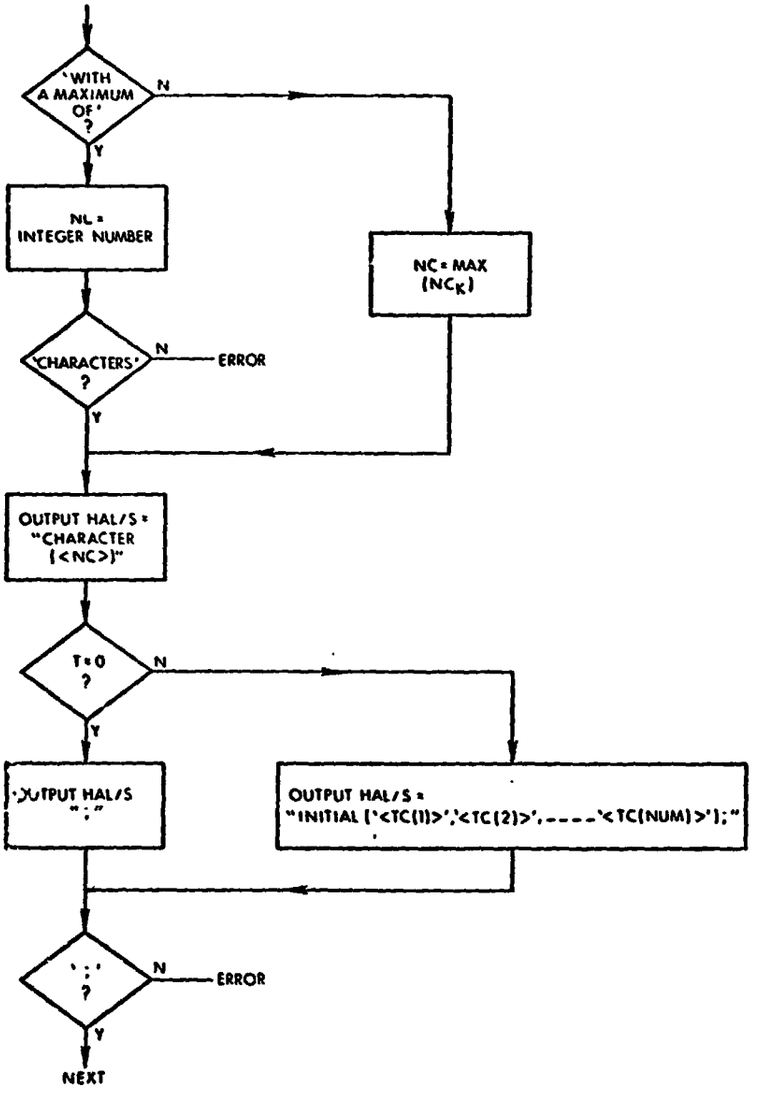


CHART 2 OF 2

DECLARE NUMERIC TABLE STATEMENT (19)

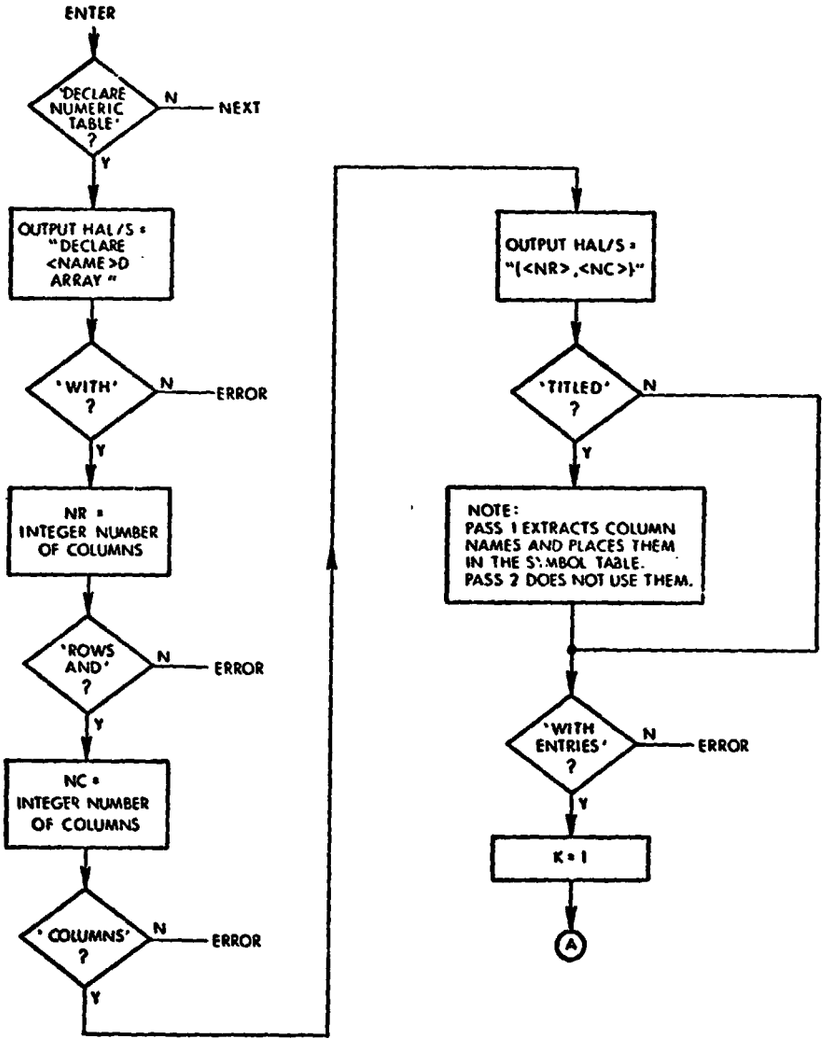


CHART 1 OF 2

DECLARE NUMERIC TABLE STATEMENT (19), (CONT.)

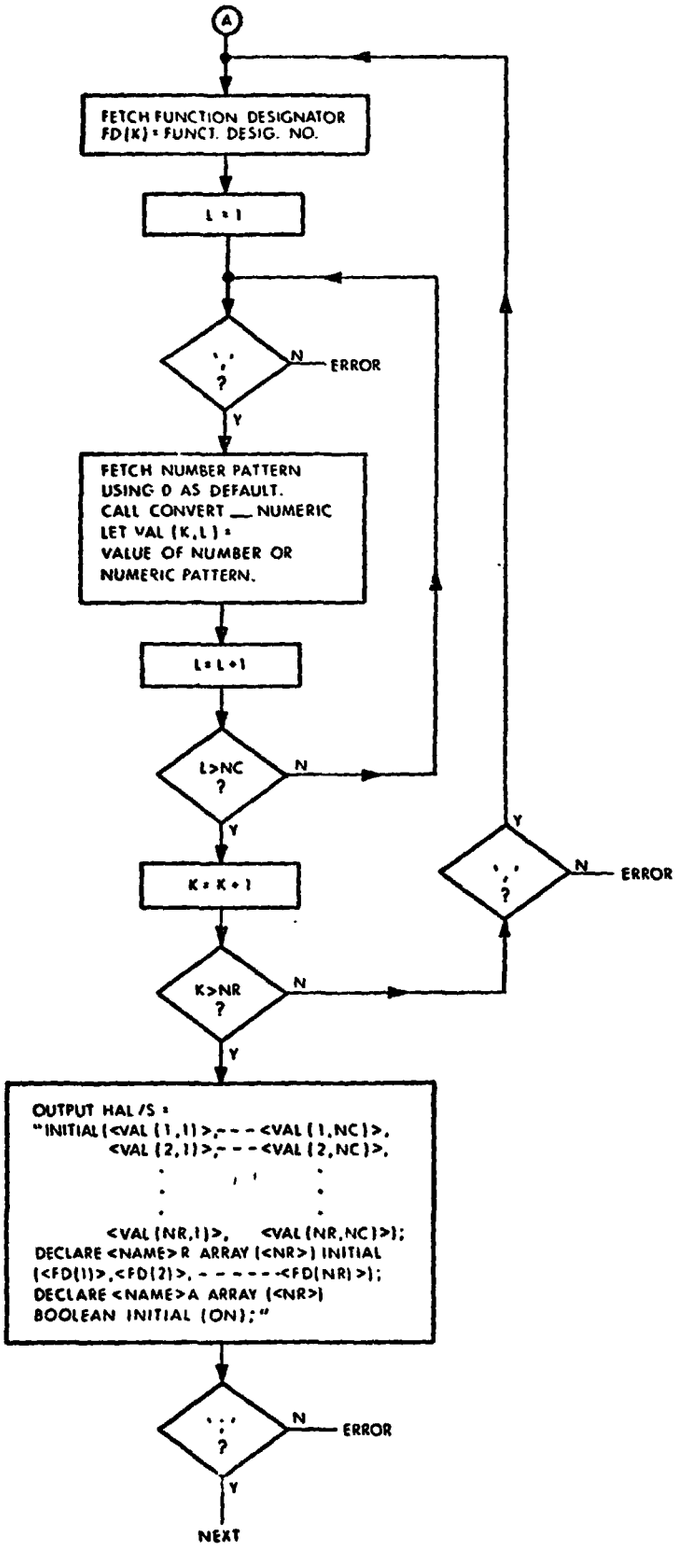


CHART 2 OF 2

DECLARE QUANTITY TABLE STATEMENT (21)

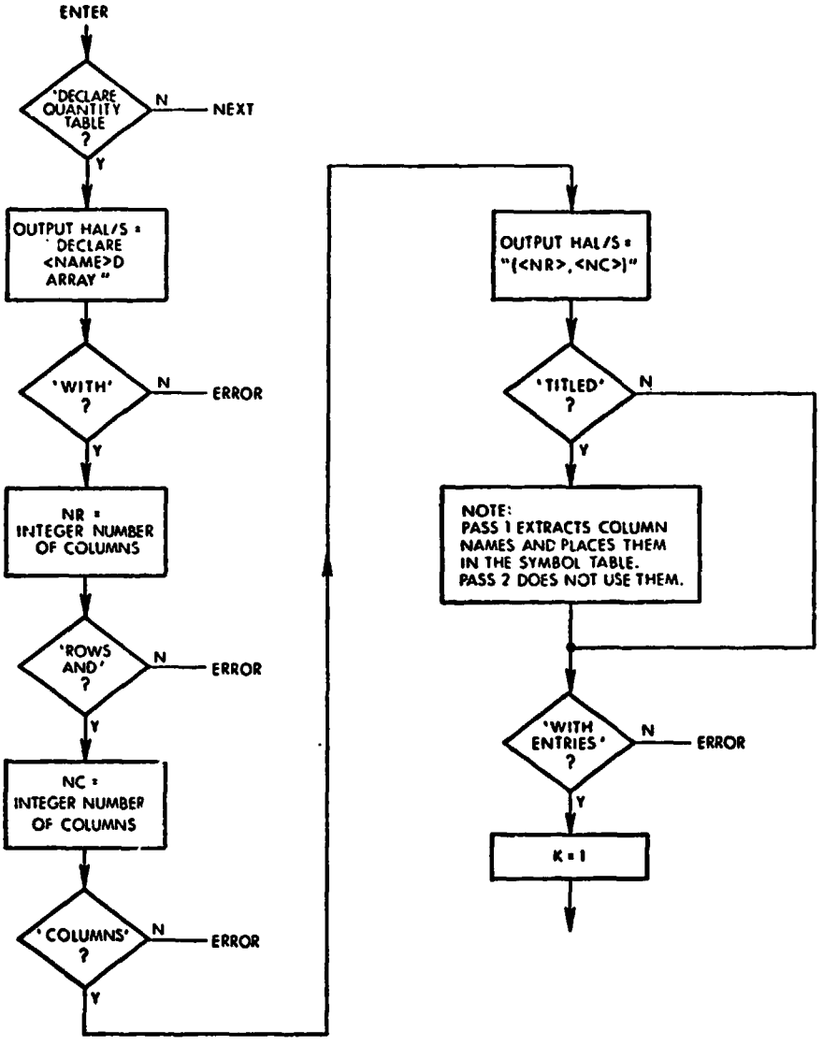


CHART 1 OF 2

DECLARE QUANTITY TABLE STATEMENT (21), (CONT.)

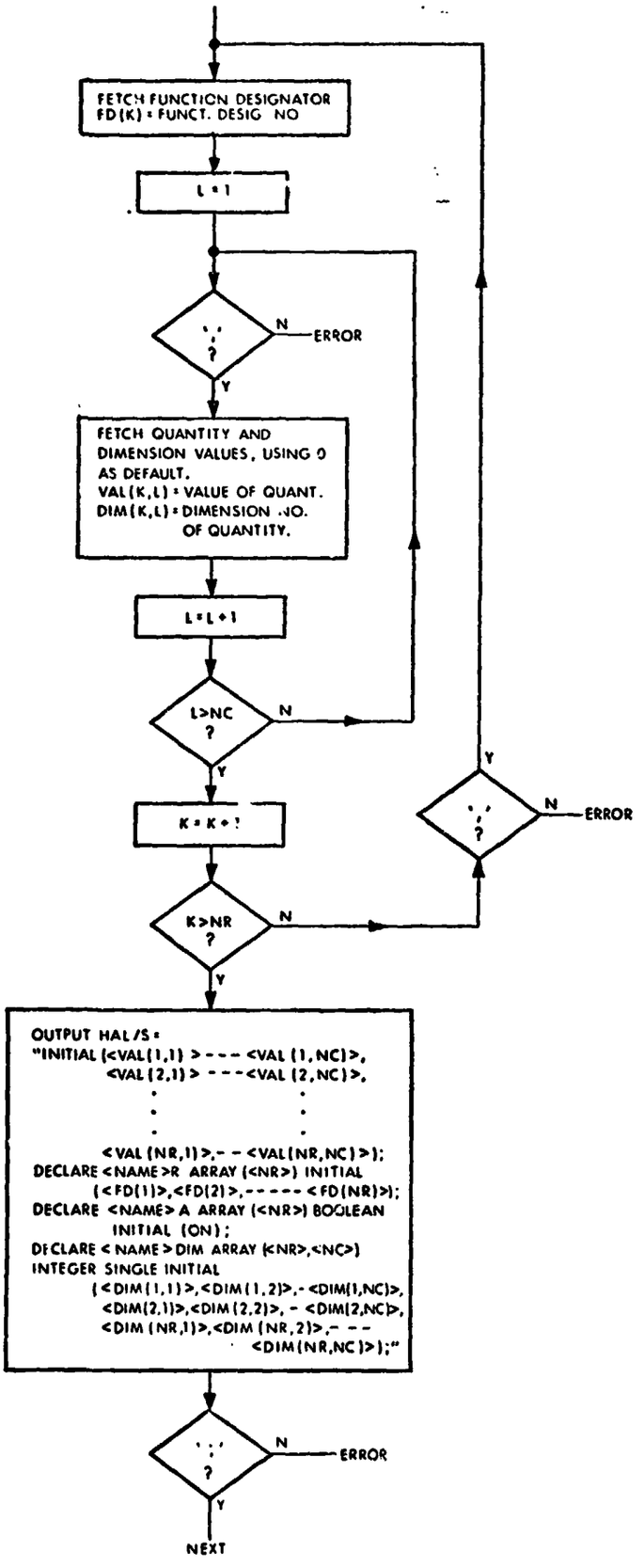


CHART 2 OF 2

DECLARE STATE TABLE STATEMENT (23)

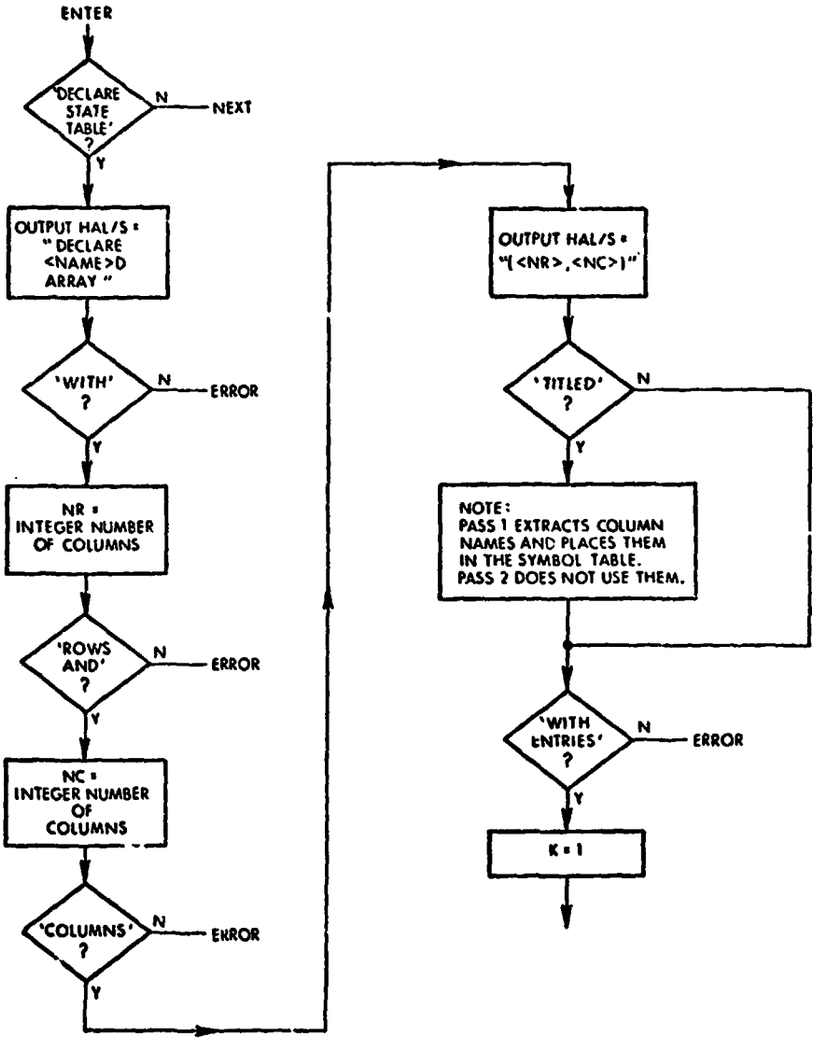


CHART 1 OF 2

DECLARE STATE TABLE STATEMENT (23), (CONT.)

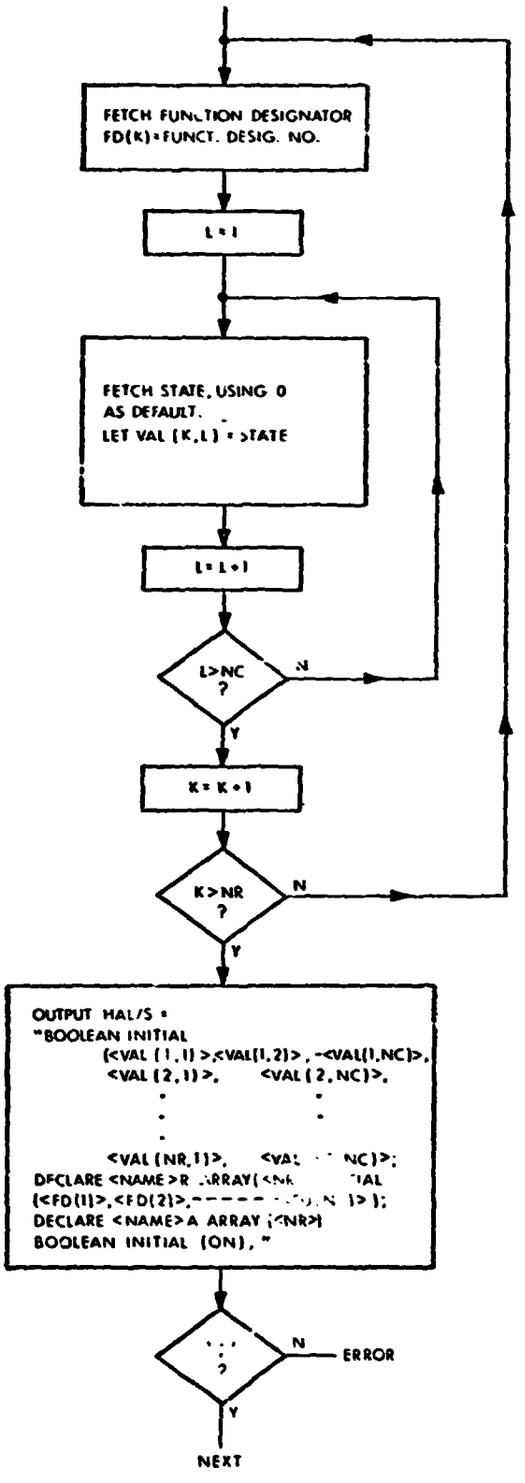


CHART 2 OF 2

DECLARE TEXT TABLE STATEMENT (25)

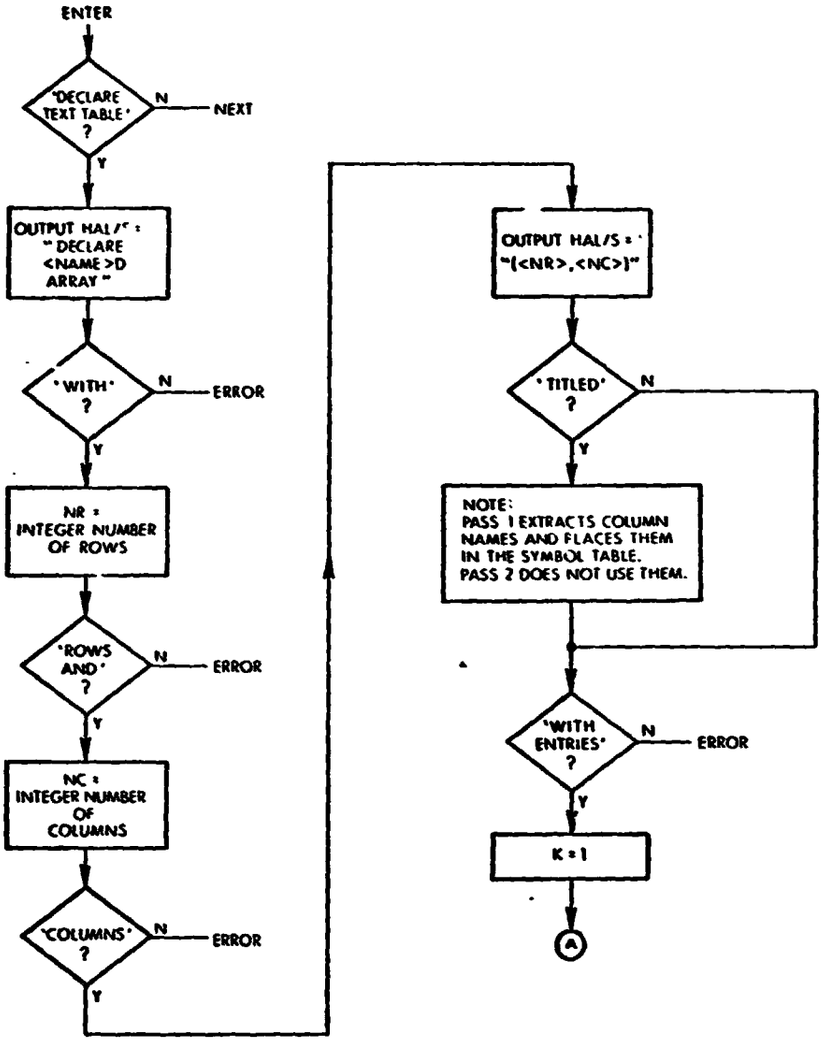


CHART 1 OF 2

DECLARE TEXT TABLE STATEMENT (25),(CONT.)

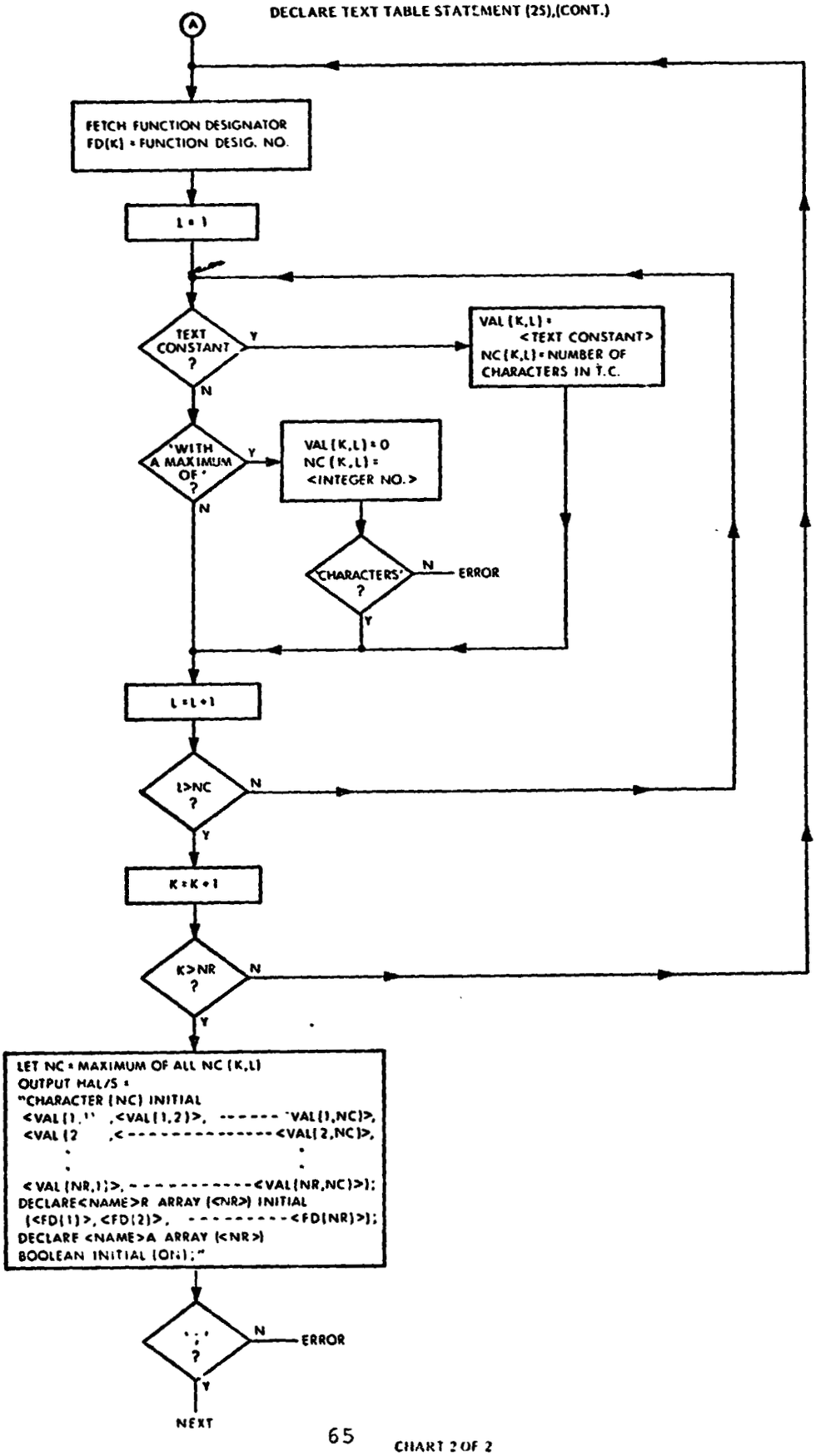
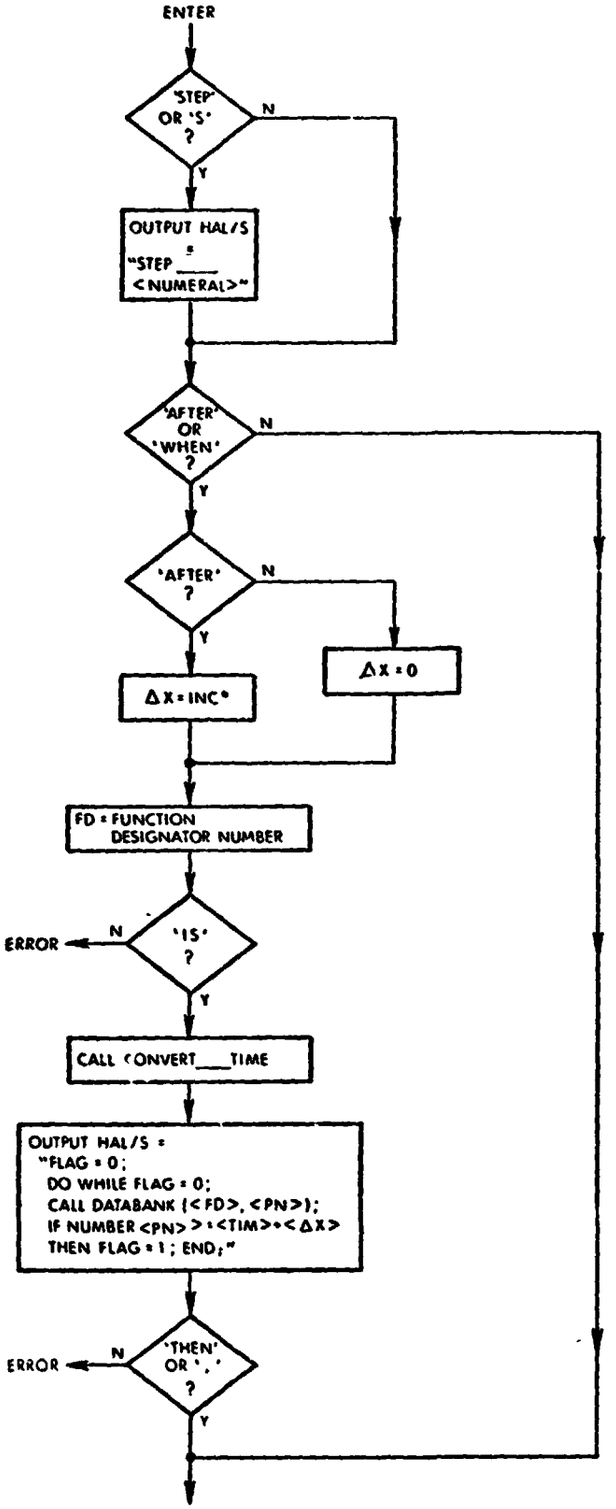


CHART 2 OF 2

THIS PAGE INTENTIONALLY LEFT BLANK.

5.3.2 PROCEDURAL STATEMENTS

PROCEDURAL STATEMENT PREFIX -
STEP NUMBER PREFIX (73),
TIME PREFIX (80), AND VERIFY PREFIX (83)



*INC = FCOS - DEPENDENT INCREMENT OF TIME.

CHART 1 OF 6

PROCEDURAL STATEMENT PREFIX (CONT.)

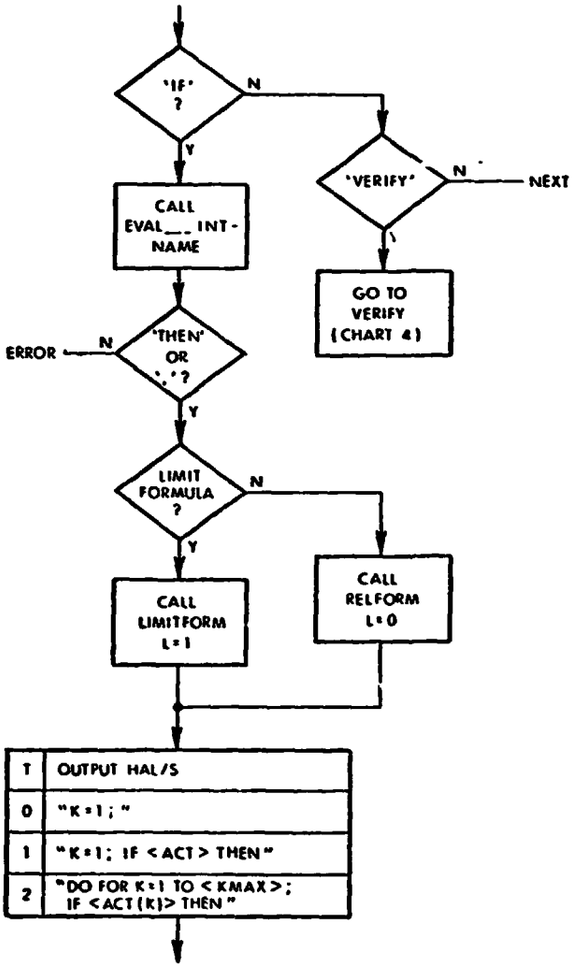


CHART 2 OF 6

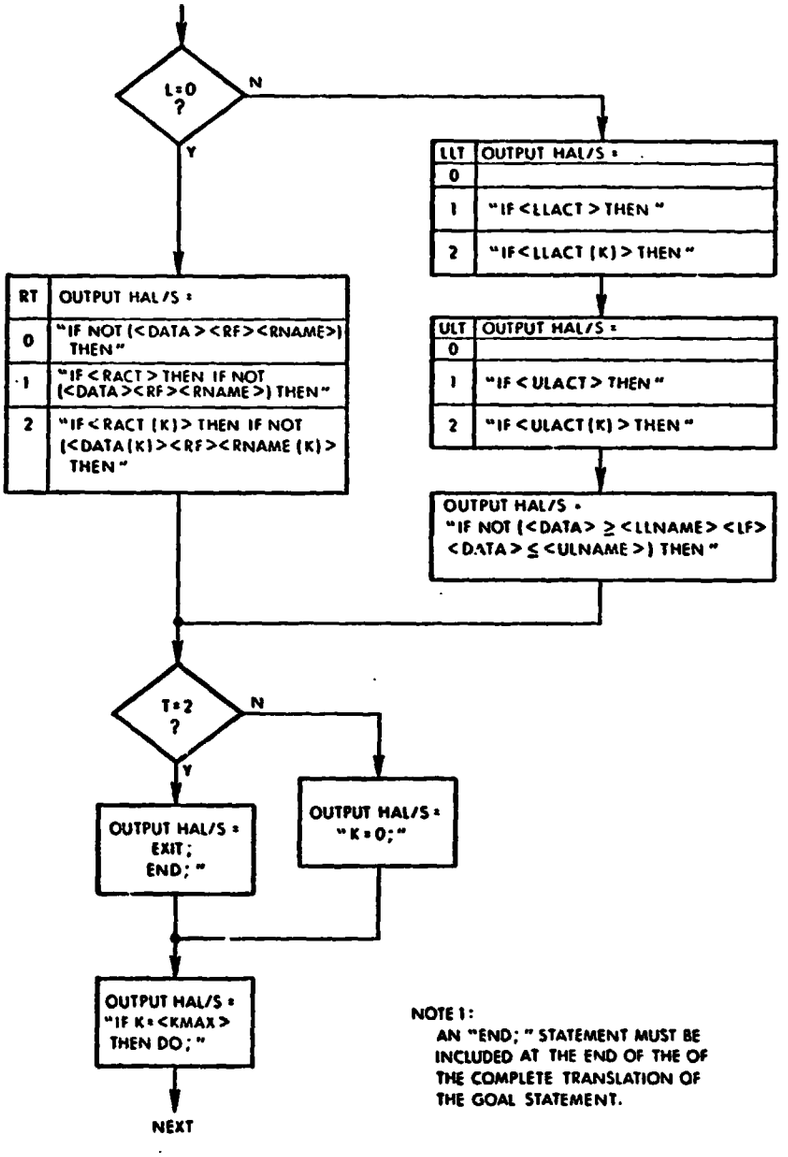

L=0 ?
N
Y
LLT | OUTPUT HAL/S =
0
1 | "IF <LLACT> THEN "
2 | "IF<LLACT(K)> THEN "
RT | OUTPUT HAL/S =
0 | "IF NOT (<DATA><RF><RNAME>) THEN"
1 | "IF <RACT> THEN IF NOT (<DATA><RF><RNAME>) THEN"
2 | "IF <RACT(K)> THEN IF NOT (<DATA(K)><RF><RNAME(K)> THEN"
ULT | OUTPUT HAL/S =
0
1 | "IF <ULACT> THEN"
2 | "IF <ULACT(K)> THEN"
OUTPUT HAL/S = "IF NOT (<DATA> ≥ <LLNAME> <LF> <DATA> ≤ <ULNAME>) THEN"
T=2 ?
N
Y
OUTPUT HAL/S = EXIT; END;"
OUTPUT HAL/S = "K=0;"
OUTPUT HAL/S = "IF K=<KMAX> THEN DO;"
NEXT
NOTE 1:
AN "END;" STATEMENT MUST BE INCLUDED AT THE END OF THE OF THE COMPLETE TRANSLATION OF THE GOAL STATEMENT.

CHART 3 OF 6

PROCEDURAL STATEMENT PREFIX (CONT.)

VERIFY PREFIX (83)

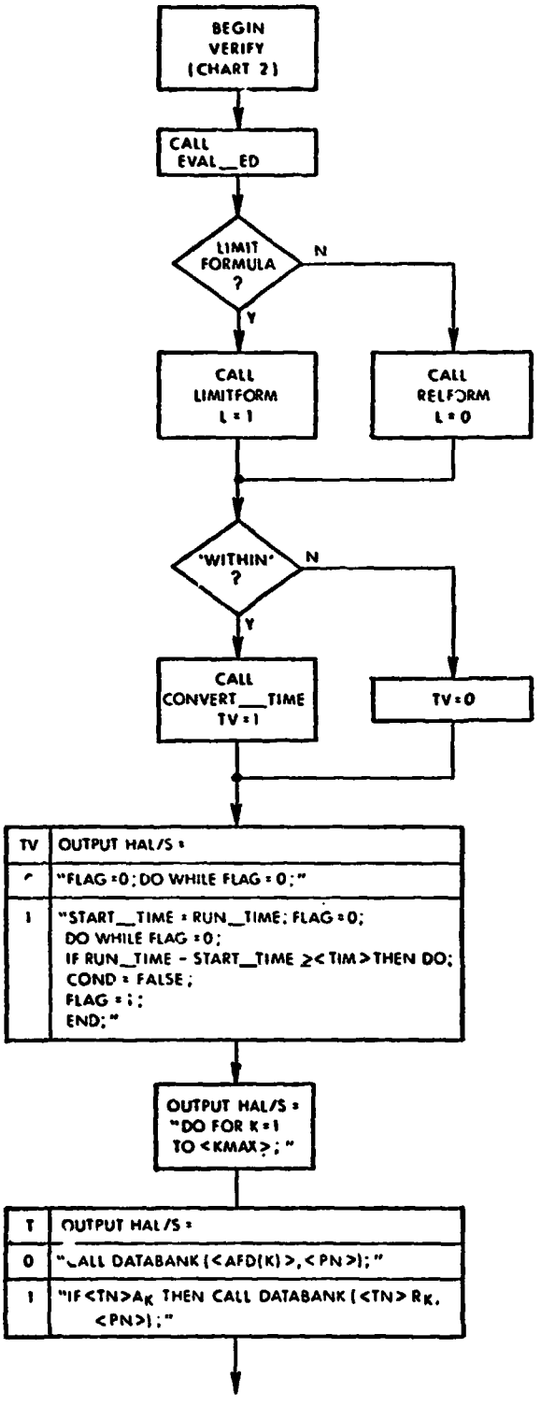


CHART 4 OF 6

PROCEDURAL STATEMENT PREFIX (CONT.)

VERIFY PREFIX (83), (CONT.)

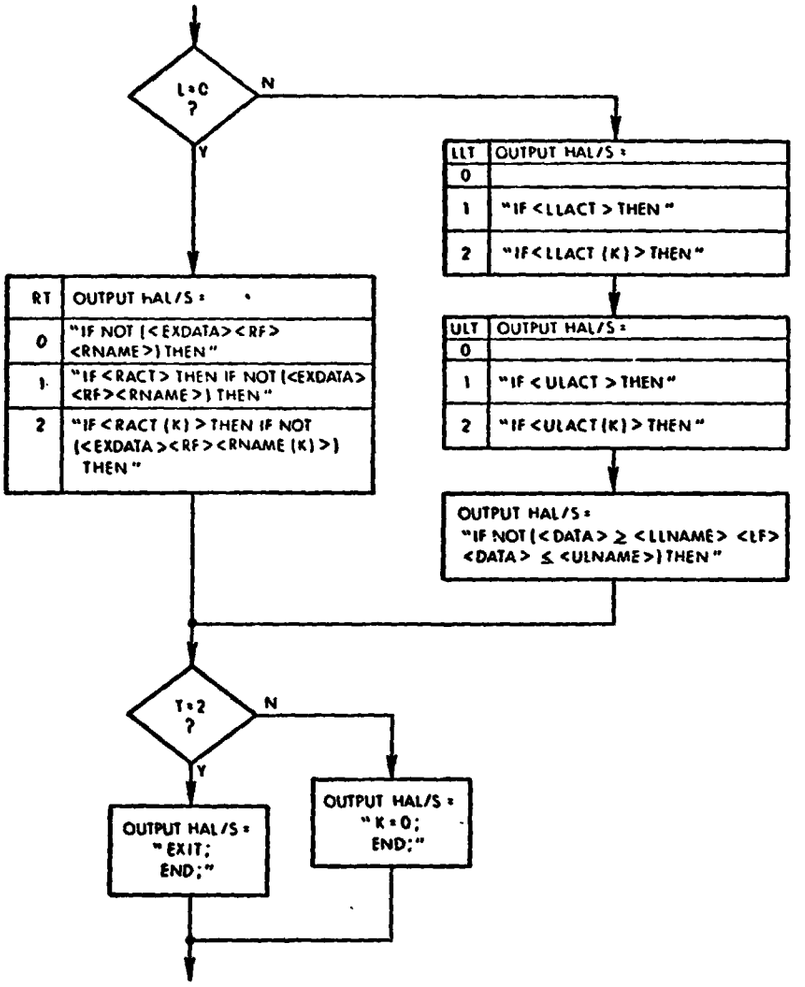


CHART 5 OF 6

PROCEDURAL STATEMENT PREFIX (CONCLUDED)

VERIFY PREFIX (83) , (CONCLUDED)

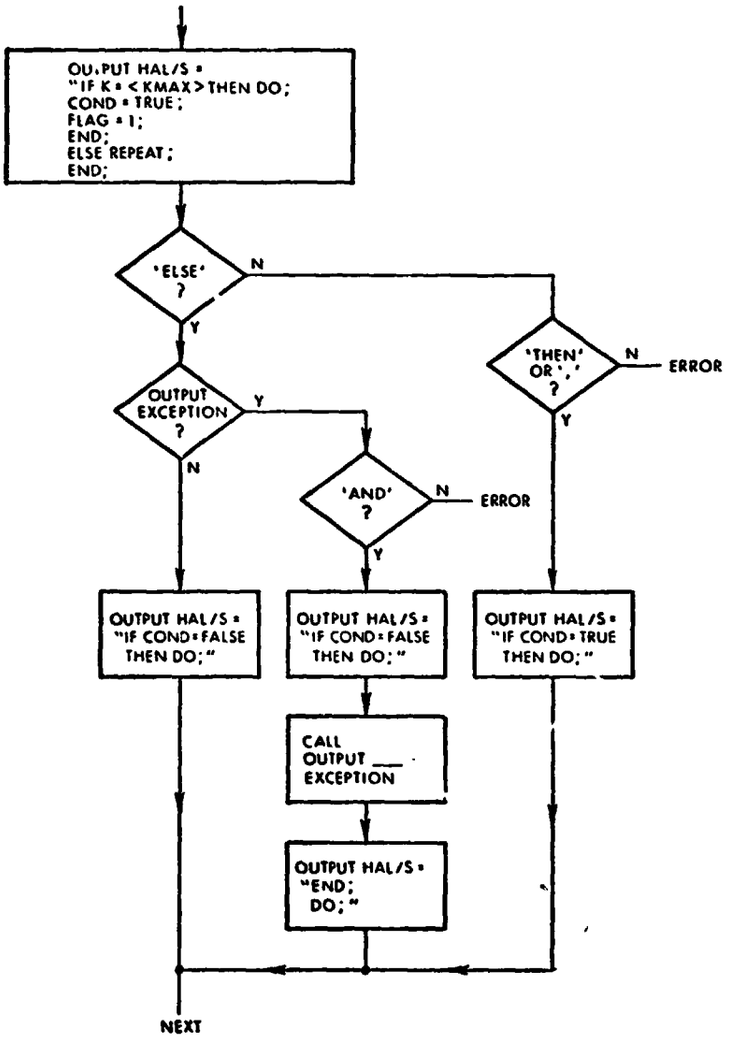


CHART 6 OF 6

ISSUE, SEND, OR APPLY ANALOG STATEMENT (40, 2)

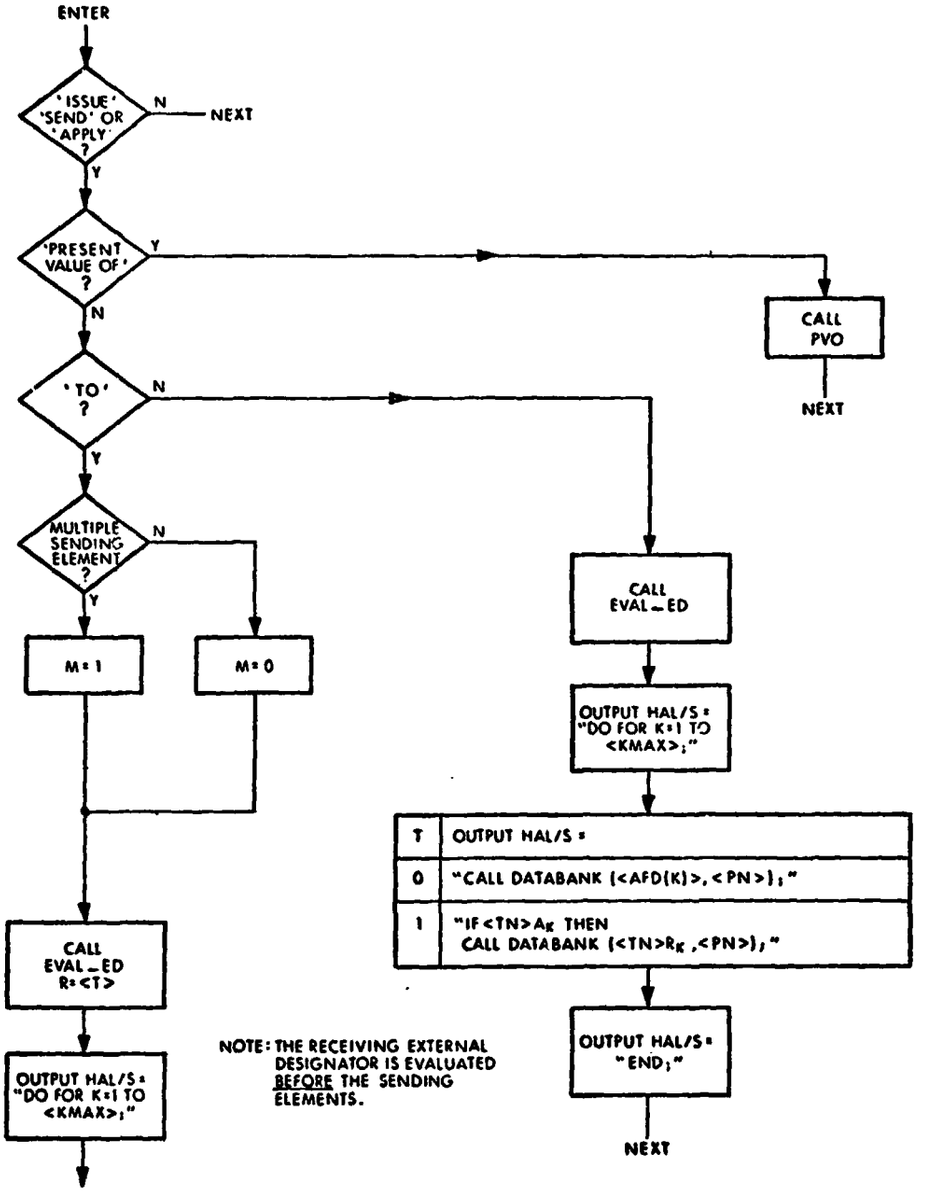
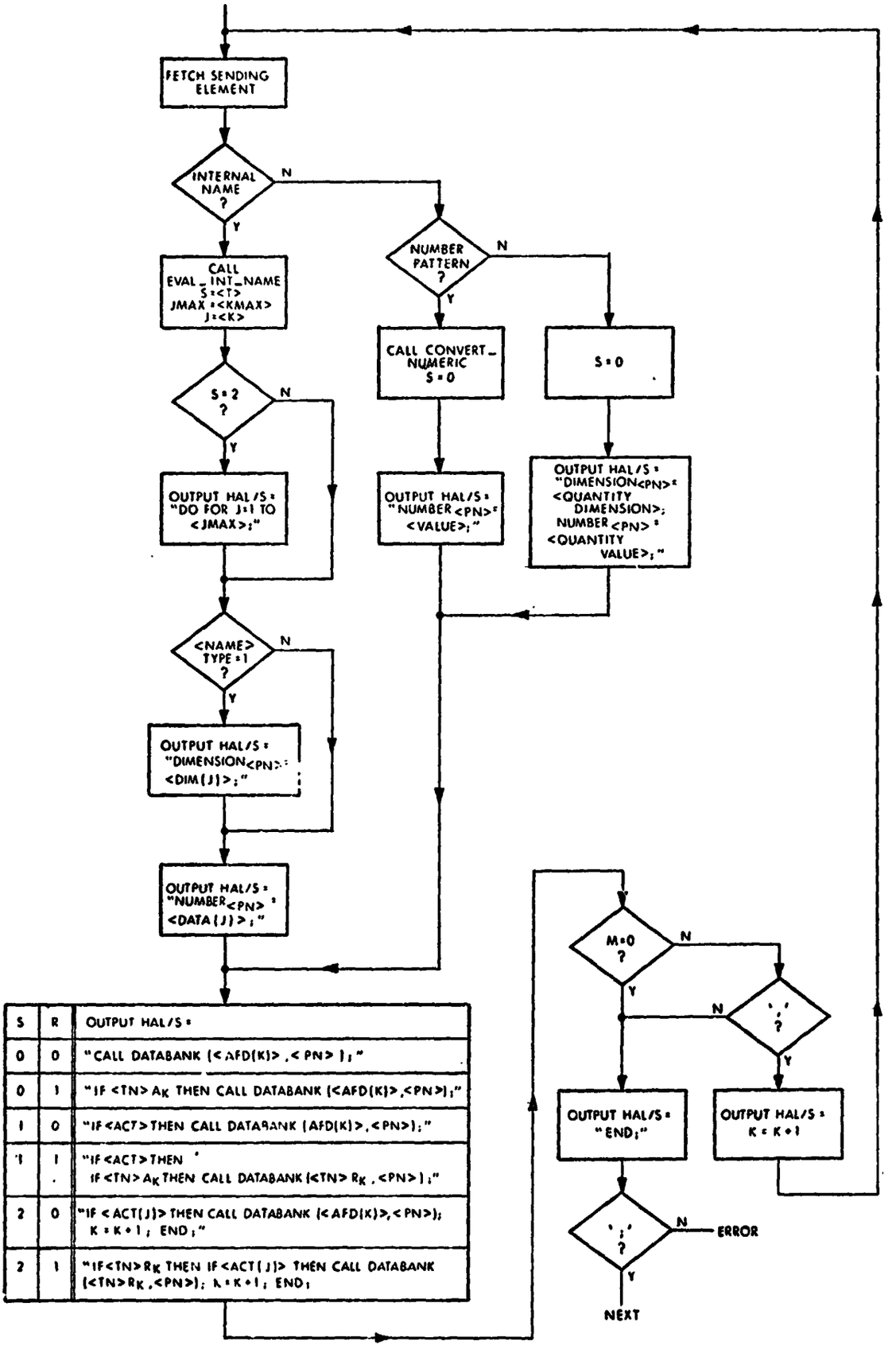
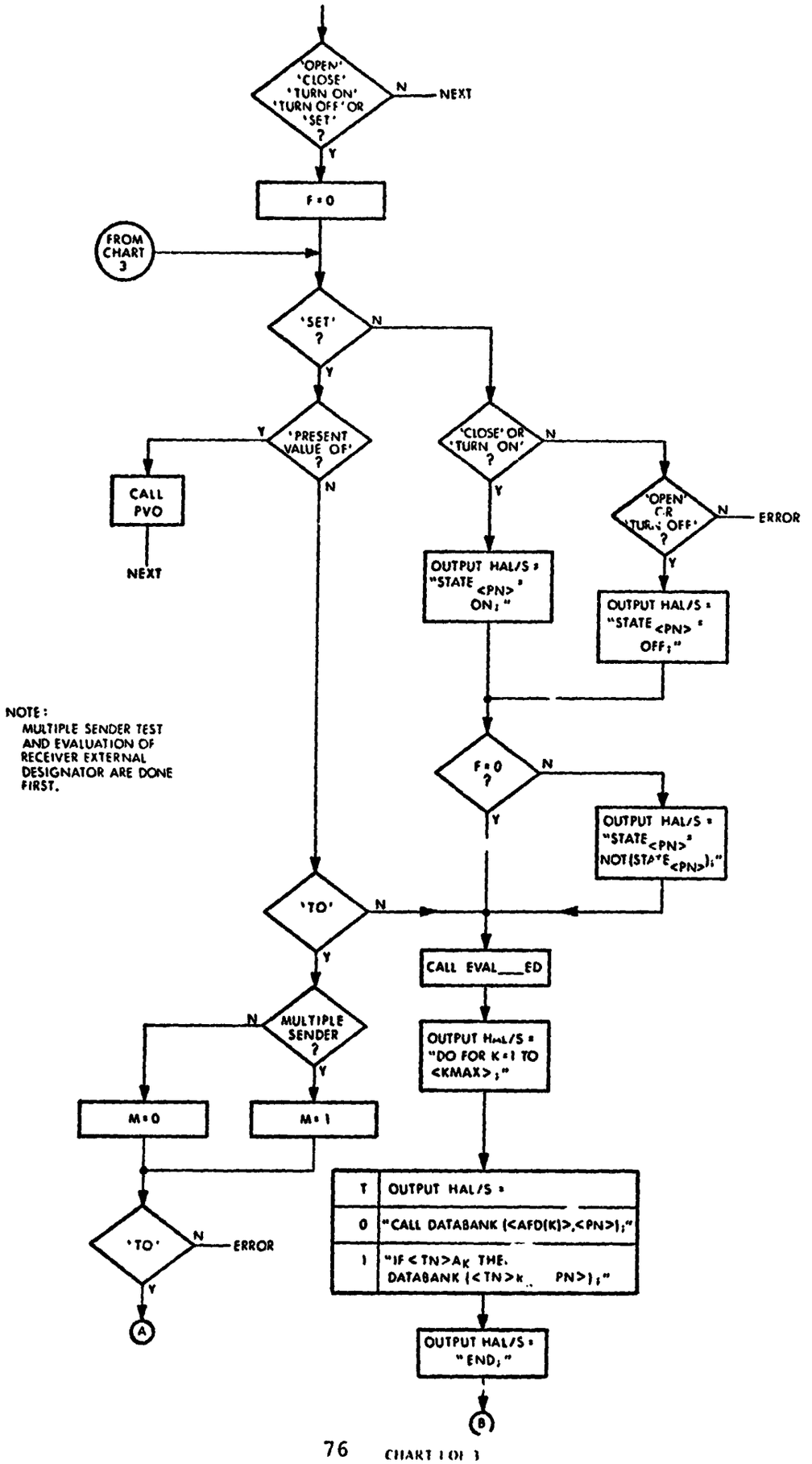


CHART 1 OF 2



S	R	OUTPUT HAL/S =
0	0	"CALL DATABANK (<AFD(K)>,<PN>);"
0	1	"IF <TN> A_K THEN CALL DATABANK (<AFD(K)>,<PN>);"
1	0	"IF <ACT> THEN CALL DATABANK (AFD(K)>,<PN>);"
1	1	"IF <ACT> THEN IF <TN> A_K THEN CALL DATABANK (<TN> R_K ,<PN>);"
2	0	"IF <ACT(J)> THEN CALL DATABANK (<AFD(K)>,<PN>); K = K + 1; END;"
2	1	"IF <TN> R_K THEN IF <ACT(J)> THEN CALL DATABANK (<TN> R_K ,<PN>); K = K + 1; END;

CHART 2 OF 2


'OPEN' 'CLOSE' 'TURN ON' 'TURN OFF' OR 'SET' ?
N
NEXT
Y
F = 0
FROM CHART 3
'SET' ?
N
Y
'PRESENT VALUE OF' ?
Y
N
CALL PVO
NEXT
'CLOSE' OR 'TURN ON' ?
N
Y
'OPEN' OR 'TURN OFF' ?
N
ERROR
Y
OUTPUT HAL/S = "STATE <PN> ON;"
OUTPUT HAL/S = "STATE <PN> OFF;"
NOTE:
MULTIPLE SENDER TEST AND EVALUATION OF RECEIVER EXTERNAL DESIGNATOR ARE DONE FIRST.
F = 0 ?
N
Y
OUTPUT HAL/S = "STATE <PN> NOT(STATE <PN>);"
'TO'
N
Y
CALL EVAL___ED
MULTIPLE SENDER ?
N
Y
OUTPUT HAL/S = "DO FOR K = 1 TO <KMAX>;"
M = 0
M = 1
T
OUTPUT HAL/S =
0
"CALL DATABANK (<AFD(K)>,<PN>);"
1
"IF <TN>AK THE
DATABANK (<TN>K PN>);"
'TO'
N
ERROR
Y
A
OUTPUT HAL/S = "END;"
B

SET DISCRETE STATEMENT (70) , (CONT.)

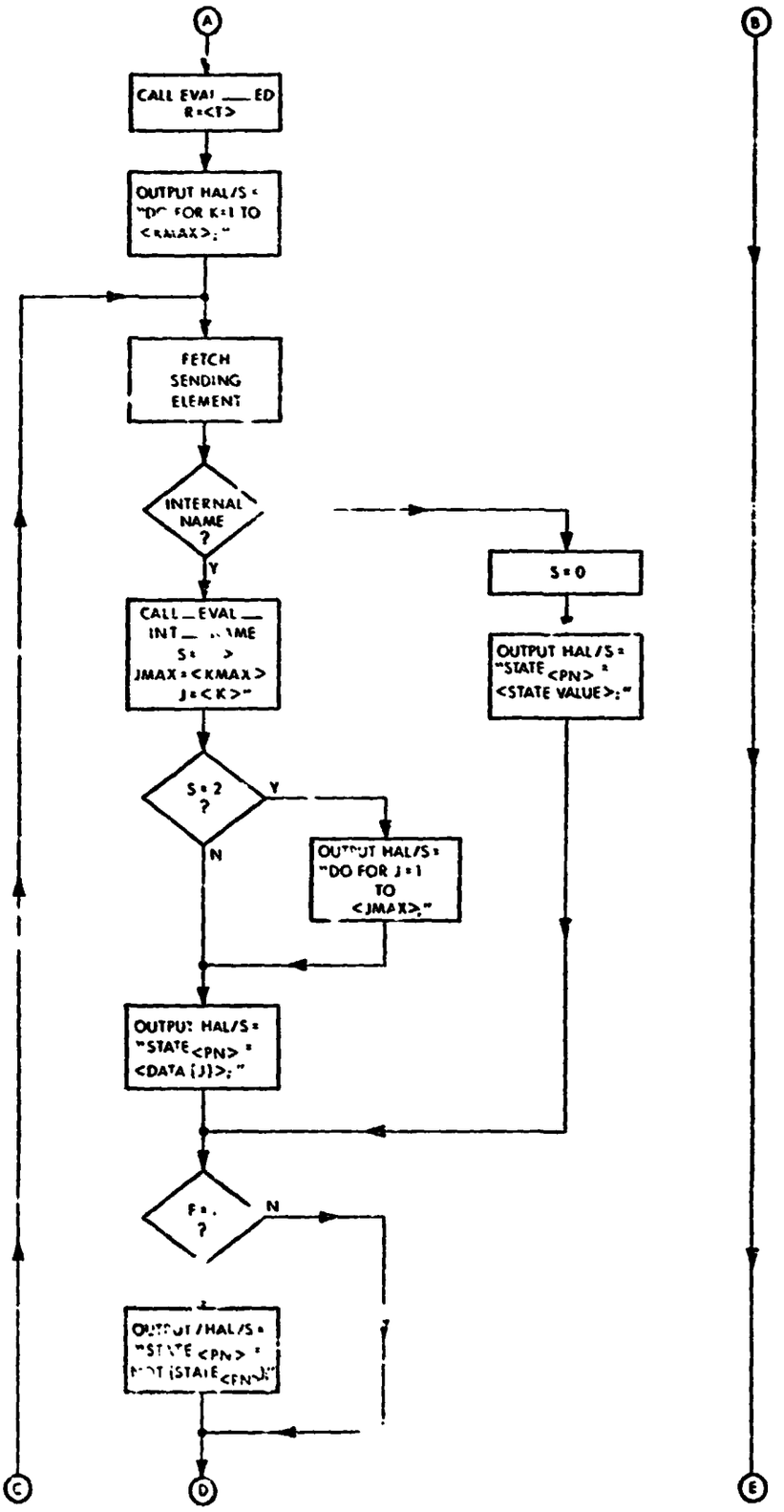
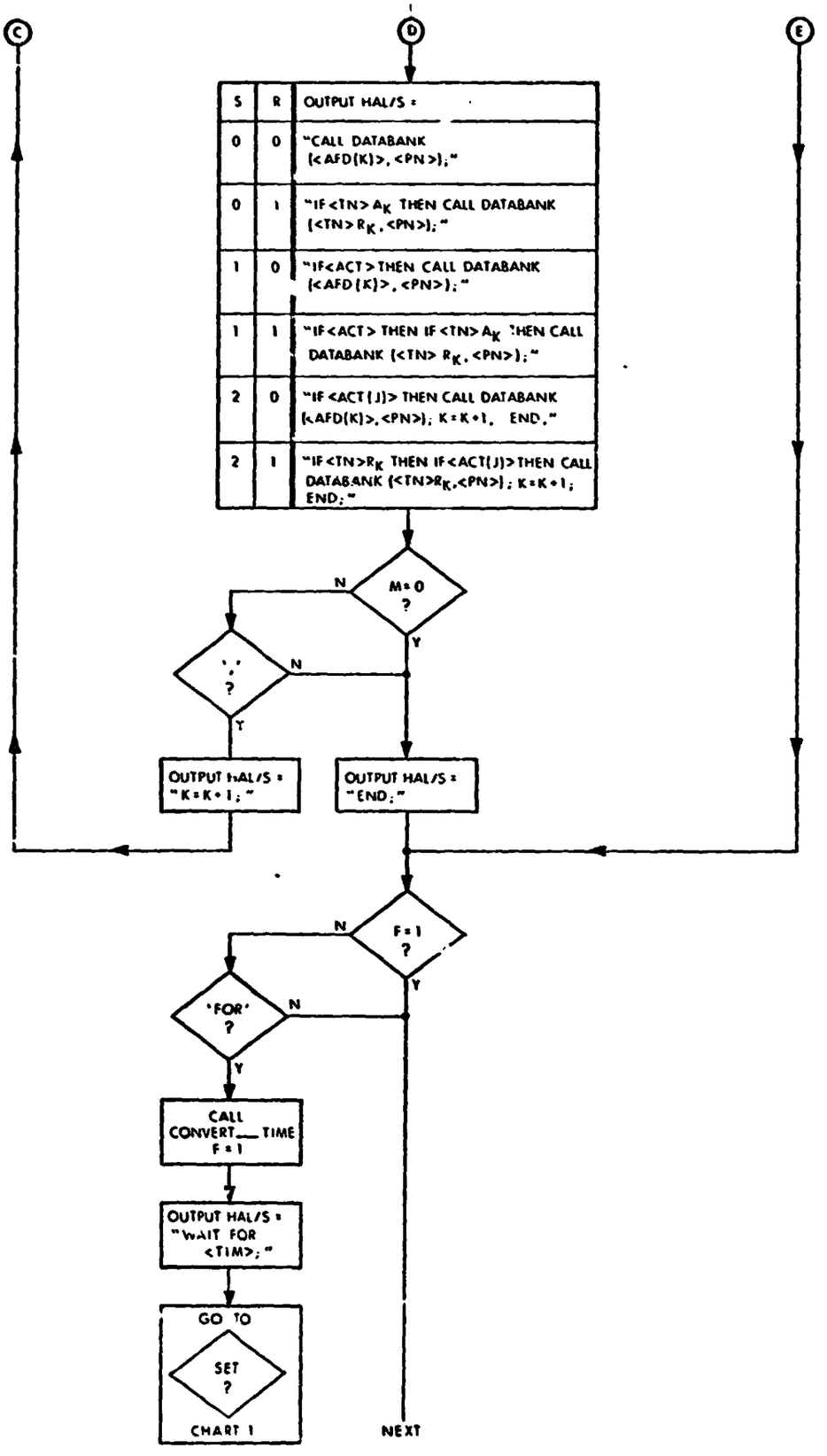


CHART 2 OF 3

SET DISCRETE STATEMENT (70) , (CONT.)



THIS PAGE INTENTIONALLY LEFT BLANK.

RECORD DATA STATEMENT (61)

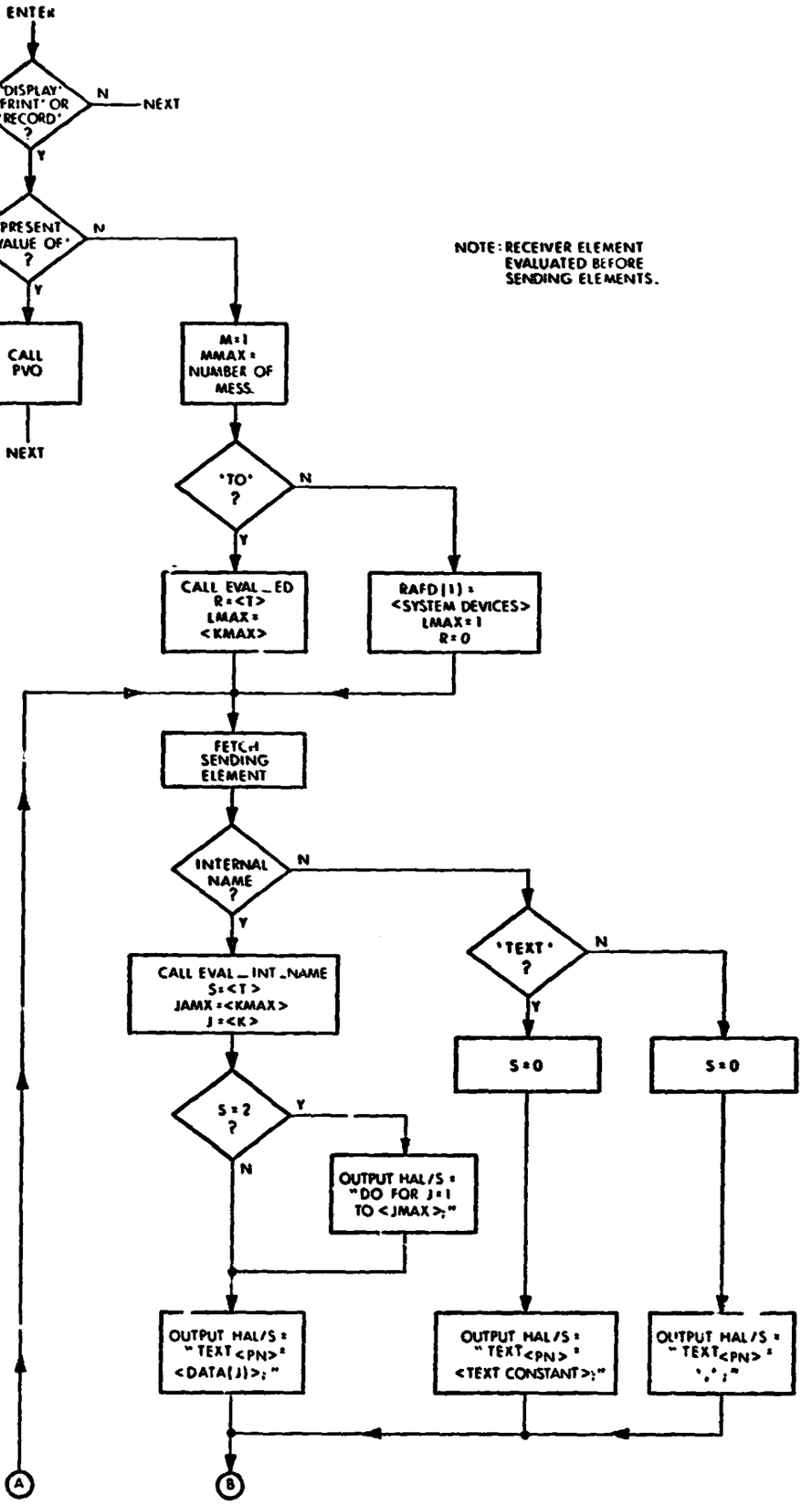


CHART 1 OF 2

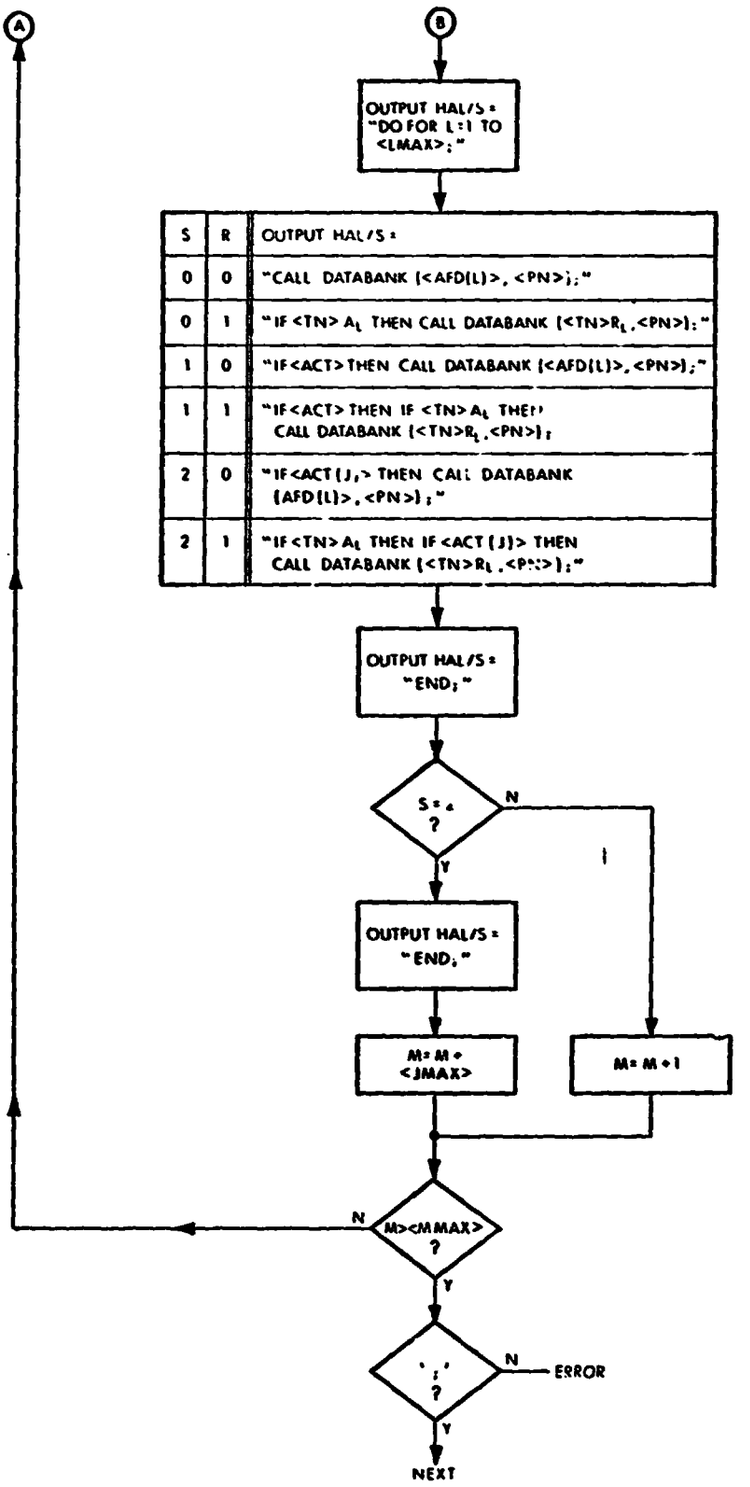

A
B
OUTPUT HAL/S =
"DO FOR L:=1 TO
<LMAX>;"
S
R
OUTPUT HAL/S =
0
0
"CALL DATABANK (<AFD(L)>, <PN>);"
0
1
"IF <TN> A_L THEN CALL DATABANK (<TN>R_L, <PN>);"
1
0
"IF <ACT> THEN CALL DATABANK (<AFD(L)>, <PN>);"
1
1
"IF <ACT> THEN IF <TN> A_L THEN
CALL DATABANK (<TN>R_L, <PN>);
2
0
"IF <ACT(J)> THEN CALL DATABANK
(AFD(L)>, <PN>);"
2
1
"IF <TN> A_L THEN IF <ACT(J)> THEN
CALL DATABANK (<TN>R_L, <PN>);"
OUTPUT HAL/S =
"END;"
S = 4
?
N
Y
OUTPUT HAL/S =
"END;"
M = M +
<JMAX>
M = M + 1
M><MMAX>
?
N
Y
';'
?
N
ERROR
Y
NEXT

CHART 2 OF 2

· AVERAGE STATEMENT (4)

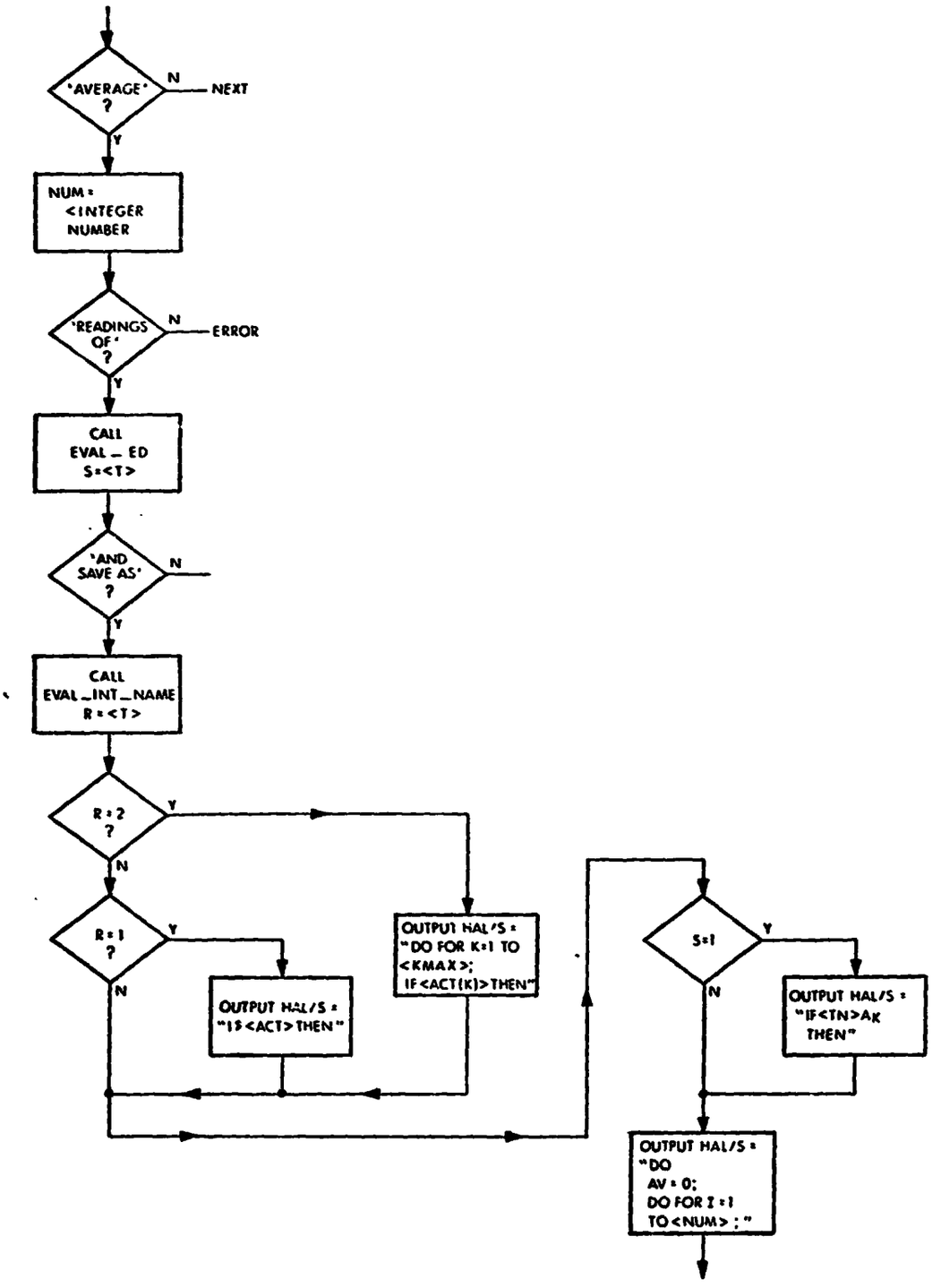


CHART 1 OF 2

AVERAGE STATEMENT (4), (CONT.)

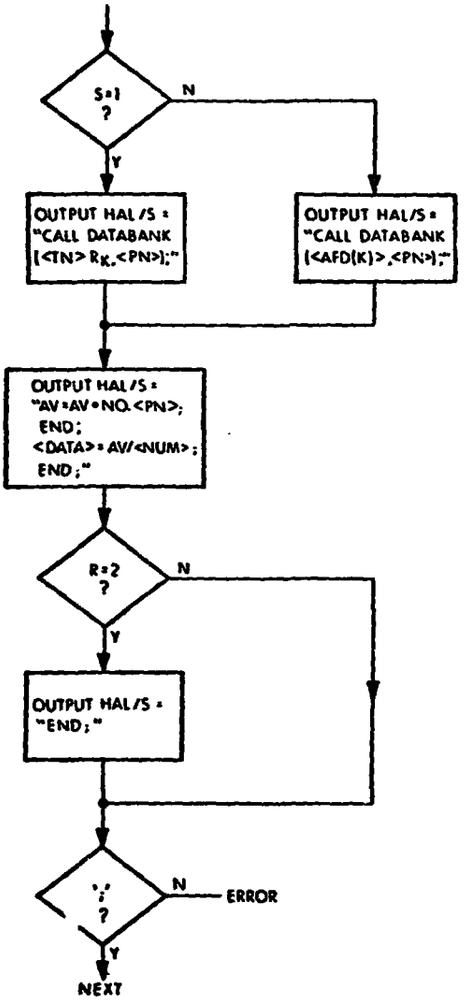


CHART 2 OF 2

READ PROCEDURAL STATEMENT (60)

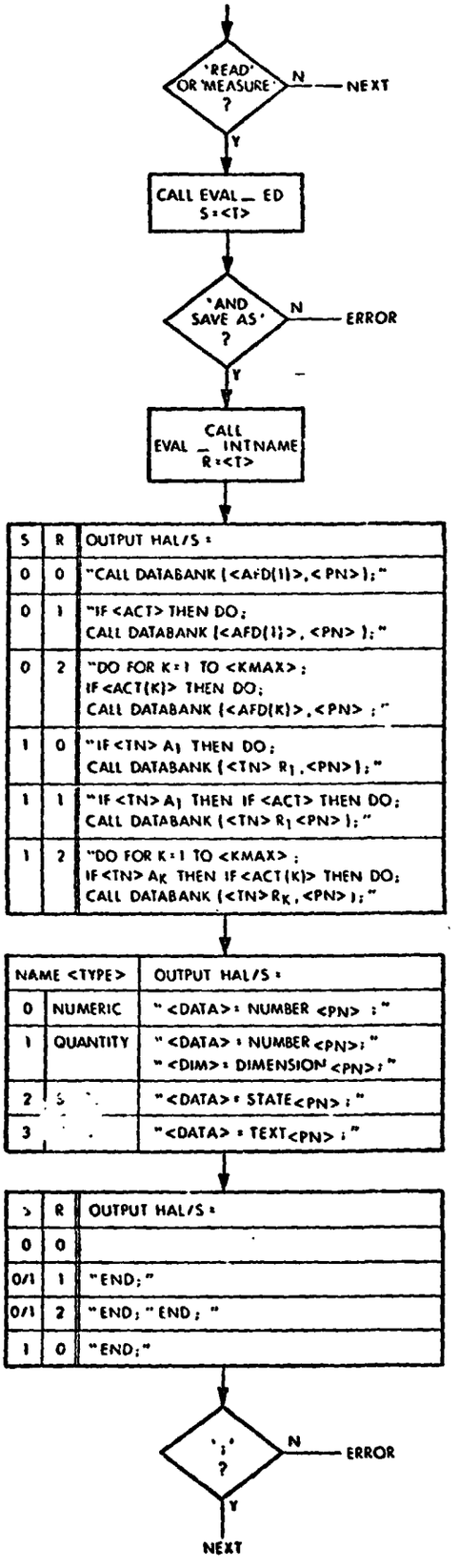
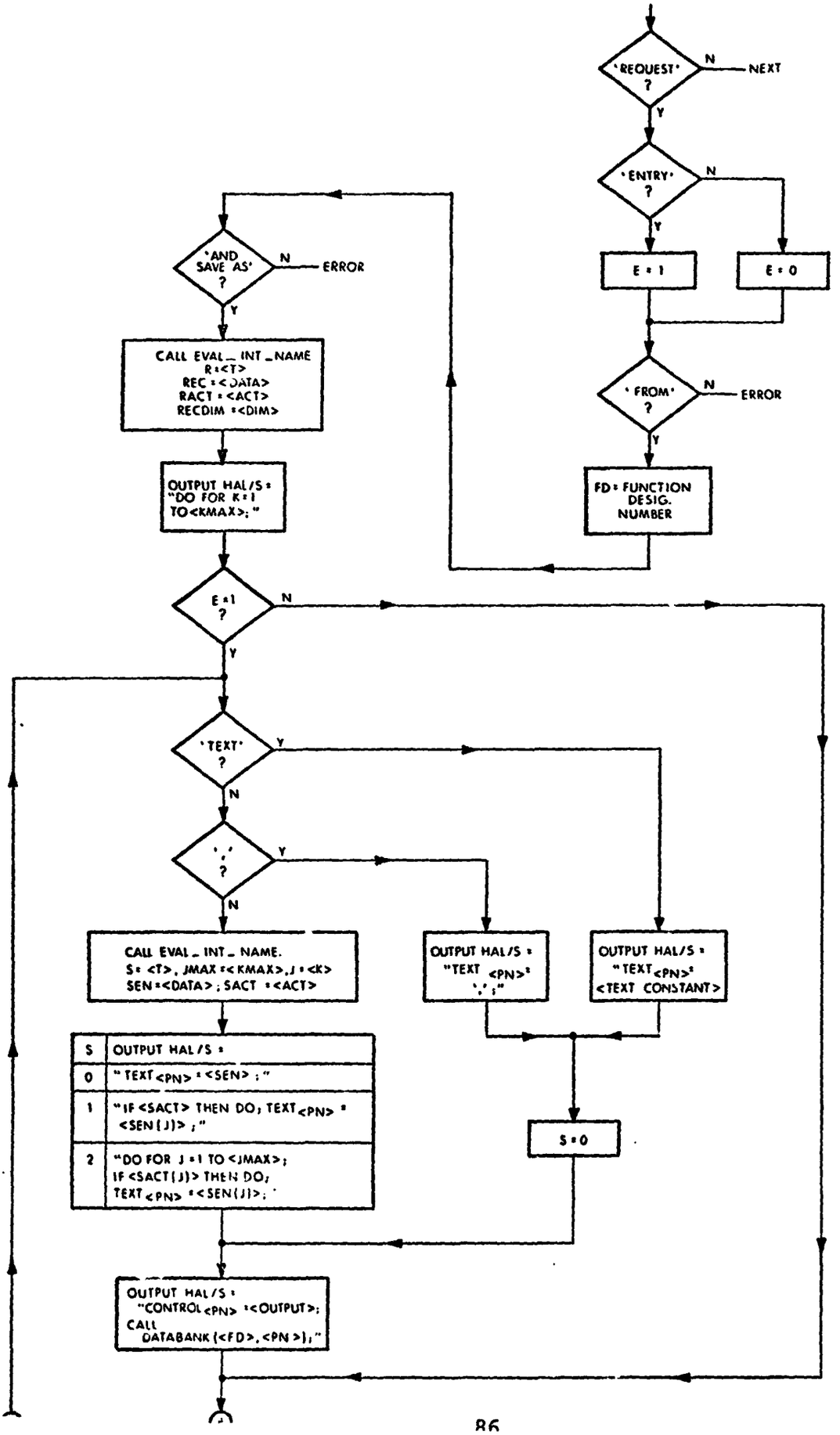


CHART 1 OF 1

THIS PAGE INTENTIONALLY LEFT BLANK.



S	OUTPUT HAL/S =
0	"TEXT<PN> = <SEN>;"
1	"IF <SACT> THEN DO; TEXT<PN> = <SEN(J)>;"
2	"DO FOR J=1 TO <JMAX>; IF <SACT(J)> THEN DO; TEXT<PN> = <SEN(J)>;

REQUEST KEYBOARD STATEMENT (66) , (CONT.)

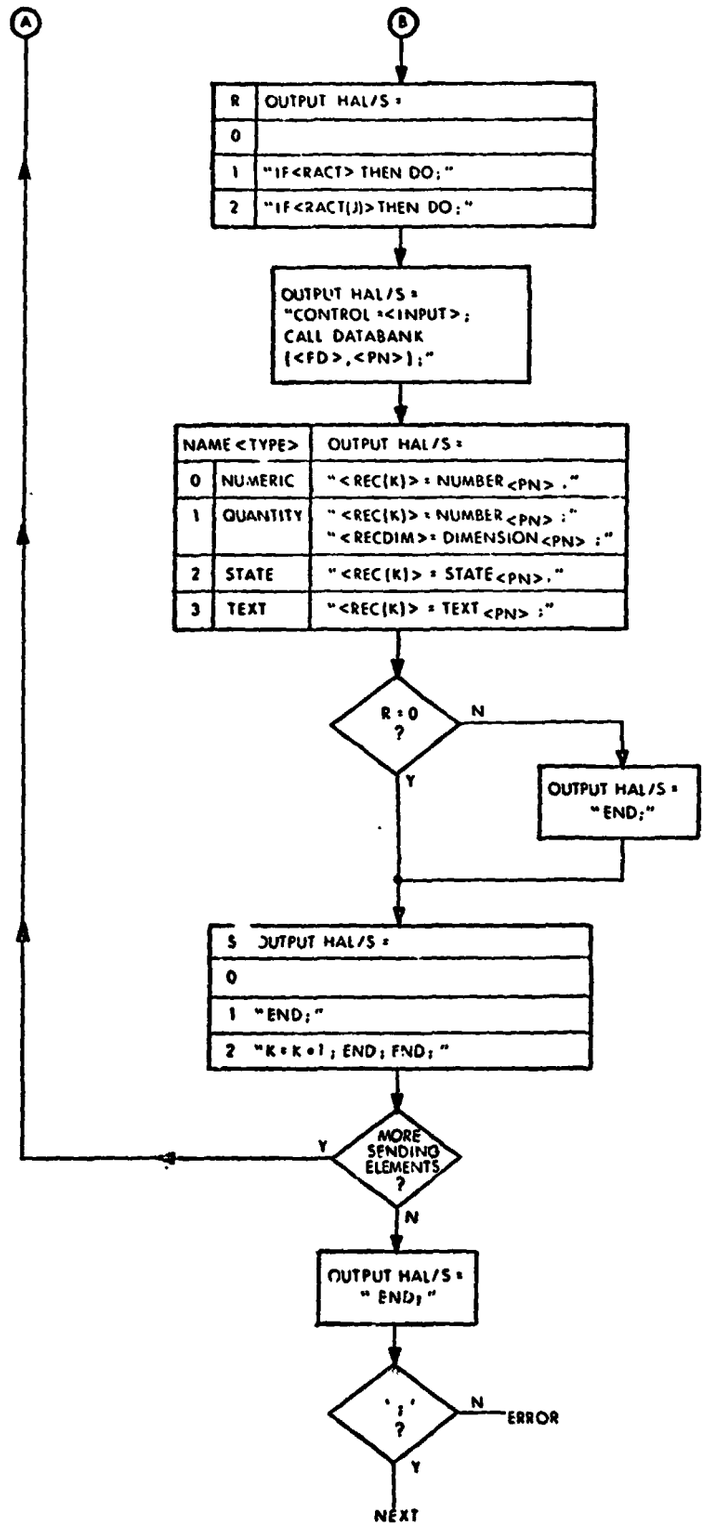


CHART 2 OF 2

"DELAY" PROCEDURAL STATEMENT (26)

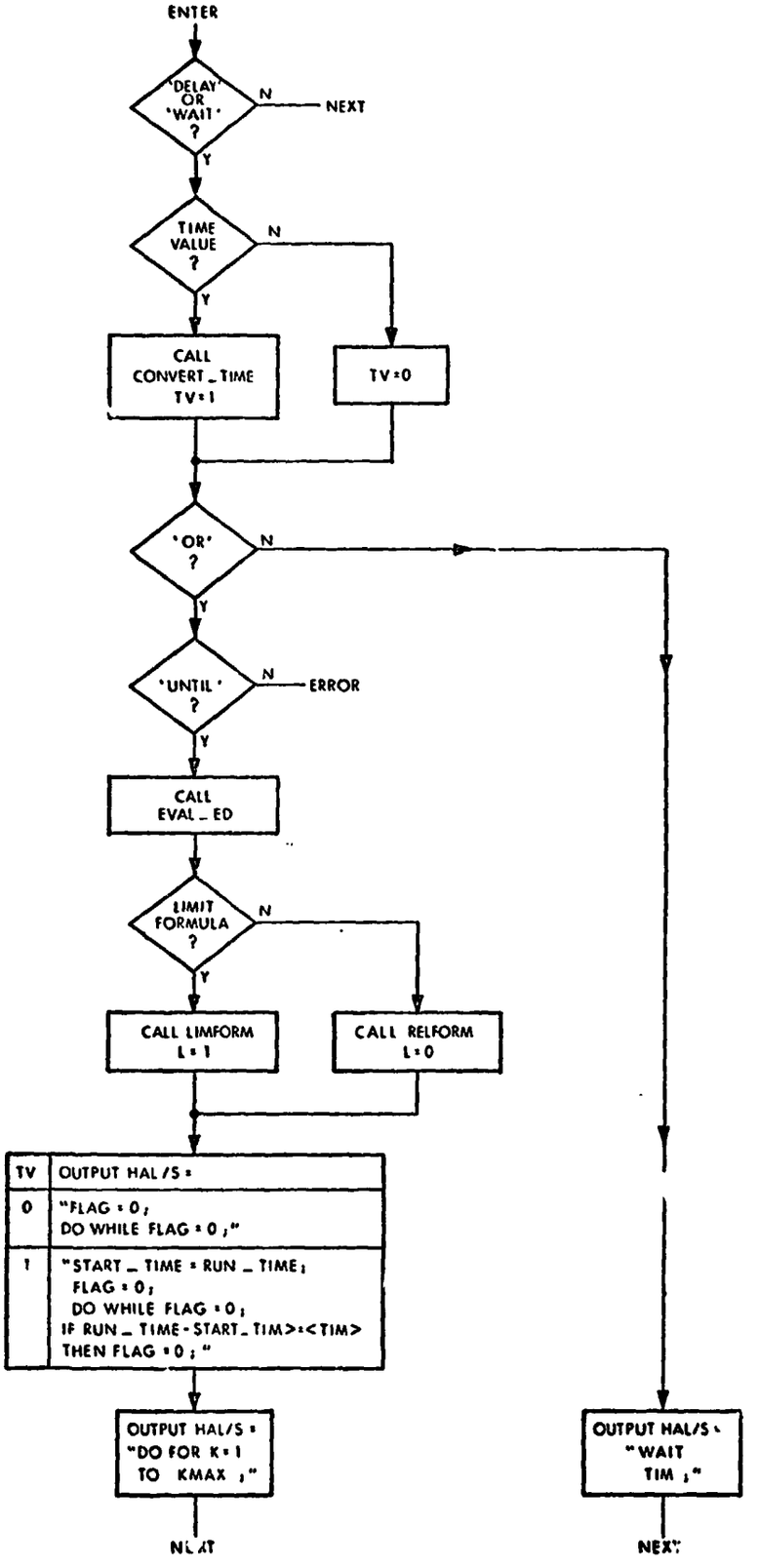


CHART 1 OF 2

"DELAY" PROCEDURAL STATEMENT (26) , (CONT.)

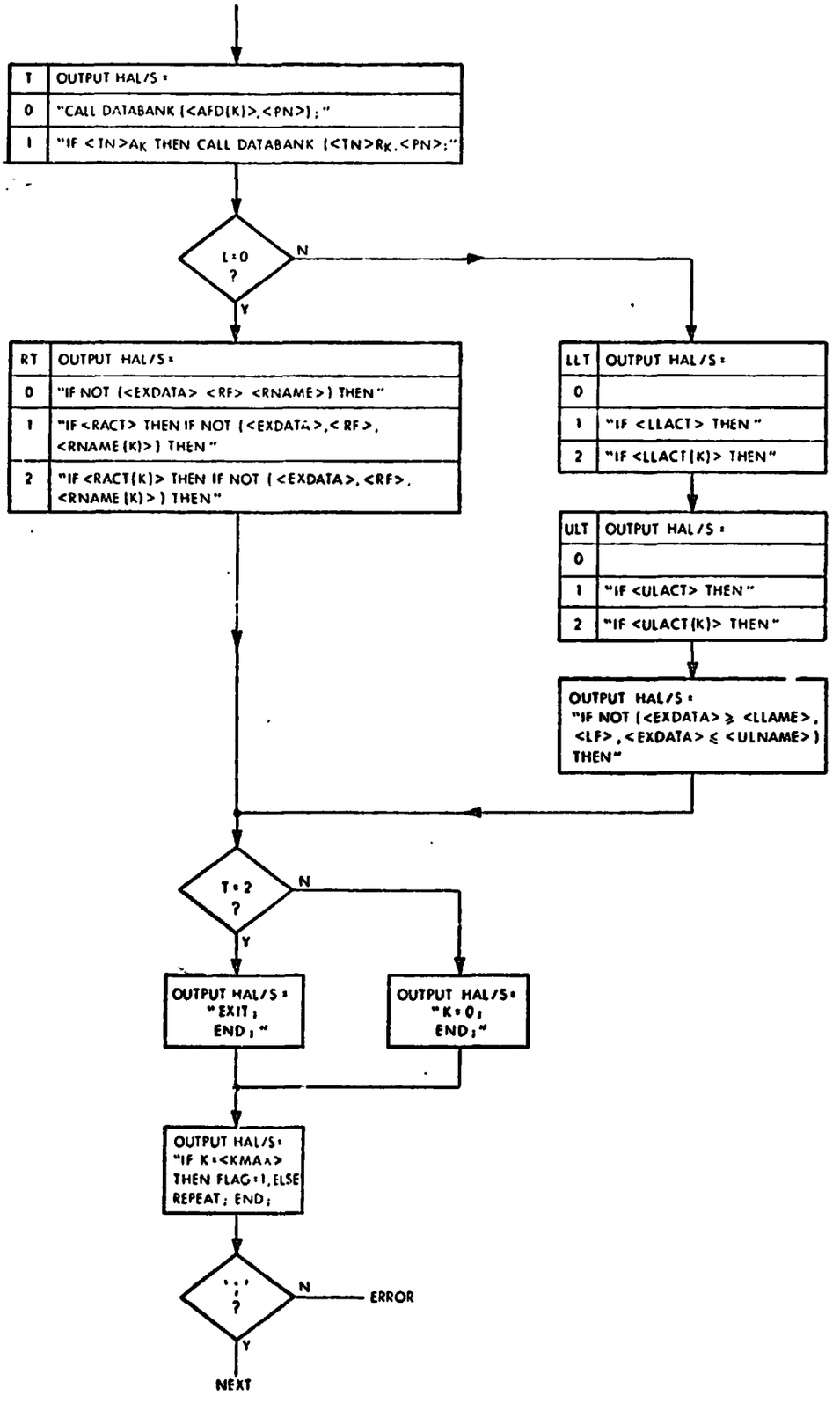


CHART 2 OF 2

"GO TO" PROCEDURAL STATEMENT (34)

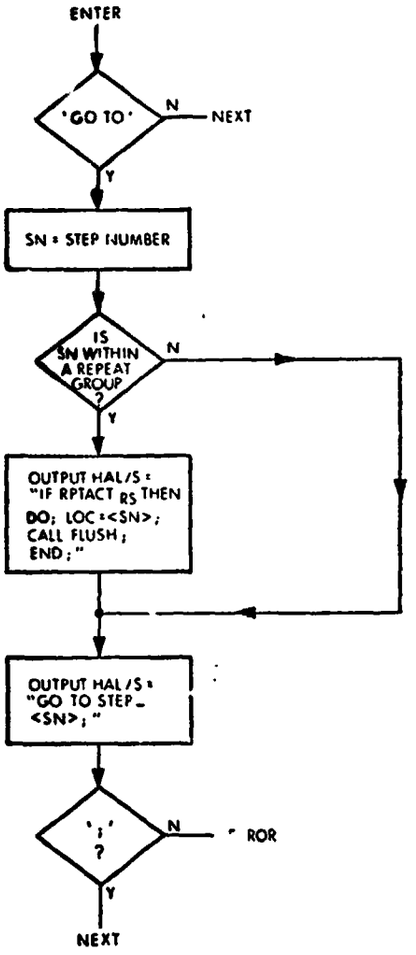
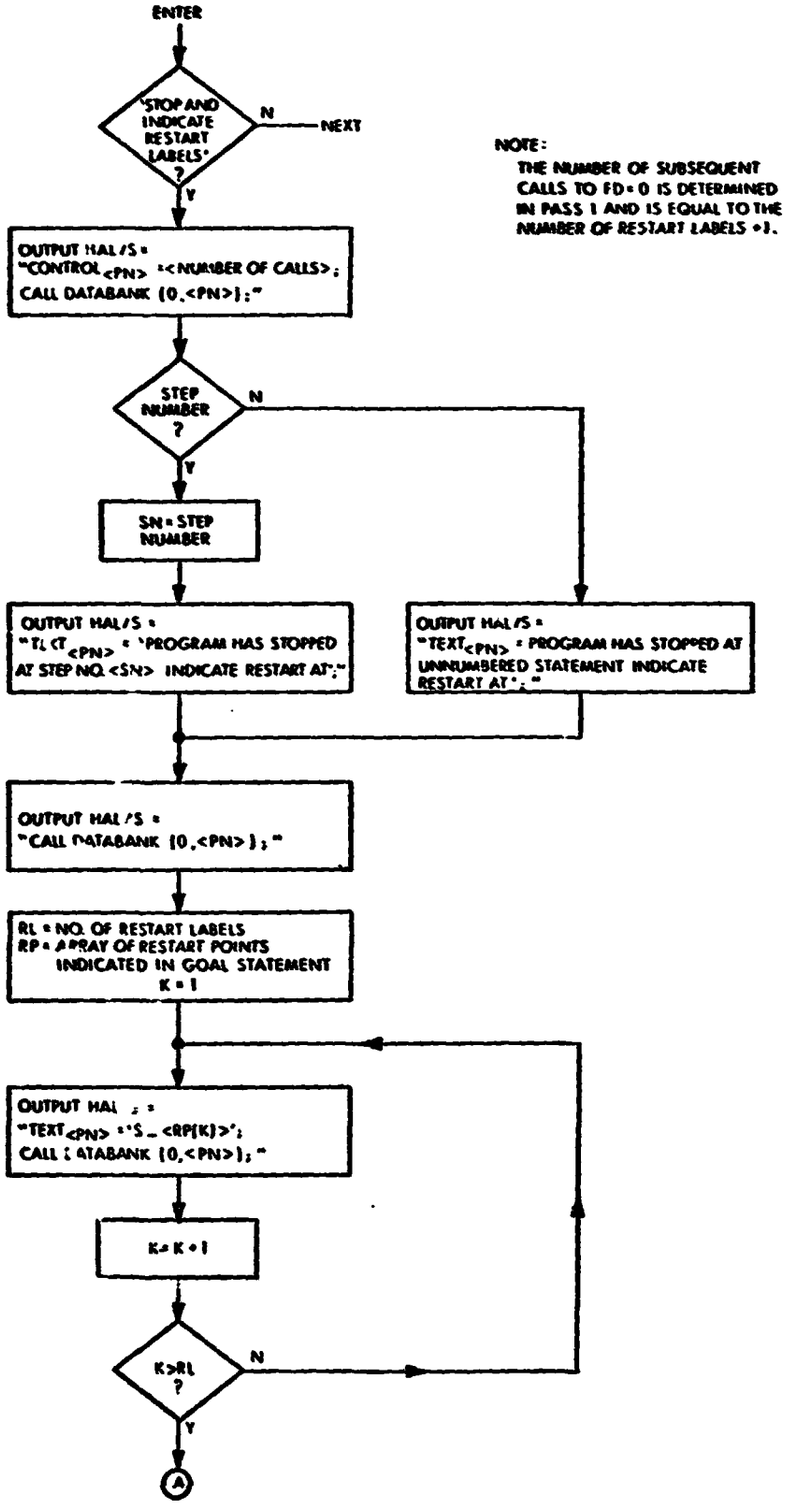


CHART 1 OF 1

THIS PAGE INTENTIONALLY LEFT BLANK.

C-2

"STOP" PROCEDURAL STATEMENT (74)



NOTE:
THE NUMBER OF SUBSEQUENT CALLS TO FD = 0 IS DETERMINED IN PASS 1 AND IS EQUAL TO THE NUMBER OF RESTART LABELS + 1.

CHART 1 OF 2

"STOP" PROCEDURAL STATEMENT (74) , (CONT.)

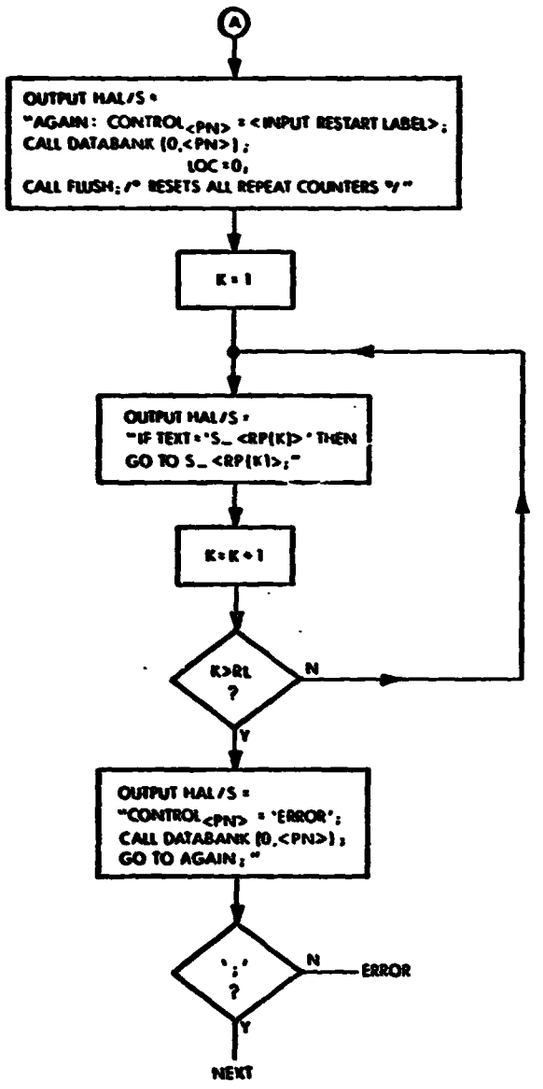
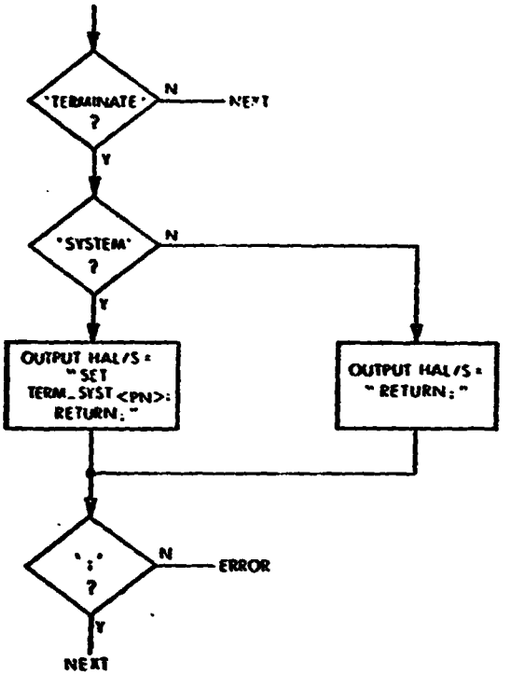


CHART 2 OF 2

"TERMINATE"
PROCEDURAL STATEMENT
(78)



NOTE: TERM_SYST<PN> IS AN ARRAY OF DECLARED BOOLEANS, ONE BIT PER PROGRAM. IT'S SIZE IS DETERMINED BY THE MAXIMUM LENGTH OF PROGRAMS ALLOWED.

CHART 1 OF 1

"REPEAT" PROCEDURAL STATEMENT (64)

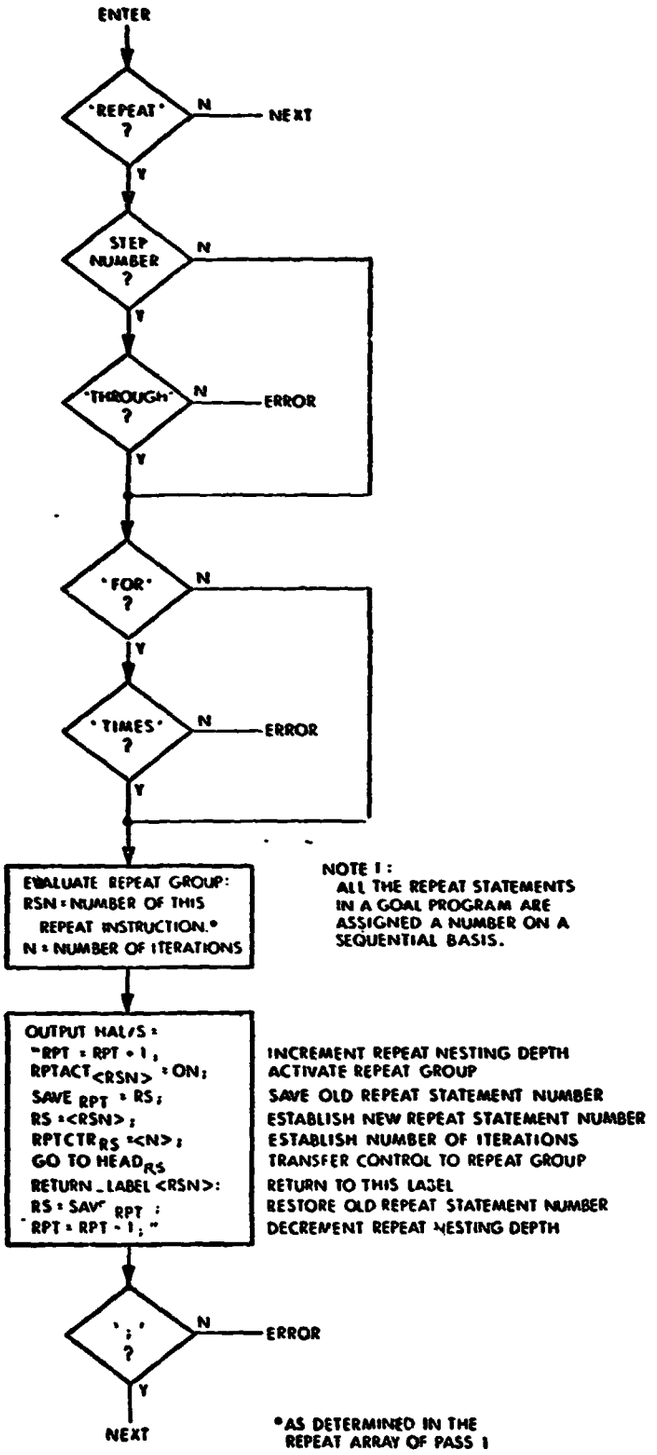


CHART 1 OF 1

ASSIGN STATEMENT (3)

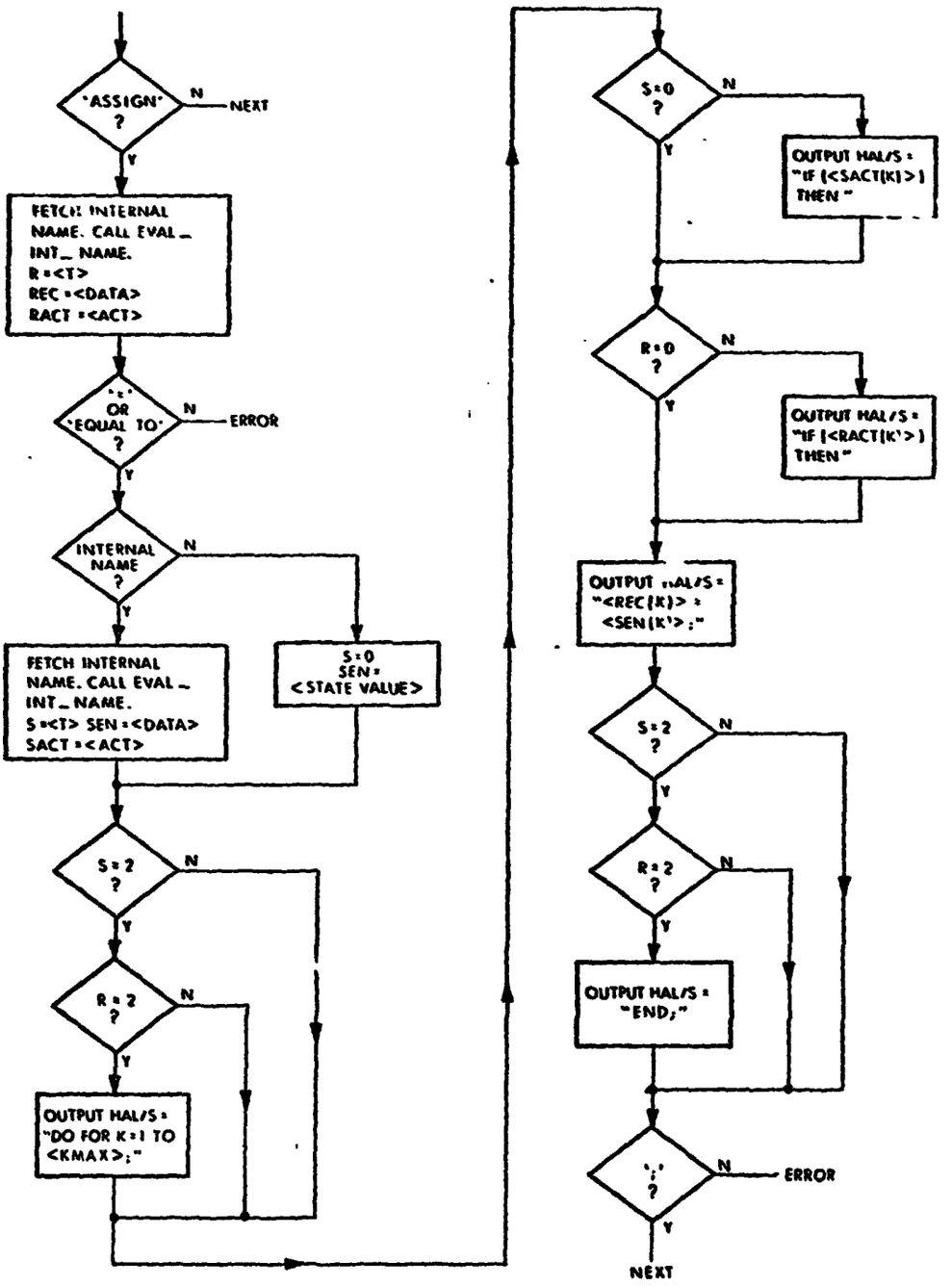


CHART 1 OF 1

LET EQUAL STATEMENT (42)

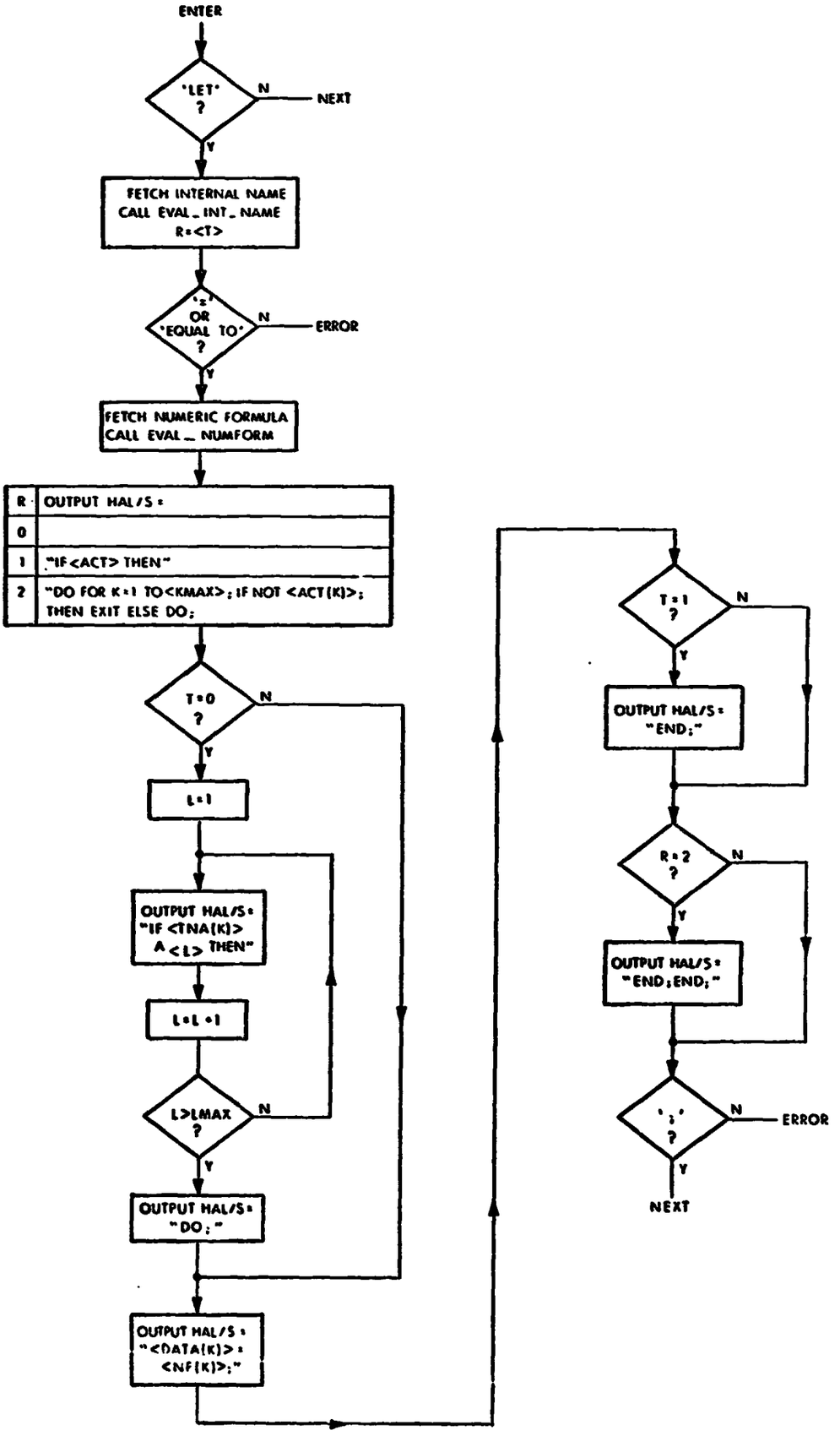
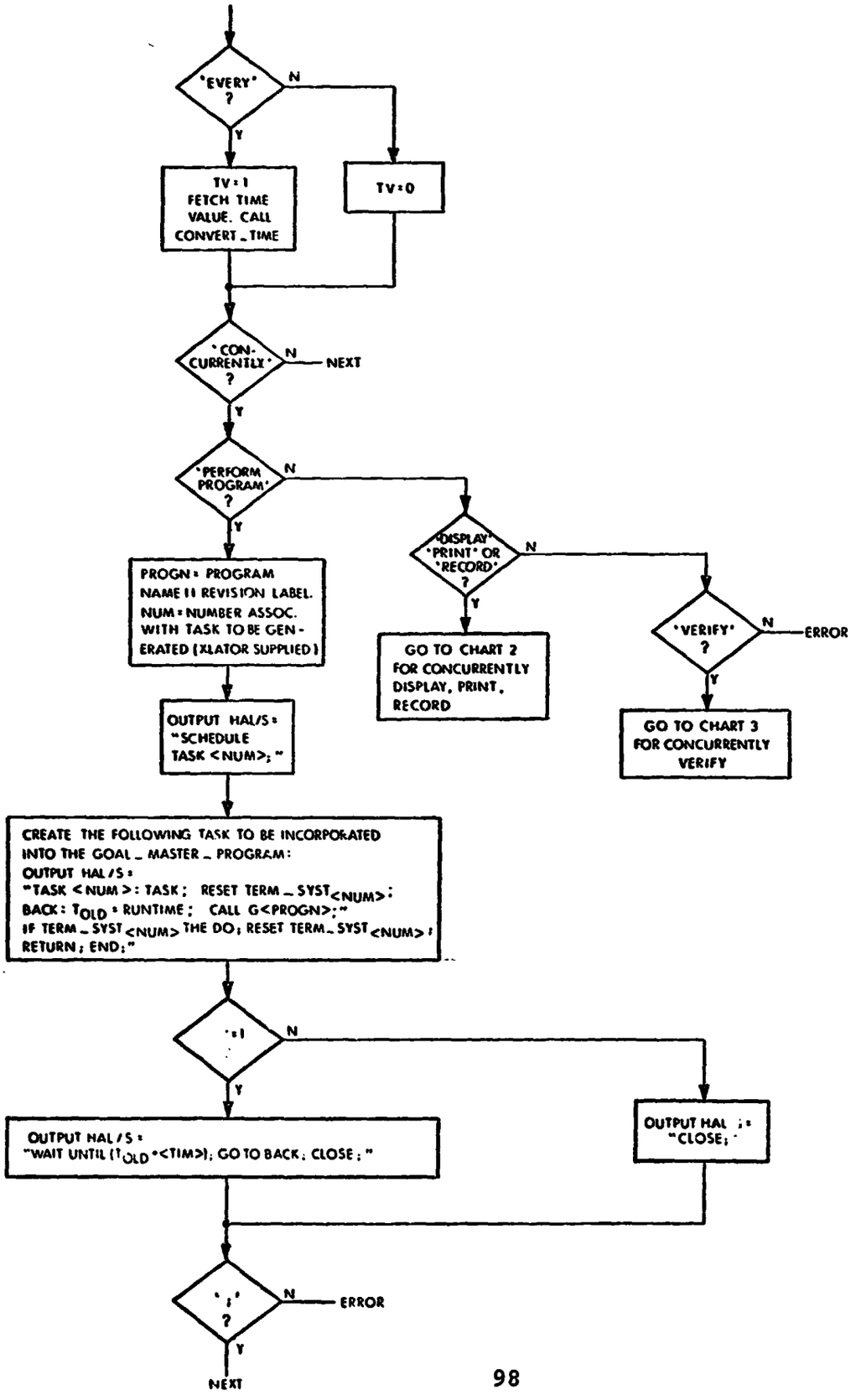


CHART 1 OF 1

CONCURRENT STATEMENT (15)



CONCURRENT STATEMENT (15) , (CONT.)

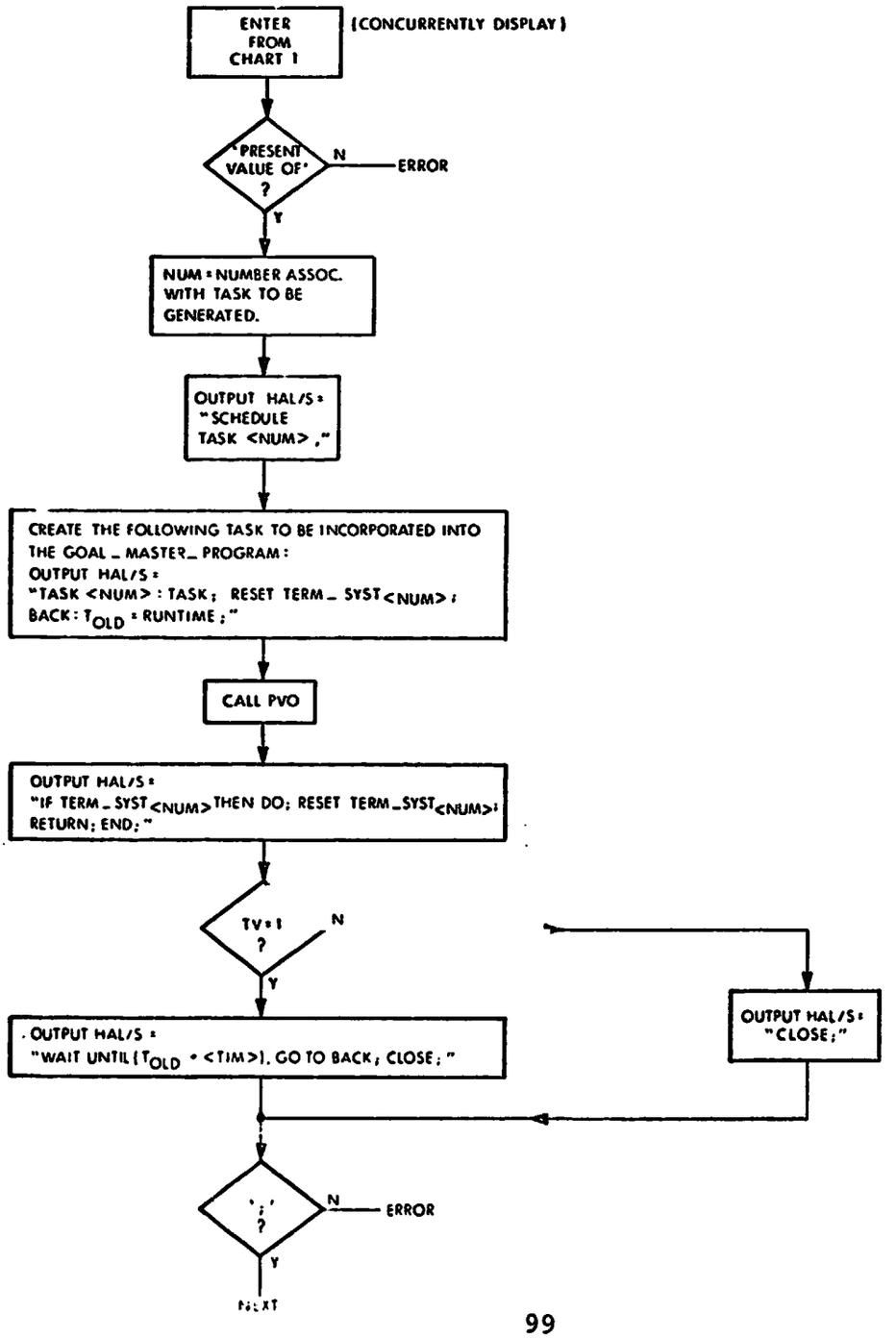


CHART 20 OF 5

CONCURRENT STATEMENT (15) , (CONT.)

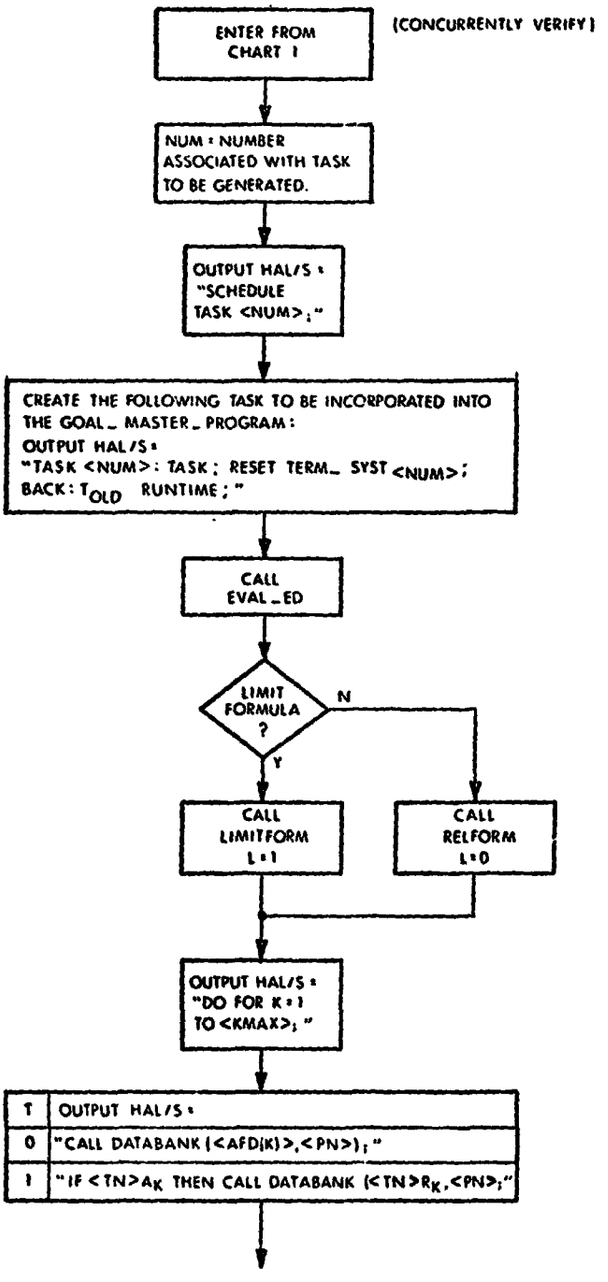


CHART 3 OF 5

CONCURRENT STATEMENT (15) , (CONT.)

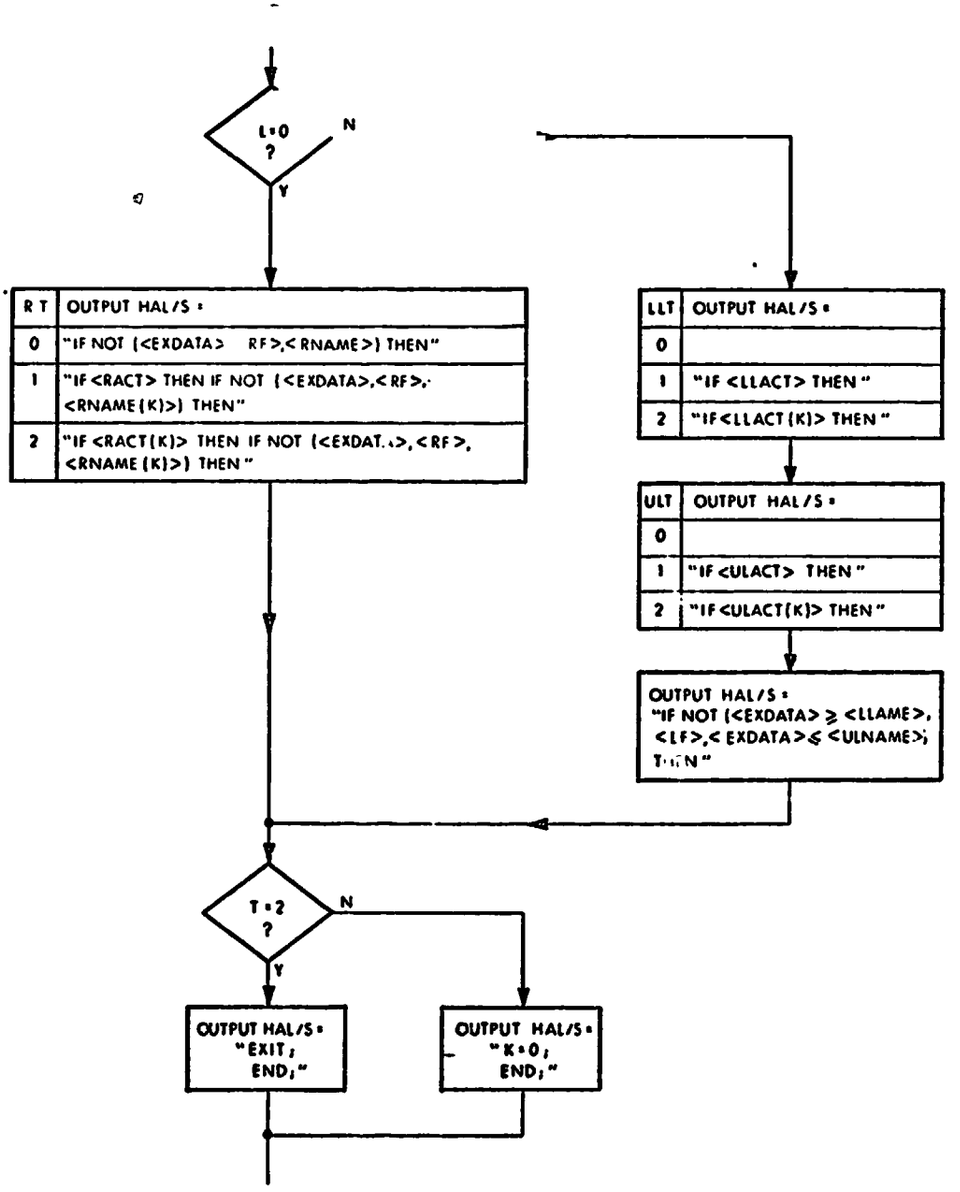


CHART 4 OF 5

CONCURRENT STATEMENT (15) , (CONT.)

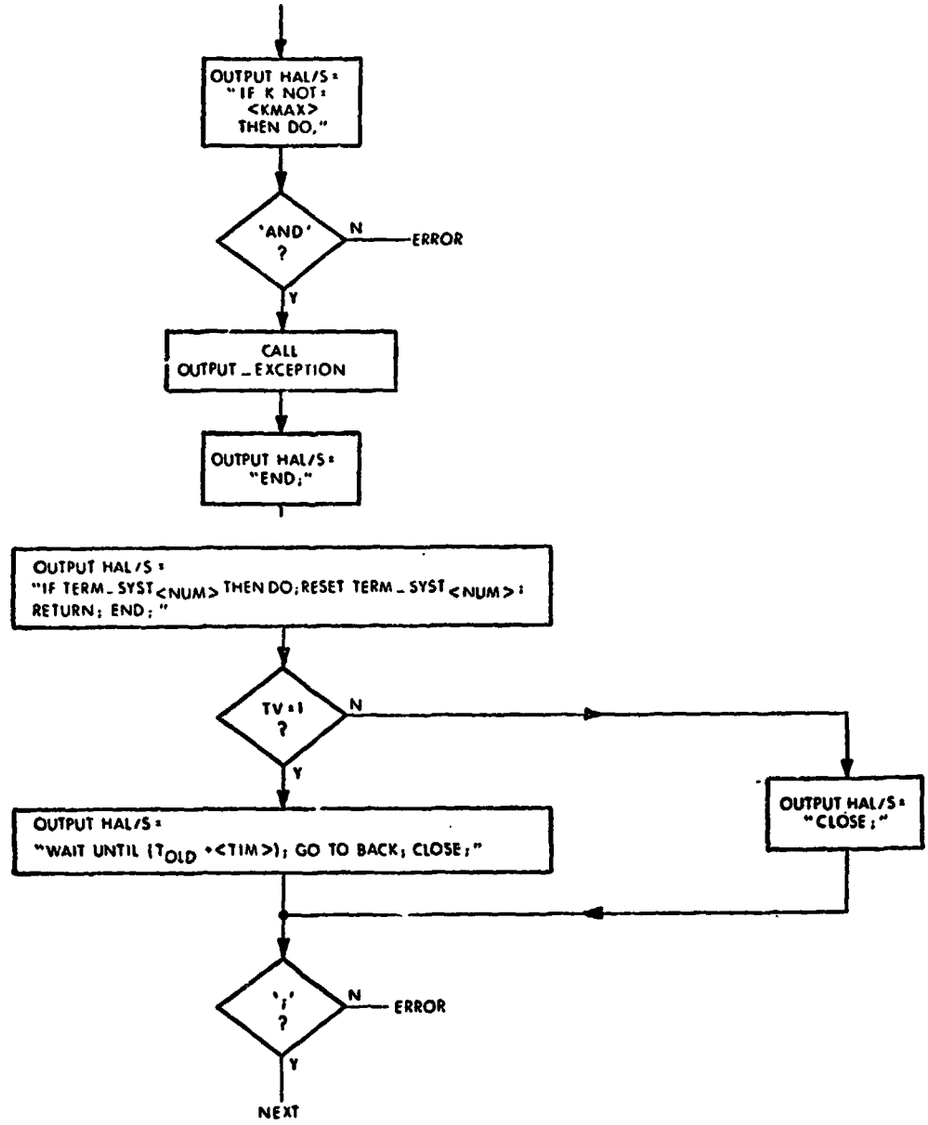


CHART 5 OF 5

RELEASE CONCURRENT STATEMENT (63)

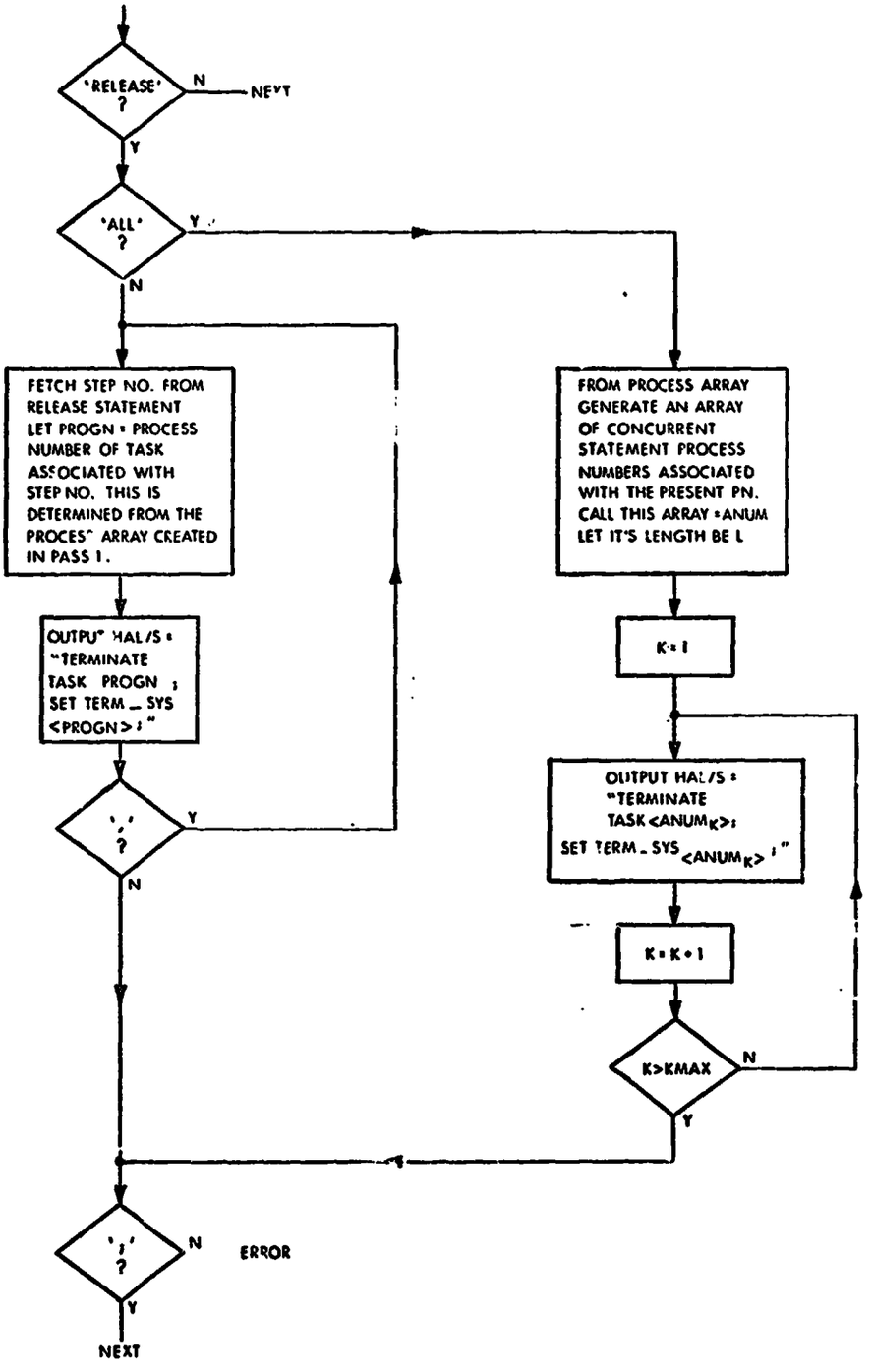


CHART 1 OF 1

PERFORM PROGRAM STATEMENT (55)

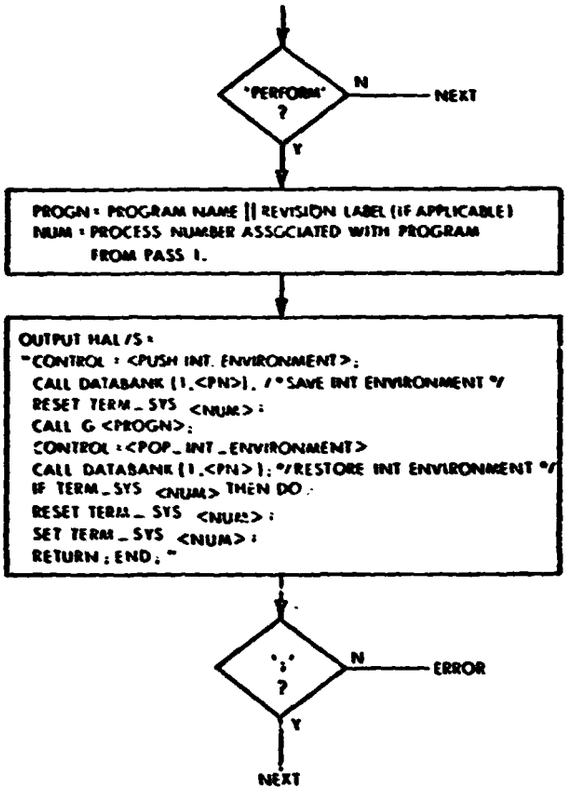


CHART 1 OF 1

PERFORM SUBROUTINE STATEMENT (56)

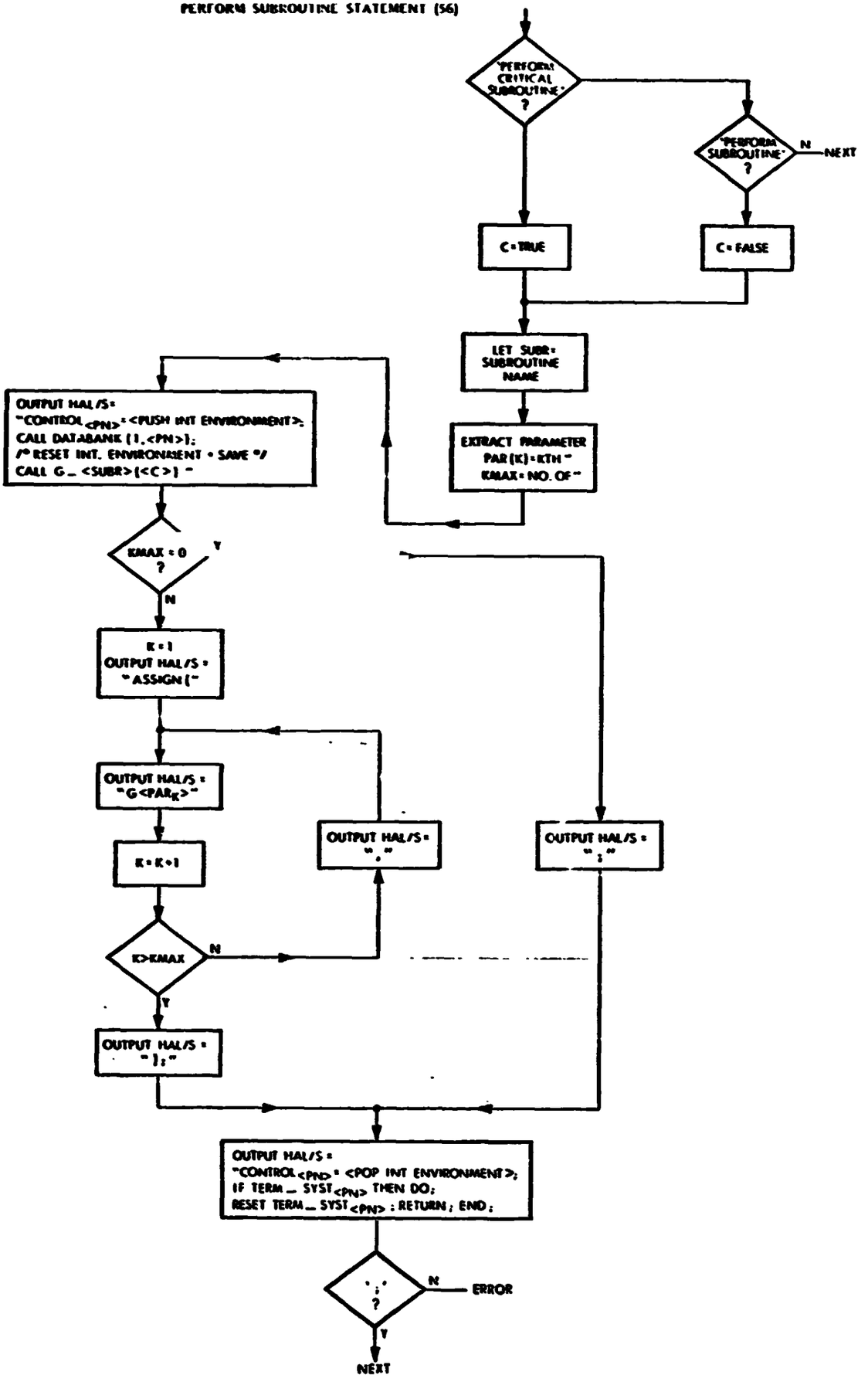
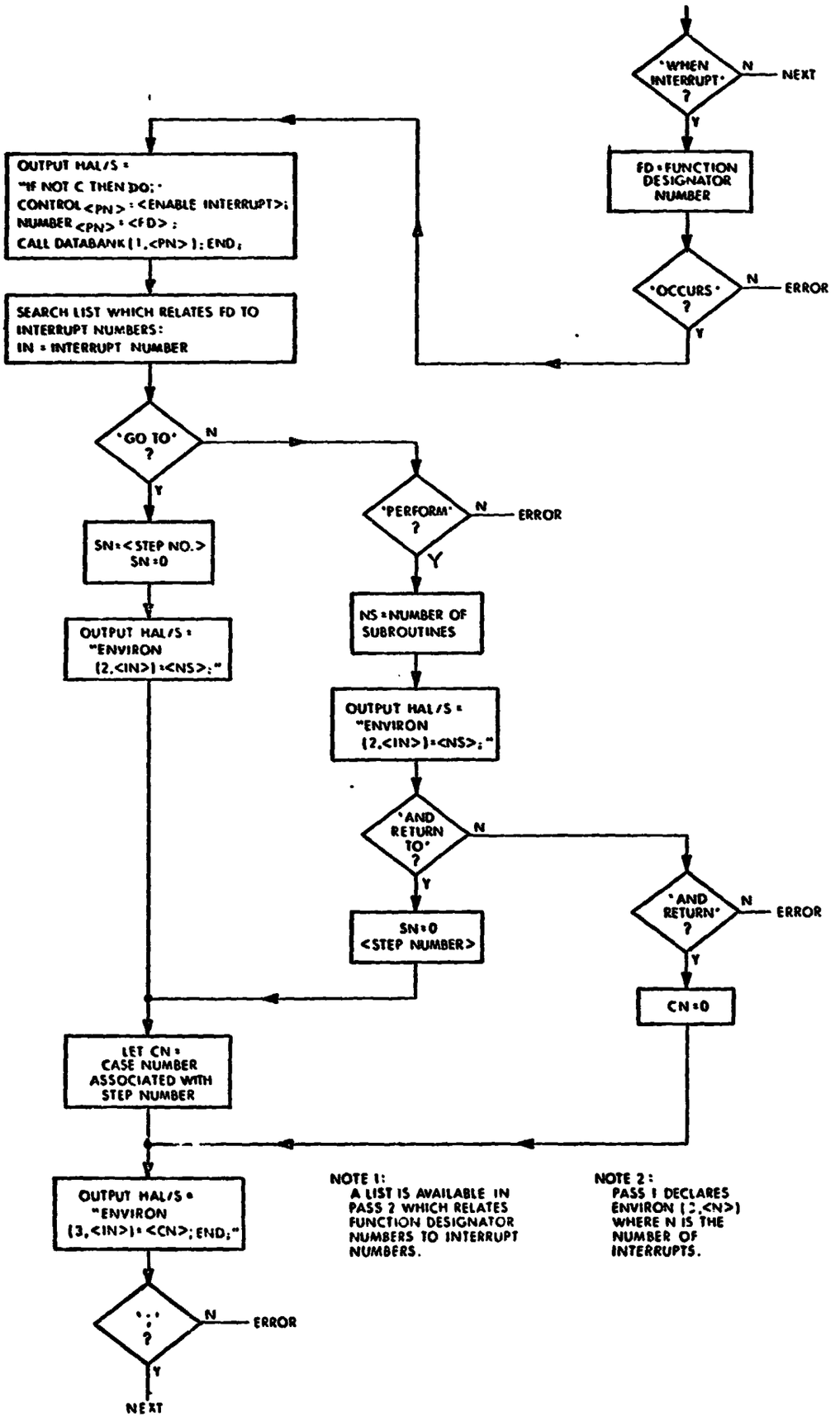


CHART 1 OF 1

WHEN INTERRUPT STATEMENT (84)



DISABLE INTERRUPT STATEMENT (28)

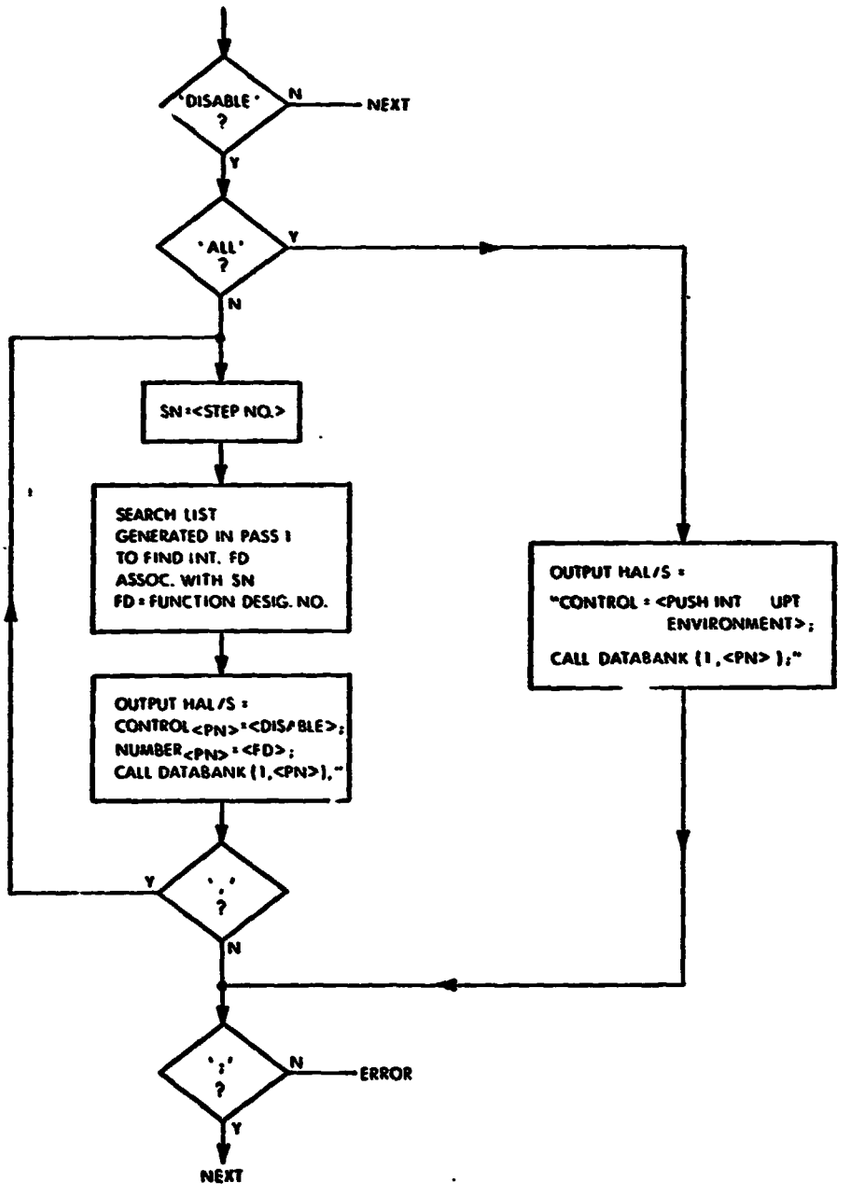


CHART 1 OF 1

ACTIVATE TABLE STATEMENT (1)

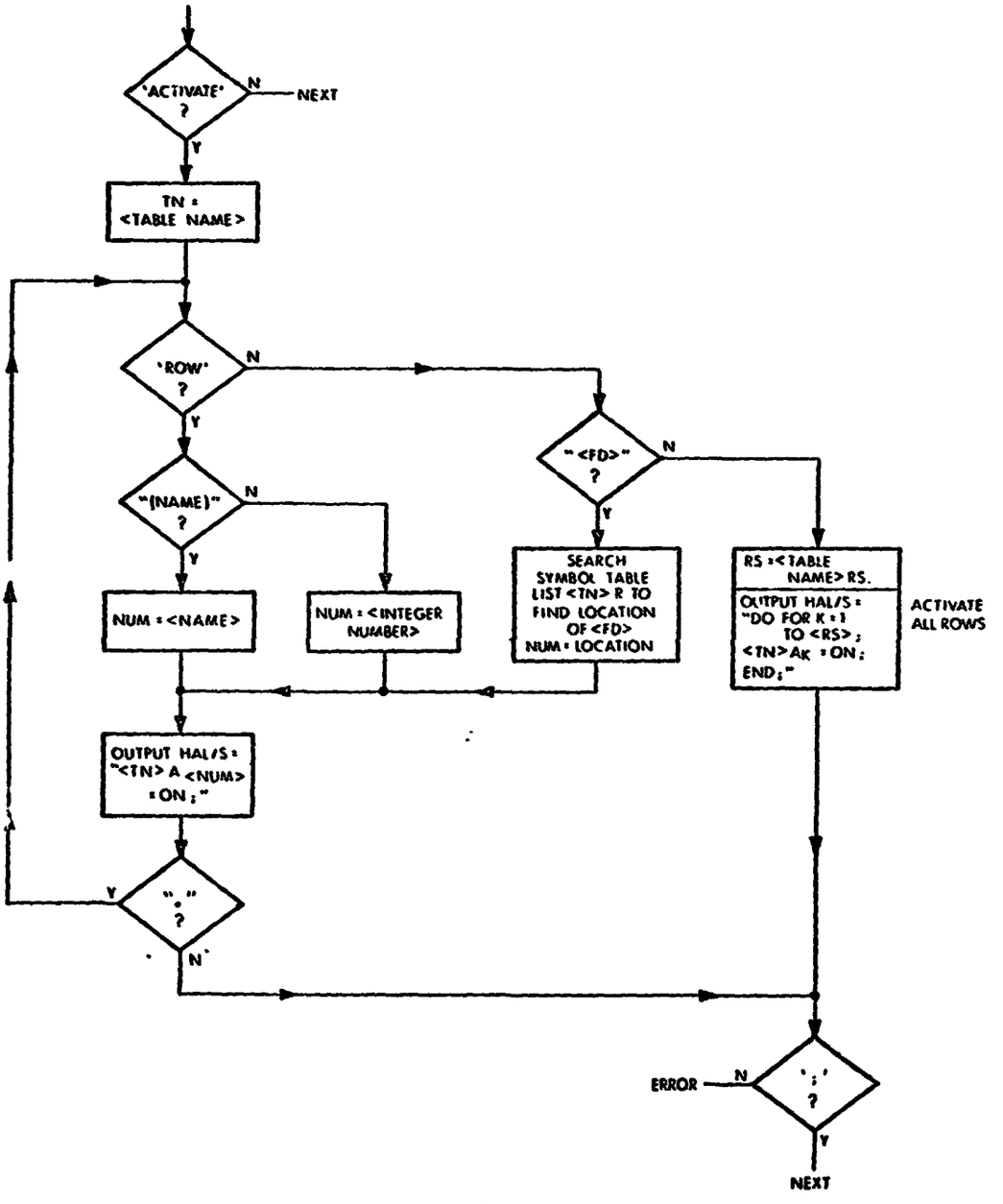


CHART 1 OF 1

INHIBIT TABLE STATEMENT (37)

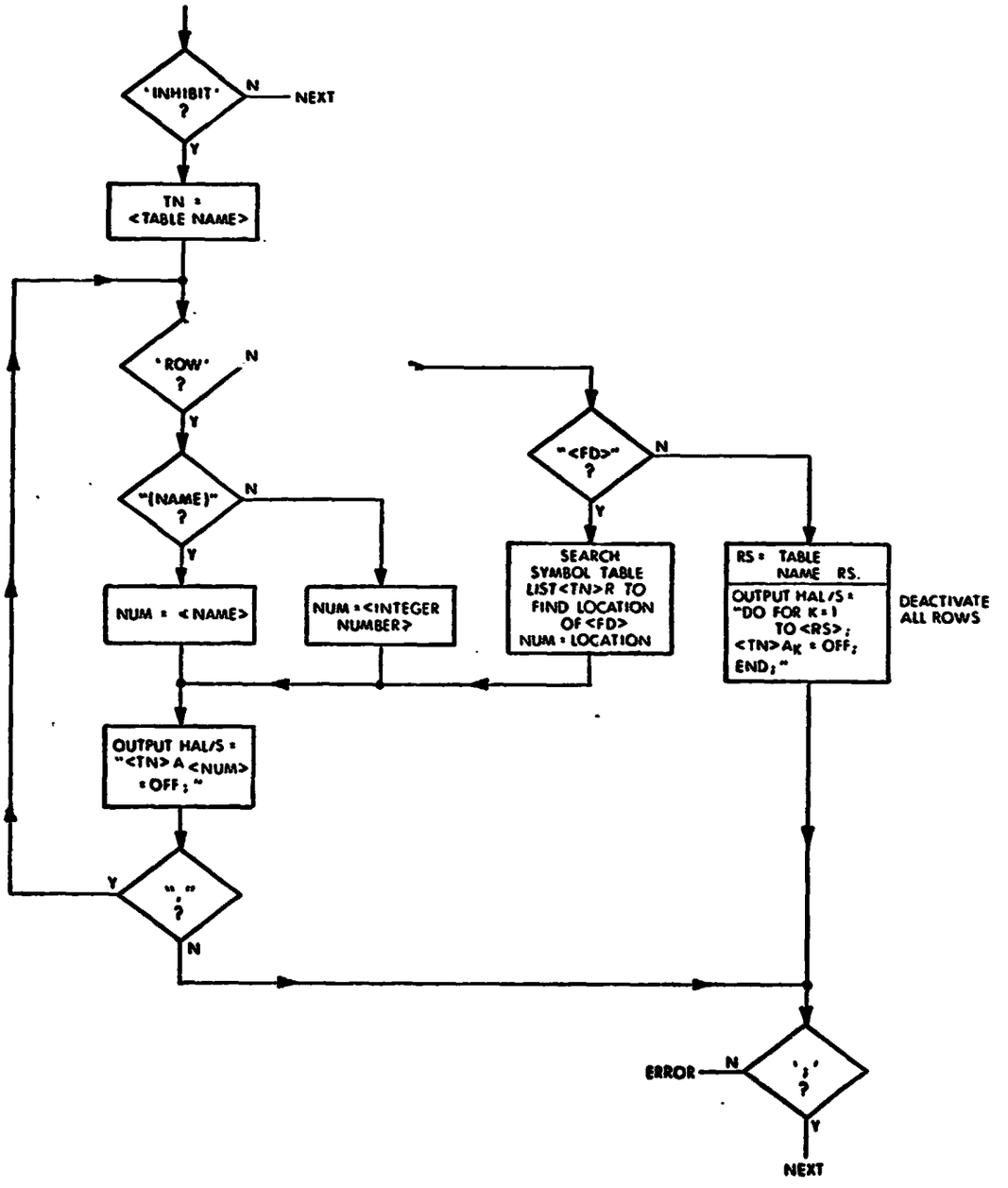
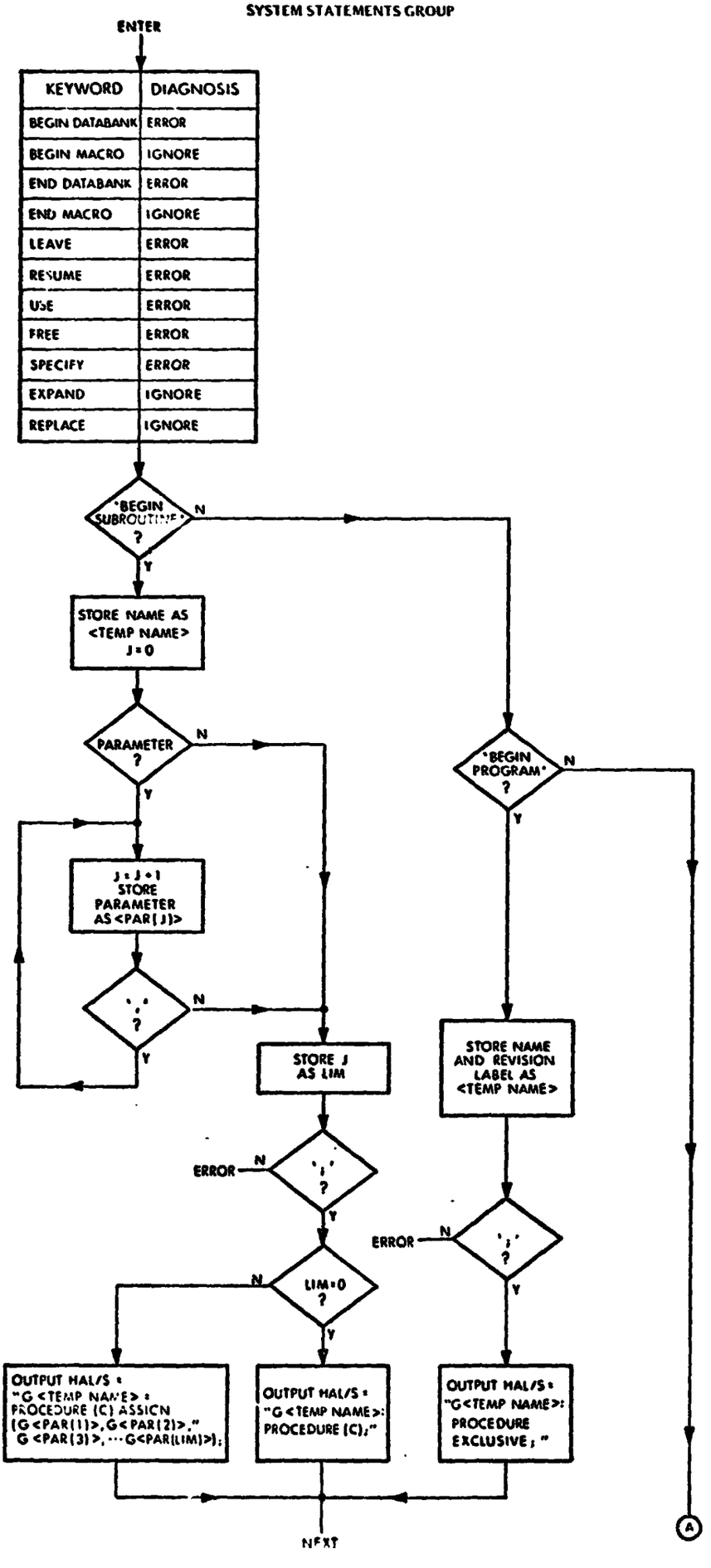


CHART 1 OF 1

THIS PAGE INTENTIONALLY LEFT BLANK.

5.3.3 SYSTEM STATEMENTS


SYSTEM STATEMENTS GROUP
ENTER
KEYWORD | DIAGNOSIS
BEGIN DATABANK | ERROR
BEGIN MACRO | IGNORE
END DATABANK | ERROR
END MACRO | IGNORE
LEAVE | ERROR
RESUME | ERROR
USE | ERROR
FREE | ERROR
SPECIFY | ERROR
EXPAND | IGNORE
REPLACE | IGNORE
'BEGIN SUBROUTINE' ?
N
Y
STORE NAME AS <TEMP NAME>
J=0
PARAMETER ?
N
Y
J=J+1
STORE PARAMETER AS <PAR(J)>
',' ?
N
Y
'BEGIN PROGRAM' ?
N
Y
STORE J AS LIM
STORE NAME AND REVISION LABEL AS <TEMP NAME>
';' ?
ERROR
N
Y
LIM=0 ?
N
Y
';' ?
ERROR
N
Y
OUTPUT HAL/S = "G<TEMP NAME> : PROCEDURE (C) ASSIGN (G<PAR(1)>, G<PAR(2)>," G<PAR(3)>, ··· G<PAR(LIM)>);
OUTPUT HAL/S = "G<TEMP NAME>: PROCEDURE (C);"
OUTPUT HAL/S = "G<TEMP NAME>: PROCEDURE EXCLUSIVE;"
NEXT
A

SYSTEM STATEMENTS GROUP , (CONT.)

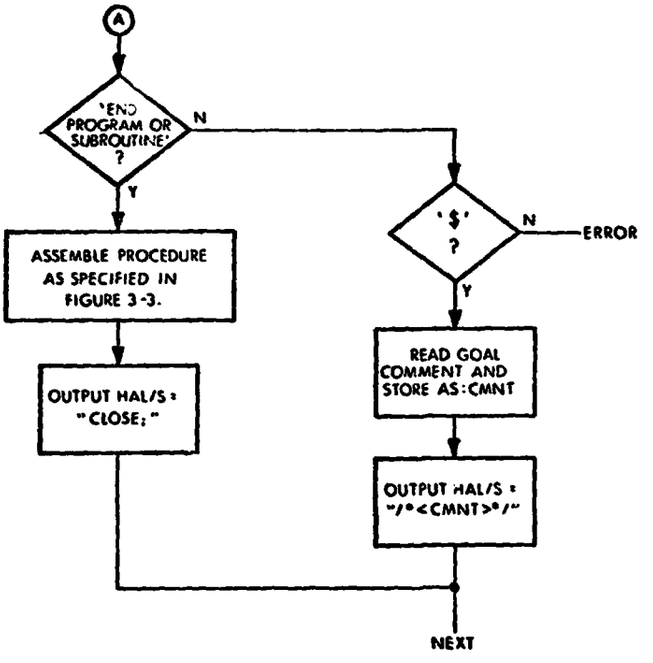


CHART 2 OF 2

5.4 Output Processor

The output processor shall be an integral part of the Translator. It performs its role last in the translation sequence. This role shall consist of two parts. First, it assembles the GOAL_MASTER_PROGRAM according to the format specified in Section 3.1 and 3.2 and depicted in Figures 3-2 and 3-3. Secondly, it generates an output file of HAL/S source whose format is compatible with the machine upon which HAL/S compilation is to take place. It also shall generate supporting outputs consisting of error messages, block summaries, program layouts, etc.

6.0 EXECUTIVE SUPPORT STRUCTURE

6.1 Introduction

Executive Support Structure is defined as the software support system required in the onboard Shuttle computers in order to permit the full potential of GOAL-in-HAL to be realized.

6.2 Recommended Approach

As set forth in Section 3.0 of the Specification, the GOAL_MASTER_PROGRAM contains all software and data required to execute. The only external software required will be the FCOS. By design, the output of the translator is an assembly of HAL/S code which contains not only translations of each of the individual GOAL programs including all relevant data-bank information, but also the procedure DATABANK used for interpreting function designator information in the Flight Computer Operating System (FCOS) environment. The output of the translator is submitted to the HAL/S compiler like any other HAL/S applications program. The result is object code for the flight computer. The GOAL Master Program is self-sufficient and, by design, requires no executive support structure other than the FCOS.

THIS PAGE INTENTIONALLY LEFT BLANK.

7.0 IMPLEMENTATION, VERIFICATION, AND DOCUMENTATION REQUIREMENTS

This section provides a plan and a recommended sequence of steps by which the GOAL-to-HAL Translator can be implemented, documented, verified, maintained, and placed into operational use. This is accomplished by presenting the topic in three sequential groupings. These groupings are implementation, verification, and operation. In each grouping. the pertinent documents are identified, and then they are set forth again in Section 7.4. Section 7.5 is a suggested schedule for the Translator.

7.1 Implementation

Two key decisions have to be part of the implementation phase of the GOAL-to-HAL Translator. The first of these is the choice of the host machine upon which the Translator is to be operated. Technical considerations involved in this choice include the host machine for the KSC GOAL compiler, the destination of the HAL/S source output, and the requirements for utility software, peripherals, operating modes, and output writer capability.

The second key decision involves the language in which the Translator is to be written. This decision could affect the host machine decision. For example, Fortran 4 will run on almost any system, but XPL (which is a good media for writing translators) is only supported on the IBM-360 and Univac 1108.

To the present document, the Translator Specification, must be added a Host Machine and Language Addendum which will specify the interface between the Translator, its language, and the host machine. This document can be brief, but it should deal with the interface questions, and the desired inputs as well as PASS 1 and PASS 2 outputs.

The implementation would now proceed to the coding phase in the language for the chosen machine. The result of this effort would be the Translator Code.

Concurrently, with the actual coding of the Translator, two subsidiary but important efforts should be going on. The first of these is to produce a Translator User's Manual which

would tell the user how to set up and run the Translator, how to prepare the input, and what the supporting and final outputs consist of.

The second document is the Translator Test Plan. This must deal with two levels. First the Test Plan must provide proof hat the Translator is generating valid HAL/S and, secondly, it must show that the logical transformations performed are valid.

7.2 Verification

The two-level Test Plan is now brought into operation. The HAL/S compiler can effectively deal with the first level of verification. That is, whether the Translator has produced valid HAL/S code. Translator output in the form of a GOAL_MASTER_PROGRAM is, by design, supposed to meet completely the necessary characteristics of a HAL/S applications software program, and can therefore be submitted as an input to the compiler.

The Test Plan will have to provide for successive levels of complexity in the GOAL source inputs. These successive levels are suggested in Table 7-1.

The HAL/S Compiler will provide completely adequate diagnosis of the HAL/S source. This diagnosis includes the items in Table 7-2. This, then, is the fundamental approach to syntactical verification via the use of the HAL/S compiler.

The second verification level involves making a valid functional check of the resulting HAL/S. This level involves the project in the very fundamental question of Shuttle software verification. It is expected that the HAL/S-360 Compiler will have, in addition to the HAL/S compiler function, some form of HAL Statement Simulator (HSS) which will simulate execution of HAL/S programs on a flight computer. If this simulator is provided with virtual function designator data which interacts with the HAL/S DATABANK procedure, then a degree of functional verification can be accomplished. A different functional v. .fication at this level can and should be accomplished by running GOAL-in-HAL on the AP-101 target machine which is scheduled to be ntr l element in the Shuttle Avionics Integr tio (SAIL).

0) Single GOAL Statements (Visual Verification against Specification)

1) Single GOAL Program, No Stops, No Repeats, No Interrupts, No Tables

2) Single GOAL Program, Add Stops, Repeats, Interrupts, and Tables

3) Multiple GOAL Programs

4) Multiple GOAL Programs with sets of CONCURRENTLY Perform, Verify and Record Statements

Table 7-1. Successive Levels of GOAL Source Complexity for Translator Verification

Block Summary	Outer Variables Outer Replaces
Program Layout	Programs, Functions and Procedures
Symbol Table Listing	Name, Type, Class, Length, Precision, Nesting, etc.
Cross Reference Listing	Flags & Statement No.
Macro Text Listing	
Etc.	

Table 7-2. HAL/S Compiler Diagnosis of Source

A third test of the overall GOAL system, which includes the GOAL compiler, Translator, and HAL/S compiler, will be in the execution of a translated GOAL program in the SDL using simulated equipment and environment.

The results of these two levels of verification testing carried out in accordance with the Translator Test Plan are documented in the Translator Test Results document. This will only be issued in final form when all aspects of the Specification have been verified. Preliminary versions would include the results of the two levels of testing previously described.

7.3 Operations, Maintenance and Update

It is important to recognize that the Translator acts as a bridge between two languages each of whom will be subject to change. Accordingly, it is necessary to have a Translator Maintenance and Update Procedure.

Under such a plan, the Translator would be under the aegis of a Change Control Board (CCB) at KSC. Computer Program Change Requests (CPCR's) would be brought to this CCB in response to three categories of events:

a) A GOAL Change or CPCR

b) A HAL/S Change approved by the Software Control Board (SCB) at JSC using a Language Change Request (LCR) as the vehicle.

c) A Translator Discrepancy Report (DR).

The Translator CCB would act on the CPCR. If it approved a Translator change, it would also specify the tests to be run and test reports to be written.

7.4 Documentation

A list of proposed GOAL-to-HAL Translator documentation is given below:

7.4.1 Specifications

7.4.1.1 Translator Specification (Per NAS 10-8385)

7.4.1.2 Host Machine and Language Addendum

7.4.2 Translator Code Documentation Listing

7.4.3 Translator User's Manual

7.4.3.1 How to Run the Translator
- Basic Procedures
- Run-Time Options

7.4.3.2 Preparing GOAL Source Input

7.4.3.3 Supporting Output
- Optional Primary GOAL Source Listing
- GMP Structure Outlines
- Translated GOAL Program Structure
- Function Designator Utilization List

7.4.3.4 Final HAL/S Source Output

7.4.4 Translator Test Plan

7.4.4.1 Valid HAL/S Check by Compilation

7.4.4.2 Valid Functional Check by:
- HAL Statement Simulator
- SAIL Test
- SDL Simulation

7.4.5 Translator Test Results

7.4.5.1 Compilation Results

7.4.5.2 Functional Results

7.4.6 Translator Maintenance Manual and Update Procedure

7.4.6.1 CCB Procedure

7.4.6.2 GOAL Changes

7.4.6.3 HAL/S Changes

7.4.6.4 Translator DR's and Host Machine Changes

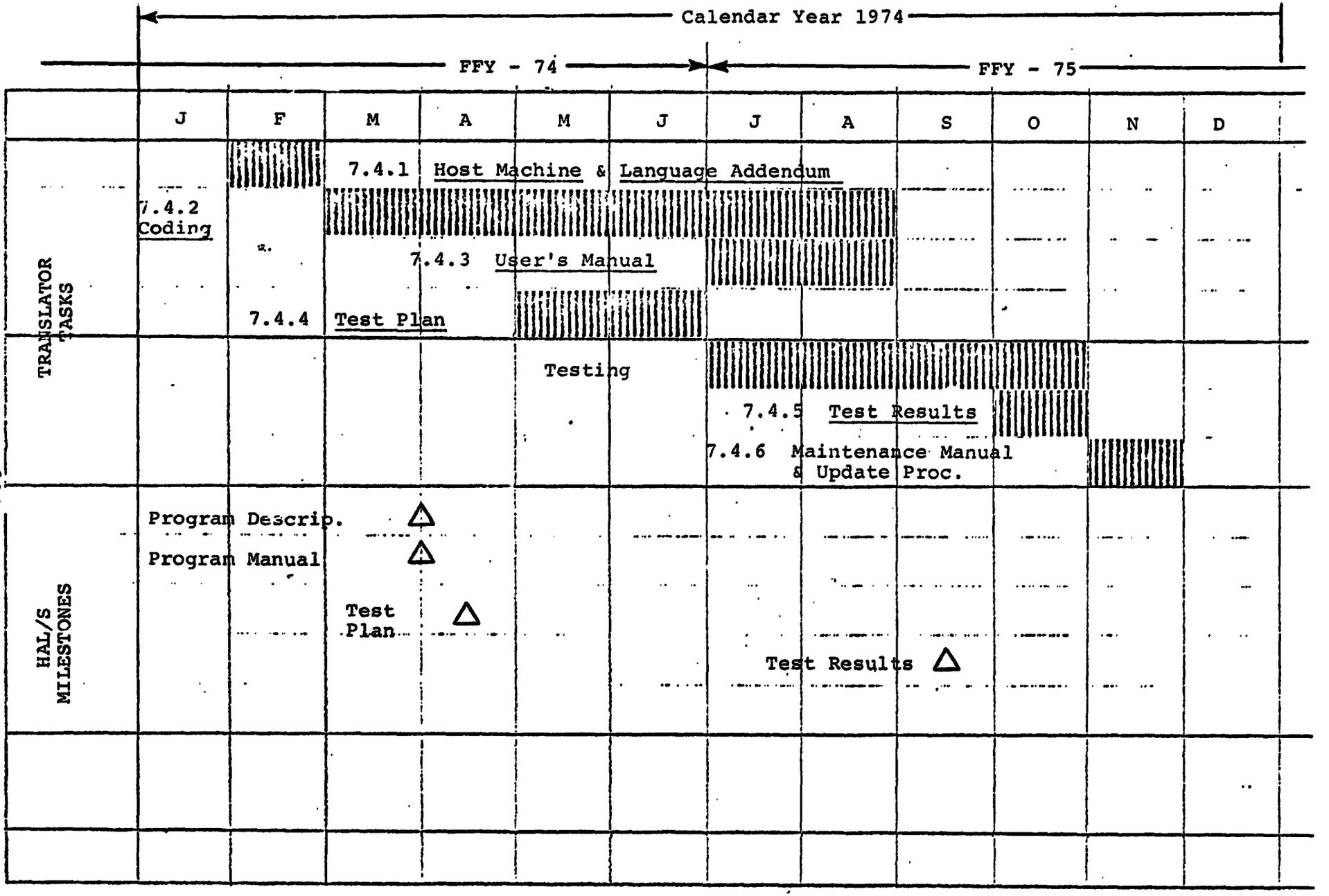
7.4.6.5 Change Validation

7.4.6.6 How to Debug the Translator

7.5 Translator Schedule

See Figure 7-1.

THIS PAGE INTENTIONALLY LEFT BLANK.


Calendar Year 1974
FFY - 74
FFY - 75
J
F
M
A
M
J
J
A
S
O
N
D
TRANSLATOR TASKS
7.4.1 Host Machine & Language Addendum
7.4.2 Coding
7.4.3 User's Manual
7.4.4 Test Plan
Testing
7.4.5 Test Results
7.4.6 Maintenance Manual & Update Proc.
HAL/S MILESTONES
Program Descrip.
Program Manual
Test Plan
Test Results

THIS PAGE INTENTIONALLY LEFT BLANK.

8.0 RECOMMENDATIONS

8.1 Choice of Host Machine

The GOAL-to-HAL Translator should be implemented on the IBM 360 or equivalent. This is because the HAL/S language, as defined in the HAL/S Language Specification, is going to be implemented on a HAL/S-360 compiler system resident on a IBM 360. The same compiler system, with minor modifications, will be compatible with the IBM 370 computer series.

By choosing the IBM 360 as the host machine, the translation and the compilation can be run end-to-end with immediate syntactical verification of the overall results. Furthermore, a HAL Statement Simulator (HSS) will be resident on the machine and it will provide a flight computer functional simulation for functional checking.

8.2 Choice of Translator Language

The Translator should be implemented using the XPL Translator Writing System (TWS) as the primary tool. TWS is a tool which is intended to assist in the writing of translator-compilers, interpreters, assemblers, etc. Its usefulness lies in its ability to supply uniform functional modules for standard functions such as text scanning and to automate the production of language-dependent portions of the Translator. This specialized language is very close to machine language form, although it has the convenient block-structure of PL/1 or HAL. It is an easy language to use for strings, indexing, etc. since it doesn't have the bulk of assembly language nor does it have Fortran's inappropriate mathematical orientation.

The HAL compiler was written in XPL and it was found to be an efficient and powerful tool.

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX A. GOAL-TO-HAL MAPPING

APPENDIX A. GOAL-TO-HAL MAPPING

In this section, the proposed relationships between the GOAL and HAL statements are described in some detail. This description has been designated "mapping" in order to distinguish it from the Translator Specification. Mapping will give an explanation or example of each complete GOAL statement. The Specification rigorously defines all variations, permutations, and combinations of each GOAL statement.

A.1 Declaration Statements

A.1.1 Single Data

GOAL Statement:

```
DECLARE NUMBER(RESULTS);
```

Purpose:

This statement declares a numeric data item with the symbolic name (RESULTS) for purposes of general computation within the GOAL program.

Equivalent HAL/S form:

DECLARE RESULTS; This statement will declare an unarrayed single precision scalar variable which can be used in the same context as the corresponding GOAL form.

GOAL Statement:

```
DECLARE QUANTITY (OFFSET) = .5 PSI, (PUMP PRESS);
```

Purpose:

This statement declares a GOAL "quantity" (a real number and an associated dimension) as a variable for use

in a program. Since the units associated with these entities exist for annotation purposes only, the equivalent HAL/S declarations include a scalar declaration and a character string dimension.

HAL/S Equivalent Form:

```
DECLARE OFFSETD INITIAL(0.5); DECLARE OFFSETDIM INTEGER
SINGLE INITIAL(3); DECLARE PUMP_PRESSD; DECLARE PUMP_PRESSDIM
INTEGER SINGLE;
```

Note that the initialization value of 0.5 was provided in the GOAL form as a compile time assignment (the equal-sign) which becomes the initial value 0.5 in the HAL/S form. The OFFSETDIM value of three (3) is presumed to correspond to the PSI label.

GOAL Statement:

```
DECLARE STATE (FLAG A) = ON, (FLAG B);
```

Purpose:

This statement is used in the context of a GOAL program to declare the existence of the single bit booleans FLAG A and FLAG B. The initial value of the variable FLAG A is set to be ON. FLAG B is not initialized.

Equivalent HAL/S Form:

```
DECLARE BOOLEAN FLAG_AD INITIAL(ON);

DECLARE BOOLEAN FLAG_BD;
```

GOAL Statement:

```
DECLARE TEXT(ERROR MESSAGE) = (6D10 BATTERY VOLTAGE LOW);
```

Purpose:

This statement is used in a GOAL program to declare a fixed character message which will be used in some I/O operation. Since there is no character manipulation or assignment in GOAL, this message must always be either fixed or defined in some input operation.

Equivalent HAL/S Statement:

```
DECLARE ERROR_MESSAGED CHARACTER(24) INITIAL('6D10 BATTERY
                                VOLTAGE LOW');
```

A.1.2 List Type

GOAL Statement:

```
DECLARE NUMERIC LIST(LIST NUM) WITH 4 ENTRIES;
```

Purpose:

This statement declares to the GOAL compiler that the programmer wishes to create a linear array of 4 numeric elements without any initialization.

Equivalent HAL/S Form:

```
DECLARE LIST_NUMD ARRAY(4);
```

GOAL Statement:

```
DECLARE NUMERIC LIST (ROOT 3) WITH 10 ENTRIES 1.000,
1.260, 1.442, 1.587, 1,710, 1.817, 1,903, 2.000, 2.080,
2.154;
```

Purpose:

This statement declares to the GOAL compiler that the programmer wishes to create a linear array of 10 numeric elements with the indicated initialization.

Equivalent HAL/S Form:

```
DECLARE ROOT_3D ARRAY(10) INITIAL(1.00,1.260,1.442,
1.587,1.710,1.817,1.903,2.000,2.080,2.154);
```

GOAL Statement:

```
DECLARE QUANTITY LIST (LIST A) WITH 3 ENTRIES;
```

Purpose:

This statement creates a list of 3 GOAL quantities in a linear array form. Each quantity has a scalar value and a physical units dimension.

HAL/S Equivalent Form:

```
DECLARE LIST_AD ARRAY(3);

DECLARE LIST_ADIM ARRAY(3) INTEGER SINGLE;
```

GOAL Statement:

```
DECLARE QUANTITY LIST(VOLTAGE LIST) WITH 6 ENTRIES
   28V,+0.5V,-0.5V,0V,50V,10 SECS;
```

Purpose:

This statement sets up a list of 6 GOAL quantities, with initialization to the values indicated.

HAL/S Equivalent Form:

```
DECLARE VOLTAGE_LISTD ARRAY(6)
    INITIAL(28,0.5,-0.5,0,50,10);

DECLARE VOLTAGE_LISTDIM ARRAY(6) INTEGER SINGLE
    INITIAL(1,1,1,1,1,2);
```

Note that the units are presented in a coded form, assuming 1 = volts and 2 = seconds.

GOAL Statement:

```
DECLARE STATE LIST (FLAG LIST) WITH 10 ENTRIES;
```

Purpose:

This statement sets up an array of 10 booleans for reference within the GOAL program. No initialization is performed.

HAL/S Equivalents:

```
DECLARE FLAG_LISTD ARRAY(10) BOOLEAN;
```

GOAL Statement:

```
DECLARE STATE LIST (LIST STATE) WITH 6 ENTRIES
    ON,ON,ON,OFF,OFF,ON;
```

Purpose:

This statement declares an array of 6 binary values (Booleans) for use within the GOAL program to store the states of discretes.

Equivalent HAL/S Form:

```
DECLARE LIST_STATED ARRAY(6) BOOLEAN INITIAL(ON,ON,ON,OFF,OFF,ON);
```

GOAL Statement:

```
DECLARE TEXT LIST(INPUT) WITH 2 ENTRIES WITH A
     MAXIMUM OF 25 CHARACTERS;
```

Purpose:

This statement sets up an array of two text strings for use as the receiver of some input followed by later use as the source of some output (no internal manipulations of text are provided by GOAL).

Equivalent HAL/S Form:

```
DECLARE INPUTD ARRAY(2) CHARACTER(25);
```

Note that in this example the original name duplicated a HAL/S keyword and thus had to be modified in some way following the translation. In the example, the letter D was appended to the original name thereby resolving the keyword conflict. The maximum length of 25 carries over directly.

GOAL Statement:

```
DECLARE TEXT LIST (OPERATOR INSTRUCTION) WITH 2 ENTRIES
   (PLACE SWITCHES INDICATED),(*PREFLIGHT TM CAL IN
   PROGRESS*);
```

Purpose:

This statement declares an array of two character strings which are initialized to the values indicated, with a maximum length determined by implication from the length of the initial values.

Equivalent HAL/S Form:

```
DECLARE OPERATOR_INSTRUCTIOND ARRAY(2) CHARACTER(30)
  INITIAL ('PLACE SWITCHES INDICATED','*PRELIGHT
  TM CAL IN PROGRESS*');
```

This is a direct translation, with the only subtlety being the determination of the maximum length found among the initial values so that the character string maximum length may be declared.

A.1.3 Table Types

The translation of GOAL table data types requires a strategy employing several HAL/S arrays to accomplish the same ends within the framework of a HAL/S program. A set of arrays which will accomplish this is the following:

1. A main data array with row and column dimensions identical to the row and column dimensions of the original GOAL table. The HAL/S data type of this table will be SCALAR, BOOLEANS, or CHARACTER depending upon the original GOAL table's data type (QUANTITY & NUMERIC, STATE, or TEXT respectively). This array will store data in the HAL/S version. For GOAL QUANTITY tables, an auxiliary character array of dimensions is required.

2. An auxiliary "activation array" of BOOLEAN elements controlling whether or not the given row is to be active at some time during execution.

These two arrays cover all the variable information about a GOAL table as translated into HAL/S.

GOAL Statement:

```
DECLARE NUMERIC TABLE(HIGH LOW RUN) WITH 3 ROWS
    AND 4 COLUMNS TITLED
        (HIGH), (LOW), (RUN), (CUR) WITH ENTRIES

<E1 GG CHAMBER P> , 1000.1,  1.0,  500.0,   ,
<E2 GG CHAMBER P> , 1001.2,   .9,  500.0,   ,
<E3 GG CHAMBER P> ,  999.8,  1.2,  500.0,   ,
```

Purpose:

This statement sets up a GOAL table with initial values in 3 columns, and 3 rows of function-designators. A fourth column is left uninitialized.

Equivalent HAL/S Form:

Main Data Array:

```
DECLARE HIGH_LOW_RUND ARRAY(3,4) INITIAL(

  1001.1,   1.0,   500.0,    0,
  1001.2,   0.9,   500.0,    0,
   999.8,   1.2,   500.0,    0);
```

Note here that the HAL/S initialization cannot embed uninitialized values within its list so a "0" has been used in the fourth-column entries.

```
DECLARE HIGH_LOW_RUNR ARRAY(3) INITIAL
  (<FD(1)>,<FD(2)>,<FD(3)>);
DECLARE HIGH_LOW_RUNA ARRAY(3) BOOLEAN INITIAL(ON);
```

GOAL Statement:

```
DECLARE QUANTITY TABLE (MAIN FUEL FLOW) WITH 5 ROWS
   AND 3 COLUMNS WITH ENTRIES

<E1 MAIN FUEL>,  0.1 PPS,  300.1 PPS,    ,
<E2 MAIN FUEL>,  0.3 PPS,  300.2 PPS,    ,
<E3 MAIN FUEL>,  0.4 PPS,  300.1 PPS,    ,
<E4 MAIN FUEL>,  0.2 PPS,  300.1 PPS,    ,
<E5 MAIN FUEL>,  0.1 PPS,  299.8 PPS,    ,
```

Purpose:

This statement sets up a GOAL quantity table with 5 function designators and 3 columns. Since this is a quantity table, and since quantity units can change as date, the HAL/S equivalent will have two main data arrays, the second containing dimensional data.

Equivalent HAL/S Form:

Main Data Arrays:

```
DECLARE MAIN_FUEL_FLOWD ARRAY(5,3) INITIAL (
```

```
            0.1,   300.1,    0,
            0.3,   300.2,    0,
            0.4,   300.1,    0,
            0.2,   300.1,    0,
            0.1,   299.8,    0);

      DECLARE MAIN_FUEL_FLOWR ARRAY(5)
        INITIAL(<E1 MAIN FUEL>, <E2 MAIN FUEL>,<E3 MAIN FUEL>,
                <E4 MAIN FUEL>,<E5 MAIN FUEL>);

      DECLARE MAIN_FUEL_FLOWA ARRAY(5)
          BOOLEAN INITIAL (ON);

      DECLARE MAIN_FUEL_FLOWDIM ARRAY(5,3)
          INTEGER SINGLE INITIAL

      (3,3,0,3,3,0,3,3,0,3,3,0,3,3,0);
```

using the coded form for dimensions 3 = PPS and 0 = unspecified.

GOAL Statements:

```
      DECLARE STATE TABLE (THRUST OK) WITH 5 ROWS AND
        3 COLUMNS TITLED (THRUST OK), (THRUST NOT OK),
        (STATE) WITH ENTRIES
          <THRUST OK 1E1> ,  ON,  OFF,   ,
          <THRUST OK 1E2> ,  ON,  OFF,   ,
          <THRUST OK 1E3> ,  ON,  OFF,   ,
          <THRUST OK 1E4> ,  ON,  OFF,   ,
          <THRUST OK 1E5> ,  ON,  OFF,   ,
```

Purpose:

This statement sets up a GOAL "state table" with 3 columns and 5 rows, initialized as shown.

```
DECLARE THRUST_OKD ARRAY(5,3) BOOLEAN INITIAL (TRUE,
   FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE,
   FALSE);

DECLARE THRUST_OKR ARRAY(5) INITIAL (<THRUST OK 1E1>,
   <THRUST OK 1E2>,<THRUST OK 1E3>,<THRUST OK 1E4>,
   <THRUST OK 1E5>);

DECLARE THRUST_OKA ARRAY(5) BOOLEAN INITIAL (ON);
```

GOAL Statement:

```
DECLARE TEXT TABLE (MESSAGE TABLE) WITH 2 ROWS
    AND 1 COLUMN TITLED

          (MESSAGE A) WITH ENTRIES

<224 DISPLAY B35> , (SWITCH SCAN IN PROGRESS),

<224 DISPLAY B42> , (PLACE ABOVE SWITCHES AS INDICATED);
```

Purpose:

This GOAL statement prepares a table of character string data associated with two function designators. The table has but a single column.

Equivalent HAL/S Form:

Main Data Array:

```
DECLARE MESSAGE_TABLED ARRAY(2,1) CHARACTER(33) INITIAL(
   'SWITCH SCAN IN PROGRESS',
   'PLACE ABOVE SWITCHES AS INDICATED');

DECLARE MESSAGE_TABLER ARRAY(2) INITIAL (<224 DISPLAY
   B35>,<224 DISPLAY B42>);

DECLARE MESSAGE_TABLEA ARRAY(2) BOOLEAN INITIAL(ON);
```

A.2 Procedural Statements

A.2.1 Prefixes

GOAL Step Number Prefix:

```
STEP 163 ... rest of GOAL statement ...
```

HAL/S Equivalent Form:

```
STEP_163: ... rest of HAL statement ...
```

GOAL Time Prefix:

```
WHEN <COUNT_DOWN_CLOCK> IS
   -80 HRS 27 MIN 00 SEC THEN
     ... rest of GOAL statement
```

Purpose:

Cause the GOAL program to wait until the <COUNT_DOWN_CLOCK> value is greater than or equal to -80:27:00.

Equivalent HAL/S Form:

```
FLAG = 0;
DO WHILE FLAG = 0;
    CALL DATABANK (<COUNT_DOWN_CLOCK>,<PN>);
IF NUMBER<PN> = -289620
    THEN FLAG = 1;
END;
```

In this example, -80 hrs., 27 minutes, was converted to absolute time in "machine units" by the Translator subroutine CONVERT_TIME.

The effect of the WAIT statement of HAL/S is independent of any following statements (it is not a "prefix" as in GOAL).

However, by delaying processing of statements which follow it, the WAIT works as if it were the prefix. The effect is also identical to the GOAL "WHEN" prefix since any clock time greater than or equal to the specified time will cause the halt to be ended and/or ignored.

Note also that all HAL/S Real Time statements assume a single real time clock. In order to allow the possibility of multiple clocks (not rules out by GOAL specification), the Translator will have to incorporate a scaling and offset algorithm so that all clock function designators can be driven by the single HAL/S Real Time Operating System clock.

GOAL Time Prefix:

```
AFTER <COUNT DOWN CLOCK> IS -80 HRS 27 MINUTES
    00 SECS THEN ...
```

Purpose:

Delay execution of the particular statement until after the time named. In any time-dependent digital system, "after" may only mean "one system clock tick" later than the specified time. Thus, this statement is the same as a WHEN statement with a time value increased by the unit of the basic clock period.

Equivalent HAL/S Form:

```
FLAG = 0;
DO WHILE FLAG = 0;
  CALL DATABANK(<COUNT_DOWN_CLOCK>,<PN>);
  IF NUMBER >=  -(289620 + <ΔX>)
  THEN FLAG = 1;
END;
```

where 289620 is the machine-unit (absolute) equivalent of the specified time and ΔX is the granularity of time in the clock, expressed in "machine units".

GOAL Statement:

```
IF (MIDDLE GIMBAL ANGLE) IS GREATER THAN (MIDDLE
   GIMBAL LIMIT) THEN
```

The HAL/S Equivalent Form is generated with the assistance of the EVAL_INT_NAME and REL_FORM Translator subroutines:

```
K = 1;
IF NOT (MIDDLE_GIMBAL_ANGLE_D > MIDDLE_GIMBAL_LIMIT)
THEN K = 0;
IF K = 1 THEN DO;
     --- ---- (Procedural Statement)
END; (Provided by Translator)
```

GOAL Verify Pre_ix:

```
VERIFY <MIDDLE GIMBAL ANGLE> IS LESS THAN
  (MIDDLE GIMBAL LIMIT) ELSE ...
```

The HAL/S Equivalent Form is generated with the assistance of the EVAL_ED and REL_FORM Translator subroutines:

```
FLAG = 0;
DO WHILE FLAG = 0;
     DO FOR K = 1 TO 1
         CALL DATABANK (<MIDDLE_GIMBAL_ANGLE>);
         IF NOT (<NUMBER_<PN>> <MIDDLE_GIMBAL_LIMIT>) THEN
             K = 0;
     END;
     IF K = 1 THEN DO;
         COND = TRUE;
         FLAG = 1;
     END;
     ELSE REPEAT;
END;
IF COND = FALSE THEN DO;
```

A.2.2 External Test Actions

GOAL Statement:

```
SEND 10V to <POWER SELECTOR 1>,<POWER SELECTOR 2>;
```

Purpose:

This statement is used to implement an I/O operation to some specific external device.

```
DO FOR K = 1 TO 2;
    DIMENSION<PN> = <QUANTITY DIMENSION>;
    NUMBER<PN> = <QUANTITY VALUE>;
    CALL DATABANK(<AFD(K)>, <PN>);
END;
```

GOAL Statement:

```
APPLY PRESENT VALUE OF <POWER BUS 1> TO
    <POWER BUS 2>;
```

Purpose:

This statement is used to implement an input operation from power bus 1 and a corresponding output operation to power bus 2 with no intermediate storage in program variables.

HAL/S Equivalent Form:

```
DO FOR K = 1 TO 1;
    CALL DATABANK (<POWER BUS 1>,<PN>);
    CALL DATABANK (<POWER BUS 2>,<PN>);
END;
```

GOAL Statement:

```
ISSUE (OCTAL SEVENS), (OCTAL ONES) TO
  <PANEL LIGHTS 32>, <PANEL LIGHTS 31>;
```

Purpose:

This statement is supposed to send a "digital pattern" in the form of a numeric internal variable to a selected I/O word identified by the selected function designators. In this case, the internal variable (OCTAL SEVENS) is sent to "PANEL LIGHTS 32" and (OCTAL ONES) is sent to "PANEL LIGHTS 31";

HAL/S Equivalent Form:

```
DO FOR K = 1 TO 2;

   NUMBER<PN> = <OCTAL SEVENS>;

   CALL DATABANK (<PANEL LIGHTS 32>,<PN>);

   K = K + 1

   NUMBER<PN> = <OCTAL ONES>;

   CALL DATABANK (<PANEL LIGHTS 31>,<PN>);

END;
```

GOAL Statement:

```
ISSUE PRESENT VALUE OF <CH 63> TO <CH 11>;
```

Purpose:

As in the above example, this is simply some I/O of data to a particular set of channel addresses in some channel or channels of the I/O hardware.

HAL/S Equivalent Form:

```
DO FOR K = 1 TO 1;
      CALL DATABANK (<CH 63>,<PN>);
      CALL DATABANK (<CH 11>,<PN>);
END;
```

GOAL Statement:

```
S 104 AFTER <CLOCK> IS -1 HRS, OPEN <HELIUM SUPPLY>;
```

Purpose:

This GOAL statement has a time prefix, and a label prefix, used to control when and under what conditions the action of sending a bit valve corresponding to an OPEN valve to an external register symbolically identified by the function designator "HELIUM SUPPLY".

HAL/S Equivalent Form:

```
STEP_104:
   WAIT UNTIL -3600.001;
   STATE<PN> = OFF;
   DO FOR K = 1 TO 1;
        CALL DATABANK(<HELIUM SUPPLY>,<PN>);
   END;
```

GOAL Statement:

```
STEP 5:  TURN ON (THRUST OK IND) FUNCTIONS;
```

Purpose:

This GOAL statement is supposed to set all the bits in all the active function designators of a STATE table to the "ON" value.

HAL/S Equivalent Form:

```
STEP 5:  STATE<PN> = ON;
DO FOR K = 1 TO <KMAX>;
   IF THRUST_OK_INDA_K THEN
       CALL DATABANK(<THRUST_OK_INDR_K>,<PN>);
END;
```

GOAL Statement:

```
RECORD (INTERNAL TIME) TO <MAG 2-5>;
```

Purpose:

This statement is supposed to send data from a GOAL internal variable named "Internal time" to the function designator identified. This means in effect that an I/O operation of writing the internal value to the implicit I/O address of the function designator is required.

HAL/S Equivalent Form:

```
TEXT<PN> =  INTERNAL_TIMED;
DO FOR L = 1 TO 1;
   CALL DATABANK (<MA62-5>,<PN>);
END;
```

GOAL Statement:

```
DISPLAY TEXT (ALL SYSTEMS READY FOR POWER TRANSFER)
       TO <CRT 9>;
```

Purpose:

This statement is supposed to write the given text out onto a character-oriented I/O device, namely a display. Since this output is character-oriented, the equivalent HAL/S form will have a character output statement form.

HAL/S Equivalent Form:

```
TEXT PN = 'ALL_SYSTEMS_READY_FOR_POWER_
    TRANSFER';
DO FOR L = 1 TO 1;
    CALL DATABANK(<CRT 9>,<PN>);
END;
```

GOAL Statement:

```
AVERAGE 10 READINGS OF <IU COOLANT TEMPERATURE> AND
    SAVE AS (COOLANT TEMP);
```

Purpose:

The purpose of this statement is to read a hardware input channel 10 times, averaging the readings. No time delay other than the response time of the software is to be employed explicitly between successive readings.

HAL/S Equivalent Form:

```
DO;
  AV = 0;
  DO FOR I = 1 TO 10;
  CALL DATABANK(<IU COOLANT  TEMPERATURE>,<PN>);
  AV = AV + NUMBER<PN>;
  END;
  COOLANT_TEMPD = AV/NUM;
END;
```

GOAL Statement:

```
READ <PC STAGE INLET PRESSURE> AND SAVE AS
  (INLET PRESSURE);
```

HAL/S Equivalent Form:

```
CALL DATABANK(<PC STAGE INLET PRESSURE>,<PN>);

INLET_PRESSURED = NUMBER<PN>;
```

GOAL Statement:

```
READ (TABLE A) FUNCTIONS AND SAVE AS (CURRENT VALUE);
```

Purpose:

The purpose of this statement is to evaluate the current value of all the function designators (which are active) in the TABLE_A and assign the results into corresponding positions in the table column indicated.

HAL/S Equivalent Form:

```
DO FOR K = 1 TO <KMAX>;

    IF TABLE_AAK THEN IF  CURRENT_VALUEAK THEN DO;

    CALL DATABANK(<TABLE_ARK>,<PN>);

    CURRENT_VALUEDK,<CI> = NUMBER<PN>;

    END;

END;
```

GOAL Statement:

```
REQUEST TEXT (DEGREES PITCH) FROM <CRT 7> AND SAVE
   AS (DEG PTCH);
```

Purpose:

Display a request on the console and then reads in the result. Since a character-oriented operation with conversion to scalar is involved, a conversion function in the HAL/S version will be used.

HAL/S Equivalent Form:

```
DO FOR K = 1 TO 1;

      CONTROL<PN> = <INPUT>;
      CALL DATABANK(<CRT 7>,<PN>);
      <DEG_PTCH> = TEXT<PN>;
 END;
```

A.2.3 Internal Sequence Control

GOAL Statement:

```
DELAY 5 SECS;
```

Purpose:

Cause the program to "go to sleep" for a time interval of 5 seconds.

HAL/S Equivalent Form:

```
WAIT 5;
```

GOAL Statement:

```
WAIT UNTIL <SIVB 3200 PSIA SUP VENT> IS OPEN;
```

Purpose:

Cause the program to "go to sleep" for a time interval of unspecified length until the boolean (state) function designator is recognized as being "OPEN".

HAL/S Equivalent Form:

```
FLAG = 0;
DO WHILE FLAG = 0;
   DO FOR K = 1 TO 1;
       CALL DATABANK(<SIVB 3200 PSIA SUP VENT>,<PN>);
       IF NOT(STATE<PN> = ON) THEN
       K = 0;
   END;
IF K = 1 THEN FLAG = 1;
ELSE REPEAT;
END;
```

GOAL Statement:

```
GO TO S 20;
```

HAL/S Equivalent Form:

```
GO TO STEP_20;
```

where the label S 20 of GOAL has been translated into a HAL/S identifier.

GOAL Statement:

```
STOP  AND INDICATE RESTART LABELS S100, S200;
```

Purpose:

To give the GOAL program an ability to cease active execution and wait for operator intervention, followed by a jump to one of the indicated labels.

HAL/S Equivalent Form:

```
CONTROL<PN> = '3';
CALL DATABANK(0,<PN>);
TEXT<PN> = 'PROGRAM HAS STOPPED AT UNNUMBERED STATEMENT.
           INDICATE RESTART AT;
CALL DATABANK(0,<PN>);
TEXT<PN> = 'STEP_100';
CALL DATABANK(0,<PN>);
TEXT<PN> = 'STEP_200';
CALL DATABANK(0,<PN>);
AGAIN:  CONTROL<PN> = <INPUT RESTART LABEL>;
CALL DATABANK(0,<PN>);
```

```
LOC = 0;

CALL FLUSH;

IF TEXT = 'STEP_100' THEN GO TO STEP_100;

IF TEXT = 'STEP_200' THEN GO TO STEP_200;

CONTROL<PN> = 'ERROR';

CALL DATABANK(0,<PN>);

GO TO AGAIN;
```

GOAL Statement:

```
TERMINATE;
```

Purpose:

The TERMINATE statement of GOAL is used to:

Return control to the calling program if it is found in a GOAL subroutine.

Stop execution of a program if found at the program level, returning control to the caller.

HAL/S Equivalent Form:

```
RETURN;
```

GOAL Statement:

```
TERMINATE SYSTEM;
```

Purpose:

This statement shuts down an entire system of programs.

HAL/S Equivalent Form:

```
SET TERMSYS<PN>;
RETURN;
```

GOAL Statement:

```
REPEAT STEP 5 THRU STEP 7;
```

HAL/S Equivalent Form:

```
OUTPUT HAL/S:
     "RPT = RPT + 1;
RPTACT<RSN> = ON;
     SAVE_RPT = RS;
     RS = <RSN>;
     RPT CTR_RS = 1;
     GO TO STEP_5;
     RETURN_LABEL<RSN>:
     RS = SAVE_RPT;
     RPT = RPT_1;
```

A.2.4 Arithmetic/Logical Operations

GOAL Statement:

```
ASSIGN (FLAG B) = ON;
```

Purpose:

Set a new value of "ON" into the (FLAG B) internal name.

HAL/S Equivalent Form:

```
FLAG_B = TRUE;
```

This equivalence assumes the convention that "ON" has a binary value of "1" or "true". FLAG_B is assumed to be a HAL/S BOOLEAN.

GOAL Statement:

```
LET (A) = (A) + 1;
```

Purpose:

Assign a new value to GOAL internal variable (A) calculated as shown.

HAL/S Equivalent Form:

```
A = A + 1;
```

A.2.5 Execution Control

GOAL Statement:

```
CONCURRENTLY PERFORM PROGRAM (BE01);
```

HAL/S Equivalent Form:

```
SCHEDULE TASK <BE01>;
```

GOAL Statement:

```
CONCURRENTLY VERIFY <PRESS ENG 102 GIMBAL>
IS BETWEEN 1665PSIA and 1465PSIA and
DISPLAY EXCEPTION TO <CRT12>;
```

HAL/S Equivalent Form:

```
SCHEDULE TASK <NUM>;

TASK <NUM>:  TASK;

RESET TERM_SYS<NUM>;

BACK:  T_OLD = RUN_TIME;

DO FOR K = 1 TO 1;

       CALL DATABANK(<PRESS ENG 102 GIMBAL>,<PN>);

       IF NOT(<NUMBER<PN>> >= <1465>) AND

            <NUMBER<PN>> =< <16657>)

       THEN K = 0;

END;

IF K NOT = 1 THEN DO;

       TEXT<PN> = <DEFAULT MESSAGE>;

       DO FOR K = 1 TO 1

            CALL DATABANK(<CRT12>,<PN>);

       END;

END;
```

GOAL Statement:

RELEASE STEP 10;

Purpose:

Step 10 had "concurrently" set up some concurrent process at a previous time. This statement "releases" the concurrent process initiated, by removing it from the implicit executive queues involved.

HAL/S Equivalent Form:

TERMINATE TASK <PROGN>;

SET TERM_SYS$_{<PROGN>}$;

GOAL Statement:

RELEASE ALL

Purpose:

Terminate all concurrently scheduled operations within the current GOAL component.

HAL/S Equivalent Form:

TERMINATE TASK <ANUM$_k$>;

SET TERM_SYST$_{<ANUM_k>}$;

where k is cycled from 1 to the maximum provided in the arrays.

GOAL Statement:

```
PERFORM PROGRAM (LVDC POWER ON);
```

Purpose:

Branch to the program selected, execute it and return.

HAL/S Equivalent Form:

```
CONTROL<PN> = <PUSH INT ENVIRONMENT>;
CALL DATABANK(1,<PN>); /*SAVE INT ENVIRONMENT*/
RESET TERM_SYST<NUM>;
CALL G_LVDC POWER ON;
CONTROL = <POP_INT_ENVIRONMENT>
CALL DATABANK(1,<PN>); /*RESTORE INT ENVIRONMENT*/
IF TERM_SYST<NUM>THEN DO;
   RESET TERM_SYST<NUM>;
      SET TERM_SYST<PN>;
   RETURN;
END;
```

GOAL Statement:

```
PERFORM CRITICAL SUBROUTINE
    (CALCULATE DELAY TIME);
```

Purpose:

Inhibit all software-level interruptions by other system components (this does not refer to physical interrupts which the OS handles) during the execution of the subroutine.

HAL/S Equivalent Form:

```
CONTROL<PN> = <PUSH INT ENVIRONMENT>;
CALL DATABANK(1,<PN>);
CALL G_CALCULATE DELAY TIME (TRUE);
CONTROL<PN> = <POP INT ENVIRONMENT>;
IF TERM_SYST<PN> THEN DO;
RESET TERM_SYST<PN>;
RETURN;
END;
```

A.2.6 Interrupt Control

GOAL Statement:

```
WHEN INTERRUPT <POWER FAILURE> OCCURS
    GO TO STEP 9000;
```

Purpose:

Send control to step 9000 when the indicated interrupt occurs.

HAL/S Equivalent Form:

```
IF NOT C THEN DO;
        CONTROL<PN> = <ENABLE INTERRUPT>;
        NUMBER<PN> = <POWER FAILURE>;
        CALL DATABANK(1,<PN>);
ENVIRON(2,<IN>) = 0;
ENVIRON(3,<IN>) = <CN>;*
END;
```

GOAL Statement:

```
WHEN INTERRUPT <CLOCK T-22 MINS> OCCURS PERFORM
   SUBROUTINE (START_TANK CHILLDOWN) AND RETURN TO
   STEP 9999;
```

Purpose:

When the indicated interrupt occurs, perform a subroutine then unconditionally branch to the indicated step number.

HAL/S Equivalent Form:

```
IF NOT C THEN DO;
   CONTROL <PN> = <ENABLE INTERRUPT>;
   NUMBER<PN> = <CLOCK T-22 MINS.;
   CALL DATABANK(1,<PN>);
   ENVIRON(2,<IN>) = <NS>;
   ENVIRON(3,<IN>) = <CN>;*
END;
```

* IN = No. Assoc. with Interrupt.
CN = Case Number Assoc. with GO TO or RETURN TO option.
NS = Number Assoc. with Subroutine

GOAL Statement:

```
DISABLE STEP 20;
```

Purpose:

Inhibit a software interrupt set up by a "WHEN INTERRUPT" at the step indicated (STEP 20);

HAL/S Equivalent Form:

$CONTROL_{<PN>}$ = <DISABLE>;

$NUMBER_{<PN>}$ = <FD>;

CALL DATABANK(1,<PN>);

A.2.7 Table Control

GOAL Statement:

ACTIVATE (TABLE A) ROW 1, ROW 3;

Purpose:

Activate the function designators associated with the indicated rows so that future I/O to this table will include the rows in question.

HAL/S Equivalent Form:

The HAL/S translation of a GOAL table involves a data array and an activation array of BOOLEANS. Assuming that "$TABLE\_AA_1$ " is TABLE A's activation array, and that "TRUE' means the corresponding row is active, then the HAL/S equivalent is simply:

$TABLE\_AA_1$ = ON;

$TABLE\_AA_3$ = ON;

GOAL Statement:

INHIBIT (TABLE A) ROW 2, ROW 3;

Purpose:

Disable the function designators associated with the indicated rows, so that future I/O to this table will exclude the rows in question.

HAL/S Equivalent Form:

. Analogous to the ACTIVATE case:

```
TABLE_AA2 = OFF;

TABLE_AA3 = OFF;
```

A.3 System Statements

The GOAL system statements serve primarily as inputs governing the course of the GOAL-HAL/S translator's operation. Some of these statements have implications which are reflected in the HAL/S code produced.

A.3.1 Boundary Statements

GOAL Statement:

```
BEGIN DATA BANK (S2 DATA BANK) REVISION 0;
```

Purpose:

Mark the beginning of the Data Bank named for input to an appropriate data bank compilation. This statement should not be in the GOAL source. The Translator will generate an error message.

GOAL Statement:

```
BEGIN PROGRAM (LV TM CAL) REVISION 0;
```

Purpose:

Mark the beginning of a GOAL program.

HAL/S Equivalent Form:

```
G_LV_TM_CAL_0:  PROCEDURE EXCLUSIVE;
```

GOAL Statement:

```
BEGIN SUBROUTINE (FORCE_TERM) (PARAMETER_1);
```

Purpose:

This statement begins a GOAL subroutine block, passing one formal parameter.

HAL/S Equivalent Form:

```
G_FORCE_TERM:  PROCEDURE (C) ASSIGN (PARAMTER_1D);
```

GOAL Statement:

```
BEGIN MACRO AZ (PARAMETER 1);
```

Purpose:

This statement specifies the start of a GOAL source language macro. Since macros only refer to the source code and are expanded within the translator, there is no HAL/S equivalent.

This does not exclude use of a HAL/S macro form as part of the generated HAL/S source code if a GOAL-to-HAL/S translation is done at the source language level. But such use is completely separate from the GOAL macro and its use.

GOAL Statement:

```
END DATA BANK;
END PROGRAM;
END SUBROUTINES:
END MACRO;
```

Purpose:

Mark the end of the particular GOAL block (or component).

HAL/S Equivalent Form:

Since MACRO forms exist only in the source inputs to the GOAL-HAL translation, the END statement for this block is devoid of a HAL/S equivalent.

The HAL/S equivalents of END PROGRAM and END SUBROUTINE are provided by the HAL/S CLOSE statement; i.e.

CLOSE;

GOAL Statement:

LEAVE:

Purpose:

Link to some other language subroutine (in object form). This is an operating system function which has no equivalent in HAL/S and is illegal.

GOAL Statement:

RESUME;

Purpose:

Return to GOAL compiling after LEAVE. See comment in the discussion of "LEAVE" above.

A.3.2 System Directive Statements

The GOAL system directive statements are USE, FREE, and SPECIFY - all of which refer to the DATA BANK.

These statements have no HAL/S equivalent (see Section 3.0) and are system-oriented inputs to the translation process. All data bank references are resolved by the translator - the data bank (as such) does not appear in the output of the translator.

A.3.3 Special Aid Statements

GOAL Comment Statement:

```
$ POWER TRANSFER SWITCH VERIFICATION;
```

Purpose:

Annotate listing.

HAL/S Equivalent Form:

a. Embedded Comment

```
/* POWER TRANSFER SWITCH VERIFICATION */
```

b. Comment Line

```
pos  1
  C      POWER TRANSFER SWITCH VERIFICATION
```

Note: Comments map directly into comments with identical text and minor syntax changes.

GOAL Statement:

```
EXPAND MACRO ADJUST, <AC SIGNAL>,
     (0.5V), 5340,;
```

Purpose:

Expand a GOAL substitution macro prio to further compilation.

HAL/S Equivalent Form:

None. All macros involve expanding GOAL source statements, so the macro itself and its expansion disappear in the translation process.

GOAL Statement:

```
REPLACE <POWER SUPPLY NO 1>
     WITH <POWER SUPPLY NO 2>;
```

Purpose:

Replace is a source level substitution of characters prior to compilation, a sort of "mini-macro" facility. As with macros, it disappears following translation because it must be expanded in order to translate.

REFERENCE LIST

1) Flanders, et al; Final Report on Shuttle Avionics and the GOAL Language, including the Impact of Error Detection and Redundancy Management, Intermetrics, Inc., Cambridge, Mass., June 1973.

2) Flanders, et al; Final Report on GOAL-to-HAL Translation Study, Intermetrics, Inc., Cambridge, Mass., 1973.