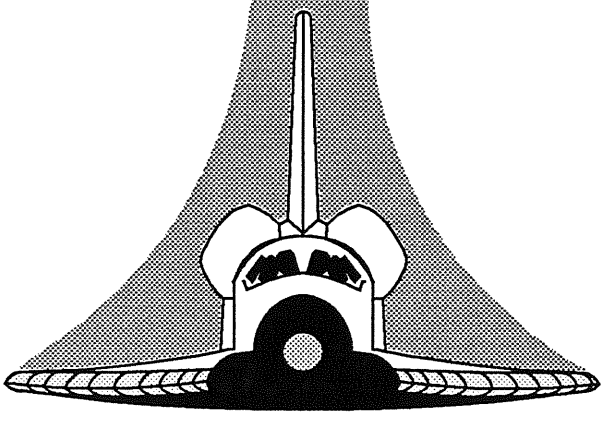
KSC-LPS-033-03
July 1996
Baseline
S G2 Release

LPS CCMS GROUND OPERATIONS AEROSPACE LANGUAGE (GOAL) USER GUIDE Part 1



LPS SYSTEM
SOFTWARE
DOCUMENTATION

Prepared By:

Lockheed Space Operations Company
Integrated Data Systems
LPS Software Quality and Data Analysis
LPS Documentation

Contract NAS 10-10900

Send corrections to this document to: Lockheed Space Operations Company, ATTN: LPS Documentation , USK-252, 1100 Lockheed Way, Titusville, FL 32780

UPDATE HISTORY

BASELINE S E3 — **June 1995**

This document has been converted into FrameMaker publication software, baselined, and includes changes released as authorized by DCN and published in conjunction with software delivery and valid only for software updated through the S E3 Release Level. The intent of the following ESR/SPR is documented in the attached pages per the following:

DCN#	ESR/SPR	IPR/SPR	Release	Section(s)
3732	CDS0960	000X-3364	S E36	17,34,54,73
3732	CDS0968	000X-3355	S E36	
3732	CMM0674	000X-2847	S E36	
3732	FR21200	000X-3171	S E36	
3790	CDS0973	000X-3629	S E36	56,57
3843	PIAMO.2202		S E37	90

BASELINE S G2 — **July 1996**

This document change released as authorized by DCN and published in conjunction with software delivery and valid only for software updated through the S G2 Release Level. This document is being baselined this release to include reformatting of the section numbering. The intent of the following ESR/SPR is documented in the attached pages per the following:

DCN#	ESR/SPR	IPR/SPR	Release	Section(s)
000438	K45252		S G23	1,7,12,16,B,C,D

THIS PAGE INTENTIONALLY LEFT BLANK.

RELEASE EFFECTIVITY OF SECTIONS

SECTION NUMBER	TITLE	G2
1	INTRODUCTION	X
2	FUNDAMENTALS OF GOAL	X
3	NOMENCLATURE	X
4	DECLARATION STATEMENTS	X
5	ARITHMETIC AND LOGIC STATEMENTS	X
6	BRANCHING STATEMENTS	X
7	RECORD AND DISPLAY STATEMENTS	X
8	COMMAND AND MONITOR STATEMENTS	X
9	PROCEDURE CONTROL STATEMENTS	X
10	EXECUTION CONTROL STATEMENTS	X
11	CCMS CONTROL STATEMENTS	X
12	INTERRUPT PROCESSING STATEMENTS	X
13	DISK FILES	X
14	IN-LINE SUBROUTINES	X
15	SYSTEM STATEMENTS	X
16	GOAL ELEMENTS	X
17	GOAL COMPILER DIRECTIVES	X
18	PROCEDURE EFFICIENCY CONSIDERATIONS	X
19	GOAL SHORT FORMS	X
A	GOAL DIMENSIONS	X
B	SYSTEM FUNCTION DESIGNATORS	X
C	GLOSSARY	X

RELEASE EFFECTIVITY OF SECTIONS

SECTION NUMBER	TITLE	G2
D	LIST OF ACRONYMS	X
E	FEP EXCEPTION RE-ROUTING	X

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Section Page

TABLE OF CONTENTS (Continued)

Section Page

TABLE OF CONTENTS (Continued)

Section Page

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Section **Page**

THIS PAGE INTENTIONALLY LEFT BLANK.

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

THIS PAGE INTENTIONALLY LEFT BLANK.

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK.

1. INTRODUCTION

THIS PAGE INTENTIONALLY LEFT BLANK.

1.1 PURPOSE

The purpose of this document is to provide the detailed information required by the test engineer to develop Ground Operations Aerospace Language (GOAL) test and control procedures to accomplish Space Shuttle Vehicle test and checkout at Kennedy Space Center (KSC).

1.2 SCOPE

The GOAL USER GUIDE provides a detailed description of GOAL including the details of each language statement and each major option as well as many examples. The following information about each statement is also presented: Statement Execution, Error Processing, Restrictions/Limitations, and Legal Function Designators.

Additional topics covered in this document include the compiler directives and instructions for developing GOAL procedures. Reference material provided in the appendices includes syntax diagrams, GOAL Dimensions, system Function Designators, and an index of GOAL statements and elements.

1.3 SYSTEM OVERVIEW

GOAL is an integral part of the Launch Processing System (LPS). GOAL is the language used by test engineers to accomplish test and checkout of the Space Shuttle Vehicle for the Shuttle Launch and Landing Project at KSC. GOAL test procedures are compiled and configured on the Central Data Subsystem (CDS) and executed on the Checkout, Control, and Monitor Subsystem (CCMS).

1.3.1 GOAL OVERVIEW

GOAL is a high order test language drawing from several languages in addition to NASA's experience in space vehicle testing. GOAL is a test engineer-oriented language, which is designed to be used as the standard procedure terminology and test programming language in performance of ground checkout operations in a space vehicle test and launch environment. It encompasses a wide range of testing, including vehicle systems and subsystems preflight checkout, ground preflight operations such as propellant transfer, support systems verification, ground power control and monitoring. The language is compatible with a wide variety of engineering design, requiring primarily command/response action (analog and digital) to the system to be tested. It may be used in the checkout of line replaceable units, both on-board

preflight, and in the shop. GOAL permits a high degree of readability and retainability by providing the necessary operators required for testing, expressed in a familiar notation. Therefore, it is easily learned and understood by personnel not necessarily skilled in programming techniques.

1.3.2 CDS FUNCTIONS

The Support Software required to prepare GOAL procedures for execution is resident in CDS. The GOAL procedures are processed into low level interpretive code by the GOAL Language Processor (GLP), which resides in CDS. The GOAL System Configurator, also resident in CDS, inserts hardware information from the CCMS Data Bank into the interpretive code and resolves references to other procedures and display skeletons. Configured GOAL procedures are integrated into a Test Configuration ID (a catalog of software required to accomplish a test from CCMS) and transferred to disks on the CCMS consoles.

1.3.3 CCMS FUNCTIONS

Vehicle and Ground Support Equipment (GSE) test and control is accomplished by CCMS. A block diagram of the CCMS Firing Room 1 Set is shown in Figure 1-1. The basic function of the Front End Processors (FEP's) is to process measurement data and issue stimuli to Vehicle and GSE systems. The Processed Data Recorder (PDR) performs recording of test data to disk and magnetic tape for near real time and historical retrieval. The Shared Peripheral Area (SPA) is used for post processing data reduction. The Printer/Plotter Subsystem provides a stripchart-recording function. The Consoles are used to execute GOAL test procedures as well as to provide a man-machine interface.

Execution of GOAL procedures in a Console is controlled by the GOAL Executor, which allows direct communication with the console operator for control and monitoring of procedures, and is the primary means of interfacing GOAL procedures with Vehicle and GSE systems as well as elements of CCMS.

1.3.3.1 GOAL/GSE FEP Interface

Testing and control of GSE is accomplished from GOAL procedures by means of the GSE FEP's. The GSE FEP's constantly poll Hardware Interface Modules (HIM's) to acquire GSE measurement data. Measurement data obtained from the HIM's are stored in the Common Data Buffer (CDBFR). GOAL procedures in any console may acquire this measurement data from the CDBFR. Whereas GOAL procedures in

any console may read measurement data, only GOAL procedures executing in the Master, Integration, or responsible console may issue stimuli to GSE. GSE FEP's may also send interrupts to GOAL procedures upon request when analog measurements exceed limits or discrete values change to exception states.

1.3.3.2 GOAL/Downlink Pulse Code Modulation (PCM) FEP Interface

The downlink PCM FEP's are the Operational Flight Instrumentation (OFI), Main Engine (ME), Swinger, and External Tank (ET) PCM FEP's. These FEP's process PCM measurement data at predefined sample rates and store this data in the CDBFR for GOAL procedures to access. The downlink PCM FEP's may also send interrupts to GOAL procedures to notify them of measurement exception conditions.

1.3.3.3 GOAL/LDB FEP Interface

GOAL procedures may communicate with on-board systems including Test Control Supervisor (TCS), Software Avionics Command Support (SACS), and Launch Sequence via the Launch Data Bus (LDB) FEP. GOAL procedures may perform such on-board interface functions as: issue commands (set discretes, apply analogs, issue digital patterns), read measurements, read and write to GPC memory, issue equivalent Display Electronics Unit (DEU) commands, control the LDB I/O function, interact with TCS sequences (STOP, RESUME, CANCEL), and perform TCS sequences and explicitly coded programs. Refer to KSC-LPS-OP-033-04 for GOAL on-board interface statements pertaining uniquely to the LDB FEP interface.

1.3.3.4 GOAL/Uplink Command FEP Interface

GOAL procedures executing in the Master, Integration, or responsible console may send commands to the on-board systems via the Uplink Command FEP. Basically the same capabilities are available via the Uplink Command interface as are available via the LDB FEP interface with the exception that read operations are not possible via the Uplink. Refer to KSC-LPS-OP-033-04 for GOAL on-board interface statements pertaining to the UPLINK COMMAND FEP interface.

1.3.3.5 GOAL/PDR Interface

GOAL procedures may send data to the PDR to be recorded on tape for historical reference. In addition, the GOAL Executor logs the start and stop times as well as any requested trace data of each procedure executed to the PDR.

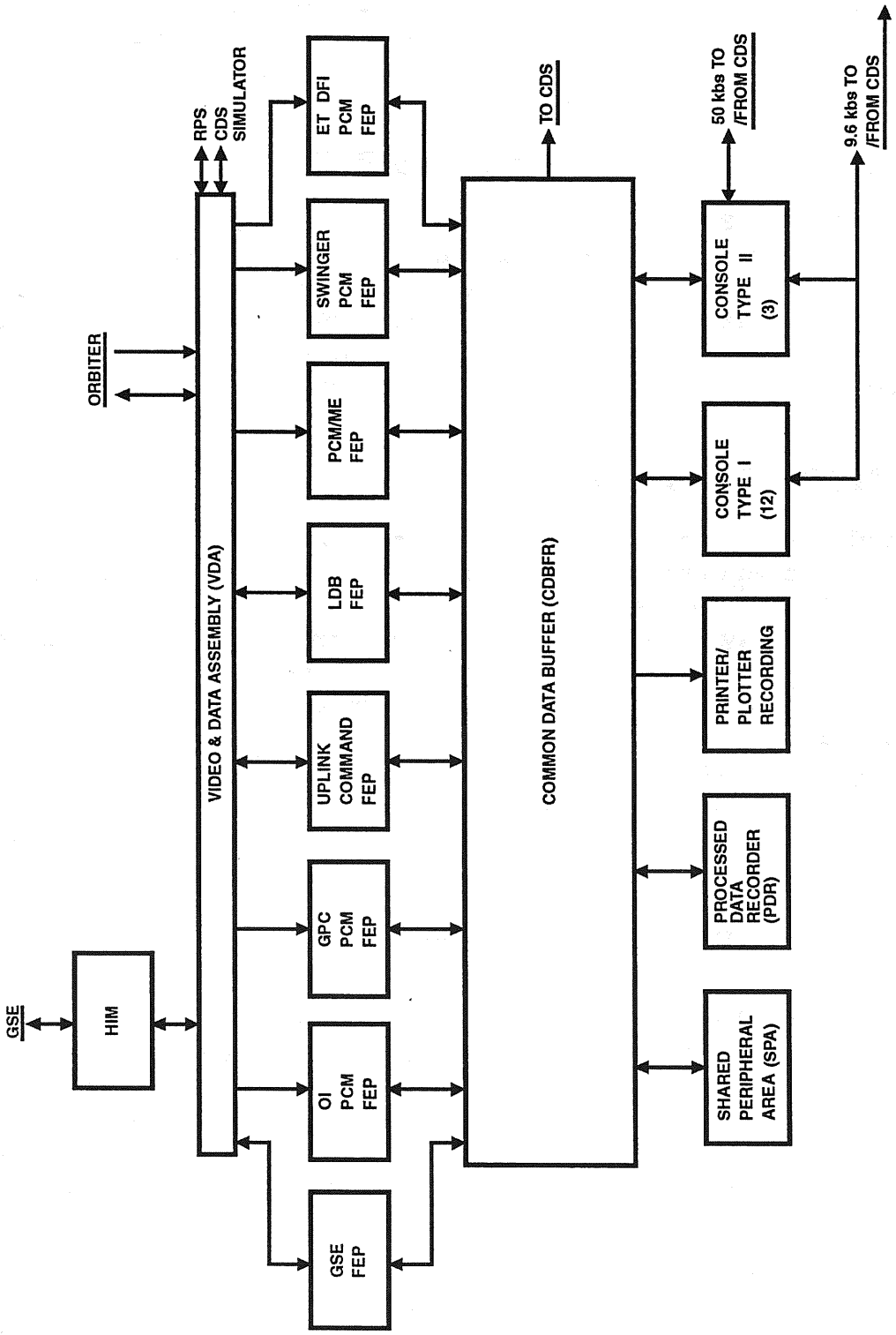


Figure 1-1 Firing Room 1 Set

1.3.3.6 GOAL/SPA Interface

GOAL procedures may send data to the SPA line printer to be printed in the form of a report. A report will be initiated when the SPA printer is first referenced in a task and added to each time the procedure requests data to be sent to the SPA. When the task is completed, the report will be printed.

1.3.3.7 GOAL/Printer/Plotter Subsystem Interface

GOAL procedures may send requests to the Printer/Plotter Subsystem to initiate and terminate plots of CDBFR resident measurements.

1.3.3.8 GOAL/Console Interfaces

The GOAL interfaces with the console include: The Data Generator (DG) CRT's and keyboards, Programmable Function Panels (PFP's), CDBFR, moving head disk, console printer/plotter, and Control Logic.

DG CRT's and Keyboards

The GOAL Executor will provide a display capability for each GOAL task, which utilizes twelve application pages and a shared application page in each console computer:

A. When a GOAL task is performed, it is assigned two display pages. These two display pages are referenced as PAGE-A and PAGE-B. PAGE-A will be used by the GOAL Executor for any messages pertinent to the execution of this task. The task name and active level name will be displayed on the status line of the CRT when the page is being viewed.

B. Each GOAL task will have the capability to display to each of the twelve application pages and the shared application page. The application pages may be referenced as PAGE-A and PAGE-B and by physical page assignment. This display capability also applies to a prompt output by a procedure and to skeleton assignments. Prompt output is provided when the procedure expects a reply from the operator in response to a RECORD statement.

C. Each GOAL procedure will have the capability to request notification of an operator action on a display page. If so requested, a single operator action may be routed to

more than one GOAL procedure. It should be noted that the exception to this is the response to a prompt. All prompt responses will be routed only to the procedure which contained the prompt request.

D. The housekeeping required for a display page is strictly under procedural control. The only time the system will initialize a page is when a skeleton is assigned to the page.

Programmable Function Panels

The GOAL Executor will supply the procedures with the capability of displaying to and receiving interrupt notifications from the PFP's.

A. When a GOAL task is initiated, it is assigned a PFP presentation. This presentation and the function keys corresponding to each half-line are referenced by the procedures using the default Function Designators.

B. Each GOAL procedure will have the capability of displaying to and receiving interrupt notifications on any of the six PFP presentations. Interrupts from a single function key will be routed to all procedures requesting notification. A PFP half-line does not have to be written to before an interrupt is processed.

C. PFP function key 7 has two functions: (1) it is used to call up a menu of PFP presentations assigned to application tasks and (2) it is used to generate an interrupt to a GOAL procedure, which may be used to provide a disarm function.

D. As with display pages, the housekeeping of the PFP LED's and lights is left to the procedure writer.

CDBFR

GOAL procedures may read measurement values from the CDBFR. They may also read and set pseudo measurements in the CDBFR. The following types of pseudo measurements are allowed: Pseudo discretes, pseudo analogs, and pseudo digital patterns. Pseudo measurements may be assigned to a responsible console, or they may be designated as global. Pseudo measurements assigned to a responsible console are also stored in page one. Global and

specific pseudo measurements are stored in page one of the CDBFR and may be read or changed by GOAL procedures in any console.

There are 128 words per console private write area and 2048 words in page one currently allocated for pseudo measurements. Pseudo discretes are packed 16 discretes per word. Pseudo analogs require three words per analog, and pseudo digital patterns require one word.

Moving Head Disk

GOAL procedures may store data on the moving head disk at the console for later retrieval by the same or another GOAL procedure. Six disk files are provided per console. Each disk file contains 512 (16-bit) words. A GOAL procedure may write data to a file using the WRITE option of the RECORD DATA Statement. Any GOAL procedure executing in the same console may read any disk file using the READ Statement.

Console Printer/Plotter

GOAL procedures may print data on the console printer/plotter.

Control Logic

The GOAL Executor will route all commands with prerequisite sequences through Control Logic.

1.3.3.9 GOAL/Remote Communication Interface

In order for one procedure to adequately communicate with another for data communication or control purposes, the capability is provided in the GOAL Language for one procedure to interrupt another. This capability is incorporated via a SEND INTERRUPT Statement which specifies a destination and a communication serial number. The communication serial number is created by defining a remote communication Function Designator in the data bank. The send interrupt may be issued to a GOAL procedure resident in the same or another console. To use this capability the Function Designator must be defined and referenced in a SPECIFY INTERRUPT Statement. The same Function Designator used on a SEND interrupt will cause the event to be passed to the procedure(s) with the interrupt specified. Any time a GOAL procedure wishes to communicate data to another, the data may be stored in the CDBFR or on the disk and an interrupt sent. The procedure receiving the interrupt must be in execution and must have a remote communication interrupt outstanding.

1.3.3.10 GOAL/CDS Interface

GOAL procedures may send data to the CDS to be recorded. This data is sent to CDS via the CDS First-In-First-Out (FIFO) interface to the CDBFR. Also, the GOAL Executor enables GOAL procedures to send commands to CDS by writing into predefined CDBFR locations (defined by pseudo measurements) and logging to CDS.

1.3.4 GOAL PROCEDURE EXECUTION

The first procedure initiated in a given partition is called the mainline, or level 1 procedure. Each mainline procedure may nest to four levels. This means that the mainline performs a procedure, which in turn performs a procedure, and so on until four procedures have been performed (counting the mainline). This process is illustrated as follows:

```
Level 1    Mainline
Level 2    Procedure A
Level 3    Procedure B
Level 4    Procedure C
```

The mainline and all the procedures performed under it is referred to as a task or concurrency. A maximum of six tasks may execute concurrently in a given console.

GOAL procedures may perform non-GOAL programs as well as TCS sequences and explicitly coded programs in the on-board, General Purpose Computer (GPC).

A method is provided for the execution of critical GOAL tasks by locking out the processing of normal tasks until the critical procedure waits for an input/output operation or terminates.

1.3.5 GOAL INTERRUPTS

The GOAL Executor will accept notification of events (interrupts) to which the procedure is to respond. These events consist of measurement exceptions, PFP function keys, timers, and system interrupts and DG function keys to include cursor transmits, programmable function keys, and execute and disarm function keys.

The GOAL Executor, upon receipt of an interrupt, will pass the notification to all procedures requiring the knowledge of the event. The interrupt will be processed by the procedure only when the procedure is active, or becomes active.

1.3.6 GOAL ERROR CLASSIFICATIONS

The GOAL Executor will detect run time errors and communicate them to the operator. There are four classes of run time errors. The error classes and the action taken by the GOAL Executor when they occur are as follows:

Class I - Terminal Errors

Example: Disk failure which disallows Executor Functions.
Action: Terminate the GOAL procedure.

Class II - Critical Errors

Example: Trying to perform more levels than allowed.
Action: Stop the GOAL procedure.

Class III - External Run Time Errors

Example: Command issue failure, prerequisite sequence failure.
Action: Allow the procedure to continue if procedure error override is specified Default to Class II if the procedure does not specify procedure error override.

Class IV - Minor Errors

Example: Invalid cursor coordinates used when referencing a display page.
Action: Output a message to PAGE-A and continue execution of the procedure.

Class V - Unsuccessful response from a Computer to Computer (CTC) error.

Example: FEP command issue failure. Results in Class IV error message from the GOAL Executor followed by a Class III error message from the FEP.
Action: Same as Class III.

THIS PAGE INTENTIONALLY LEFT BLANK.

2. FUNDAMENTALS OF GOAL

THIS PAGE INTENTIONALLY LEFT BLANK.

The basic fundamentals of GOAL are based upon a natural language for procedure development. The format and basic elements utilized by GOAL are described in the following paragraphs.

2.1 GOAL FORMAT

The "statement" format was selected as the most natural form for preparing test procedures. An unrestricted free field format was adopted for the flexibility it afforded in statement positioning and paragraphing during procedure development. Judiciously used, this feature should greatly enhance the readability and the logical layout of the procedure. This format is also readily adaptable to various input media: computer cards, remote terminals, etc. The free field concept minimizes the importance of blanks or spaces within a statement. Free field format is maintained in the GOAL Source Listing but not in the Expanded Source Listing.

2.2 CHARACTER SET

The GOAL character set is compatible with the USA Standard Code for Information Interchange (ASCII) and the Extended Binary Coded Decimal Interchange Code (EBCDIC).

The GOAL character set consists of capital letters, numerals, and special characters.

CAPITAL LETTERS:

A-Z

NUMERALS:

0-9

SPECIAL CHARACTERS:

ASTERISK	*	SLASH	/
BLANK	b	SEMICOLON	;
COMMA	,	DECIMAL POINT	.
CURRENCY	$	LEFT ANGLE BRACKET	<
EQUAL	=	RIGHT ANGLE BRACKET	>
PLUS	+	LEFT PARENTHESIS	(
MINUS	-	RIGHT PARENTHESIS	)

Only these characters are used in forming the language words and parameters. However, special characters, such as engineering graphics, may be used in the generation of display formats. See

the syntax for Symbols (Figure 16-5) for special characters legal in TEXT CONSTANT only.

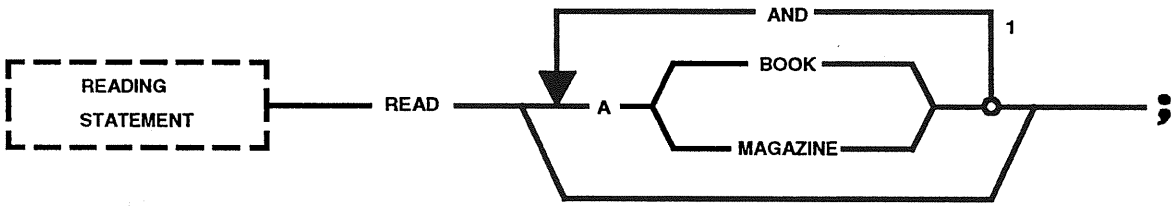
2.3 EXTERNAL/INTERNAL NAMES

A convention was adopted that allows recognition of system names that are unique in the procedure because they are system software or hardware dependent. These items are enclosed in angle brackets; e.g., <MAIN POWER>. Items so named are centrally maintained and are made available to the Language Processor through the Data Bank. These special items are called Function Designators (refer to Section 16.4 for a description of the Function Designator Types). All other names used in the language are enclosed in parentheses; e.g., (PRESS SAVE). Because blanks are not significant when the processor is building or checking a Name or Function Designator, care must be used in selecting Names. For instance, (A B) and (AB) would be interpreted as the same Name.

2.4 SYNTAX DIAGRAMS

To illustrate the different allowable variations of each statement, a presentation method using syntax diagrams was selected. Syntax diagrams identify legal sequences of items in a GOAL statement, including alternate branches, optional entries, and feedback loops. The following is an example of a syntax diagram illustrating a "READING STATEMENT". Note the use of the semicolon as a statement terminator.

EXAMPLE 1

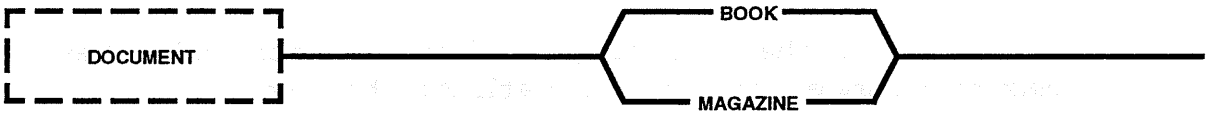


This allows any of the following sentences to be written:

```
READ;
READ A BOOK;
READ A MAGAZINE;
READ A BOOK AND A MAGAZINE;
READ A MAGAZINE AND A BOOK;
READ A BOOK AND A BOOK;
READ A MAGAZINE AND A MAGAZINE;
```

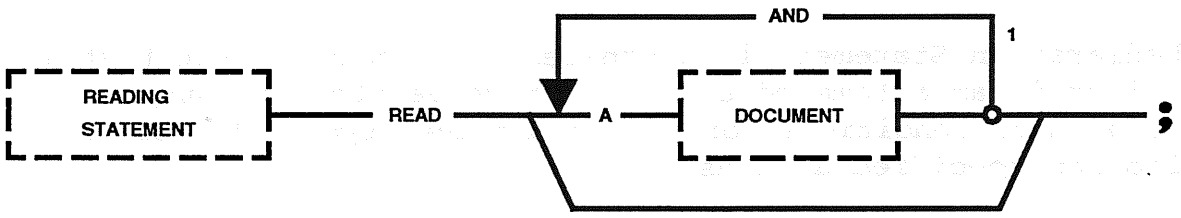
If the use of "book" and "magazine" appeared the same way in several diagrams and represented a logical grouping, then a new syntax unit could be created as illustrated in Example 2.

EXAMPLE 2



Example 1 would then become:

EXAMPLE 3



The dashed box represents a syntax unit. The syntax unit on the left is being defined in terms of "characters" and other syntax units.

Some basic rules for using syntax diagrams are:

A. Syntax diagrams are read from left to right except for feedback loops.

B. _____ is a connecting path.

C. Blanks and comments may be added freely to enhance readability.

D. Capital letters must be used as shown.

E. Diagonal lines are alternate forward paths.

F. A bubble indicates the start of a return (feedback) path.

G. A numeral at the beginning of a return path indicates the maximum number of times a path may be taken.

H. Syntax notes provide semantical explanation.

I. GOAL statements terminate with a semicolon.

2.5 STATEMENT CLASSIFICATIONS

The principal types of GOAL Statements are: Declaration Statements, Procedural Statements, and System Statements.

2.5.1 DECLARATION STATEMENTS

A Declaration Statement is a non-test action statement that is required in an automated system to reserve storage, supply initial data conditions, or to declare the "type" of data that is valid for specified actions.

GOAL declarations may be considered as either simple data declarations, list declarations, or table declarations. The simple data declaration provides a unique name for a data entry. The list is used to assign a single name to a string of data entries. The table is used to specify a "rows and columns" format.

For each of the declaration structures, the data "type" must also be indicated. The allowable types are Numeric, Quantity, State, or Text.

The following list identifies the Declaration Statements:

DECLARE DATA Statement
DECLARE NUMERIC LIST Statement
DECLARE NUMERIC TABLE Statement
DECLARE QUANTITY LIST Statement
DECLARE QUANTITY TABLE Statement
DECLARE STATE LIST Statement
DECLARE STATE TABLE Statement
DECLARE TEXT LIST Statement
DECLARE TEXT TABLE Statement

The general format of a Declaration Statement is:

DECLARE "TYPE" "STRUCTURE" "ENTRIES"

Example

DECLARE NUMERIC LIST (LIST 1) WITH 4 ENTRIES 1,2,3,4;

The Declaration Statements will be discussed in Section 4.

2.5.2 PROCEDURAL STATEMENTS

Procedural Statements are the GOAL test action statements that specify the commands to be issued, measurements to be tested, and control of the execution sequence. The general structure of a Procedural Statement is the same as a simple imperative English sentence, with the subject understood to be the computer. The minimum requirement for a Procedural Statement is a verb; however, most statements also contain an object to receive the action. An optional phrase may be used to modify the action: that is, tell when, how often, or how long to perform the action.

Example

OPTIONAL MODIFIER	VERB	OBJECT
AFTER <CLOCK> IS		
-3 HRS 30 MIN	OPEN	<INLET SUPPLY VALVE>;

2.5.3 SYSTEM STATEMENTS

System Statements act as directives to the Language Processor or serve as Boundary Statements in signaling the Language Processor of the beginning, or the end, of a component. The general format is:

Example

```
VERB          OBJECT

BEGIN         PROGRAM (POWER UP);
```

The following list identifies the System Statements:

- BEGIN PROGRAM Statement
- END PROGRAM Statement
- COMMENT Statement
- BEGIN MACRO Statement
- END MACRO Statement
- EXPAND MACRO Statement
- BEGIN SEQUENCE Statement
- END SEQUENCE Statement
- DEFINE Statement
- SPECIFY INTERRUPT Statement

The System Statements will be discussed in Sections 15-15.5.2 with the exception of the SPECIFY INTERRUPT Statement which is discussed in Section 12.4 and the DEFINE Statement which is discussed in Section 9.3.

The Procedural Statements are presented in Sections 8-12.8. The Procedural Statements along with some of their related GOAL elements are classified as follows for presentation in this document:

ARITHMETIC AND LOGIC STATEMENTS

- ASSIGN Statement
- LET EQUAL Statement
- NUMERIC FORMULA
- LOGICAL FORMULA
- AVERAGE Statement

BRANCHING STATEMENTS

- GO TO Statement
- VERIFY PREFIX
- REPEAT Statement
- PERFORM STATEMENT GROUPS Statement
- END STATEMENT GROUPS Statement
- STATEMENT GROUP LABEL

RECORD AND DISPLAY STATEMENTS

- RECORD DATA Statement
- MODIFY DISPLAY Statement
- CLEAR DISPLAY Statement
- DISPLAY SKELETON Statement
- OUTPUT EXCEPTION

COMMAND AND MONITOR STATEMENTS

- APPLY ANALOG Statement
- SET DISCRETE Statement
- TEST AND SET Statement
- ISSUE DIGITAL PATTERN Statement
- READ Statement
- READ FEP STATUS Statement

PROCEDURE CONTROL STATEMENTS

- CONCURRENT Statement
- PARAMETER
- PSEUDO PARAMETER
- DEFINE Statement
- RELEASE CONCURRENT Statement
- PERFORM PROGRAM Statement
- STOP Statement
- TERMINATE Statement

EXECUTION CONTROL STATEMENTS

- DELAY Statement
- CONTINUE Statement
- ACTIVATE TABLE Statement
- INHIBIT TABLE Statement

CCMS CONTROL STATEMENTS

TIMER CONTROL Statement
ACTIVATE SYSTEM Statement
INHIBIT SYSTEM Statement
CHANGE Statement
LOAD SAFING SEQUENCE Statement
ACTIVATE PLOT Statement
INHIBIT PLOT Statement
CHANGE ANALOG PLOT Statement
ACTIVATE CRITICAL MODE Statement
INHIBIT CRITICAL MODE Statement

INTERRUPT PROCESSING STATEMENTS

SPECIFY INTERRUPT Statement
ACTIVATE NOTIFICATION Statement
INHIBIT NOTIFICATION Statement
WHEN INTERRUPT Statement
SEND INTERRUPT Statement
RELAY INTERRUPT Statement
RETURN INTERRUPT Statement

2.6 GOAL PROGRAM ORGANIZATION

The first statement of a GOAL PROGRAM must be a BEGIN PROGRAM Statement which defines the name of the program and specifies any required input parameters. This statement may be followed by DECLARE and DEFINE Statements which are used to reserve space for variable data and to describe the input parameters. The SPECIFY INTERRUPT Statements must appear before any Procedural Statement. After all Declaration Statements have been specified, the Procedural Statements of the program are given. No Declaration Statements may be embedded in the Procedural Statements. The last statement of a program must be the END PROGRAM Statement. This statement must be preceded by a TERMINATE Statement or a statement which causes a branch back into the program.

3. NOMENCLATURE

THIS PAGE INTENTIONALLY LEFT BLANK.

In order to provide meaningful examples, the following standard nomenclature will be used in the examples in this document. The notation in parentheses following each Function Designator type is the data bank type which corresponds to it.

FUNCTION DESIGNATORS

Analog Measurements (AM)	< AM1 >	, < AM2 >	. . .
Analog Stimuli (AS)	< AS1 >	, < AS2 >	. . .
Discrete Measurements (DM)	< DM1 >	, < DM2 >	. . .
Discrete Stimuli (DS)	< DS1 >	, < DS2 >	. . .
Digital Pattern Measurements (DPM)	< DPM1 >	, < DPM2 >	. . .
Digital Pattern Stimuli (DPS)	< DPS1 >	, < DPS2 >	. . .
Pseudo Analogs (PA)	< PA1 >	, < PA2 >	. . .
Pseudo Digital Patterns (PDP)	< PDP1 >	, < PDP2 >	. . .
Pseudo Discretes (PD)	< PD1 >	, < PD2 >	. . .
Time Homogeneous Data Set (THDS)	< THDS1 >	, < THDS2 >	. . .
Countdown Clock (CDT)	< CDT >		
Greenwich Mean Time Clock (GMT)	< GMT >		
Interval Timer (TIMR)	< TIMER >		
Mission Elapsed Time Clock (CDT)	< METIME >		
Julian Time of Year (CDT)	< JTOY >		

INTERNAL NAMES

A. NAMES

1. Number Patterns

(A)	Integer	(N1) ,(N2) , . . .
(B)	Binary	(B1) ,(B1010) , . . .
(C)	Octal	(T1) ,(T377) , . . .
(D)	Hexadecimal	(X1) ,(X3FF) , . . .

2. Quantity (Q1) , (Q2) , . . .

3. State (STATE1) , (STATE2) , . . .

4. Text (TEXT1) , (TEXT2) , . . .
(GRAPHIC1) , (GRAPHIC2) , . . .

B. LISTS

1. Numeric (NLIST1) , (NLIST2) , . . .

2. Quantity (QLIST1) , (QLIST2) , . . .

3. State (SLIST1) , (SLIST2) , . . .

4. Text (TLIST2) , (TLIST2) , . . .

C. TABLES

1. Numeric (NTABLE1) , (NTABLE2) , . . .

2. Quantity (QTABLE1) , (QTABLE2) , . . .

3. State (STABLE1) , (STABLE2) , . . .

4. Text (TTABLE1) , (TTABLE2) , . . .

4. DECLARATION STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The Declaration Statements will be discussed in this section. These statements are used to define data which a GOAL procedure may use in calculations, may update or compare against other data, etc. In other words, these statements define working space for the GOAL procedure's use in execution. The Declaration Statements are alphabetized by class and are as follows:

DECLARE DATA STATEMENTS (Section 4.1)
DECLARE NUMBER STATEMENT (Section 4.1.1)
DECLARE QUANTITY STATEMENT (Section 4.1.2)
DECLARE STATE STATEMENT (Section 4.1.3)
DECLARE TEXT STATEMENT (Section 4.1.4)

DECLARE LIST STATEMENTS (Section 4.2)
DECLARE NUMERIC LIST STATEMENT (Section 4.2.1)
DECLARE QUANTITY LIST STATEMENT (Section 4.2.2)
DECLARE STATE LIST STATEMENT (Section 4.2.3)
DECLARE TEXT LIST STATEMENT (Section 4.2.4)

DECLARE TABLE STATEMENTS (Section 4.3)
DECLARE NUMERIC TABLE STATEMENT (Section 4.3.1)
DECLARE QUANTITY TABLE STATEMENT (Section 4.3.2)
DECLARE STATE TABLE STATEMENT (Section 4.3.3)
DECLARE TEXT TABLE STATEMENT (Section 4.3.4)

THIS PAGE INTENTIONALLY LEFT BLANK.

4.1 DECLARE DATA STATEMENTS

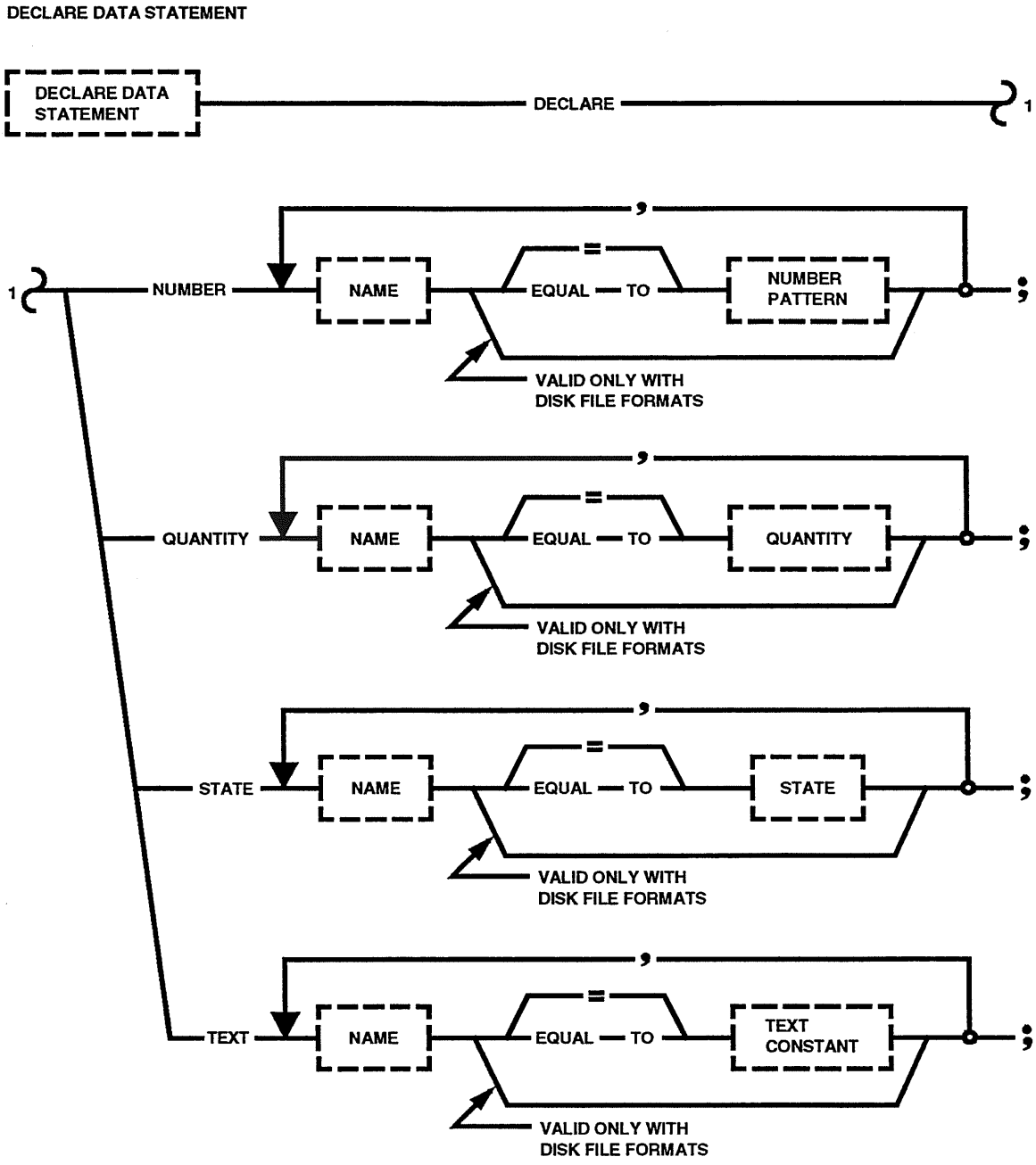


Figure 4-1 DECLARE DATA Statement

Function

The DECLARE DATA Statement is used to define the name and characteristics of single data items. The Data Declarations to be discussed in this section are as follows:

DECLARE NUMBER STATEMENT (Section 4.1.1)
DECLARE QUANTITY STATEMENT (Section 4.1.2)
DECLARE STATE STATEMENT (Section 4.1.3)
DECLARE TEXT STATEMENT (Section 4.1.4)

Description

The DECLARE DATA Statement is used to define Internal Names. All Internal Names used in a GOAL procedure must be explicitly declared with Declaration Statements. The reference to Name in the syntax diagram refers to an Internal Name which is a single data item (as opposed to a list or table). A Name may refer to Internal Names which have the following data types: Number (or Numeric), Quantity, State, and Text. The DECLARE DATA Statement may be used to define any of these data types.

Declarations For Pseudo Parameters

A pseudo parameter is a Name or a Function Designator used to identify input data received by a program from another program. All pseudo parameters which are Names must be defined by Declaration Statements. These pseudo parameters must use the following default declaration options:

NUMERIC: B, X, I, T.

QUANTITY: Dimension only.

STATE: ON/OFF, OPEN/CLOSED, TRUE/FALSE, WET/DRY.

TEXT: Character Count.

A disk file record item entry is a name used to identify a data item or data items in a given position in each disk file record. The type and attributes of each disk file record item must be defined by Declaration Statements. The disk file record format entries must use the following default declaration options:

NUMERIC: B, X, I, T.

QUANTITY: Dimension only.

STATE: ON/OFF, OPEN/CLOSED, TRUE/FALSE, WET/DRY.

TEXT: Character Count.

A disk file record item name that has already been used in a BEGIN DISK FILE record format may be subsequently used within a BEGIN DISK FILE record format by specifying the type and name only. No attribute data (i.e., subtype, engineering units, character, count, etc.) will be allowed on the subsequent declaration.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Internal Names of different data types may not be declared in the same DECLARE DATA Statement.

B. Initial values must not be provided if the declared Name is a pseudo parameter.

C. In a Text Constant, blanks and commas are significant.

D. Any ASCII character may be used in a Text Constant with the exception of parenthesis and ampersand.

E. The maximum length of a Text Constant is 80 characters.

F. Initial values must not be defined if the Internal Name is in a disk file record format specification.

G. If an Internal Name is used within a BEGIN DISK FILE record format specification, the name can be subsequently used within a BEGIN DISK FILE record format by specifying the type and name only. The Internal Name will carry the attributes of the original declaration. No attribute data (i.e., subtype, engineering units, character count, etc.) can be specified with the subsequent declares.

4.1.1 DECLARE NUMBER STATEMENT

The DECLARE NUMBER option is used to define a Name which refers to a number. The allowable types of numbers are defined by the Number Pattern element; i.e., binary, hexadecimal, integer, and octal numbers. A Name may be defined to have an initial value EQUAL TO (=) a number. A Name may optionally be defined as one of the four number types without an initial value. This is accomplished by declaring Name equal to B, X, I, or T, which refer to binary, hexadecimal, integer, and octal respectively. When this type of declaration is used, the value of Name defaults to zero.

Examples

```
DECLARE NUMBER (N5) EQUAL TO 5;
DECLARE NUMBER (B2) = B10,
               (X1) = X 3AF,
               (T9) = T 375,
               (N9) = -33;
DECLARE NUMBER (B1) = B,
               (X1) = X,
               (T1) = T,
               (N1) = I;
```

The first example defines the Name (N5) as an Integer, which is initialized to 5. The second example defines and initializes binary, hexadecimal, octal, and integer numbers. The third example defines binary, hexadecimal, octal, and integer numbers without initial values.

4.1.2 DECLARE QUANTITY STATEMENT

The DECLARE QUANTITY option is used to define a Name equal to a Quantity. Quantity may be a Decimal Number followed by a Dimension, a Decimal Number, a Dimension, or a Time Value. If a Decimal Number without a Dimension is specified, it refers to a dimensionless constant; e.g., a ratio or a coefficient. If only a Dimension is specified, the value of Name is initialized to zero with the Dimension specified.

Examples

```
DECLARE QUANTITY (Q5) = 5V;
DECLARE QUANTITY (Q15) = 7.55,
                 (Q16) = 1 MIN 5 SEC,
                 (Q18) = - 20 MIN CDT,
                 (Q19) = GMT,
                 (Q10) = AMP;
```

The first example declares the Quantity Name (Q5) initialized to 5 volts. The second example declares multiple Quantity Names. The Name (Q15) is initialized to the decimal value 7.55. The Names, (Q16) and (Q18), are initialized to time values. The Names (Q19) and (Q10) are declared as variables with attributes of time and amperes, respectively, with default initial values of zero.

4.1.3 DECLARE STATE STATEMENT

The DECLARE STATE option is used to define a Name equal to a State. Allowable values for State are ON, OFF, OPEN, CLOSED, TRUE, FALSE, WET, DRY. State may optionally be specified using the defaults ON/OFF, TRUE/FALSE, OPEN/CLOSED, and WET/DRY to indicate the operational sense without giving an initial value. State declarations using the defaults should be initialized to a specific state by procedural statements prior to use in the GOAL procedure.

Examples

```
DECLARE STATE (STATE1)  = ON;
DECLARE STATE (STATE6)  = OPEN,
              (STATE7)  = CLOSED,
              (STATE11) = TRUE,
              (STATE2)  = ON/OFF;
```

The first example declares the State Name (STATE1), initialized to ON. The second example declares multiple State Names. The last State Name in this declaration (STATE2) is declared with the default values ON/OFF.

4.1.4 DECLARE TEXT STATEMENT

The DECLARE TEXT option is used to define a Name equal to a Text Constant. Text Constant is used to describe a character string of text or a group of graphic characters. For the character string of text, all ASCII characters are valid except parenthesis. The graphic characters are named by their key cap numbers. Successive graphic character names must be separated by commas. Two successive commas are used to represent a blank.

When an initial text value is not provided, a character count may be used to specify the length of the Text Constant. In this case, the compiler initializes a character field of length specified with blanks. These Text Constants should be initialized prior to any reference by the programs.

Examples

```
DECLARE TEXT (TEXT1) = TEXT(OPTION 3);
DECLARE TEXT (TEXT2) = TEXT(ABC 123*,/),
             (TEXT7) = GRAPHIC(36U,,) TEXT(A),
             (TEXT8) = 25 CHARACTERS;
```

The first example declares the Text Name (TEXT1) initialized with the text value (OPTION 3). The second example declares multiple Text Names. The second Text Name (TEXT7) is declared initialized to a Text Constant consisting of a Graphic, a blank, and a Text character. Consecutive commas are used to declare blanks. The last Text Name (TEXT8) is declared with the default of 25 CHARACTERS, causing the name to be initialized to 25 blanks.

4.2 DECLARE LIST STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The list declarations to be discussed in this section are as follows:

DECLARE NUMERIC LIST STATEMENT (Section 4.2.1)
DECLARE QUANTITY LIST STATEMENT (Section 4.2.2)
DECLARE STATE LIST STATEMENT (Section 4.2.3)
DECLARE TEXT LIST STATEMENT (Section 4.2.4)

A list is used to assign a single name to a group of related data.

Example

```
DECLARE STATE LIST (SLIST1) WITH 3 ENTRIES OFF,
                                           ON,
                                           OFF;
```

The example illustrates the declaration for a State List with Name, (SLIST1). The List has 3 entries with values OFF, ON, OFF.

All of the entries of a List must be of the same data type, but subtypes may be mixed within a list. For example, for State data, the following subtypes may be within a List: ON, OFF, OPEN, CLOSED, TRUE, FALSE, WET, DRY.

The LIST DECLARATION statement may be used to initialize an entire list to a single value (e.g., 5 V) or to a subtype (e.g., ON). Lists or list elements may also be defined using default values without initializing the List or List element.

The default values are:

NUMERIC:	B, X, I, T.
QUANTITY:	Dimension only.
STATE:	ON/OFF, OPEN/CLOSED, TRUE/FALSE, WET/DRY
TEXT:	Character count.

Default values for Numeric and Quantity data are initialized to zero by the compiler. Defaults for Text data are initialized to blanks. Defaults for State data are not initialized by the compiler. They must be initialized by procedural statements in the GOAL procedure.

List References

Once a list has been defined with a Name, it may be referenced as a List Name, which is an Internal Name; e.g., (SLIST1). A list element is referenced by specifying the List Name and the element number. For example, (SLIST1)2 refers to the second element in (SLIST1), which has a value of ON. In referencing the element, a numeric variable (i.e., LIST INDEX) may also be used; for example (SLIST1) (N2) may also refer to the second element in (SLST1) if (N2) is equal to 2.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The minimum number of entries in a list is one and the maximum number is 100.

B. If initial values or subtypes are specified for two or more entries in the list, then the same must be done for all entries in the list.

C. Initial values must not be defined if the declared Name is a pseudo parameter.

D. A list may not contain Function Designators. A list is only for data.

E. If individual items of a list are to be selected by an Internal Name index, the first element of the list will be the criterion used for compatibility checking.

F. Initial values must not be defined if the declared Name is a disk file record format specification.

G. If an Internal Name is used within a BEGIN DISK FILE record format specification, the name can be subsequently used within a BEGIN DISK FILE record format by specifying the Type and Name only. The Internal Name will carry the attributes of the original declaration. No attribute data (i.e., subtype, engineering units, character count, etc.) can be specified with the subsequent declares.

THIS PAGE INTENTIONALLY LEFT BLANK.

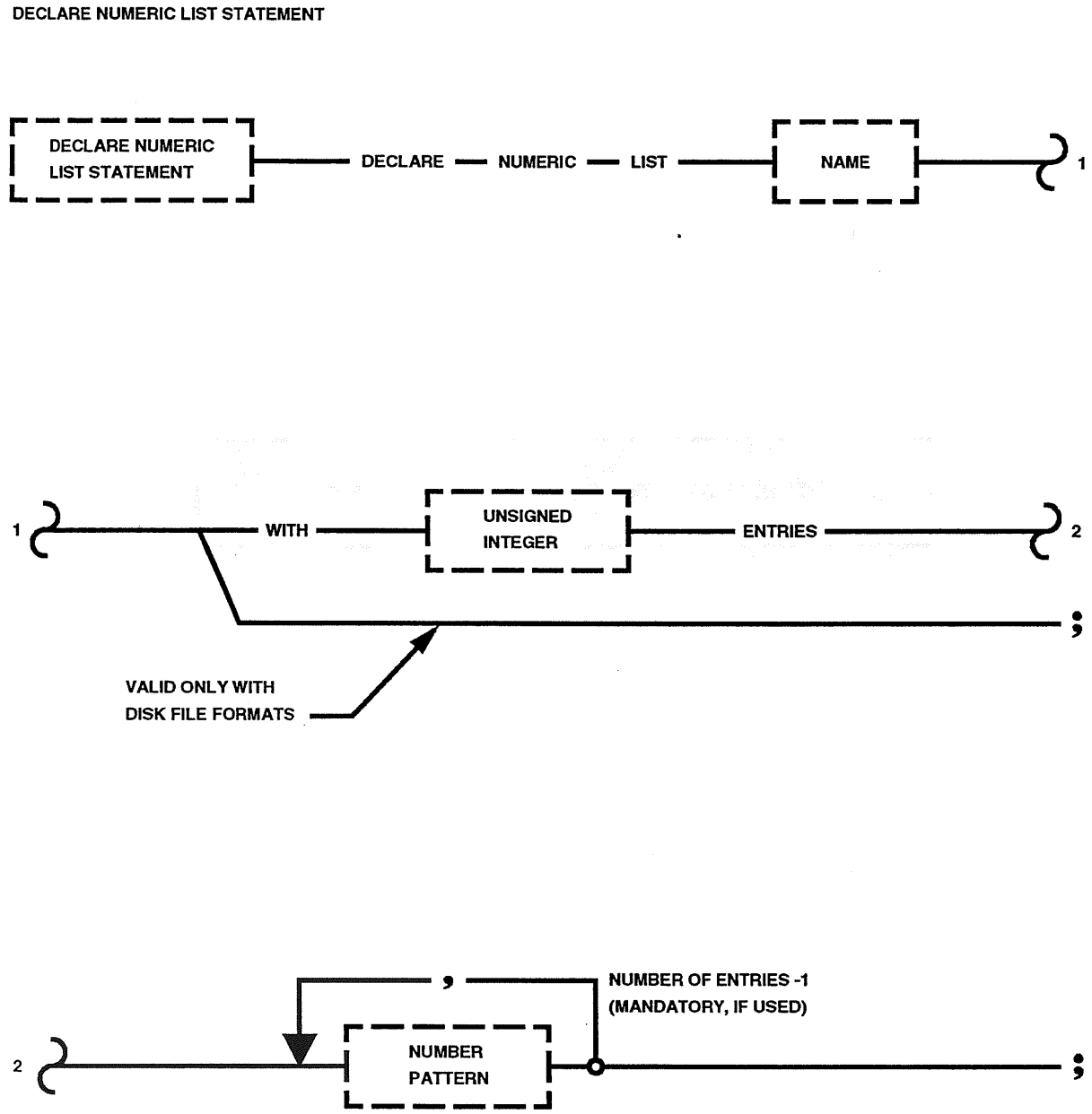


Figure 4-2 DECLARE NUMERIC LIST Statement

4.2.1 DECLARE NUMERIC LIST STATEMENT

Function

The DECLARE NUMERIC LIST Statement is used to define the Name and characteristics of a list of Numeric data items.

Description

The data items in a Numeric List must be Number Patterns; i.e., binary, hexadecimal, integer, and octal numbers. Various Number Pattern formats may be mixed in a given list. The entire list may be initialized to a single value or subtype (B, X, I, T).

Examples

```
DECLARE NUMERIC LIST (NLIST1) WITH 3 ENTRIES B;
DECLARE NUMERIC LIST (NLIST2) WITH 3 ENTRIES I,
    B  101,
    T  377;
DECLARE NUMERIC LIST (NLIST3) WITH 3 ENTRIES
    X 3FAB,
    B 1100,
    32760;
```

The first example defines the Numeric List (NLIST1), as having three binary entries without initial values specified. The result is that the values default to zero. The second and third examples illustrate Numeric Lists with mixtures of Number Pattern formats.

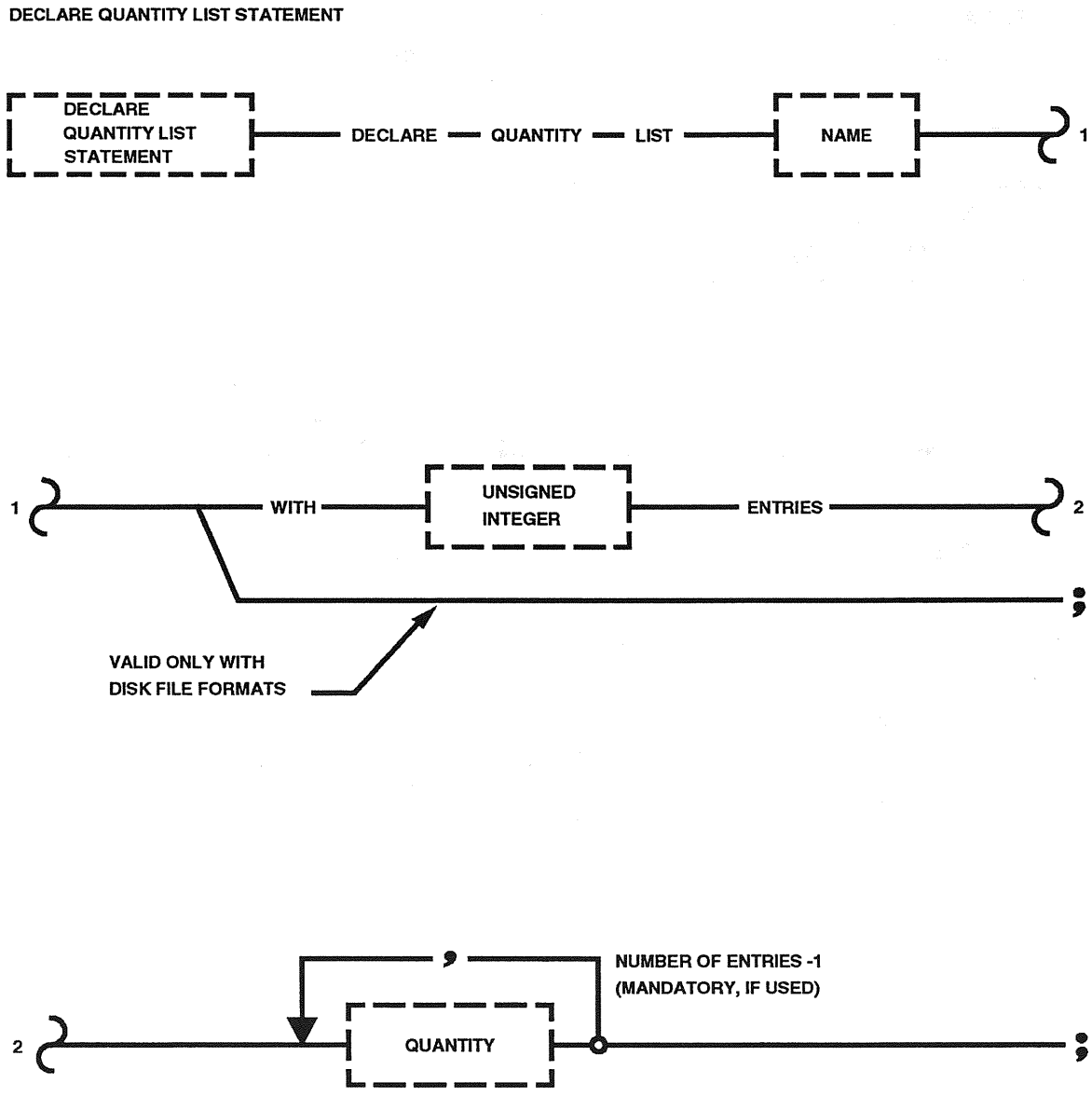


Figure 4-3 DECLARE QUANTITY LIST Statement

4.2.2 DECLARE QUANTITY LIST STATEMENT

Function

The DECLARE QUANTITY LIST Statement is used to define the Name and characteristics of a list of Quantity data items.

Description

The entries in a Quantity List must conform to the specifications for Quantity type data. Various dimensional types may be mixed in a given list. The whole list may be initialized to a single value or subtype (i.e., by specifying a dimension only).

Examples

```
DECLARE QUANTITY LIST (QLIST1) WITH 3 ENTRIES  5 V;
DECLARE QUANTITY LIST (QLIST2) WITH 3 ENTRIES 35.5,
                                              V,
                                          7 AMP;
DECLARE QUANTITY LIST (QLIST3) WITH 50 ENTRIES V;
```

The first example defines a Quantity List, (QLIST1), with 3 entries initialized to 5 volts. The second example defines a Quantity List with mixed dimensional types. The third example defines a Quantity List with 50 entries of the subtype, volts.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Time Value dimensional types may not be mixed with non-time value dimensional types in a given Quantity List. All non-time value dimensional types may be mixed in a given list.

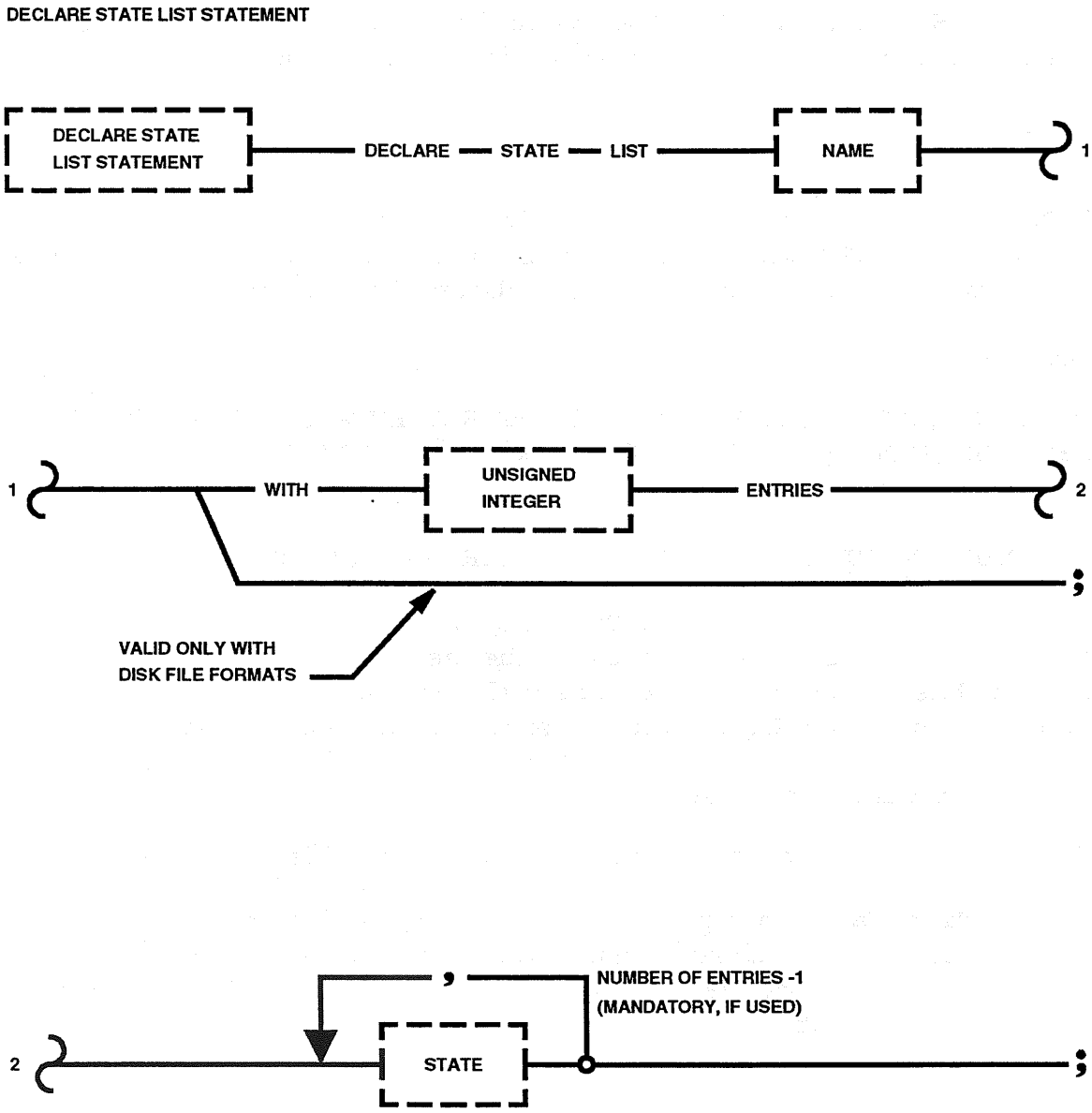


Figure 4-4 DECLARE STATE LIST Statement

4.2.3 DECLARE STATE LIST STATEMENT

Function

The DECLARE STATE LIST Statement is used to define the Name and characteristics of a list of State data items.

Description

The entries in a State List must be State type. Various State subtypes may be mixed in a given list. The whole list may be initialized to a single value or default subtype (ON/OFF, OPEN/CLOSED, TRUE/FALSE, WET/DRY).

Examples

```
DECLARE STATE LIST (SLIST1) WITH 3 ENTRIES ON;
DECLARE STATE LIST (SLIST2) WITH 3 ENTRIES OPEN,
                                           FALSE,
                                             DRY;
DECLARE STATE LIST (SLIST3) WITH 50 ENTRIES ON/OFF;
```

The first example defines the State List, (SLIST1), with three entries initialized to ON. The second example defines the State List, (SLIST2), with mixed state subtypes. The third example defines the State List, (SLIST3), with 50 entries of the subtype, ON/OFF.

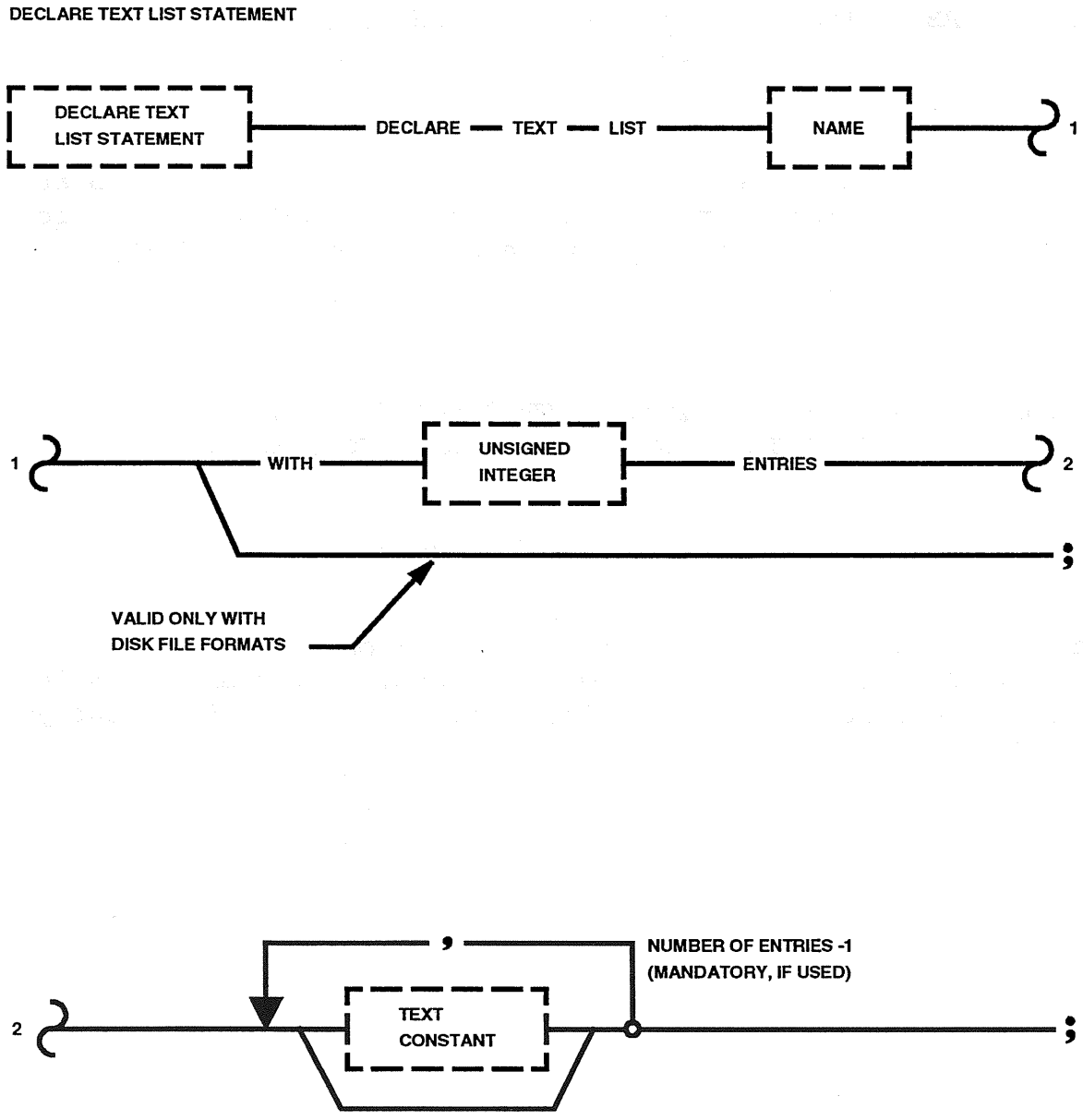


Figure 4-5 DECLARE TEXT LIST Statement

4.2.4 DECLARE TEXT LIST STATEMENT

Function

The DECLARE TEXT LIST Statement is used to define the Name and characteristics of a list of Text data items.

Description

The entries in a Text List must be Text Constants (i.e., Text or Graphic characters). The whole list may be initialized to a single Text Constant or character string length. Various length Text Constants may be mixed in a given list.

A default entry in a list may be indicated by specifying a comma. It will have a length equal to the maximum specified entry. All entries in a list will have a length equal to that of the longest entry. Shorter entries are padded, with blanks, to this length. The contents or length of the first entry of a list must be specified.

Examples

```
DECLARE TEXT LIST (TLIST1) WITH 3 ENTRIES TEXT (**ABC123);
DECLARE TEXT LIST (TLIST2) WITH 3 ENTRIES
    TEXT(FEEDBACK LOOP OPTION),,
    GRAPHIC(36U,,,,36L,44L);
DECLARE TEXT LIST (TLIST3) WITH 50 ENTRIES 7 CHARACTERS;
```

The first example declares the Text List, (TLIST1), with three entries equal to the character string, **ABC123. The second example declares a List, (TLIST2), with three entries; i.e., a Text Constant, a default text entry (20 blanks), and a group of graphic characters padded with blanks. The third example declares a List, (TLIST3), with 50 entries, each of which has 7 characters reserved for it.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The contents or length of the first entry of a Text List must be specified. All entries in a Text List have a length equal to the longest item in the list.

THIS PAGE INTENTIONALLY LEFT BLANK.

4.3 DECLARE TABLE STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The table declarations to be discussed in this section are as follows:

DECLARE NUMERIC TABLE STATEMENT (Section 4.3.1)
DECLARE QUANTITY TABLE STATEMENT (Section 4.3.2)
DECLARE STATE TABLE STATEMENT (Section 4.3.3)
DECLARE TEXT TABLE STATEMENT (Section 4.3.4)

All of the table declarations have basically the same characteristics. The similarities will be discussed in the introduction, and the same material will not be duplicated for each TABLE DECLARATION Statement; rather, the differences will be pointed out.

Example

```
DECLARE STATE TABLE (STABLE1) WITH 3 ROWS AND 2 COLUMNS TITLED

   WITH ENTRIES    (COLUMN A), (COLUMN B)

  < DS1 > ,          ON ,        OFF,
  < DS2 > ,          OFF,        ON ,
  < DS3 > ,          OFF,        ON ;
```

The basic format of the TABLE DECLARATION Statements is illustrated by the example. The name of a table is an Internal Name and therefore must be unique within a GOAL procedure. The name of the table in the example is (STABLE1). Following the table name, the number of rows and columns in the table must be defined. The "S" on ROWS and COLUMNS is optional so that a table may be defined as having one row or one column. The columns of the table may be titled if desired, but this is not required.

The columns in the example are titled (COLUMN A) and (COLUMN B). A column name must be unique with respect to all other Internal Names except for column names of another table.

Each row of a table must begin with a unique Function Designator called a Row Designator, to identify that row. A Row Designator may not be repeated within a table. The Row Designators must be of the same Function Designator type. For example, load and sensor discrete Function Designators may not be mixed within a table.

The data items in a table may be initialized. For example, the data items in the example are initialized to ON and OFF. Default values may also be specified for the data items in a table in

order to reserve storage within the table. Examples of default values are: ON/OFF, OPEN/CLOSED, etc.

The data items in a table must be of the same type as the table data type. For example, State data items may only appear in State tables, and Quantity data items may only appear in Quantity tables. Also, Function Designator types must be compatible with the table type. However, the subtypes of the data items in a table do not have to be the same as the subtypes of the Row Designators. For example, a Row Designator may be defined in the Data Bank as OPEN/CLOSED, but it may have a row entry in a table which is specified as ON. Taking another example, a Row Designator, which is a Load Analog, may have row entries specified as SEC.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Tables may have from 1 to 100 rows and from 0 to 10 columns.

B. Column titles may not be used for tables with zero columns.

C. Column names must be unique within a table.

D. The data items in a table must be of the same data type as the table data type.

E. Function Designator types in a table must be compatible with the table data type.

F. Load and Sensor Function Designators may not be mixed in the same table; however, Pseudo Function Designators may be mixed with either of the two.

Table Referencing

A row of a table may be referred to in a GOAL statement by specifying the row number or the Row Designator (refer to the Internal Name element for additional details). Row numbers begin with 1. For example, the row number corresponding to < DS1 > in the example is number 1. ROW 1 may also be referenced by specifying the Row Designator. Columns may be referred to by number or title. In the example, the title (COLUMN A) is the same as COLUMN 1 and (COLUMN B) is the same as COLUMN 2.

An element of a table may be specified by referring to the table name, row, and column. The following references are used to specify the first data item in the table, (STABLE1).

Examples

```
(STABLE1) ROW 1 COLUMN 1
(STABLE1) < DS1 > COLUMN 1
(STABLE1) ROW 1 (COLUMN A)
```

The term, FUNCTIONS, is used to refer to all of the Row Designators in a table. This provides a handy notation for working with groups of Function Designators.

Examples

```
SET (STABLE1) FUNCTIONS TO OFF;
SET < DS1 >
    < DS2 >
    < DS3 > TO OFF;
SET (STABLE1) FUNCTIONS TO ROW 1 (COLUMN B);
```

All of the preceding examples accomplish the same result; i.e., to SET the discrete stimuli referenced to OFF. Similarly, an entire column of a table may be referenced by its column name or number.

Examples

```
SET (STABLE1) FUNCTIONS TO (COLUMN B);
SET (STABLE1) FUNCTIONS TO COLUMN 2;
SET (STABLE1) FUNCTIONS TO OFF,
                           ON,
                           ON;
```

The preceding examples all accomplish the same result.

Table Formatting

Function Designator expansion during compilation may result in skewing rows of the table or in wraparound of the rows onto the next line. This problem can be prevented if 34 character positions are skipped between the Row Designators and the commas to allow space for insertion of the 34-character descriptors. If the Row Designators in the table do not have 34-character descriptors, then the problem of skewing and wraparound will not exist.

Another problem which may be encountered when working with large tables is overrunning the compiler limit of 110 cards per statement. Tables of over 50 rows should be single-spaced for this reason.

THIS PAGE INTENTIONALLY LEFT BLANK.

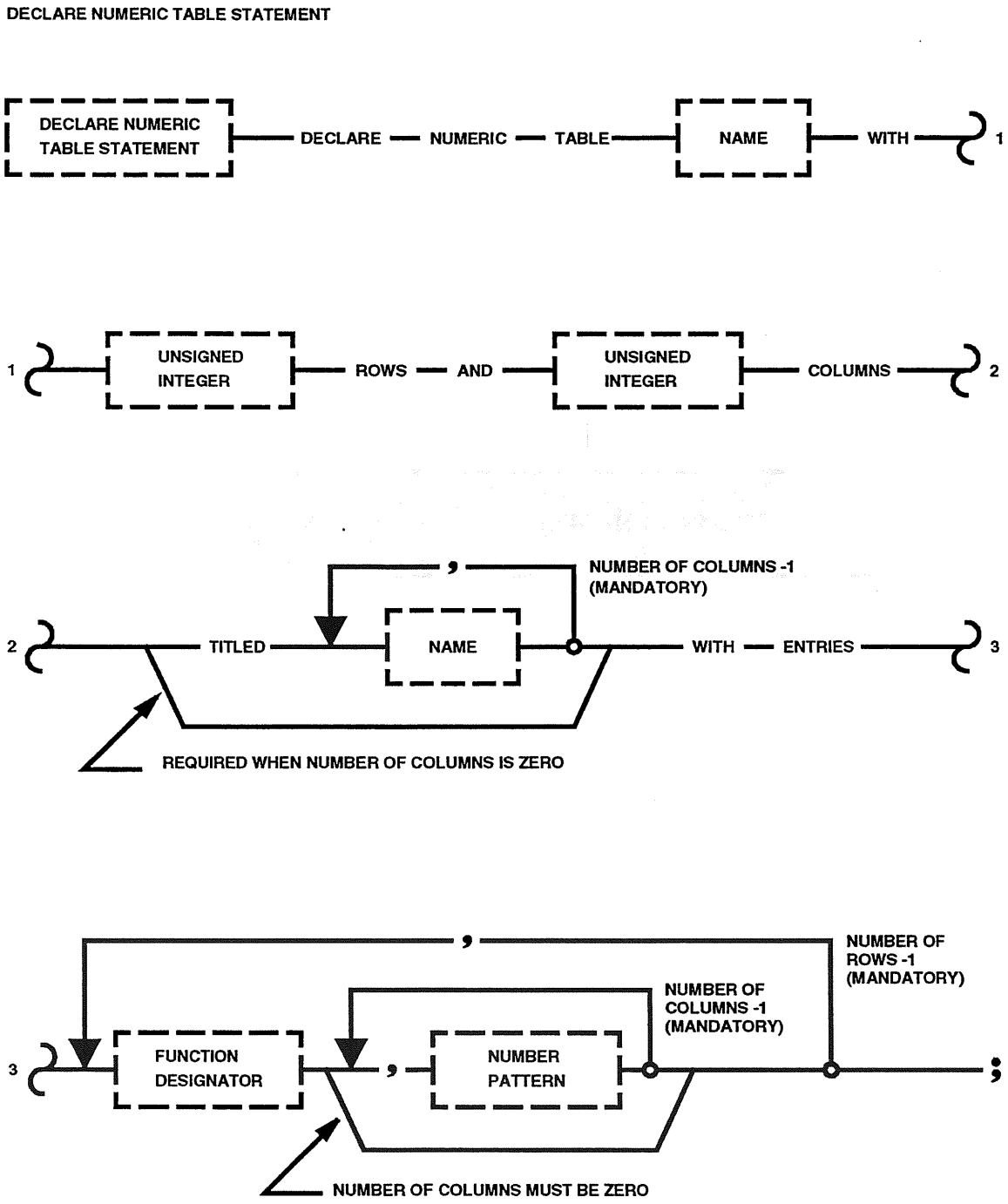


Figure 4-6 DECLARE NUMERIC TABLE Statement

4.3.1 DECLARE NUMERIC TABLE STATEMENT

Function

The DECLARE NUMERIC TABLE Statement is used to define the name and characteristics of a table of Numeric data items.

Description

The DECLARE NUMERIC TABLE Statement is used to declare a table of Numeric data items. The data items must be Number Patterns. The data items may be initialized or a default subtype (B, X, I, T) may be specified.

Examples

```
DECLARE NUMERIC TABLE (NTABLE1) WITH 3 ROWS AND 2 COLUMNS TITLED

                        (COLUMN A),     (COLUMN B)

    WITH ENTRIES

    < DPS1 >,           B101,           B,
    < DPS2 >,           X3AB,           X,
    < DPS3 >,           T377,           T;

DECLARE NUMERIC TABLE (NTABLE2) WITH 3 ROWS AND 0 COLUMNS

    WITH ENTRIES

    < DPM1 >,
    < DPM2 >,
    < DPM3 >;
```

Legal Function Designators

Digital pattern measurement (DPM)
Digital pattern stimulus (DPS)
Pseudo digital pattern (PDP)
Digital pattern stimulus with data (DPSD)
System status digital pattern (SSA2)

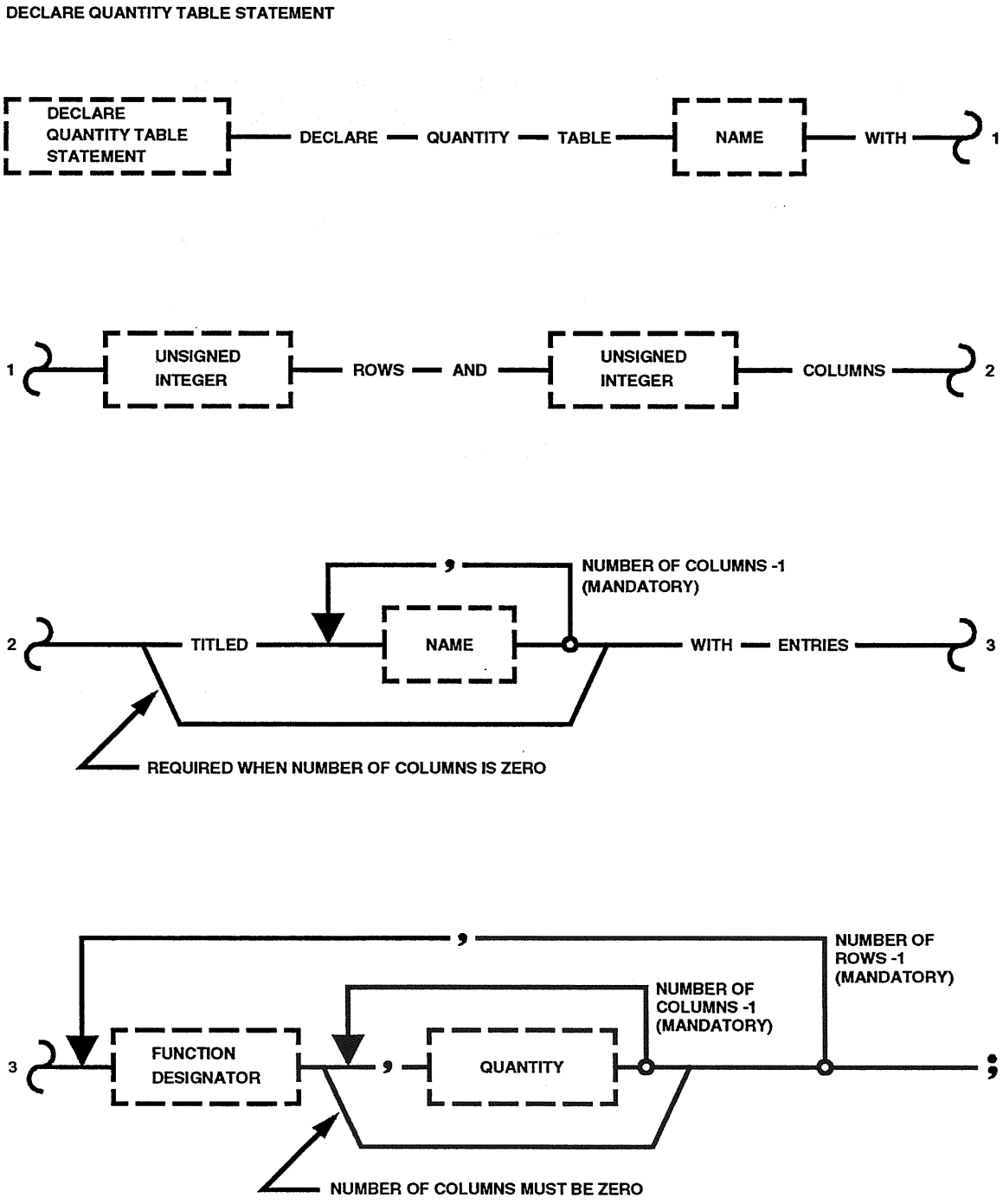


Figure 4-7 DECLARE QUANTITY TABLE Statement

4.3.2 DECLARE QUANTITY TABLE STATEMENT

Function

The DECLARE QUANTITY TABLE Statement is used to define the Name and characteristics of a table of Quantity data items.

Description

The DECLARE QUANTITY TABLE Statement is used to declare a table of Quantity data items. The data items must conform to the specifications for Quantity data. The data items may be initialized or a default subtype (i.e., Dimension with no decimal number) may be specified.

Example

```
DECLARE QUANTITY TABLE (QTABLE1) WITH 3 ROWS AND 2 COLUMNS

    WITH ENTRIES

    < AM1 >,        35.689,      5 V,
    < AM2 >,        10 V,        8 AMP,
    < AM3 >,           V,          AMP;
```

Legal Function Designators

Analog measurement (AM)
Analog stimulus (AS)
Pseudo analog (PA)
Double precision analog (AMDP)
Floating point (FP)
Interval Time (TIMR)
Countdown Time (CDT)
Greenwich Mean Time (GMT)
Mission Elapsed Time (MET)
Julian Time of Year (JTOY)

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Time value dimensional types may not be mixed with non-time value dimensional types in a given Quantity table.

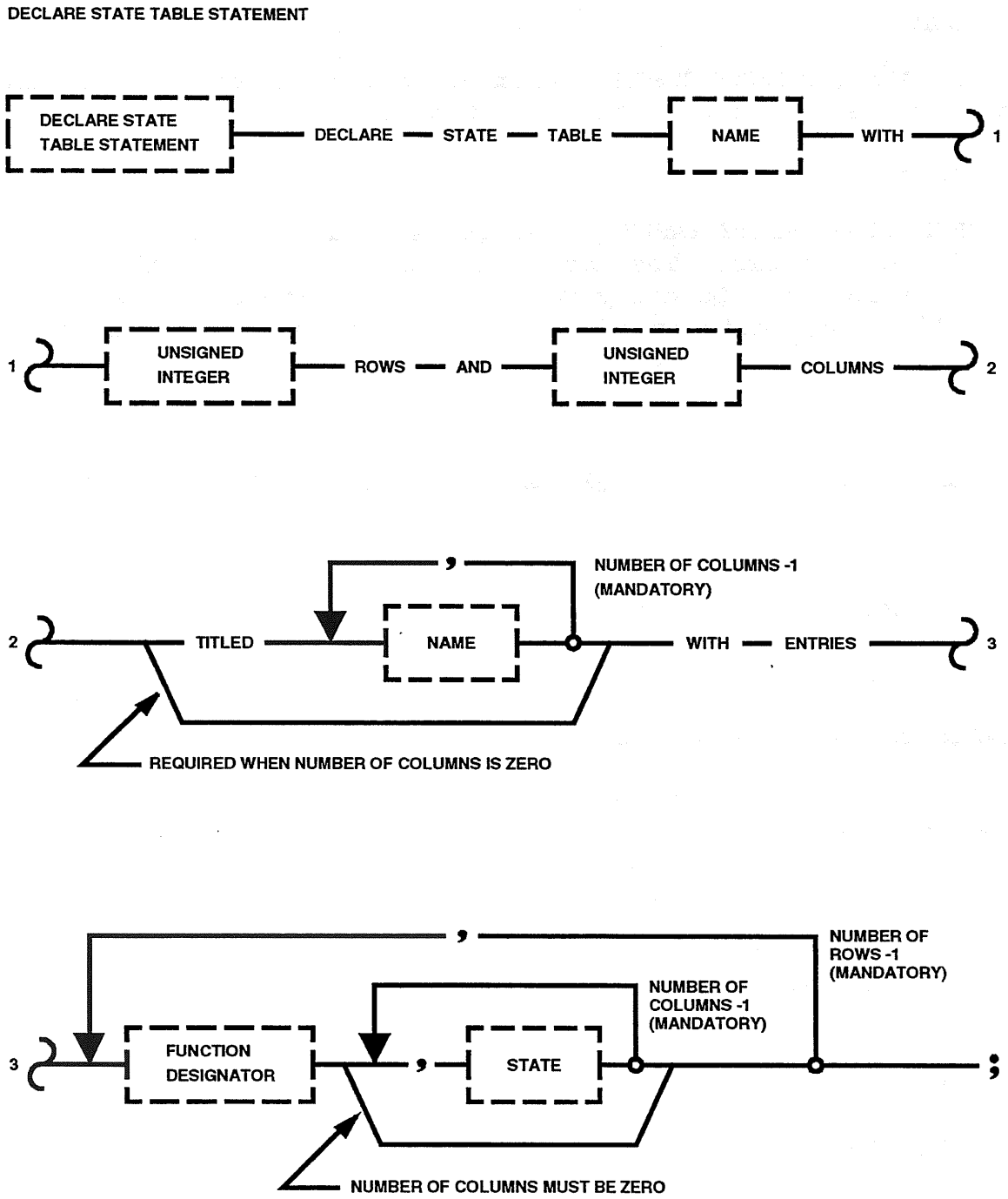


Figure 4-8 DECLARE STATE TABLE Statement

4.3.3 DECLARE STATE TABLE STATEMENT

Function

The DECLARE STATE TABLE Statement is used to define the Name and characteristics of a table of State data items.

Description

The DECLARE STATE TABLE statement is used to declare a table of State data items. The data items may be initialized or a default subtype (i.e., ON/OFF, OPEN/CLOSED, TRUE/FALSE, WET/DRY) may be specified.

Example

```
DECLARE STATE TABLE (STABLE2) WITH 3 ROWS AND 2 COLUMNS TITLED

                    (COLUMN A),    (COLUMN B)

    WITH ENTRIES

    < DM1 >,          OPEN,          CLOSED,
    < DM2 >,          ON/OFF,        ON,
    < DM3 >,          DRY,           ON/OFF;
```

Legal Function Designators

Discrete stimulus (DS)
Discrete measurement (DM)
Pseudo discrete (PD)
PFP indicators (PFPL)
Alarm (ALRM)
System status discrete (SSA1)

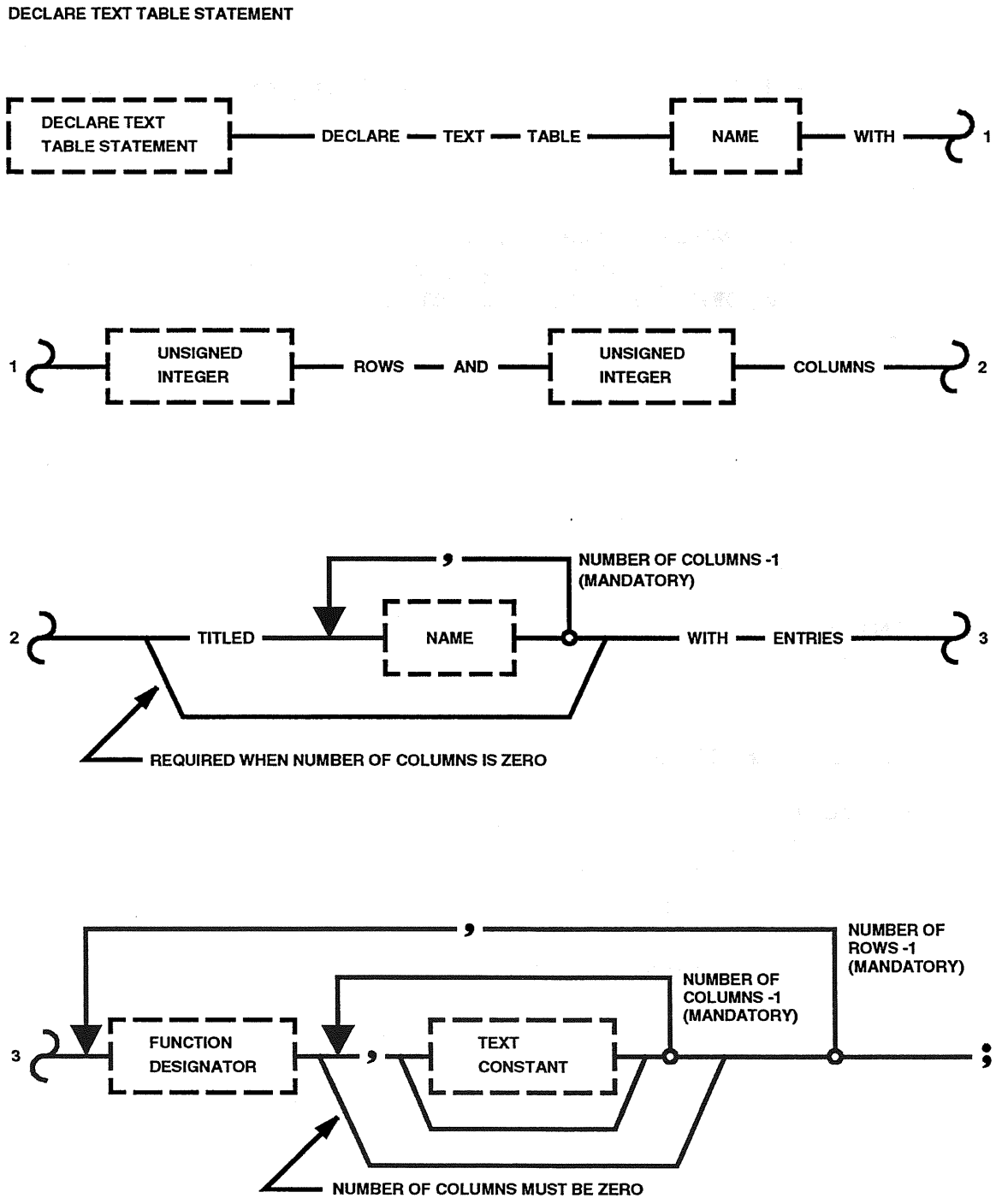


Figure 4-9 DECLARE TEXT TABLE Statement

4.3.4 DECLARE TEXT TABLE STATEMENT

Function

The DECLARE TEXT TABLE Statement is used to define the name and characteristics of a table of Text data items.

Description

The DECLARE TEXT TABLE Statement is used to declare a table of Text data. The data items of the table must be Text Constants. A default entry may be indicated by specifying a comma. It will have a length equal to the maximum specified entry. The contents or length of the first entry of the table must be specified. Subsequent entries may be treated according to the preceding rules.

Example

```
DECLARE TEXT TABLE (TTABLE2) WITH 2 ROWS AND 2 COLUMNS

    WITH ENTRIES

    < PAGE-A >,        25 CHARACTERS,      TEXT (ABC),
    < PAGE-B >,        GRAPHIC(44L) ,      TEXT (ABC);
```

Legal Function Designators

Printer (PRTR)
Console printer plotter (CPP)
Application pages (PAGE)
Recorders - ASCII (PDRR)
LED's (LED)
Disk files (FILE)

These Function Designators may be mixed in a given table.

THIS PAGE INTENTIONALLY LEFT BLANK.

5. ARITHMETIC AND LOGIC STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The Arithmetic and Logic statements and related elements to be discussed in the indicated sections are as follows:

ASSIGN STATEMENT (Section 5.1)
Assign Format (Section 5.1.1)
LET EQUAL STATEMENT (Section 5.2)
Numeric Formula (Section 5.2.1)
Logical Formula (Section 5.2.2)
AVERAGE STATEMENT (Section 5.3)

The Arithmetic and Logic statements are used to perform arithmetic computations and to equate the various data types. The ASSIGN Statement is used to equate state and text data. The LET EQUAL Statement is used to perform numerical calculations on numeric and quantity constants and Internal Names, and bit manipulations on numeric constants and Internal Names. The AVERAGE Statement is used to calculate averages of analog measurement data.

THIS PAGE INTENTIONALLY LEFT BLANK.

5.1 ASSIGN STATEMENT

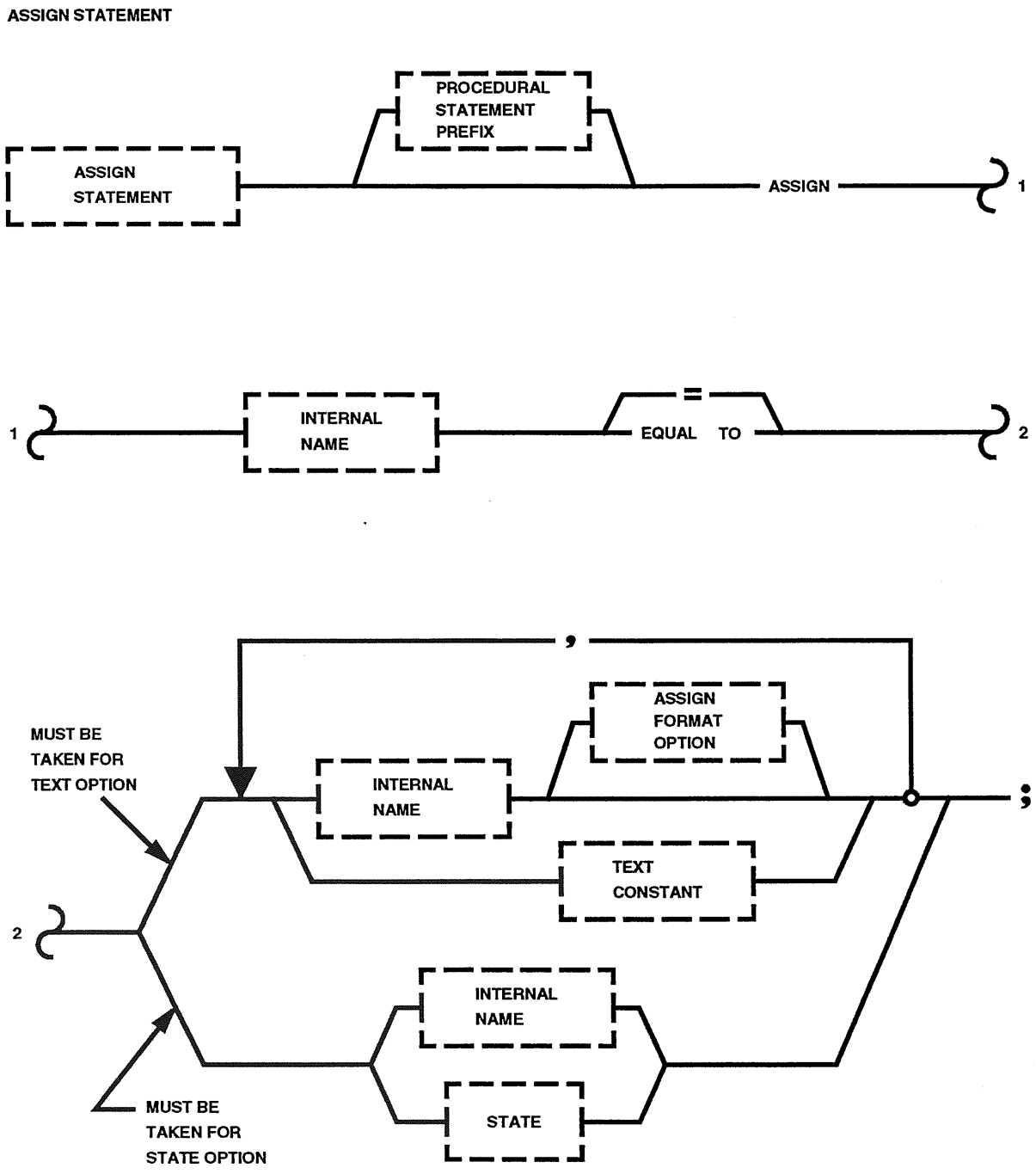


Figure 5-1 ASSIGN Statement

Function

The ASSIGN Statement replaces the contents of state or text type Internal Names with values of state or text type data.

Description

The ASSIGN Statement is the direct counterpart of the LET EQUAL Statement. The ASSIGN Statement is used to equate state and text type data, while the LET EQUAL Statement is used to equate quantity and numeric type data. The ASSIGN Statement will cause the contents of the Internal Name on the left side of the equal to be replaced by the data specified on the right side of the equal.

The ASSIGN Statement may be used to assign one or more values to Internal Names that are capable of accepting one or more values. The following repetitive combinations are allowed:

One-to-One: A single Internal Name is assigned equal to a single state or text value.

Many-to-One: Many internal variables (i.e., a list or table column) are assigned equal to one state or text value.

Many-to-Many: Many internal variables (i.e., a list or table column) are assigned to many state or text values of a list or table column.

A text type Internal Name may be assigned the value of a Text Constant, a text type Internal Name, a numeric type Internal Name, quantity type Internal Name, or a series of the preceding items separated by commas. The values of items in a series are concatenated into the text type Internal Name on the left side of the ASSIGN Statement to provide the capability of inserting the current value of variables into text messages. This is similar to the RECORD DATA Statement except that the formatted data is saved into an Internal Name instead of being recorded to the CRT.

A numeric or quantity type Internal Name is converted from the numeric or quantity value to a text value as described in Section 5.1.1, ASSIGN FORMAT option. If the Assign Format is not used with a numeric or quantity type Internal Name then a default format will be as described in Figure 5-2. Function Designators are not allowed in the ASSIGN Statement. The ASSIGN FORMAT option may not be used with a text type Internal Name or a Text Constant.

(1) TEXT CONSTANT - The characters of a Text Constant are output exactly as specified. Note that engineering graphics cannot be printed.

(2) NUMERIC DATA - The internal data is converted to a default Number Pattern format according to the way the data was declared.

Declared as Numeric Type:

DEFAULT FORMAT:	Character Field Positions 11111111112 01234567890	
Integer ASSIGN (TEXT) = (N10);	10	Field = Sign + 5 digits with leading zeroes suppressed, left justified, with 1 leading blank
Hexadecimal ASSIGN (TEXT) = (N10);	X000A	Field = ØX + 4 digits with leading zeros inserted, right justified
Octal ASSIGN (TEXT) = (N10);	T000012	Field = ØT + 6 digits with leading zeros inserted, right justified
Binary ASSIGN (TEXT) = (N10);	B0000000000001010	Field = ØB + 16 digits with leading zeros inserted, right justified

(3) QUANTITY DATA - The internal data is converted to a Quantity format. Engineering units are appended according to their declaration.

Figure 5-2 Default Formats (1 of 2)

Declared as Quantity Type:

DEFAULT FORMAT:

	1111111111 01234567890123456789	
- ASSIGN (TEXT) = (Q8)	SXXXXXXXX.ℬ UUUUUUUU	Sign + 8 digits + decimal point + 8 characters of units

Note: Decimal Point floats per the fractional number's value.

(4) STATE DATA - The internal data is converted to a State format according to its declaration and abbreviated to three output characters: OFF, ON, OPN, CLS, TRU, WET, DRY N/F, O/C, T/F, or W/D.

(5) TEXT DATA - The internal data is assigned directly as text characters; leading and trailing blanks are included. Note that engineering graphics can be assigned.

(6) TIME DATA - A quantity data declared as time is assigned as follows:

Declared as Quantity - Time Type:

DEFAULT FORMAT:

	111111 0123456789012345
- ASSIGN (TEXT) = (GMT);	+HHMM/SS.ZZZ
- ASSIGN (TEXT) = (CDT); or ASSIGN (TEXT) = (MET);	±DD:HHMM/SS
- ASSIGN (TEXT) = (JTOY);	DDD:HHMM/SS
- ASSIGN (TEXT) = (INTERVAL);	+HHMM/SS.ZZZ

Where: D = days, H = hours, M = minutes, S = seconds, and Z = milliseconds.

Figure 5-2 Default Formats (2 of 2)

ASSIGN INTERNAL NAME EQUAL TO STATE Option

Assign Name Equal to State

Examples

```
ASSIGN (STATE1) EQUAL TO ON;
ASSIGN (STATE7) = OPEN;
```

The first example results in assigning the contents of the Internal Name, (STATE1), equal to ON. The second example results in assigning (STATE7) equal to the OPEN state.

Assign List or List Element to State

Examples

```
ASSIGN (SLIST3) = TRUE;
ASSIGN (SLIST1) = ON;
ASSIGN (SLIST1)2 = WET;
```

The first two examples assign entire lists equal to state values. The third example assigns a single List element to a state value.

Assign Table Column or Table Element to State

Examples

```
ASSIGN (STABLE1) ROW 1 COLUMN 2 = FALSE;
ASSIGN (STABLE2) (COLUMN 5) = WET;
ASSIGN (STABLE1) < DS1 > COLUMN 2 = CLOSED;
```

The first example assigns the element of the table (STABLE1) located at Row 1 Column 2 to FALSE. The second example assigns Column 5 of (STABLE2) to WET. The third example assigns the element of (STABLE1) defined by Row Designator < DS1 > Column 2 to CLOSED.

ASSIGN INTERNAL NAME EQUAL TO TEXT CONSTANT Option

Assign Name Equal to Text Constant

Examples

```
ASSIGN (TEXT2) = TEXT(**ABC**);
ASSIGN (GRAPHIC1) EQUAL TO GRAPHIC(15L,27U);
ASSIGN (TEXT3) = TEXT(**), TEXT(XYZ), TEXT(**);
```

The first example assigns the Internal Name (TEXT2) equal to the Text Constant (**ABC**). The second example assigns the Internal Name (GRAPHIC1) equal to the graphic characters, 15L and 27U. The third example assigns the Internal Name (TEXT3) equal to the Text Constant (**XYZ**).

Assign List or List Element Equal to Text Constant

Examples

```
ASSIGN (TLIST1)3 = TEXT(AB12);
ASSIGN (TLIST1) = TEXT (AB 12);
ASSIGN (TLIST3) = (TLIST4);
ASSIGN (TLIST5) = TEXT(AB), TEXT(12);
```

The first example assigns the third element of the Text List (TLIST1) to the Text Constant (AB12). The second example assigns the entire list (TLIST1) equal to the Text Constant (AB 12). The third example assigns each of the three elements of Text List (TLIST3) equal to the corresponding element of Text List (TLIST4). The fourth example assigns the entire list (TLIST5) equal to the Text Constant (AB12).

Assign Table Column or Table Element Equal to Text Constant

Examples

```
ASSIGN (TTABLE1) ROW 1 COLUMN 2 = TEXT(**ABC**);
ASSIGN (TTABLE1) COLUMN 1 EQUAL TO TEXT(VALUE3);
ASSIGN (TTABLE1) COLUMN 1 = TEXT(**), TEXT(  ), TEXT(AB),
                            TEXT(DE), TEXT(  ), TEXT(**);
```

The first example assigns an element of the Text Table (TTABLE1) equal to the Text Constant (**ABC**). The second example assigns Column 1 of (TTABLE1) equal to the Text Constant (VALUE3). The third example assigns Column 1 of (TTABLE1) equal to the Text Constant (** ABDE **).

ASSIGN INTERNAL NAME EQUAL TO INTERNAL NAME Option

Assign Internal Name Equal to Name

Examples

```
ASSIGN (STATE1) = (STATE5);
ASSIGN (TEXT1) = (TEXT2);
ASSIGN (SLIST1) = (STATE5);
ASSIGN (STABLE1) COLUMN 2 = (STATE5);
```

```
ASSIGN (TLIST1) = (TEXT1), (N1), (TEXT2);
ASSIGN (TTABLE) COLUMN 1 = (TEXT1), (N10) FORMAT (I3)
                           (TEXT2);
```

The first example assigns the State Name (STATE1) equal to the value of the State Name (STATE5). The second example assigns a Text Name equal to a Text Name. The third example assigns a State List equal to a State Name. The fourth example assigns Column 2 of the State Table equal to the State Name (STATE5). If the text type Internal Names (TEXT1) and (TEXT2) have the values (ABC) and (XYZ) respectively, and the numeric type Internal Names (N1) and (N10) have the values 1 and 10 respectively, then the fifth example assigns the entire Text List (TLIST1) equal to (ABCbbbbb1XYZ). The sixth example assigns all rows or the first column of the Text Table (TTABLE) equal to (ABC10XYZ).

Assign Internal Name Equal to List or List Element

Examples

```
ASSIGN (STATE1) = (SLIST1)1;
ASSIGN (SLIST1) = (SLIST3)2;
ASSIGN (SLIST3) = (SLIST1);
ASSIGN (STABLE1) COLUMN 1 = (SLIST1)1;
ASSIGN (TLIST) 1 = (TLIST2) 1, (N10) FORMAT(X3), (TLIST) 2;
```

The first example assigns the State Name equal to the first element of the State List, (SLIST1). The second example assigns an entire list equal to an element of a list. The third example assigns a list equal to a list. The last example assigns a table column equal to a list element. If the text type Internal Names (TLIST2) 1 and (TLIST3) 2 have the values (ABC) and (XYZ) respectively, then the fifth example assigns the first element of the Text List (TLIST) equal to (ABCX00AXYZ).

Assign Internal Name Equal to Table Column or Table Element

Examples

```
ASSIGN (STATE1) = (STABLE2) < DM7 > COLUMN 1;
ASSIGN (SLIST2) = (STABLE1) COLUMN 2;
ASSIGN (STABLE1) COLUMN 2 = (STABLE2) COLUMN 1;
ASSIGN (TTABLE1) COLUMN 2 = (TTABLE2) ROW 1 COLUMN 2,
                            (NTABLE3) ROW 15 COLUMN 1
                            FORMAT (F6.2, NO UNITS);
```

The first example assigns a Name equal to a table element. The second example assigns the entire list (SLIST1) equal to Column 2 of (STABLE1). The third example assigns Column 2 of (STABLE1) equal to Column 1 of (STABLE2). If the Text Table (TTABLE2) ROW 1 COLUMN 2 has the value (ABC) and the Numeric Table (NTABLE3) ROW 15 COLUMN 1 has the value 10.32, then the fourth example assigns all rows of the second column of the Text Table (TTABLE1) equal to (ABCb10.32).

Statement Execution

The ASSIGN Statement is a CPU bound statement. The data specified on the right of the equal is obtained and stored in the Internal Name(s) referenced on the left. If the formatted numeric or quantity data does not fit into the text variable provided, then that portion of the text variable at which the data would go is filled with asterisks (*).

Error Processing

An invalid index into a list or table column will result in a Class III error.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. More than one value may not be assigned to a single internal data item.

B. In a Many-to-Many operation, the number of values must equal the number of internal variables.

C. Numeric and quantity type internal variables may only be used in a One-to-One or One-to-Many operation. They may not be used in a Many-to-Many operation.

D. The data type of the value being assigned must be compatible with the data type of the internal variable where it is received.

E. The receiving internal variable must be text type if numeric or quantity type internal variables are used.

F. The Assign Format option may only be used with numeric or quantity type internal variables. It may not be used with any type of constant or with text or state type Internal Names.

G. If a list or table column is used to specify the data values, no additional data items may be specified.

H. For Text assignments, if the length of the variable on the left is greater than the one on the right, the left variable will be padded with blanks. If the variable on the left is shorter, the value from the right will be truncated to fit the shorter length.

5.1.1 ASSIGN FORMAT OPTION

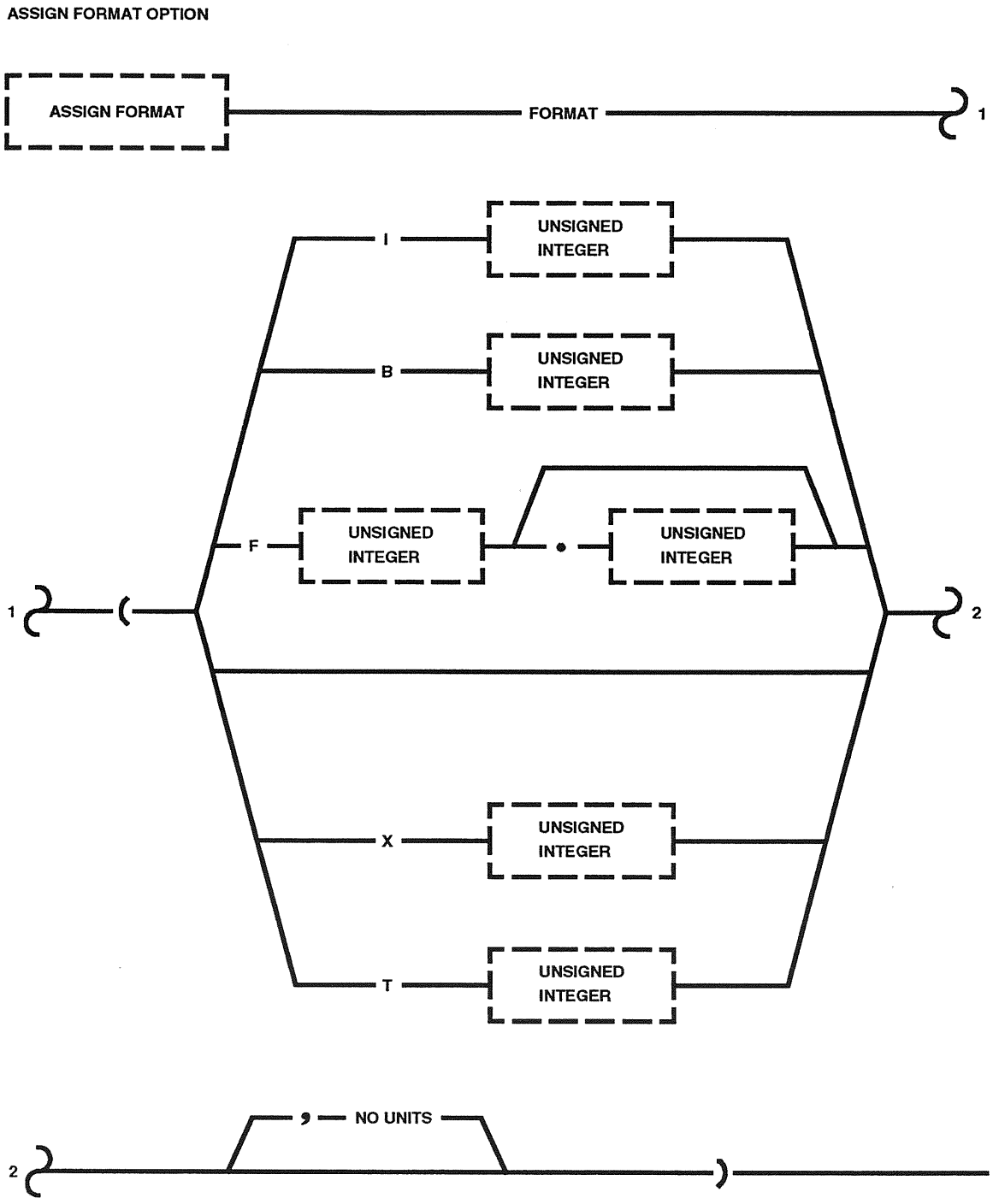


Figure 5-3 Assign Format Option

Function

The ASSIGN FORMAT option is used to specify conversion and editing information for data which is to be assigned to a text type Internal Name.

Description

The ASSIGN FORMAT option is used with the ASSIGN Statement to format numeric and quantity data into text data. Format specifications may be used for Internal Names only. The data formats are:

I	-	Integer format
B	-	Binary format
X	-	Hexadecimal format
T	-	Octal format
F	-	Decimal format

I, B, X, and T are used to assign numeric type data. F is used to assign Quantity type data.

I, B, X, and T data formats must be followed by an Unsigned Integer which indicates the field width of the data to be assigned.

The F format is written in GOAL as Fx.y, where x represents the number of integer characters in the text and y represents the number of fractional characters to the right of the decimal point. If y is zero, the decimal point will not be a part of the text. If y is not specified, the format will be treated as if y had been specified as zero. If y is specified and there are no non-zero fractional digits in the actual data, the text will contain zeroes in the fractional locations. The value of the y + 1 fractional digit (the most significant fractional digit not assigned) is used in rounding the formatted number. For example, when using an F1.1 format, "1.55" and "1.56" are assigned as "b1.6" and "1.54" is assigned as "b1.5". This also means that if y is not specified or is zero (F1.0 format), "1.5" is assigned to "b2". If the number cannot be contained in the specified field width, the entire text field will contain asterisks (*). If the number is positive, the "+" sign will not be in the text field, but a blank will be reserved for it. Leading zeroes will be suppressed, and the number will be right justified in the text field.

The parameter NO UNITS may be selected to suppress insertion of engineering unit text information when assigning Internal Name quantity type data.

The parameter NO UNITS may also be selected to suppress the insertion of '⊬B', '⊬X', or '⊬T' when assigning Internal Name numeric type data.

If NO FORMAT option is specified, data will be assigned according to the default formats given in Section 7.1, RECORD DATA Statement, as they apply to Internal Name numeric type data only.

In the examples on the following page, the GOAL source statements and the assigned text will be shown. Character positions are shown with the text. The "S" in character position zero refers to the sign.

GOAL SOURCE STATEMENT EXAMPLES

GOAL Source Statements	Text
A. ASSIGN Name - Default Format ASSIGN (TEXT) = (N10);	**0 1 2 3 4 5 6 7 8 9** 1 0
B. ASSIGN Name - I,B,T,X Formats ASSIGN (TEXT) = (N10) FORMAT(I3); ASSIGN (TEXT) = (N10) FORMAT(B5); ASSIGN (TEXT) = (N10) FORMAT(T3); ASSIGN (TEXT) = (N10) FORMAT(X3); ASSIGN (TEXT) = (N10) FORMAT (X3,NO UNITS);	**0 1 2 3 4 5 6 7 8 9 10 11** 1 0 B 0 1 0 1 0 T 0 1 2 X 0 0 A 0 0 A
C. ASSIGN Quantity Name - F Format ASSIGN (TEXT) = (Q18) FORMAT(F3.2); ASSIGN (TEXT) = (Q19) FORMAT(F6.2);	**0 1 2 3 4 5 6 7 8 9 10 11** - 5 . 3 5 N U L L 7 . 0 0 V
D. ASSIGN Quantity Name - F Format ASSIGN (TEXT) = (Q7) FORMAT(F2.2); ASSIGN (TEXT) = (Q7) FORMAT(F2.2, NO UNITS);	**0 1 2 3 4 5 6 7 8 9 10 11** - 5 . 3 5 A M P - 5 . 3 5

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Assign Format specifications may only be used with the ASSIGN Statement.

B. Function Designators may not be used in the ASSIGN Statement.

C. Format specification information always applies to the data item which immediately precedes it. No other assign options may be embedded between a format specification and its associated data.

D. Format specifications may not be specified for Text or Time Internal Name data.

E. The maximum number of significant digits which should be specified are:

8 for F format

5 for I format

16 for B format

4 for X format

6 for T format

If more digits are specified, blanks will be filled in front of the number.

THIS PAGE INTENTIONALLY LEFT BLANK.

5.2 LET EQUAL STATEMENT

LET EQUAL STATEMENT

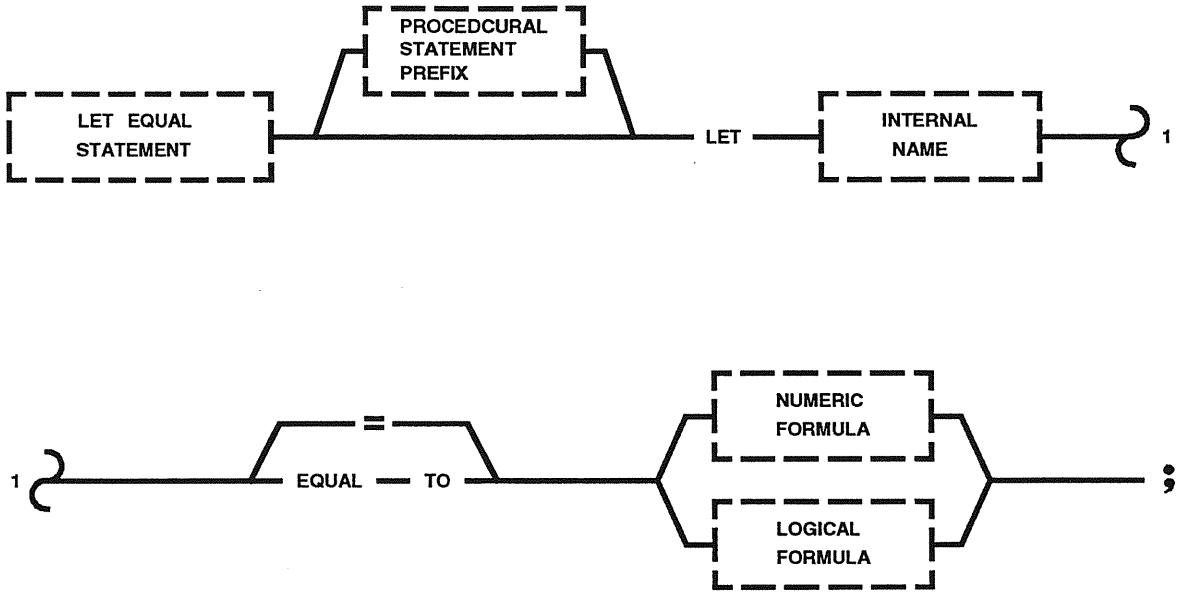


Figure 5-4 LET EQUAL Statement

Function

The LET EQUAL Statement is used to replace the contents of a Numeric or Quantity type Internal Name with a single value derived from the NUMERIC FORMULA or to replace the contents of a numeric type Internal Name with a single value derived from the Logical Formula. The LET EQUAL Statement may also be used to replace the contents of a numeric or quantity list or table column with the contents of another list or table column.

Description

The Numeric Formula is evaluated as an algebraic expression to obtain a resultant value which may be an integer or floating point value. The content of the Internal Name on the left side of the equal is replaced with the resultant value. If the resultant value is stored into a Numeric data item, it is truncated and only the integer part is stored.

The Internal Name on the left side of the equal may be either Numeric or Quantity type. However, dimensions are ignored in evaluation of the LET EQUAL Statement. Only the arithmetic values of the Internal Name Numeric Formula are used to evaluate the LET EQUAL expression.

The Numeric Formula is evaluated according to the following rules:

A. If the resultant value is stored into a non-time quantity Internal Name, then all operations are performed using floating point arithmetic (accurate to six significant figures).

B. If the resultant value is stored into a time quantity Internal Name, then all operations are performed using double precision (64-bit intermediate 32-bit final result) arithmetic unless a floating point operand is encountered in the expression. Operations will be performed in double precision up to the point where the floating point operation must be done. At that point, any intermediate results are converted to floating point and all ensuing operations will be performed using floating point arithmetic. The final result will be converted back to a 32-bit integer before storing it into the time quantity Internal Name.

C. If the resultant value is stored into a numeric Internal Name (single precision mode in effect - single precision

mode is the default condition or may be explicitly requested via compiler directive), then all operations are performed using single precision (32-bit intermediate 16-bit final result) arithmetic unless a double precision or floating point operand is encountered in the expression. Operations will be performed in single precision until the double precision or floating point operation is performed. At that time, all intermediate values are converted to either double precision or floating point and ensuing operations will be performed using double precision or floating point arithmetic. The final result will be converted back to a 16 bit integer value before it is stored into the numeric Internal Name.

D. If the resultant value is stored into a numeric Internal Name (double precision mode in effect - double precision mode is requested via compiler directive), then all operations are performed using double precision (64-bit intermediate; 32-bit final result) arithmetic unless a floating point operand is encountered in the expression. Operations will be performed in double precision up to the point where the floating point operation must be done. At that point, any intermediate results are converted to floating point and all ensuing operations will be performed using floating point arithmetic. The final result will be converted to a 16-bit integer before storing it into the numeric Internal Name.

The Logical Formula is evaluated left-to-right according to the rules of the Logical Formula in order to obtain a resultant numeric data value. The internal variable that receives the data must be declared to be a numeric type.

The following repetitive operations are allowed:

One-to-One: A single internal data item is set equal to the resultant value.

Many-to-One: Each item in a list or table column is set equal to the resultant value.

Many-to-Many: Each item in a list or table column is set equal to the corresponding value in a list or table column.

LET EQUAL With Names

Examples

```
LET (N1) = 5;
LET (N9) = B101;
LET (N2) = 5 V + 3 UNITS;
LET (Q1) = (Q2) + 4V - (Q2)/5.1;
LET (N1) = (1/10) * 10;
LET (N2) = (N3) AND XF;
LET (Q1) = 3 SEC + 5 SEC;
```

The first example sets the Numeric Name (N1) equal to the integer 5. The second example sets (N9) equal to binary 101. The third example sets the Numeric Name (N2) equal to the value 8.

Dimensions are ignored in the calculation. The fourth example illustrates a calculation using Quantity data. The fifth example is included to illustrate the consequences of truncation error when using Numeric type data. The Numeric Name (N1) will be set equal to zero since the result of the expression is less than one due to truncation. The sixth example sets the variable (N2) equal to the last four bits of the variable (N3). The last example illustrates a calculation of quantities using integer arithmetic.

LET EQUAL With Lists

Examples

```
LET (QLIST1)1 = 1 V;
LET (QLIST1)  = 5 V;
LET (NLIST2)  = 2 + 2 - (QLIST1)2 -  (N2);
LET (NLIST2)2 = NOT (N3);
LET (NLIST2)  = (NTABLE 1) COLUMN 1;
LET (NLIST1)  = (NLIST2);
```

The first example sets the List Element (QLIST1)1 equal to 1 (the dimension of the value is as defined on the DECLARE Statement for QLIST1). The second example sets the entire list (QLIST1) equal to 5. The third example sets Numeric List (NLIST2) equal to the numeric value of the expression on the right side of the equal. The fourth example sets the second element of Numeric List (NLIST2) equal to the complement of numeric variable (N3). The fifth example sets each element of (NLIST2) to the value of the corresponding element in (NTABLE 1) COLUMN 1, while the last example sets each element of (NLIST1) to the value of the corresponding element in (NLIST2).

LET EQUAL With Tables

Examples

```
LET (NTABLE1) ROW 1 COLUMN 1 = 5;
LET (NTABLE1) COLUMN 2 = (N1);
LET (QTABLE1) COLUMN 1 = 2 V;
LET (NTABLE1) ROW 2 COLUMN 2 = SHIFT RIGHT 3 BITS (NUMB);
LET (NTABLE1) COLUMN 2 = (NTABLE 1) COLUMN 4;
LET (NTABLE1) COLUMN 1 = (NLIST2);
```

The first example sets a table element of a Numeric Table equal to the number 5. The second example sets a column of a table equal to the value of the Name (N1). The third example sets a column of a Quantity Table equal to 2 volts. The fourth example sets an element of a Numeric Table equal to the number (NUMB) shifted right 3 bits. The fifth example sets each element of (NTABLE 1) Column 2, to the value of the corresponding element of Column 4. The sixth example sets each element of (NTABLE1) Column 1 to the value of the corresponding element in (NLIST2).

Statement Execution

The GOAL Executor evaluates the Numeric or Logical Formula and stores the result in the Internal Name in the procedure's variable data area (VDA). Literals are computed by the compiler.

Error Processing

Floating point violations (overflow, underflow, zero divide) which occur during execution of the LET EQUAL Statement will be processed as Class III errors at CCMS. (The valid range is 2^{256}.)

Overflow occurs when the result of a fixed-point arithmetic operation is out of the range of -32,768 (X8000) to 32,767 (X7FFF), the legal range of the numbers permissible in a 16-bit CCMS word.

For example, the following operations will result in an overflow condition and thus a Class III error:

```
EXAMPLE 1:  X4001 * X0002 (16,385 * 2)
EXAMPLE 2:  X7FFF + X0001 (32,767 +1)
EXAMPLE 3:  XFFFF + X8000 ([-1]+[-32,768])
```

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Dimensions are ignored in the evaluation of LET EQUAL expressions.

B. A single internal variable may not be set equal to more than one data item.

C. Time-valued Quantity lists/table columns may be used only with time-valued Quantity lists/table columns.

D. When assigned to a numeric internal variable, a numeric literal has a range of -32767 to 32767. Hexidecimal, octal, and binary numeric literals have a decimal range of -32768 to 32767.

E. When asigned to a quantity internal variable, a numeric literal has a range of -99999 to 99999. Hexidecimal, octal, and binary numeric literals have a decimal range of 0 to 65535.

F. Numeric literals entered as hexidecimal, octal, or binary have ranges of:

hexadecimal:	0 to FFFF
octal:	0 to 177777
binary:	0 to 1111111111111111

Legal Function Designators

None

THIS PAGE INTENTIONALLY LEFT BLANK.

5.2.1 NUMERIC FORMULA

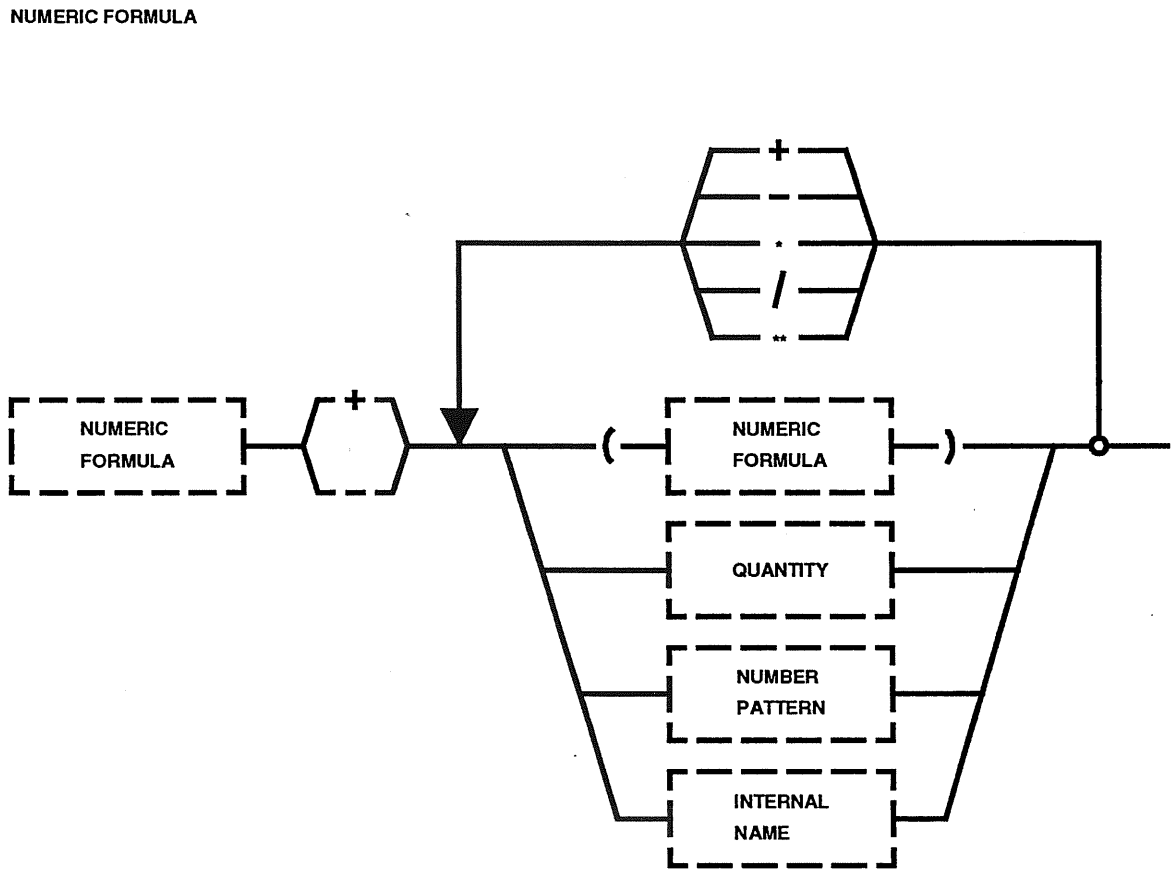


Figure 5-5 Numeric Formula

Function

The Numeric Formula element is used to specify an algebraic expression to be evaluated on the right side of the LET EQUAL Statement.

Description

The following mathematical operations may be specified in a Numeric Formula:

Multiplication	- denoted by a single asterisk (*)
Division	- denoted by a slash (/)
Addition	- denoted by a plus (+)
Subtraction	- denoted by a minus (-)
Exponentiation	- denoted by two asterisks (**)

Both Numeric and Quantity data may be used in a Numeric Formula. Entire Tables are not allowed.

Numeric Formulas are evaluated using double precision and floating point arithmetic. Operations are performed using double precision (64-bit intermediate 32-bit final result) calculations until a Quantity value is encountered in the expression. As soon as a Quantity value is encountered, all further operations will be carried out using floating-point arithmetic. Floating point calculations are accurate to six significant figures. When the resultant value of a Numeric Formula is stored in a Numeric variable, it is truncated to an integer value. Dimensions are ignored in evaluation of a Numeric Formula.

Rules

A. Parentheses may be used where required to indicate the order in which calculations are to be performed. Arithmetic operations within the innermost parentheses are performed first.

B. When the ordering of operations in an expression is not completely specified by use of parentheses, the sequence of operation is as follows:

   1. all exponentiations are performed

   2. all multiplications and divisions are performed

   3. all additions and subtractions are performed

Numeric Formula With Quantity Data

Examples

```
LET (Q8) = 5.0 + (Q7) - 1.2579;
LET (Q9) = 5 * (N2) + (Q7)/10;
LET (Q7) = 5 ** 2;
```

In the first example, the Decimal Number, 5.0, is added to the arithmetic value of the Name (Q7) and 1.2579 is subtracted from the result. In the second example, the Integer 5 is multiplied by the arithmetic value of the Numeric Name (N2) and the result is added to the value of the Quantity (Q7) divided by 10. In the third example, the number 5 is squared.

Numeric Formula With Numeric Data

Examples

```
LET (N1) = X FF - (N2) + B 101;
LET (N2) = T 367 * 5 + (N3)/4 * 2;
```

Numeric Formula With Parentheses

Examples

```
LET (N1) = (  (5+1) - (5 +1)  ) * 1.578;
LET (N2) = (  5 - (N1)  )/2;
LET (Q8) = (NLIST1)1**((N3) - 1);
```

These examples illustrate the use of parentheses to indicate the order in which calculations are to be performed.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Consecutive arithmetic operators are not permitted in a Numeric Formula.

B. Entire Tables are not allowed in the Numeric Formula.

C. If a list or table column is used in the Numeric Formula, it must appear alone.

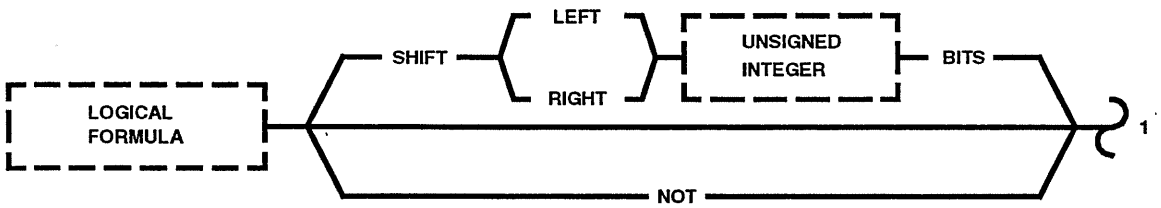
D. Any Quantity may be exponentiated by a value, M/N, where M and N are integers. If the exponent has a fractional part, a Class III run time error will occur if the Quantity is negative.

E. Zero to the zero power will give an answer of zero.

THIS PAGE INTENTIONALLY LEFT BLANK.

5.2.2 LOGICAL FORMULA

LOGICAL FORMULA



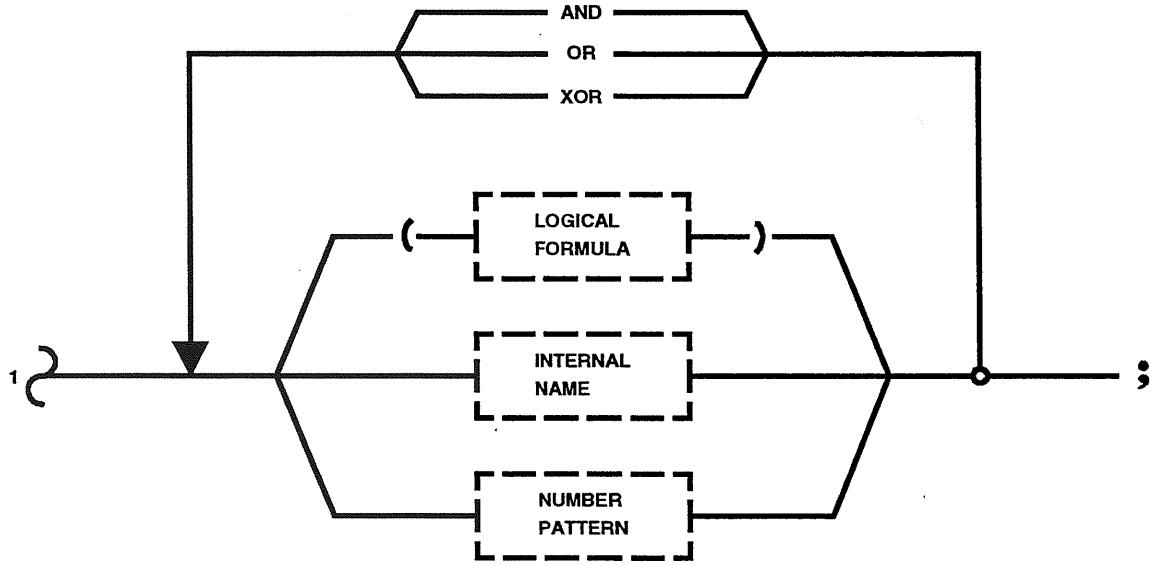


Figure 5-6 Logical Formula

Function

The Logical Formula element is used to specify a logical expression to be evaluated on the right side of the LET EQUAL Statement.

Description

The following logical operations may be specified in a Logical Formula:

Shift Left	- denoted by SHIFT LEFT n BITS
Shift Right	- denoted by SHIFT RIGHT n BITS
And	- denoted by AND
Or	- denoted by OR
Exclusive Or	- denoted by XOR
Not	- denoted by NOT

NOTE: *BIT and BITS can be used interchangeably*

Rules

A. Parentheses may be utilized where required to indicate the order in which logical operations are to be performed. Logical operations within the innermost parentheses are performed first.

B. When the ordering of operations in an expression is not completely specified by use of parentheses, the sequence in which the operations are evaluated is as follows:

   1. All SHIFT and NOT operations are performed.

   2. All AND, OR, and XOR operations are performed.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. If a list or table column is used in the Logical Formula, it must appear alone.

B. Entire tables are not allowed in the Logical Formula.

C. All Internal Names used must be numeric.

D. The number of bits which can be shifted is $0 \leq n \leq 16$.

E. The NOT operation results in a 1's complement operation whereby the settings or state of each bit in the data value is changed to its opposite state.

F. A shift operation is all vacated bit positions replaced with zeroes.

Logical Formula With Logical Connectors

```
LET (N1) = (N2) AND (N3);
LET (N1) = (N2) OR (N3);
LET (N1) = (N2) XOR (N3);
```

In the first example the Number Pattern value of the variables (N2) and (N3) are processed through a logical AND operation with the resultant value stored into the variable (N1). The second example processes the values of the variables (N2) and (N3) through a logical OR operation with the resultant value stored into the variable (N1). The third example processes the values of the variables (N2) and (N3) through a logical XOR (exclusive OR) operation with the resultant value stored into the variable (N1).

Logical Formula With Shift Operations

```
LET (N1) = SHIFT RIGHT 5 BITS (N2);
LET (N1) = SHIFT LEFT 15 BITS (N2);
```

In the first example, the contents of the variable (N2) are shifted to the right by 5 bit positions and the resultant value is stored into the variable (N1). The second example shifts the contents of the variable (N2) to the left by 15 bit positions with the resultant value stored into the variable (N1).

Logical Formula Not Operations

```
LET (N1) = NOT (N2);
LET (N1) = NOT ((N2) AND (N3)) ;
```

In the first example, the bit settings for the variable (N2) are "flipped" to the opposite state, and the resultant value is stored into (N1). The second example follows the following operations: the values of the variables (N2) and (N3) are processed through an AND operation, the bit settings of the resultant value are "flipped", and the resultant value is stored into the variable (N1).

5.3 AVERAGE STATEMENT

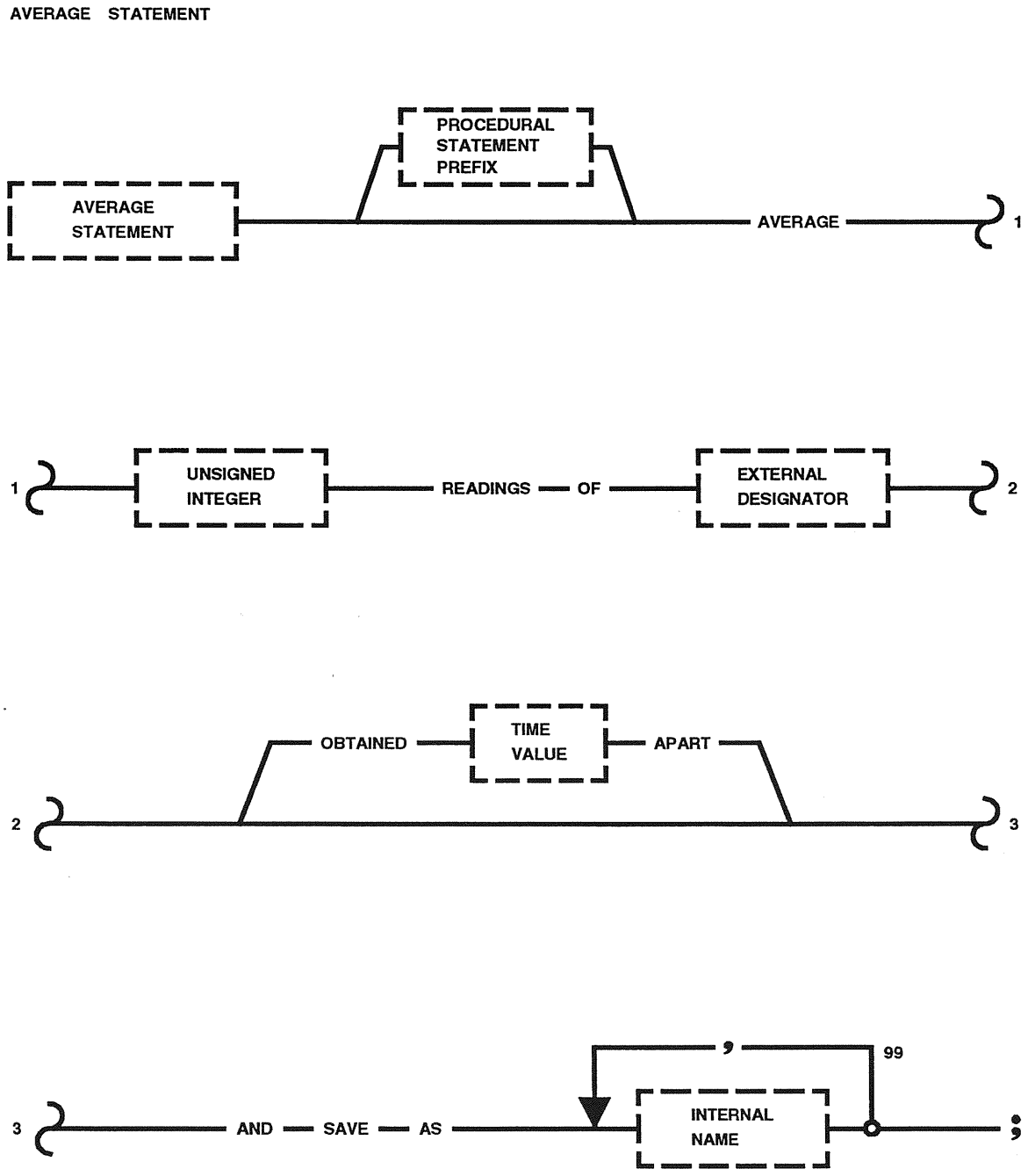


Figure 5-7 AVERAGE Statement

Function

The AVERAGE Statement is used to average readings of measurements acquired from the system under test and store the averages in specified internal variables.

Description

The AVERAGE Statement is used to average readings of Analog Measurement, Analog Measurement Double Precision, and Floating Point Function Designators only. The averaging operation is a simple mathematical average: the sum of the measurements of a test point divided by the number of times the test point is read. The value obtained from an averaging operation is stored in the specified internal variable identified by Internal Name.

Statement Options

If an interval delay is specified, the averaging operation may be described as follows:

A. The current value of each Function Designator is read and saved.

B. Processing is delayed, as required, for the specified time interval.

C. Steps (1) and (2) are repeated until the required number of samples have been taken.

D. The mathematical average is computed and saved for each measurement.

If an interval delay is not specified, the averaging operation may be described as follows:

A. The interval between successive samples of each measurement is determined by the current sample rate of the first measurement specified.

B. The current value of each Function Designator is read and saved.

C. Step (2) is repeated until the required number of samples has been taken.

D. The mathematical average is computed and saved for each measurement.

E. Unpredictable time deltas may occur as a result of changing sample rates.

Repetitive Operations

This statement may be used to average one or more Function Designators and store to one or more internal variables. Its operation is repeated until all Function Designators are processed. The following repetitive combinations are allowed:

One-to-One: A single Function Designator is averaged and stored in a single internal variable.

One-to-Many: A single Function Designator is averaged and stored in many internal variables.

Many-to-Many: Many Function Designators are averaged and stored in many internal variables (1st to 1st, 2nd to 2nd, etc.). The number of Internal Names (or list/table column length) may exceed the number of Function Designators to be read. The excess variables are unaffected.

AVERAGE With Time Value Option

Examples

```
AVERAGE 25 READINGS OF < AM1 > OBTAINED .5 SEC APART AND
    SAVE AS (Q1);

AVERAGE 25 READINGS OF < AM2 > OBTAINED 1 SEC APART
    AND SAVE AS (Q1), (Q2), (Q3);

AVERAGE 10 READINGS OF < AM3 > OBTAINED 1 SEC APART
    AND SAVE AS (QLIST1);

AVERAGE 20 READINGS OF < AM1 > < AM2 > < AM3 >
    OBTAINED 1 SEC APART AND SAVE AS (QLIST1);
```

AVERAGE Without Time Value Option

Examples

```
AVERAGE 100 READINGS OF < AM1 >
    AND SAVE AS (Q1);

AVERAGE 100 READINGS OF < AM2 >
    AND SAVE AS (Q1), (Q2), (Q3);
```

```
AVERAGE 50 READINGS OF < AM1 > < AM2 > < AM3 >
    AND SAVE AS (QTABLE1) COLUMN 2;
```

Statement Execution

The values of the measurements to be averaged will be obtained from the CDBFR at the required time interval by the GOAL Executor. After the required number of samples have been acquired, the average will be calculated. The resultant value will be stored in the specified Internal Name(s) in the variable data area.

Error Processing

If the required measurement values cannot be acquired from the CDBFR, a Class III error will be processed.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. More than one Function Designator may not be averaged and stored in a single internal variable.

B. In a Many-to-Many operation, the number of Function Designators must be less than or equal to the number of internal variables.

C. The Function Designator type being averaged must be compatible with the internal variable type in which it is received.

D. When a units mismatch occurs while using the TABLE FUNCTIONS option, a compiler warning message will be output.

E. The constant value specified for the number of readings to be performed must be greater than 1 and less than or equal to 32,767.

F. Measurements accessed via the Launch Data Bus may not be specified in the AVERAGE Statement.

G. Analog measurement type Function Designators and analog measurements double precision and floating point type Function Designators may not be mixed on a given AVERAGE Statement.

H. Limit of 80 analog measurement, double precision, and floating point numbers per statement.

Legal Function Designators (Must be resident in CDBFR)

Analog Measurement (AM)
Double Precision Analog (AMDP)
Floating Point (FP)

6. BRANCHING STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The Branching Statements provide the capability to change the execution paths of a GOAL procedure and are discussed in the indicated sections.

A. GO TO STATEMENT (Section 6.1)

   The GO TO Statement provides unconditional branching.

B. VERIFY PREFIX (Section 6.2)

   The VERIFY PREFIX provides conditional branching capability as well as testing capability.

C. REPEAT STATEMENT (Section 6.3)

   The REPEAT Statement provides a looping capability within a GOAL procedure.

D. PERFORM STATEMENT GROUPS STATEMENT (Section 6.4)

E. END STATEMENT GROUPS STATEMENT (Section 6.5)

F. STATEMENT GROUP LABEL (Section 6.6)

   The PERFORM STATEMENT GROUPS along with its END STATEMENT GROUPS and the STATEMENT GROUP LABEL provide branching capability within predefined boundaries.

Branching may also be achieved by means of the STOP Statement via keyboard input using the option to STOP AND INDICATE RESTART LABELS. However, the STOP Statement will be discussed with the Procedure Control Statements.

THIS PAGE INTENTIONALLY LEFT BLANK.

6.1 GO TO STATEMENT

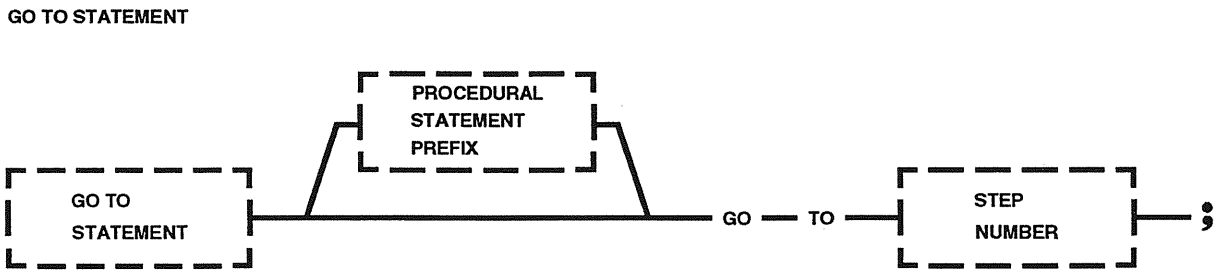


Figure 6-1 GO TO Statement

Function

The GO TO Statement provides an unconditional branching capability to change the execution path of a GOAL procedure.

Description

Execution of the GO TO Statement causes a branch to the statement referenced by Step Number. A Step Number is a label on a GOAL statement which provides a symbolic reference capability. After a GO TO Statement is executed, the next statement to be executed will be the statement referenced by the GO TO.

Example

```
        GO TO STEP 5;
        .    .
        .    .
        .    .
STEP 5 ASSIGN (STATE1) = ON;
```

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. A Step Number referenced by a GO TO must be defined within the program.

B. A Step Number referenced by a GO TO may not be within a repeat loop or a sequence unless the GO TO is also located within the same loop or sequence.

C. The statement immediately following a GO TO Statement must be given a Step Number.

THIS PAGE INTENTIONALLY LEFT BLANK.

6.2 VERIFY PREFIX

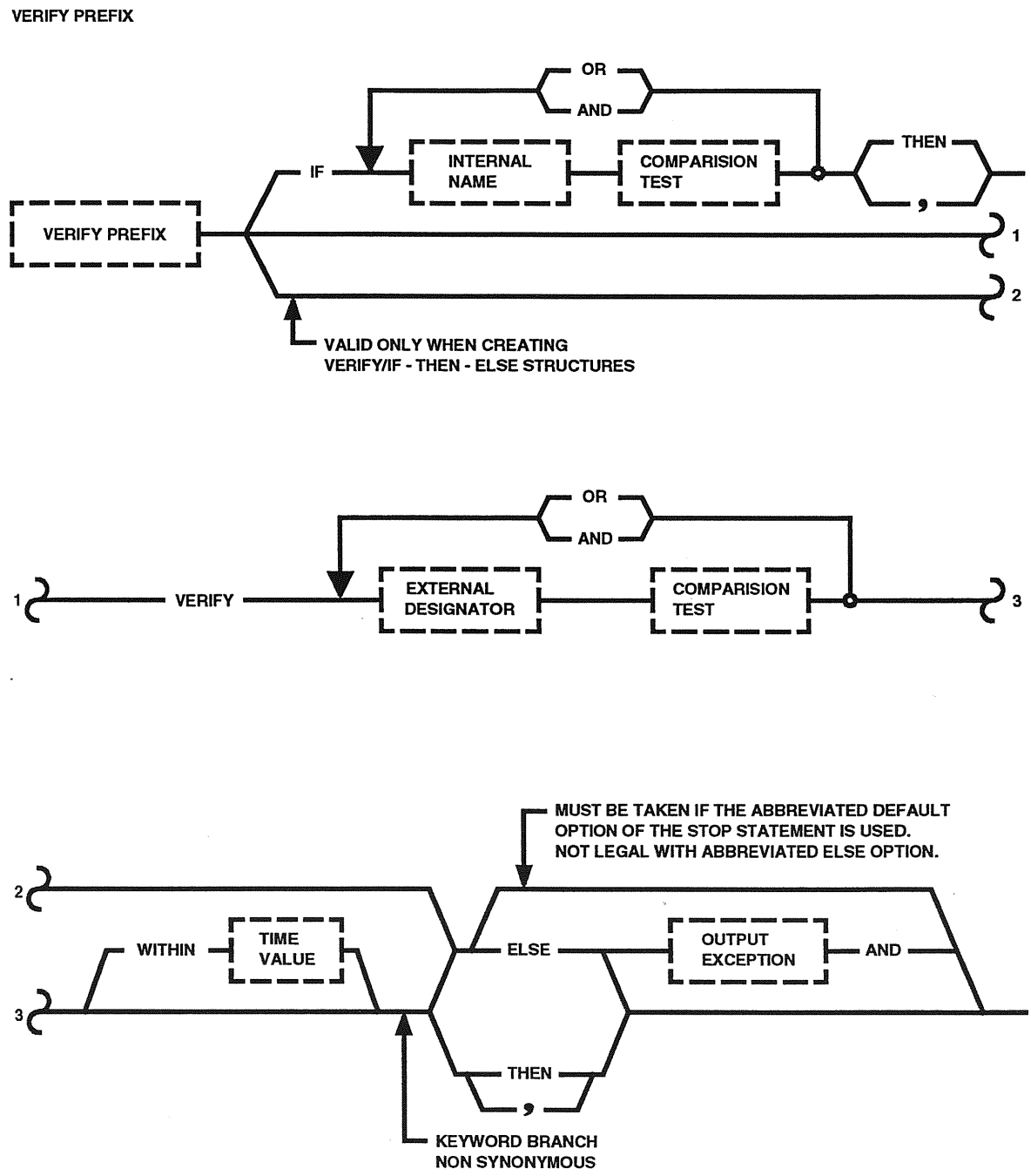


Figure 6-2 Verify Prefix

Function

The VERIFY PREFIX is used to cause the execution of a procedural statement or group of statements based upon the result of one or more Comparison Tests.

Description

The VERIFY PREFIX is an optional prefix for a procedural statement, providing a test and conditional transfer capability. It may be used to test Internal Names as well as Function Designators.

The IF option provides the capability to perform conditional branching based upon the value of an Internal Name. Testing the value is performed by one or more Comparison Tests. If the Comparison Test is successful, the remaining part of the statement will be executed. If the Comparison Test is not successful, the remaining part of the statement will not be executed; the next statement in the procedure will be executed.

The VERIFY option provides a conditional branching capability based upon the value of the specified Function Designators. If the Comparison Test is followed by the keyword THEN or a comma, the remainder of the statement will be executed only if the test is successful. The name of the CDT count/hold status pseudo Function Designator and its values can be found in the INTERRUPT PROCESSING STATEMENTS section of this document.

The VERIFY option also provides a negative Comparison Test by use of the keyword ELSE. In this case, if the result of the Comparison Test is not successful, the remainder of the statement will be executed. The keyword ELSE may be optionally followed by Output Exception. This option is used to record messages which describe the condition resulting in the failure to the operator. All exceptions are reported, even if they did not directly result in the failure of the test, but would have resulted in failure if they had actually participated in the test (subsequent tests are not performed after a failure is encountered).

The abbreviated ELSE option of the Verify Prefix (first keyword is ELSE) is used to create a VERIFY/IF-THEN-ELSE Statement structure, and may be used only in conjunction with an associated VERIFY/IF-THEN prefix. The abbreviated ELSE option specifies which statement or group of statements is to be executed when the result of the Comparison Test in the associated VERIFY/IF-THEN prefix is false. The ELSE option is used on the first procedural

statement following the statement or group of statements containing the associated VERIFY/ IF-THEN prefix. If omitted, the preceding VERIFY/IF-THEN is assumed to have a null ELSE clause which has the same effect as specifying "ELSE CONTINUE". A group of statements is specified by the use of the BEGIN and END SEQUENCE Statements.

The Comparison Test for both the IF and VERIFY options may be connected by the keywords AND or OR which provide a Boolean logical testing capability. If AND is used to connect Comparison Tests, then all of the Comparison Tests must be satisfied in order to have a successful test. If OR is used, any Comparison Test which is satisfied will result in a successful test. Both keywords, AND and OR, may not be used in the same statement.

The VERIFY option also allows an optional Time Value to be specified. If the Comparison Test is not successful, it will be repeated until the specified time interval has expired. At this time the test is considered unsuccessful and the remainder of the statement is processed according to the THEN/ELSE option. If the test is successful within the specified time interval, then the remainder of the statement will be processed according to the THEN/ELSE option.

When the THEN or ELSE options are not used, the VERIFY PREFIX then becomes an abbreviated form of the STOP Statement. The effect is the same as if the keywords ELSE RECORD EXCEPTIONS AND STOP had been specified. In this case, the VERIFY PREFIX is used as a statement. Execution will stop if the Comparison Test is not successful and an exception will be reported to Page-A. Otherwise, the next sequential statement in the procedure is executed.

Examples

```
IF (Q11) IS GREATER THAN (Q2) THEN GO TO STEP 5;
```

This example illustrates the IF option by doing a compare between the two Internal Names Q11 and Q2. If the Comparison Test is true, control is transferred to the statement identified by STEP 5. If the test is false, control is passed to the next sequential statement in the procedure.

```
IF (STATE1) IS ON, GO TO STEP 5;
```

This example is similar to the above example with the comma replacing THEN.

```
IF (QLIST1) IS BETWEEN (QLIST3) AND (QLIST4), SET < DS1 > TO
OFF;
```

This example also performs a Comparison Test to determine if each item of the list QLIST1 is between the corresponding items of QLIST3 and QLIST4. If the test is true, the Function Designator DS1 is set to OFF. If it is false, control falls through to the next sequential statement of the procedure.

```
IF (NLIST) = 5, GO TO STEP 5;
```

This example tests each element of the list NLIST to see if it is equal to 5. If all elements are equal to 5, control is transferred to STEP 5. Otherwise, control falls through to the next sequential statement.

IF Option With Lists

Examples

```
IF (NLIST1) = 5, GO TO STEP 5;
IF (QLIST1) IS BETWEEN (QLIST3) AND (QLIST4), SET < DS1 > TO
OFF;
IF (QLIST1) 2 IS LESS THAN 5 V, APPLY 2 V TO < AS1 > ;
```

IF Option With Tables

Examples

```
IF (NTABLE1) ROW 1 COLUMN 2 = ROW 1 COLUMN 3, GO TO STEP 5;
IF (STABLE1) COLUMN 2 IS OFF, GO TO STEP 5;
```

IF Option - Boolean

Examples

```
IF (N1) = 1
    AND (N2) = 2
    AND (N3) = 3
    THEN GO TO STEP 5;
IF (Q1) IS LESS THAN -5V
    OR (Q1) IS GREATER THAN 5V
    THEN GO TO STEP 5;
```

VERIFY Option With Constants

Examples

```
VERIFY < AM1 > IS GREATER THAN 5 V THEN GO TO STEP 5;
VERIFY < DM1 > IS ON THEN GO TO STEP 5;
VERIFY < DPM1 > IS EQUAL TO B11011 ELSE GO TO STEP 5;
VERIFY < DM1 >
       < DM2 >
       < DM3 > ARE OFF THEN GO TO STEP 5;
```

VERIFY Option With Names

Examples

```
VERIFY < AS1 >
       < AS2 >
       < AS3 > ARE GREATER THAN (Q1) THEN GO TO STEP 5;
VERIFY < DM1> IS EQUAL TO (STATE1) ELSE GO TO STEP 5;
```

VERIFY Option With Lists

Examples

```
VERIFY < AM1 >
       < AM2 >
       < AM3 > ARE BETWEEN (QLIST1) AND (QLIST3);
VERIFY < DM1 >
       < DM2 >
       < DM3 > ARE EQUAL TO (SLIST1) THEN GO TO STEP 5;
```

VERIFY Option With Tables

Examples

```
VERIFY < AM1 >
       < AM2 >
       < AM3 > ARE BETWEEN (QTABLE1) COLUMN 1 AND COLUMN 2
    ELSE RECORD EXCEPTIONS TO < PAGE-B > AND GO TO STEP 5;
VERIFY (STABLE1) FUNCTIONS ARE EQUAL TO COLUMN 1 THEN
    GO TO STEP 55;
```

VERIFY Option - Boolean

Examples

```
VERIFY < DM1 > = (STATE1) AND
       < DM2 > = (STATE2) THEN GO TO STEP 5;
```

```
VERIFY < DM1 > = (STATE1) OR
       < DM1 > = (STATE2) ELSE RECORD EXCEPTIONS AND STOP;
```

VERIFY Option Within Time Value

Examples

```
VERIFY < DM1 > IS ON WITHIN 5 SEC ELSE GO TO STEP 5;
VERIFY < DM1 > IS OFF WITHIN (TIME) THEN GO TO STEP 5;
```

VERIFY Option - Abbreviated STOP

Examples

```
VERIFY < DM1 > IS ON;
VERIFY < AM1 > IS LESS THAN 5 V;
```

Abbreviated ELSE Option

```
IF (STATE) IS ON
    THEN LET (NUMBER3) = 6;
    ELSE LET (NUMBER4) = 10;

VERIFY <DM3> IS EQUAL TO (STATE1)
    THEN BEGIN SEQUENCE;
        SET <DS1> TO OFF;
          .
          .
          .
        GO TO STEP 100;
    END SEQUENCE;
    ELSE BEGIN SEQUENCE;
        LET (STATE3) = OFF;
          .
          .
          .
        APPLY (Q4) to <AS1>;
    END SEQUENCE;
```

Statement Execution

The values of the Function Designators requested in the VERIFY PREFIX are obtained by the GOAL Executor, and the Comparison Test is executed.

A. If the Function Designator is a measurement or command resident in the CDBFR, the GOAL Executor will read the value of the Function Designators from the CDBFR.

B. If the Function Designator references a measurement to be obtained via the Launch Data Bus FEP, the GOAL Executor will request the LDB FEP to format a TCS-1/SACS operator to the GPC on board. The GPC will obtain the value and return it to the CCMS. This communication with the LDB takes place via the CDBFR.

C. Pseudo Function Designators are read directly from the CDBFR by the GOAL Executor.

D. If the interval timer, Greenwich Mean Time or countdown time is referenced, the Time Value is obtained from memory in the console.

E. If a PFP function key indicator is referenced by the Verify Prefix, the CCMS Operating System is requested by the GOAL Executor to return the light status.

F. If a VERIFY OPTION WITHIN TIME VALUE is specified, the comparison will be performed as above. In addition the time will be checked each time the Comparison Test is performed. When the time is exhausted, the test will fail.

If the Table Functions option of External Designator is used on a VERIFY option and rows of the table are inhibited, then the VERIFY will not be performed on the inhibited rows. Tests on inhibited rows are regarded as successful.

The VERIFY may be used with External Designator to obtain the values of many Function Designators with a single CDBFR request. Thus, the procedure does not have to wait once for each Function Designator.

Error Processing

When executing a VERIFY, if an error is encountered on the LDB or CDBFR read requests, a Class III error will be processed. If the error occurs when using External Designator to VERIFY many Function Designators, the Class III error will be processed on the Function Designator that caused the error. If procedure error override is active, the procedure will continue; if procedure error override is inhibited, the procedure will be placed in a Class II error stop. Invalid indexing into lists or tables will cause a Class III error.

Restrictions/limitations

Refer to the following for restrictions and limitations:

The following comparisons are legal:

One-to-One: A single item is tested against a single condition.

Many-to-One: Many items are tested against a single condition. For relational formulas other than NOT EQUAL and for limit formulas other than NOT BETWEEN, the condition must be satisfied for each item in order to have a successful comparison. For the NOT EQUAL and NOT BETWEEN formulas, if any item satisfies the condition, the comparison test will be successful.

One-to-Many: A single item is tested against many conditions. For relational formulas other than NOT EQUAL and for limit formulas other than NOT BETWEEN, all conditions must be satisfied for a successful comparison. For the NOT EQUAL and NOT BETWEEN formulas, if any condition is satisfied for the item, the comparison is successful.

Many-to-Many: Many items are tested against many conditions. Each item is tested against its related condition, and for all formulas other than the NOT EQUAL and NOT BETWEEN, all tests must be satisfied in order to have a successful comparison. For the NOT EQUAL and NOT BETWEEN formulas, if any test is satisfied the comparison will be successful.

CAUTION

NOT EQUAL and NOT BETWEEN comparisons work differently than other comparisons.

Acceptable Conditions

When multi-word digital patterns are mixed with regular digital patterns, the multi-word digital patterns should be coded first.

A One-To-Many comparison joined, by the OR operation, with another relational expression will return true if any condition is satisfied.

Examples

NOT Comparison Tests

```
VERIFY <DM1> <DM2> <DM3> ARE NOT EQUAL TO (STATE1)
    THEN GO TO STEP 100;

VERIFY <AM1> <AM2> <AM3> ARE NOT BETWEEN (QVALUE1) AND (QVALUE2)
    THEN GO TO STEP 200;
```

In the two examples above, if any Function Designator satisfies the condition, the THEN part will be executed.

```
VERIFY <DM1> IS NOT EQUAL TO (DLIST2)
    THEN GO TO STEP 300;

IF (NUM1) IS NOT BETWEEN (NLIST2) AND (NLIST3)
    THEN GO TO STEP 400;
```

In the two examples above, if any of the conditions are satisfied for the item, the THEN part will be executed.

NOTE: *An analog measurement may not be tested as an equal or not equal condition.*

Output exceptions may be used with the abbreviated ELSE option (see statement description for abbreviated ELSE) only in a Verify/Else structure, not Verify/Then/Else.

The maximum number of items which may be verified in one statement is 224 (the number of exception bits allowed by the GOAL Executor).

An If or Verify clause may not immediately follow an ELSE or THEN option. However, if BEGIN SEQUENCE is used after an Else or Then, If or Verify may follow.

Legal Function Designators

Analog Measurements (AM)
Analog Stimuli (AS)
Countdown Time (CDT)
Digital Pattern Measurements (DPM)
Digital Pattern Stimuli (DPS)
Discrete Measurements (DM)
Discrete Stimuli (DS)
Greenwich Mean Time (GMT)
Interval Timer (TIMR)
Julian Time of Year (CDT)
Mission Elapsed Time (CDT)
PFP Lights (PFPL)
Pseudo Analogs (PA)
Pseudo Discretes (PD)
Pseudo Digital Patterns (PDP)
System Status Discrete (SSA1)
System Status Digital Pattern (SSA2)

THIS PAGE INTENTIONALLY LEFT BLANK.

6.3 REPEAT STATEMENT

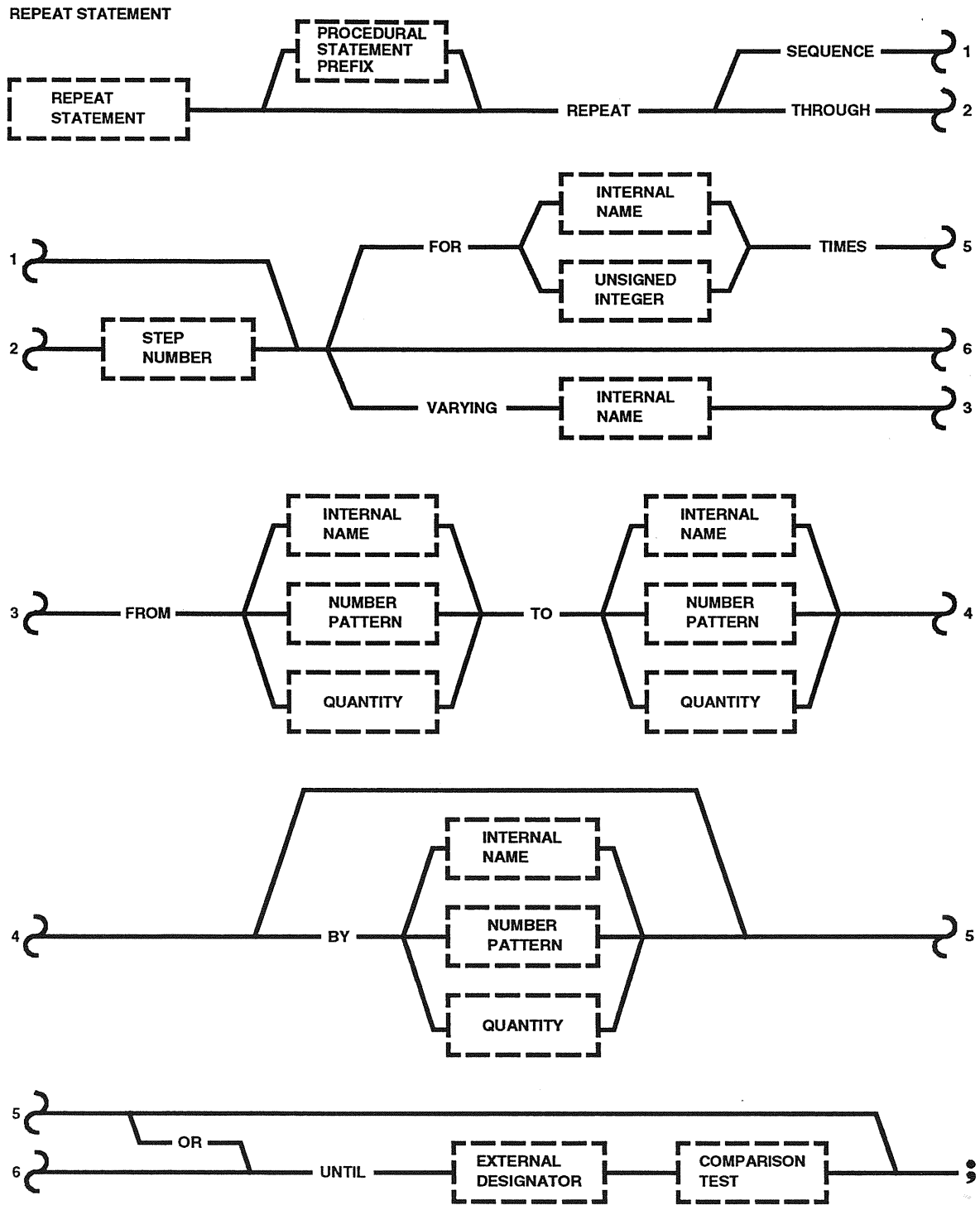


Figure 6-3 REPEAT Statement

Function

The REPEAT Statement allows repetitive execution of a group of procedural statements.

Description

The REPEAT Statement is used to repeat a group of procedural statements a specified number of times or until a specified condition is met. The SEQUENCE option may be used if a sequence is defined immediately following the REPEAT Statement. The REPEAT Statement is the first statement in a repeat loop and defines the beginning of the repeat loop. The statement referenced in the REPEAT Statement by Step Number is the last statement of the repeat group unless the SEQUENCE option is specified. If the SEQUENCE option is specified, the entire sequence, starting with the BEGIN SEQUENCE Statement and ending with the END SEQUENCE Statement, will be repeated.

The THROUGH STEP NUMBER option is used to identify the last statement of the repeat group which follows the REPEAT Statement. The last statement of the repeat group must be prefixed by the Step Number specified in the REPEAT Statement following the keyword THROUGH. The Step Number specified cannot be referenced by any other REPEAT Statements.

The number of times a repeat group is to be executed may be specified in the FOR option with an unsigned integer or an internal variable. The value specified must be greater than zero and the initial pass is included.

The VARYING option allows specification of a repeat index with an initial value (after keyword FROM), a final value (after keyword TO) and an increment value (after keyword BY). The repeat index can be either a numeric or non-time quantity value. If the increment value is omitted, the value will default to 1 if the repeat index is a numeric value, or 1.0 if the repeat index is a quantity value. The index is incremented by the specified increment value following each execution of the repeat group. The increment value can be positive or negative. However, polarity must be such that the final value may be reached by adding the increment value to the initial value. Execution of the repeat group is terminated when the repeat index has reached the specified final value.

The UNTIL option allows a conditional loop control for the REPEAT Statement. The condition is specified by the External

Designator/Comparison Test. After each execution of the loop, the Comparison Test is performed. If the test is successful, processing will continue at the next sequential statement following the end of the repeat group. If the test fails, the loop is repeated.

The name of the CDT count/hold status pseudo Function Designator and its values can be found in the INTERRUPT PROCESSING STATEMENTS section of this document.

Both a loop count and a conditional test may be specified by using the OR UNTIL option. In this case the loop will be executed until the loop count is exhausted or until the conditional test is satisfied, whichever occurs first.

The conditional test may also be used with a repeat index. The repeat group is executed until either the conditional test is satisfied or the index reaches the final value, whichever comes first.

A repeat group may be embedded in another group to form a nested repeat group. A nested repeat group must be contained within the boundaries of its outer repeat group. Nine levels of nesting of repeat groups are allowed.

Examples

```
          REPEAT THROUGH STEP 10 FOR 5 TIMES;
            .
            .
            .
  STEP 10 CONTINUE;
```

This example will repeat the block of code through step 10 five times.

```
          REPEAT SEQUENCE UNTIL < DM2 > IS ON;
          BEGIN SEQUENCE;
            .
            .
            .
            .
            .
          END SEQUENCE;
```

This example will repeat the defined sequence until Function Designator DM2 is on. The conditional check is made at the end of

the repeat group's execution for each pass and will continue executing until the specified condition is true.

```
        REPEAT THROUGH STEP 5 FOR 25 TIMES;
          .         .
          .         .
          .         .
          .         .
        REPEAT THROUGH STEP 4 FOR (N9) TIMES;
          .         .
          .         .
          .         .
          .         .
          .         .
STEP 4 CONTINUE;
STEP 5 CONTINUE;
```

The example illustrates a nested repeat loop. The outer repeat loop will be executed 25 times before continuing to the next sequential statement following STEP 5. The nested repeat loop will be executed the number of times specified by (N9) each time it is entered. The nested repeat loop will be entered 25 times, and each time the statements in the repeat loop will be executed (N9) times.

```
        REPEAT SEQUENCE VARYING (VOLT25) FROM 20 TO 30
            BY 2 OR UNTIL < AM25 > = 2 AMPS;
        BEGIN SEQUENCE;
          .         .
          .         .
          .         .
          .         .
        END SEQUENCE;
```

This example illustrates two options of the REPEAT Statement. The sequence will be repeated varying the Internal Name VOLT25 from 20 to 30 in increments of two or until the value of Function Designator is equal to 2 amps. Execution of the sequence will continue until VOLT25 equals 30 or until AM25 equals 2 amps, whichever comes first. If the value of AM25 is equal to 2 amps initially, then one execution of the sequence will occur.

Statement Execution

When the GOAL Executor encounters the operators generated by the REPEAT Statement, it sets up the Loop Control Table (LCT) in the VDA.

A fixed number of repeat loop counters are set aside in the loop control table for nested repeats. When using the REPEAT GROUP DEPTH directive, it is possible to exceed the size of the LCT if interrupt routines do repeat loop nesting after interrupting from nested loops. If the size of the LCT is exceeded, the exceeding repeat loops' counters are written over the first LCT entry, the second, and so on. An error will occur at the bottom of any loop which has had its counters overwritten.

If the VARYING option of the statement is used, a check is made at the end of each pass through the repeat loop to ensure the increment value will eventually cause the initial value to be equal to the final value. There is always at least one pass made through the loop.

If the Comparison Test of the statement is taken, the test is performed after each execution of the loop. If the test is successful, control will be passed to the next sequential statement following the end of the repeat loop. If the test fails, the loop is repeated.

Error Processing

A Class II error will be processed if the LCT is out of range. Failure of CDBFR reads will result in a Class III error.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. If the SEQUENCE option of the statement is not used, the last statement of the repeat loop must be preceded by the Step Number which is referenced by the REPEAT Statement.

B. A Step Number may not be referenced by more than one REPEAT Statement.

C. The last statement of a repeat loop cannot be a GO TO, STOP, TERMINATE, or REPEAT Statement.

D. Only nine levels of nesting of repeat loops are allowed.

E. Branching into a repeat loop is not permitted. Branching within a repeat group is permitted.

F. The number of times specified for a repeat loop to execute must be greater than zero.

G. Due to the manner in which the VERIFY PREFIX is executed, if this prefix is a part of the last statement of a repeat loop and the comparison test fails, execution of the procedure will continue outside the repeat loop.

H. The VERIFY PREFIX is not allowed on the REPEAT Statement. If the Then or Else clause is not executed, loop counters are not properly initialized.

I. When the VARYING option is used with Internal Names, the repeat index, initial value, final value and increment value must be of the same type (a quantity repeat index may have a literal numeric initial value, final value or increment value). The type must be a numeric or non-time quantity single-data item. The absolute value of the difference of the initial value and final value cannot be greater than 32767 when using either literals or internal names of numeric type.

J. When the FOR option is used with an Internal Name, the Internal Name must be a numeric single-data item.

K. An ACTIVATE...AND RETURN Statement is not allowed within a Repeat Group.

Legal Function Designators

The following Function Designators are legal for use by the REPEAT Statement.

Analog Measurements (AM)
Analog Stimuli (AS)
Countdown Time (CDT)
Digital Pattern Measurements (DPM)
Digital Pattern Stimuli (DPS)
Discrete Measurements (DM)
Discrete Stimuli (DS)
Greenwich Mean Time (GMT)
Mission Elapsed Time (CDT)
Interval Timer (TIMR)
Julian Time of Year (CDT)
PFP Lights (PFPL)
Pseudo Analogs (PA)
Pseudo Discretes (PD)
Pseudo Digital Patterns (PDP)

THIS PAGE INTENTIONALLY LEFT BLANK.

6.4 PERFORM STATEMENT GROUPS STATEMENT

PERFORM STATEMENT GROUPS STATEMENT

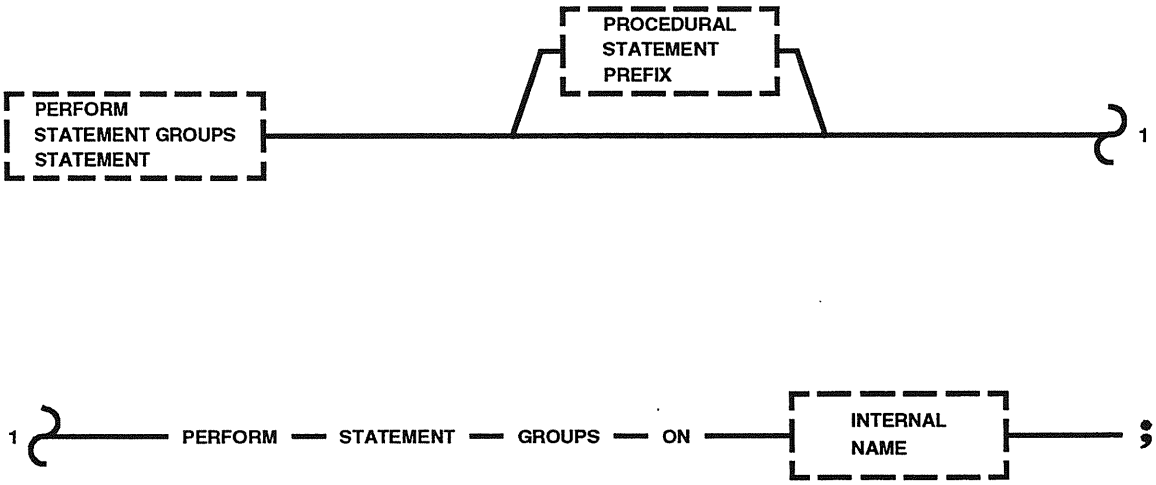


Figure 6-4 PERFORM STATEMENT GROUPS Statement

Function

The PERFORM STATEMENT GROUPS Statement is used to indicate that one of the following series of statement groups is to be executed depending on the value of a specified internal variable and to identify the start of a statement groups structure.

Description

The PERFORM STATEMENT GROUPS Statement is the first statement of a statement groups structure and defines the beginning of the statement groups structure.

The PERFORM STATEMENT GROUPS Statement can be used only once within any one statement group structure unless it defines the beginning of an embedded statement groups structure.

All procedure statements following the PERFORM STATEMENT GROUPS Statement must be a part of a specified group. Each statement group is identified by a Statement Group Label.

Special Considerations

Statement group structures may be embedded within one of the statement groups of another statement group structure. The embedded statement group structure must be completely contained within the statement group of the other structure.

An END STATEMENT GROUPS Statement must be specified for each PERFORM STATEMENT GROUPS Statement to define the end of the statement groups structure.

To insure proper return from error conditions, each case of the statement group structure will have a second ISN operator with an ISN of zero. This second operator is transparent during execution except when single-stepping, at which time two ISN operators will appear for each statement, the first with the actual ISN, and the second with an ISN of zero (which will not appear on the expanded source).

Restrictions

A. Branching into a statement group structure is not permitted.

B. Branching out of a statement group structure is permitted.

C. Branching within a statement group structure is permitted only within the same statement group.

D. The Internal Name specified must be a single-numeric quantity or text data item.

Examples

```
PERFORM STATEMENT GROUPS ON (N1);
    STATEMENT GROUP FOR (N1) EQUAL TO (1)
        BEGIN SEQUENCE;
            APPLY 6.0 V TO <A0-2001>;
            APPLY 10.0 V TO <A0-02>;
        END SEQUENCE;
    STATEMENT GROUP FOR (N1) EQUAL TO (2)
        BEGIN SEQUENCE;
            LET (A) = 1;
            ISSUE (A) TO <DPO-01>;
        END SEQUENCE;
    STATEMENT GROUP FOR (N1) EQUAL TO OTHER VALUES
        BEGIN SEQUENCE;
            SET <D0-01> TO OFF;
            SET <D0-02> TO ON;
        END SEQUENCE;
END STATEMENT GROUPS FOR (N1);
```

6.5 END STATEMENT GROUPS STATEMENT

END STATEMENT GROUPS STATEMENT

Figure 6-5 END STATEMENT GROUPS Statement

Function

The END STATEMENT GROUPS Statement is used to indicate the end of a statement groups structure associated with the specified internal variable.

Description

The END STATEMENT GROUPS Statement is the last statement of a statement groups structure and defines the end of the statement groups structure.

The END STATEMENT GROUPS Statement can appear only once within any statement groups structure unless it defines the end of an embedded statement groups structure.

The Internal Name specified must be identical to the Internal Name specified in the associated PERFORM STATEMENT GROUPS Statement.

One and only one END STATEMENT GROUPS Statement must be specified for every PERFORM STATEMENT GROUPS Statement and must follow the PERFORM STATEMENT GROUPS Statement and the procedural statements which comprise the statement groups structure.

THIS PAGE INTENTIONALLY LEFT BLANK.

6.6 STATEMENT GROUP LABEL

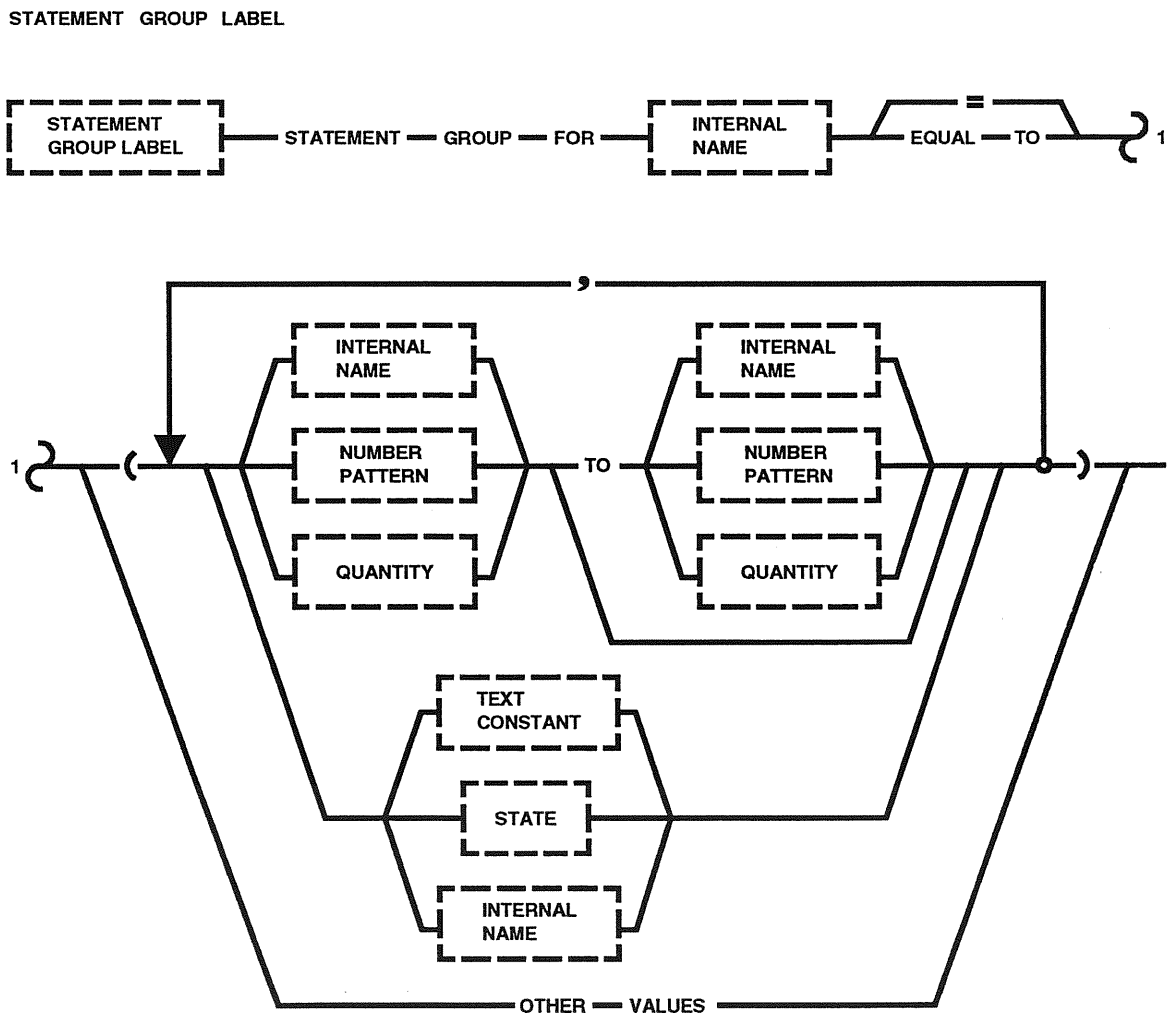


Figure 6-6 STATEMENT GROUP LABEL

Function

The Statement Group Label element is used to identify the different statement groups used in a PERFORM STATEMENT GROUPS STRUCTURE that is bounded by PERFORM STATEMENT GROUPS and END STATEMENT GROUPS Statements.

Description

The STATEMENT GROUP LABEL acts as a prefix to a procedural statement (or group of statements) in a statement group structure. The procedural statement to which it is attached is executed only when the Internal Name specified meets the criteria of the comparison tests unless the OTHER VALUES option is used.

The TO keyword is used to generate a between test for a range of values. The between test is inclusive of endpoint values.

The comma is used to include additional comparison tests in a single statement group label.

The OTHER VALUES option is used to designate a procedural statement (or group of statements) to be executed when none of the other comparison tests specified in previous STATEMENT GROUP LABELS (within the same structure) are successful.

If the OTHER VALUES option is used, it must appear as the last STATEMENT GROUP LABEL in the structure.

Statement Group Labels may appear only within the bounds of PERFORM STATEMENT GROUPS and END STATEMENT GROUPS Statements.

To specify a group of statements, the STATEMENT GROUP LABEL may preface a BEGIN SEQUENCE Statement in which case the entire sequence will be executed as a result of satisfaction of the statement group comparison tests.

Special Considerations

If overlapping or identical tests appear in more than one STATEMENT GROUP LABEL, only the statement group whose comparison tests are satisfied first will be executed.

Restrictions

A. The Internal Name specified on the STATEMENT GROUP LABEL must be identical to the Internal Name specified on the associated PERFORM STATEMENT GROUPS Statement.

B. When the Internal Name to be compared is a text constant or state, the TO option cannot be specified.

C. The range specification of the TO option must be consistent. If the beginning range and ending range are incompatible (e.g., 1 TO (TEN), where (TEN) is declared a quantity variable), a syntax error will result.

Example

```
PERFORM STATEMENT GROUPS ON (N1);
    STATEMENT GROUP FOR (N1) EQUAL TO (1 TO 3)
        BEGIN SEQUENCE;
            APPLY 6.0 V TO <A0-2001>;
            APPLY 10.0 V TO <A0-02>;
            PERFORM STATEMENT GROUPS ON (N2);
                STATEMENT GROUP FOR (N2) EQUAL TO (0)
                    LET (N3) = 1;
                STATEMENT GROUP FOR (N2) EQUAL TO OTHER VALUES
                    LET (N3) = 2;
            END STATEMENT GROUPS FOR (N2);
        END SEQUENCE;
    STATEMENT GROUP FOR (N1) EQUAL TO (4 TO 10, 15 TO 20, 25)
        APPLY 2.0 V TO <A0-2001>;
    STATEMENT GROUP FOR (N1) EQUAL TO OTHER VALUES
        APPLY 2.0 V TO <A0-02>;
END STATEMENT GROUPS FOR (N1);
```

7. RECORD AND DISPLAY STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The RECORD and DISPLAY Statements are used to format and output data to the DG color displays, programmable function panel, SPA printer, console printer/plotter. These statements may also be used to log data to the CDS or PDR for historical retrieval. The following Record and Display Statements will be discussed in the sections indicated:

A. RECORD DATA STATEMENT (Section 7.1)

The RECORD DATA Statement is used to display, print, and write data from a GOAL procedure to peripheral devices such as CRT terminals, printers, and data files. Major options of the RECORD DATA Statement are as follows:

RECORD DISPLAY OPTION (Section 7.1.1)

RECORD PRINT OPTION (Section 7.1.2)

RECORD WRITE OPTION (Section 7.1.3)

RECORD FORMAT OPTION (Section 7.1.4)

B. MODIFY DISPLAY STATEMENT (Section 7.2)

The MODIFY DISPLAY Statement is used to modify the blink, invert, and color characteristics of data on a display page or to modify the blink characteristics of the PFP LED's.

C. CLEAR DISPLAY STATEMENT (Section 7.3)

The CLEAR DISPLAY Statement is used to clear data from display pages of PFP LED's.

D. DISPLAY SKELETON STATEMENT (Section 7.4)

The DISPLAY SKELETON Statement is used to display skeletons to display pages.

E. OUTPUT EXCEPTION (Section 7.5)

The Output Exception element is used to output messages which describe discrepancies detected during Comparison Tests performed in the VERIFY PREFIX.

THIS PAGE INTENTIONALLY LEFT BLANK.

7.1 RECORD DATA STATEMENT

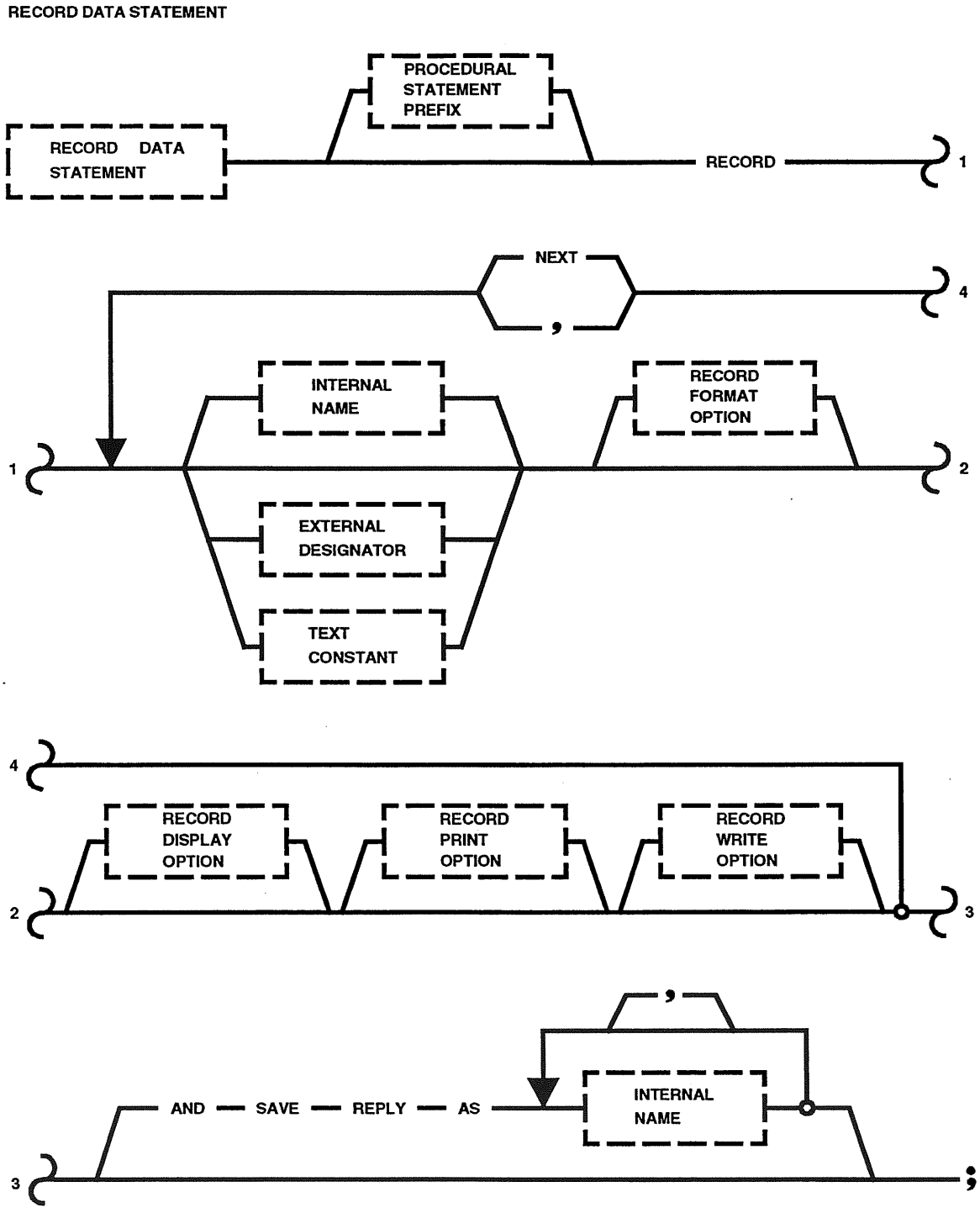


Figure 7-1 RECORD DATA Statement

Function

The RECORD DATA Statement is used to:

A. Print, display, and/or write data from GOAL procedures to peripheral devices such as CRT's, printers, and tape files.

B. Request keyed-in data from the on-line CRT terminal and save it for subsequent processing.

C. Control the formatting of data.

Refer to the GOAL On-board Interface Language User Guide, KSC-LPS-OP-033-04, for the on-board interface RECORD DATA Statement.

Description

The data which may be output by the statement includes: Text Constants, Internal Names, and Function Designators. The types of output operations which may be selected are:

A. RECORD DISPLAY Option

Output data is converted to text for display on an on-line CRT terminal. This option also supports the LED matrix on the Programmable Function Panel.

Cursor coordinates will be specified in a format compatible with the Display Generator. Character position on a line may be specified as Character 0 through 71. Lines on the screen may be addressed as line 4 through 33.

B. RECORD PRINT Option

Output data is converted to text for printing on the SPA Printer or on the console Printer/Plotter.

C. RECORD WRITE Option

Output data is written as raw data records to tape files for historical retrieval.

D. RECORD FORMAT Option

Output data is edited for display according to specified formats.

E. AND SAVE REPLY AS Option

A user-defined message is displayed and an operator supplied, keyed-in response is to be entered. Procedure execution is suspended until the response is provided by the operator and saved in the designated internal variable(s).

The RECORD Statement allows a series of data items to be grouped together for use in an output operation. Consecutive data items in a group must be separated by either a comma or the keyword NEXT. The comma is used for readability only and does not indicate a new line of output. The null path on the syntax diagram (which includes use of blanks) produces the same result as the specification of a comma. The keyword NEXT is used to indicate the end of a logical grouping of data items. It is considered as a begin-new-line indicator for the output operation and provides the capability of inserting one or more blank lines prior to or immediately following the recorded data.

Multiple output options may be specified for a given group of output data items. This specification causes the data to be sent to the devices selected in each of the output options. Care must be taken to ensure that the message length, contents, and usage are compatible with all of the output devices specified for a given group of data items. If no output option is selected, the output data is directed to the next available line on PAGE-A.

When the output is converted for display or printing, a standard format is used for each different type of data. These data types are as follows:

A. TEXT CONSTANT

The characters of a Text Constant are output exactly as specified. Note that engineering graphics cannot be printed.

B. NUMERIC DATA

The internal data is converted to a default Number Pattern format according to the way that the data was declared.

Declared as Numeric Type:

DEFAULT FORMAT:

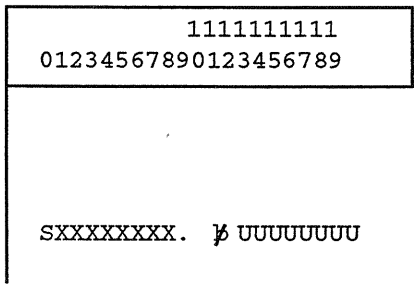
		Character Field Positions 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0	
Integer	RECORD (N10);	5 1 0	Field = Sign + numeric value with leading zeros suppressed, left justified.
Hexadecimal	RECORD (N10);	X 0 0 0 A	Field = ⊬X + 4 characters with leading zeros inserted, right justified.
Octal	RECORD (N10);	T 0 0 0 0 1 2	Field = ⊬T + 6 characters with leading zeros inserted, right justified.
Binary	RECORD (N10);	B 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0	Field = ⊬B + 16 characters with leading zeros inserted, right justified.

C. QUANTITY DATA

The internal data is converted to a Quantity format. Engineering units are appended according to their declaration.

Declared as Quantity Type:

DEFAULT FORMAT:



Note: Decimal point floats per the fractional number's value.

D. STATE DATA

The internal data is converted to a State format according to its declaration and abbreviated to three output characters as: OFF, ON, OPN, CLS, TRU, FLS, WET, DRY, N/F, O/C, T/F, or W/D.

E. TEXT DATA

The internal data is output directly as text characters; leading and trailing blanks are included. Note that engineering graphics cannot be printed.

F. TIME DATA

A quantity data declared as time is output as follows:

Declared as Quantity - Time Type

DEFAULT FORMAT

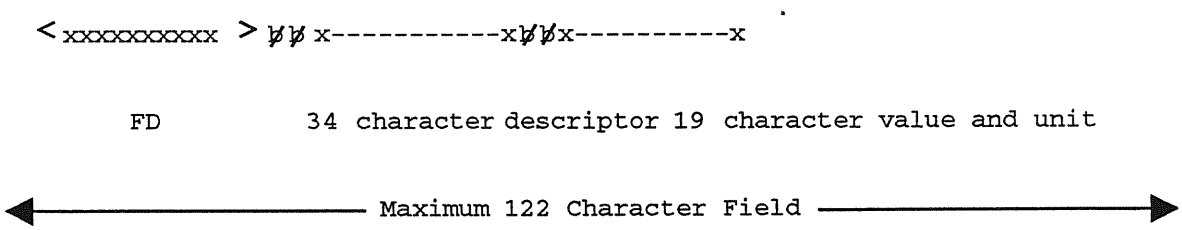
	111111 0123456789012345
- RECORD (GMT)	+HHMM/SS.ZZZ
- RECORD (CDT) or RECORD (MET)	±DD:HHMM/SS⊘
- RECORD (JTOY)	DDD:HHMM/SS⊘
- RECORD (INTERVAL)	+HHMM/SS.ZZZ

Where: D = days, H = hours, M = minutes, S = seconds, and Z = milliseconds.

G. PRESENT VALUE OF A FUNCTION DESIGNATOR

The name, descriptor, and value of the specified Function Designator are given. The value is converted for output in the format of Number, Quantity, State, or Text, depending on type (refer to Table 7-1).

Present Value of Function Designator default output is:



When Table Functions are recorded, the default is to follow each row with a carriage return. Cursor control information is only used for positioning the initial row.

TABLE 7-1 FUNCTION DESIGNATOR OUTPUT VALUES

Function Designator Type	Output Format
Analog Measurement/Stimulus (AM/AS)	Quantity
Discrete Measurement/Stimulus (DM/DS)	State
Digital Pattern Measurement/Stimulus (DPM/DPS)	Numeric
GMT, CDT, MET, Interval Timer (GMT, CDT, MET, TIMR)	Time
PFP Lights (PFPL)	State
Pseudo Analog (PA)	Quantity
Pseudo Discrete (PD)	State
Pseudo Digital Pattern (PDP)	Numeric
Date (DATE)	Text
Fixed Page 1 (FPG1)	Text
Analog Measurement Double Precision (AMDP)	Quantity
Multiword Digital Pattern (MWDP)	Numeric [1]
Floating Point (FP)	Quantity
System Status Area 1 (SSA1)	State
System Status Area 2 (SSA2)	Numeric

1. MWDP data is displayed as two to four individual numeric fields.

The RECORD FORMAT option may be used to format internal or external data items which are output via the DISPLAY AND PRINT options. This option allows for selected conversion and edit parameters to be specified for formatting of output data. Care must be taken to ensure that conversion and edit parameters are compatible with the data item referenced. If an internal data list, table column, table functions, or a string of Function Designator names is specified in the output data list, a series of output records are generated to present the data in a readable format.

The DISPLAY Option may reference an on-board CRT type Function Designator when displaying text data to the on-board CRT using the TCS 1-to-1 Text operator. See KSC-LPS-OP-033-04 for a description of the on-board RECORD.

The AND SAVE REPLY AS option may be used to request an operator response via the on-line CRT terminal. In order to use this option, an explicit message must first be output to the display page on which the response is to be entered. When the response (limited to one line) is provided by the operator, it is converted and stored in the specified internal variables. The conversion is performed according to the characteristics of the Internal Name(s). The response is entered by positioning the cursor to column zero, entering the response, and depressing the TRANSMIT RESPONSE function key, or by positioning the cursor to the right of a previously displayed response (left justified), then depressing the TRANSMIT RESPONSE function key.

There is no restriction on the amount of output data which may be directed to any output device, except for the PFP LED, (limited to no more than 17 characters) and disk files - Sector option (limited to 512 words). For other output devices, a series of output requests will be generated until all of the output has been performed. The amount of output data in a single I/O request is dependent on which of the following conditions occurs first:

A. 512 words less space for DG attribute information, printer carriage control, and format control for NEXT; or

B. The end of a series of Function Designators or a table function output. For example, if the following data was to be output:

   TEXT (Record two Function Designators)

<AI-01>
<AI-02>

TEXT (Followed by two more Function Designators)

<AI-03>
<AI-04>

The result would be to display/print the initial text constant, AI-01, and AI-02 in one I/O request, and then to display/print the remaining text constant, AI-03 and AI-04.

Although there is no restriction on the amount of output data directed to an output device, other than those identified, there are a few peculiarities when displaying an unformatted series of Function Designators or table functions to a display page. These peculiarities are as follows:

A. The GLP will generate a new write request whenever the executor's internal work area becomes full (1024 characters = 512 words) or near full (the next output data will cause the 512 maximum to be exceeded). When recording a table function (series or single) or a series of Function Designators which exceed the 512 words, the GLP will issue two or more write requests depending on the amount of data to be recorded. The number of Function Designators that can fit into 512 words are as follows:

QUANTITY (SP,DP,FP)	- 13 Function Designators (ANALOG)
NUMERIC (SP)	- 16 Function Designators (DIGITAL PATTERN)
NUMERIC (DP)	- 14 Function Designators (MULTIWORD)
STATE	- 17 Function Designators (DISCRETE)

If these limits are exceeded, i.e., more Function Designators are contained in the table, the output can then produce overwritten Function Designator data on the display, dependent upon the location on the screen where the output starts (next available line location for default line/column) and the amount of data in the subsequent output write.

B. For any default output (next available line write), the GOAL Executor checks to see if it can write the whole output to the screen without wrapping back to the top of the display page. If it cannot, the Executor will then output all this new data beginning at the first line on the page.

C. For any specific cursor coordinate records, each set of broken up Function Designators will be output beginning at the coordinate specified, thus consistently overwriting those Function Designators in the previous write.

NOTE: *No matter how you code a record to a Display Screen, the limitation for output is still 512 words; thus, if other text data is intermixed with a series of Function Designators or table functions, the above values for the number of Function Designators will diminish according to the size of this text.*

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The WRITE option may not be mixed with the DISPLAY OR PRINT options in a given RECORD DATA Statement.

B. If the External Designator following the keyword TO is specified in the form of Table Functions, output will not be performed for any Row Designator which is inhibited.

C. The keyword NEXT must be specified at least once if the TEXT, INTERNAL NAME, or EXTERNAL DESIGNATOR bypass option is selected.

D. <PAGE-A> and <PAGE-B>, used together in a table when table inhibits are used, must appear as consecutive entries.

E. If a list or table column is specified when using the AND SAVE REPLY AS option, the operator will be required to enter no more than a single value which will be used to initialize the entire list or table column.

F. When a units mismatch occurs while using the TABLE FUNCTIONS option, a compiler warning message will be output.

G. If the keyword NEXT is used in conjunction with explicit position references (Line/Column options in RECORD DISPLAY and Skip/Space options in RECORD PRINT), the explicit positioning is done before the execution of "NEXT". See the RECORD DISPLAY option section (Section 7.1.1) for an example of RECORD Text Constant to specific Line and Column with NEXT.

H. Default output formats cannot be determined for numeric, quantity, and time elements obtained from a list or table using an Internal Name as an index. However, state elements will be displayed correctly. Any element obtained from a list or table using an Internal Name as an index should have output formats explicitly stated. If the output format is not stated for numeric and time elements, the default format will be obtained from the first element of the list or table. For quantity, "NO E.U." will be displayed as the default output format.

I. If a series of Function Designators or table function(s) that exceed 13 Function Designators for Analogs, 16 Function Designators for Single-Precision Digital Patterns, 14 Function Designators for Multiword Digital Patterns, or 17 Function Designators for Discretes are used in a RECORD Statement to output data to the display page in an unformatted form, then the output may overwrite part of itself.

J. If the lines of output exceed the number of lines left on the display page, the Executor will wrap output at the top of the page unless it is a specific line write (cursor coordinates specified in the statement) which will also wrap the data to line 4 of the next page.

Legal Function Designators

The allowable output device Function Designators, grouped per option, are as follows:

A. DISPLAY Option types

1. Application pages (PAGE)

2. PFP LED's (LED)

B. PRINT Option types

1. SPA Printer (PRTR)

2. Console Printer/Plotter (CPP)

3. PDR Recorder (PDRR)

4. CDS Recorder (PDRR)

C. WRITE Option types

1. PDR Recorder - Raw Data (FILE)

2. CDS Recorder - Raw Data (FILE)

For Function Designator types whose present values may be recorded, refer to Table 7-1.

7.1.1 RECORD DISPLAY OPTION

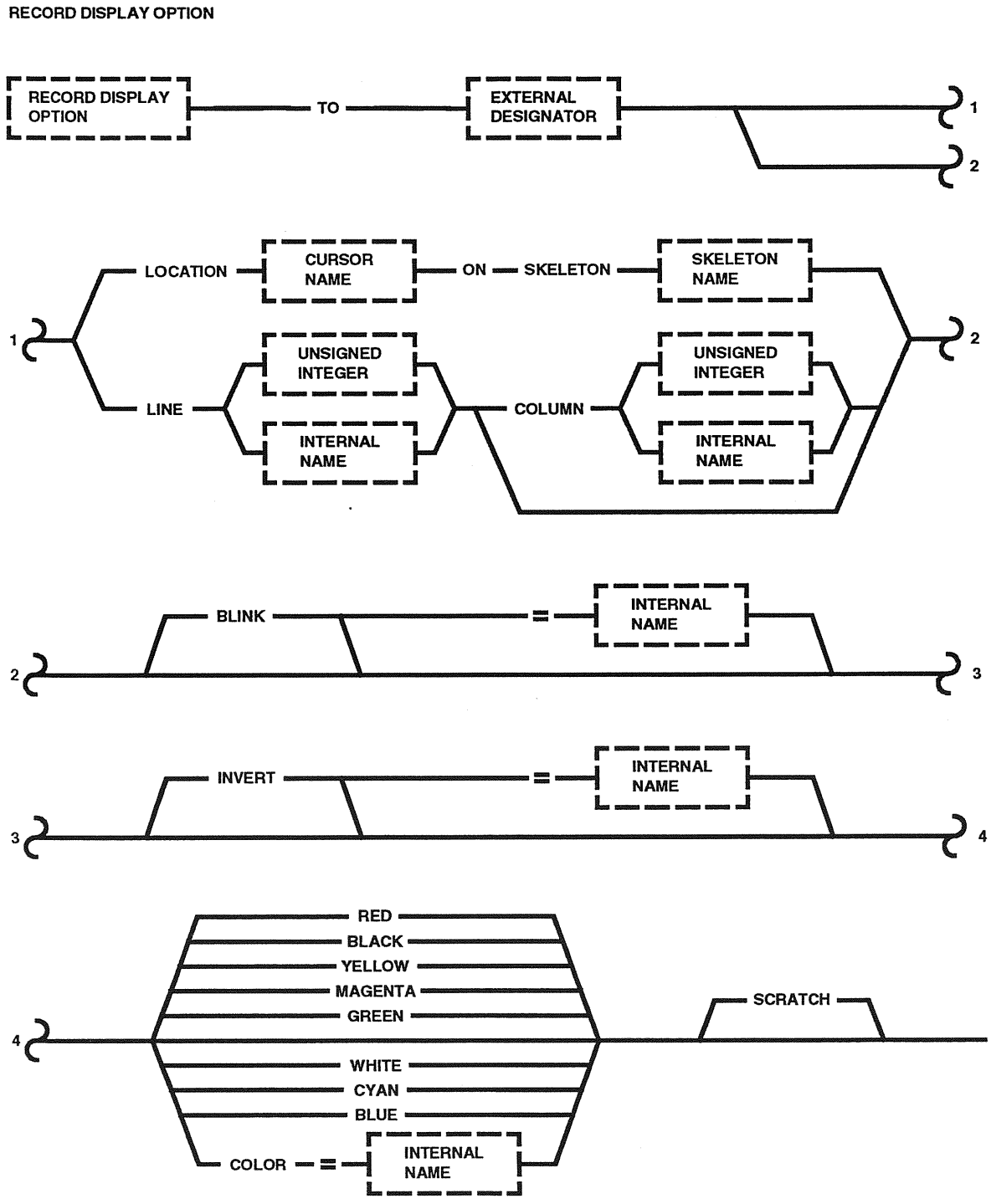


Figure 7-2 RECORD DISPLAY Option

Function

The RECORD DISPLAY option is used to control the formatting of data which is output to an on-line CRT or LED display matrix on the Program Function Panel.

Description

The RECORD DISPLAY option is used with the RECORD DATA Statement to specify output to on-line CRT's and PFP LED's. Output is directed to the appropriate device by means of the Function Designator specified, which may be Display type or PFP LED type Function Designators. One or more Function Designators may be specified. The output is directed to all indicated devices.

A. Display Type Function Designators

Output may be directed to either Default or Specific Application Pages.

1. Default Pages:

The Default pages are PAGE-A and PAGE-B. When the Default Pages are referenced, data is output only to the A or B Page assigned to the GOAL task. GOAL Executor messages will be displayed to PAGE-A, while PAGE-B is reserved for the use of the GOAL task.

2. Specific Pages:

The Specific Pages are PAGE-1A through PAGE-6B and the Shared Application Page. Specific Page references allow any procedure to write to any page.

CAUTION

Data recorded to Specific Display Pages will overwrite pages assigned to other procedures.

The Shared Application Page provides a common display page for all tasks.

B. PFP LED's

Output may be directed to either Default or Specific LED's following the same concept used with Default and Specific Application Pages.

1. Default LED's:

The Default LED's are LED1 through LED6, which are used to reference the six LED half lines of the PFP Presentation assigned to a task.

2. Specific LED's:

The Specific LED's are referenced by Function Designators of the form < LEDx-Py > where x represents the LED half line and y represents the Specific Presentation.

CAUTION

Data recorded to Specific LED Presentations will overwrite LED's assigned to other tasks.

Output data may be displayed to a specific line, column, or line and column using the LINE AND COLUMN options. If this option is not selected, the next available line is used. The line and column numbers may be specified as integer constants or Internal Names which specify a single numeric value. A RECORD to the next available line causes the remainder of the line following the displayed message to be cleared as well as the next two lines. A RECORD using the LINE AND COLUMN options inserts the output into the location specified without disturbing the surrounding display. If the keyword NEXT is used in conjunction with the LINE AND COLUMN options, the LINE AND COLUMN options are executed before NEXT.

Output data may be displayed to a predefined cursor coordinate that is defined along with the display skeleton by using the SYMBOLIC CURSOR option. When this option is selected, the Skeleton Name and the symbolic name of the cursor position must be specified. The Skeleton Name follows the keyword SKELETON and the Cursor Name follows the keyword LOCATION.

Symbolic Cursor Names are defined external to the GOAL procedure when display skeletons are built (Reference System Generation Users Guide, KSC-LPS-OP-033-05).

Validity checks on Symbolic Cursor Names and Skeleton Names are not performed until the configuration process.

The BLINK option may be selected for display or PFP LED. When this option is used, the output data will be displayed blinking at a rate determined by the output device. The INTERNAL NAME option, which is a state variable with a subtype of ON/OFF, is used to variably define the blink for display page type Function Designators only.

The COLOR option may be selected for display page type Function Designators only. This option is used to specify the color which is used to display the output data. The INTERNAL NAME option, which is a numeric variable, is used to variably define the color as follows:

0 - BLACK
1 - RED
2 - GREEN
3 - YELLOW
4 - BLUE
5 - MAGENTA
6 - CYAN (Default value if the value of the Internal Name is out of range.)
7 - WHITE

The INVERT option may be selected for display page type Function Designators only. When this option is used, the output data will appear with the background and character image reversed for each character. The INTERNAL NAME option, which is a state variable with the subtype of ON/OFF, is used to variably define the invert.

The SCRATCH option may be selected for display page type Function Designators only. This option is used to temporarily output data to a display page that is currently being viewed. The display data will be discarded under the following conditions.

A. The referenced display page was not being viewed when the RECORD Statement was processed.

B. The referenced display page is overlaid by selection of another display page.

The RECORD AND SAVE REPLY AS option may be used with the RECORD DISPLAY option. In order to use this option, an explicit message must first be output to the Display Page on which the response is to be entered. When the response (limited to one line) is provided by the operator, it is converted and stored in the specified Internal Name(s). The SCRATCH option is ignored on the RECORD AND SAVE REPLY AS option.

RECORD Text Constant to Display Page

Examples

GOAL Source

```
RECORD TEXT (SAMPLE MESSAGE) TO < PAGE-B > ;
RECORD TEXT (SAMPLE MESSAGE 1) NEXT
       TEXT (SAMPLE MESSAGE 2) NEXT
       TEXT (SAMPLE MESSAGE 3)
       TO < PAGE-B > ;

RECORD TEXT(A), TEXT(B), TEXT(C)
       TO < PAGE-B > ;
```

Output

```
SAMPLE MESSAGE
SAMPLE MESSAGE 1
SAMPLE MESSAGE 2
SAMPLE MESSAGE 3
ABC
```

The examples illustrate recording text constants to PAGE-B. The GOAL source and the resulting display output on PAGE-B are shown. In the first example, SAMPLE MESSAGE is displayed to the next available line. In the second example, SAMPLE MESSAGE 1 is displayed to the next available line, and the keyword NEXT is used to cause SAMPLE MESSAGE 2 and 3 to be displayed on the next two available lines. The use of the comma to separate the text constants A, B, C in the third example causes the text ABC to be displayed on one line.

RECORD Text Constant to a Symbolic Cursor Position

Examples

```
RECORD TEXT (SAMPLE MESSAGE) TO < PAGE-B >
    LOCATION (LOC1) ON SKELETON (DSKEL1);

RECORD TEXT (SAMPLE MESSAGE) TO < PAGE-A >
    LOCATION (LOC2) ON SKELETON (DSKEL2);
```

The first example records a message to.the symbolic cursor position identified as (LOC1) on Display Skeleton (DSKEL1) that is currently being viewed on PAGE-B. The second example records a message to the symbolic cursor position identified as (LOC2) on Display Skeleton (DSKEL2) that is currently being viewed on PAGE-A.

RECORD Text Constant to Specific Line and Column

Examples

```
RECORD TEXT (SAMPLE MESSAGE)
    TO < PAGE-B > LINE 5;
RECORD TEXT (SAMPLE MESSAGE)
    TO < PAGE-B > LINE 5 COLUMN 7;
RECORD TEXT (SAMPLE MESSAGE)
    TO < PAGE-B > LINE (N5);
```

The first example records a message to line 5 of PAGE-B. The second example records a message to line 5, column 7. The third example records a message to the line defined by the numeric Internal Name (N5).

RECORD Text Constant With BLINK, INVERT, and COLOR Options

Examples

```
ASSIGN (STAT1) = ON;
LET (NUM1) = 4;
RECORD TEXT (SAMPLE MESSAGE)
    TO < PAGE-B > BLINK;
RECORD TEXT (SAMPLE MESSAGE)
    TO < PAGE-B > INVERT;
RECORD TEXT (SAMPLE MESSAGE)
    TO < PAGE-B > GREEN;
RECORD TEXT (SAMPLE MESSAGE)
    TO < PAGE-B > BLINK = (STAT1) COLOR = (NUM1);
```

```
RECORD TEXT (SAMPLE MESSAGE)
    TO < PAGE-B > BLINK YELLOW;
```

The first example causes a message to be displayed to PAGE-B blinking. The second example causes a message to be displayed to PAGE-B inverted. The third example causes a message to be displayed to PAGE-B in green. The fourth example causes a message to be displayed to PAGE-B in blue and blinking. The last example causes a message to be displayed to PAGE-B in yellow and blinking. All cause a display to the next available line.

RECORD Text Constant to a Specific Line and Column with NEXT

Example

```
RECORD TEXT (MESSAGE) TO <PAGE-A> NEXT TEXT
    (NEW MESSAGE) TO <PAGE-A> LINE 6 COLUMN 20;
```

In this example, if "MESSAGE" was displayed on line 10, the keyword NEXT would normally cause "NEW MESSAGE" to be displayed on line 11. However, the LINE and COLUMN options will be executed before the NEXT. This will cause the output to be positioned on line 6, column 20 and then be repositioned on the next line in column 0. "NEW MESSAGE" will be displayed on line 7 starting in column 0.

RECORD Text Constant with Scratch Write Option

Example

```
RECORD TEXT (SAMPLE MESSAGE)
    TO < PAGE-B > SCRATCH;
```

This example causes a message to be displayed to PAGE-B using the SCRATCH WRITE option.

RECORD Text Constant to PFP LED's

Examples

```
RECORD TEXT (SAMPLE MESSAGE)
    TO < LED1 >;
RECORD TEXT (SAMPLE MESSAGE)
    TO < LED2 > < LED3 > < LED4 >;
```

The first example causes a message to be displayed on the default LED1. The second example causes a message to be displayed on LED2, LED3, and LED4.

RECORD Text Constant to Display and Printer

Examples

```
RECORD TEXT (SAMPLE MESSAGE)
    TO < PAGE-B > TO < CNSL-PP >;
RECORD TEXT (SAMPLE MESSAGE)
    TO < PAGE-B > TO < SPA-PRNTR > PAGE  SKIP 3;
```

The first example records to PAGE-B and the Console Printer/ Plotter. The second example records to PAGE-B and the SPA Printer. Note the use of the DISPLAY option followed by the PRINT option.

RECORD Name to Display Page

The examples show GOAL source and display output. Output columns are indicated. The "S" in column zero represents the sign.

Examples

GOAL Source	Output
	0 1 2 3 4 5 6 7 8 9 0 1 2 3
RECORD (N5) TO < PAGE-B >COLUMN 4;	- 5
RECORD (N5), (N10) TO < PAGE-B > COLUMN 4;	- 5 3 1 0 0
RECORD (STATE1) TO < PAGE-B > ;	O F F
RECORD (STATE7), (STATE8) TO < PAGE-B > ;	O P N C L S
RECORD (Q10) TO < PAGE-B > ;	2 . 9 9 9 9 5 3 2 A M P
RECORD (TEXT2) TO < PAGE-B > ;	S A M P L E M E S S A G E

The examples illustrate the display of Numeric, State, Quantity, and Text Names using the default output formats. The line under each output total denotes default field required.

RECORD List or Table Column to Display Page

Examples

GOAL Source	Output
	0 1 2 3 4 5 6 7 8 9 0 1 2 3
RECORD (NLIST1) FORMAT (I3);	5 - 8 8 1 0 0
RECORD (NTABLE1) COLUMN 2 ;	1 0 1 0 0

Blanks are inserted between the elements of a list or Table column.

Blanks are inserted between the elements of a list or Table column.

Record And Save Reply As Option

This option is a prompt to the operator which requires that an entry must be made from the DG Keyboard. The associated program will stop executing until the keyboard TRANSMIT RESPONSE key is depressed signifying that an entry has been selected or entered from the keyboard. The entry is saved in the designated Internal Name, and program execution continues.

Examples

```
RECORD TEXT (ENTER DESIRED OPTION AND DEPRESS TRANSMIT
    RESPONSE) NEXT
    TEXT (OPTIONS ARE:  1=CONTINUE, 2=PERF ABC, 3=TERMINATE)
    TO < PAGE-B > AND SAVE REPLY AS (NOPT2);

RECORD TEXT (ENTER LIST INITIAL VALUE FROM OPTIONS) NEXT
     TEXT (26) NEXT
     TEXT (28) NEXT
     TEXT (32) TO < PAGE-B > AND SAVE REPLY AS (NLIST1);
```

Both examples record a message to PAGE-B, and they must be replied to from PAGE-B. The first example requires the operator to enter a reply from the keyboard prior to depressing the TRANSMIT RESPONSE key. The second example allows inserting the reply from the keyboard, or positioning the cursor to the right of the desired input to select the value to be entered in all entries of the list (NLIST1) prior to depressing the TRANSMIT RESPONSE key.

The first example records a message to PAGE-B which must be replied to from PAGE-B. The operator must enter a reply from the DG keyboard. The second example illustrates initialization of a Numeric List by entering an initial value from the keyboard. The numeric value entered by the operator will be read and saved in all entries of the list (NLIST1).

RECORD Present Value Option

Examples

```
RECORD < DM1 > TO < PAGE-B >;
RECORD < GMT > < CDT > TO < PAGE-B >;
RECORD (QTABLE2) FUNCTIONS TO < PAGE-B >;
```

These examples all cause the present value of the specified Function Designators to be displayed. The Function Designator name, 34-character descriptor, and present value will be displayed.

Statement Execution

For RECORDs to display pages, the GOAL Executor requests the Control and Display component of the Operating System to display the specified data. The GOAL Executor waits until the display is completed before continuing execution of the GOAL program.

For RECORDs to PFP LED's, the GOAL Executor requests the PFP Interface Processor to display the specified data. The GOAL Executor waits until the display is completed before continuing execution of the GOAL Program.

For recording of present value of Function Designators, the GOAL Executor will obtain the value of the Function Designator from the CDBFR, LDB, and other sources depending on the Function Designator, and request that the value be displayed. The GOAL Executor will continue execution of the GOAL program when the display is completed.

Present value of GMT, CDT, MET, JTOY and Interval Timer will be obtained from Console memory. Present value of PFP Lights will be obtained from the PFP's via the PFP Interface Processor.

Error Processing

When an error is encountered on a display request, a Class IV error will be processed. An error message will be displayed to PAGE-A, and the execution of the GOAL procedure will continue.

When more than one RECORD AND SAVE REPLY AS is recorded to a page, a Class II error will result.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Unless specified otherwise, the display output will use the following default options: next available line, first character position of a line, no blink, no invert, and cyan for the color.

B. If the COLOR = INTERNAL NAME option is used and the Internal Name is out of range (0-7) at execution time, the color will be defaulted to cyan and no error message will be generated by the GOAL Executor.

C. If the FORMAT option is not used, default output formats will be used (refer to Section 7.1).

D. The default for the LED's is no blink.

E. Because of the limited line size, the LED's should only be used to display single text data items. No more than 17 characters may be displayed. The LED's cannot display engineering graphic characters.

F. If the LOCATION or SKELETON option is used, the specified skeleton must have previously been output using the DISPLAY SKELETON Statement. Failure to meet this requirement will result in a Class IV error.

G. The RECORD AND SAVE REPLY AS option is not permitted from the Shared Application Page.

H. If the External Designator following the keyword TO is specified in the form of Table Functions, output will not be performed for any Row Designator which is inhibited.

I. Display pages and LED's may not be mixed in a single Record Display option.

Legal Function Designators

Application Pages (PAGE)
LED's (LED)

7.1.2 RECORD PRINT OPTION

RECORD PRINT OPTION

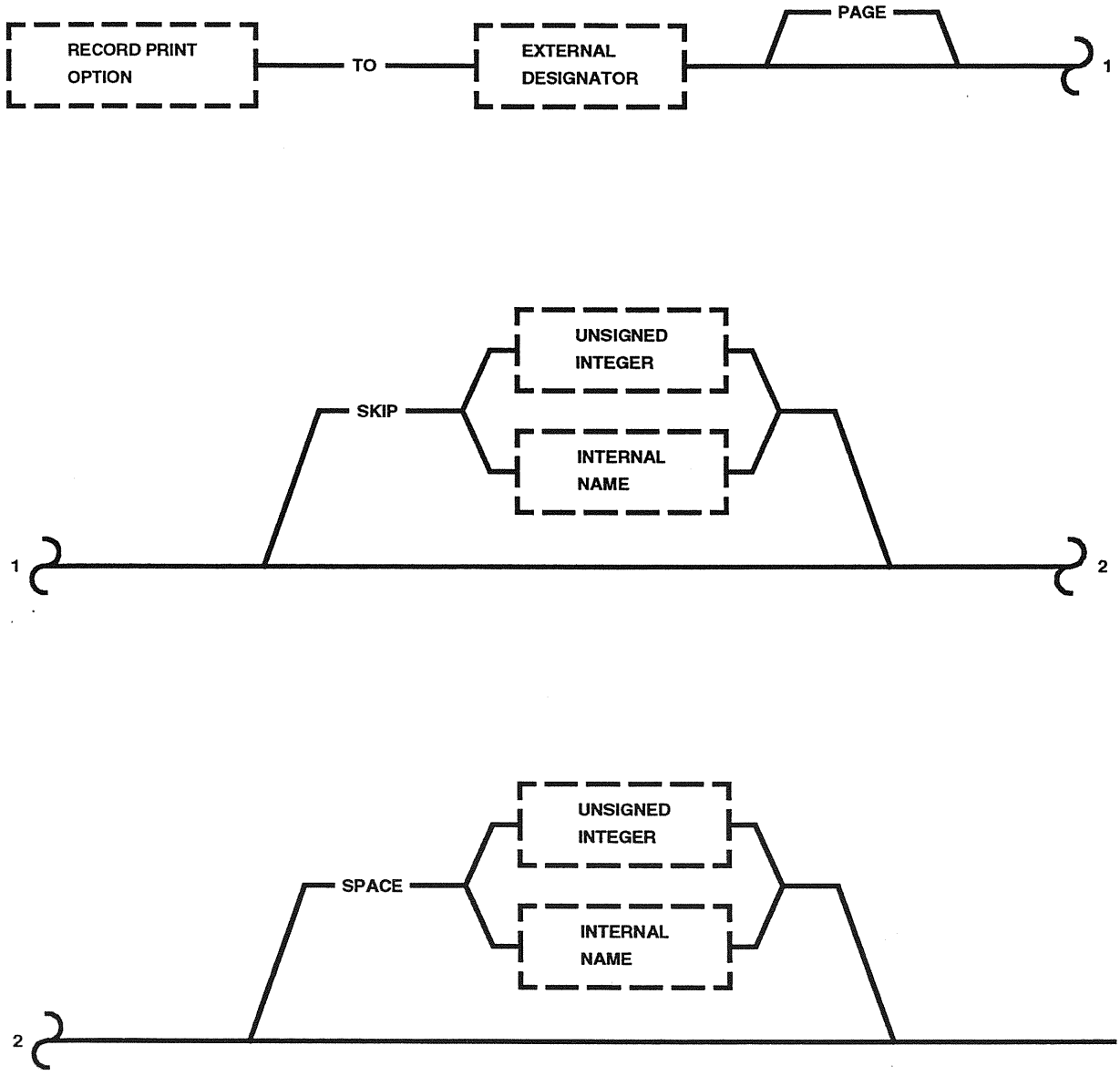


Figure 7-3 RECORD PRINT Option

Function

The RECORD PRINT option is used to control the formatting of data which is output to the SPA printer, console printer/plotter, PDR magnetic tapes or CDS magnetic tapes.

Description

The RECORD PRINT option is used with the RECORD DATA Statement to specify output to the SPA printer, PDR, CDS, and the console printer/plotter.

External Designator must be printer type. One or more Function Designators may be specified. The output is directed to all indicated devices.

OPTION	DATA RECORDED TO:			
	CONSOLE	SPA	PDR	CDS
CONSOLE	X			
SPA		X	X	X
PDR			X	X
CDS			X	X

The PAGE option will cause the printer to advance to the top of the next print page.

If this option is not specified, the output will be printed on the current page.

The SKIP option will cause a specified number of blank lines to appear prior to the printed output. The number of lines to be skipped may be specified as an integer constant or an Internal Name which specifies a single numeric value. If this option is not selected, the next available line is used.

The SPACE option will cause the output data to be indented a specified number of character positions. The number of positions may be specified by an integer constant or an Internal Name which

specifies a single numeric value. If this option is not specified, the output will begin at the first character position of the print line.

All data printed to the SPA printer will be printed as report data. The data is spooled to disk so that the entire report (consisting of all print requests from a GOAL task) may be printed without being mixed with system messages and data from other tasks.

Data which is printed on the console printer/plotter will be mixed with data printed by other GOAL tasks and other software in the console CPU.

Data which is recorded to the PDR is not printed but is retrievable with $SPBLOK (described in System Operation User Guide, KSC-LPS-OP-033-01).

Data which is recorded to CDS is not printed but is retrievable using CDS retrievals.

Examples

```
RECORD TEXT (SAMPLE MESSAGE) TO < CNSL-PP > SKIP 4;
RECORD TEXT (SAMPLE MESSAGE) TO < SPA-PRNTR > PAGE SPACE 25;
RECORD TEXT (SAMPLE MESSAGE) TO < CNSL-PP > < SPA-PRNTR >
    SKIP (N4) SPACE (N10);
```

The first example records a message to the console printer/plotter. Four lines are skipped prior to printing the message. The second example records a message to the SPA printer. The message will be printed at the top of the next page, indented 25 character positions. The last example illustrates the use of the Internal Name for specifying the number of lines to skip, (N4), and for specifying the number of spaces to indent, (N10). The output is directed to both the console printer/plotter and the SPA printer.

RECORD to CDS and PDR Recorders

Example

```
RECORD TEXT(SAMPLE MESSAGE) TO < CDS-RCDR > < PDR-RCDR >;
```

This example records the text message in ASCII to both CDS and PDR tape recorders.

Statement Execution

For print requests to the console printer/plotter, the GOAL Executor requests Printer/Plotter IOS to print the specified data. After the data is printed, the GOAL Executor will continue the execution of the GOAL procedure. If data is printed using multiple RECORD DATA Statements, or if the PRINT option is used more than once in a single RECORD DATA Statement, print requests from other GOAL procedures may be merged on the console printer/plotter. Data may also be merged under the conditions described in Section 7.1 pertaining to multiple I/O requests.

Print requests to the SPA Printer are sent to the SPA Subsystem via block transfer across the CDBFR. These print requests are spooled so that all of the print requests for a task may be printed as a report. When the first print request to the SPA printer for a task is processed, the GOAL Executor sends a request to the SPA Subsystem to initialize a spool for the task. The data will also be recorded to the CDS and PDR via the CDBFR. When the task terminates, a request is made to the SPA to terminate and print the spool.

Requests to record to the CDS and PDR recorders using the Print option will be handled by sending the data to be recorded to the CDS or PDR via the CDBFR. The GOAL procedure will not proceed until the data transfer is completed.

Print requests to the SPA printer, CDS, or PDR will also record the program's TCB number, the top level program LSN, and the current level program LSN in addition to the specified data.

Error Processing

Errors encountered on print requests will be processed as Class IV errors.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The number of lines skipped may not exceed 15.

B. Indentation may not exceed the width of a print page. If data is displayed as well as printed, the maximum indentation (and maximum size of a printed line) is 72 characters. For a print request only, indentation may not exceed 132 characters.

C. Neither the SPA printer nor the console printer is able to print engineering graphic characters.

D. Unless specified otherwise, the printed output will use the following defaults:

1. current page

2. next available line

3. no indentation

E. If the External Designator following the keyword TO is specified in the form of Table Functions, output will not be performed for any Row Designator which is inhibited.

Acceptable Conditions

When RECORD Statements are broken into multiple write requests, refer to Section 7.1, each request will contain the print options. For example, the PAGE option will cause the printer to advance to the top of the next print page for every write request.

Legal Function Designators

Console Printer/Plotter (CPP)
SPA Printer (PRTR)
PDR Recorder (PDRR)
CDS Recorder (PDRR)

7.1.3 RECORD WRITE OPTION

RECORD WRITE OPTION

Figure 7-4 RECORD WRITE Option

Function

The RECORD WRITE option is used to direct raw data to tape files for historical retrieval.

Description

The RECORD WRITE option is used with the RECORD DATA Statement to specify output as raw data files on the PDR tapes and CDS tapes. The output data is written in the form of binary records (referred to as raw data). No format control options are provided. Files written to PDR and CDS tapes may not be retrieved using GOAL statements.

Examples

```
RECORD (QLIST2) TO < PDR-RCDRR >;
RECORD (NLIST1) TO < CDS-RCDRR >;
```

The first example writes the Quantity List (QLIST2) to the PDR tape recorder. The second example writes the Numeric List (NLIST1) to the CDS tape recorder and also to the PDR tape recorder.

Statement Execution

Requests to record raw data (binary) to the CDS and PDR tape recorders are sent to the CDS and PDR via the CDBFR.

Error Processing

Errors encountered on write requests will be processed as Class IV errors.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. No format control options are provided. The output data is written in the form of binary records.

B. The WRITE option may not be used in combination with the DISPLAY and PRINT options for the same collection of data.

C. When using the WRITE option, no data conversion will be performed; only raw data will be written.

D. The keyword NEXT cannot be used when output option WRITE is used.

E. All data recorded to CDS will also be routed to PDR.

F. If the External Designator following the keyword TO is specified in the form of table functions, output will not be performed for any Row Designators which are inhibited.

Legal Function Designators

PDR Recorder - Raw Data (FILE)
CDS Recorder - Raw Data (FILE)

7.1.4 RECORD FORMAT OPTION

RECORD FORMAT OPTION

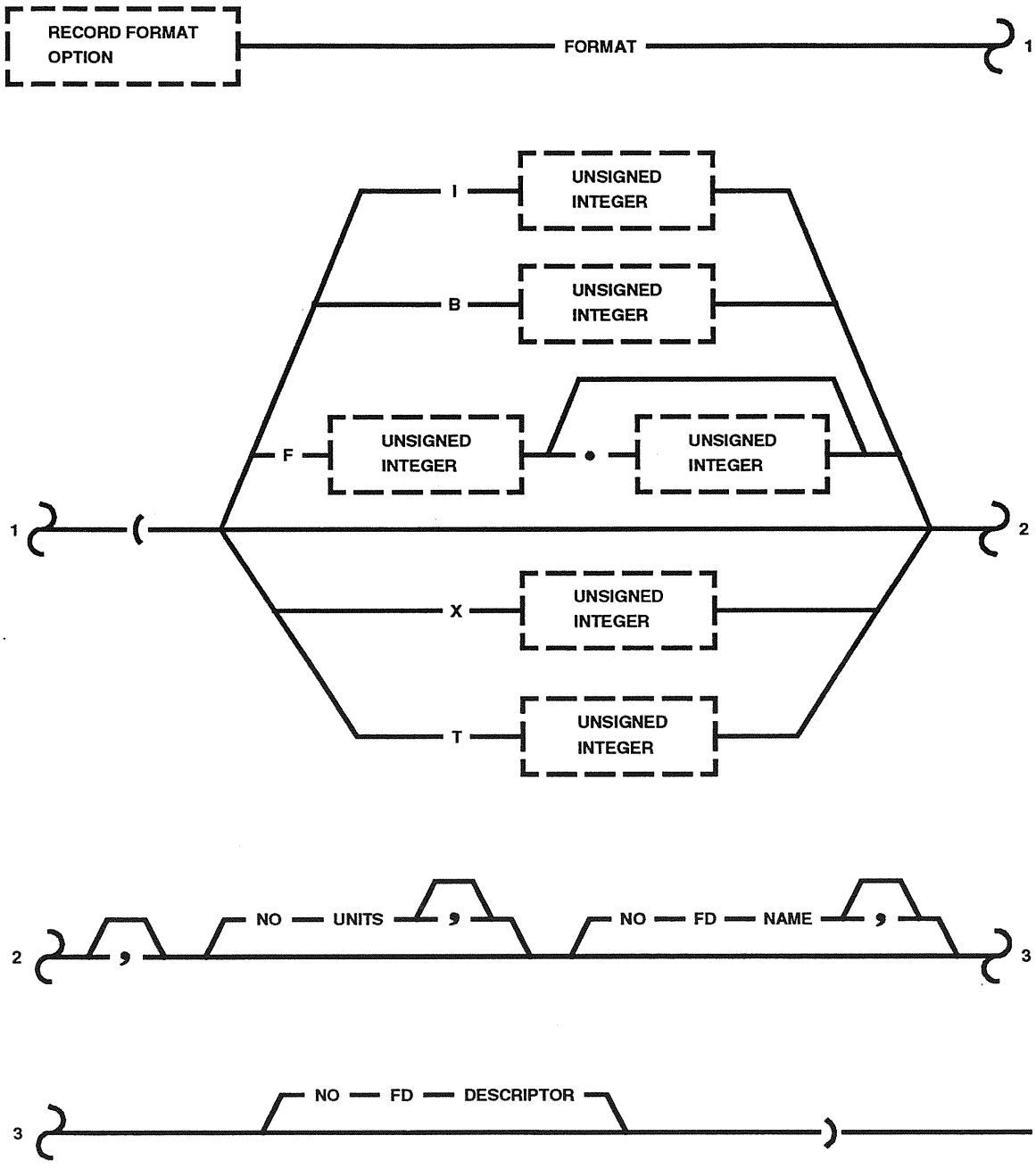


Figure 7-5 RECORD FORMAT Option

Function

The RECORD FORMAT option is used to specify conversion and editing information to be specified for data which is to be output to an on-line CRT terminal or a printer.

Description

The RECORD FORMAT option is used with the RECORD DATA Statement to format output data. Format specifications may be used for Internal Name or Function Designator outputs. The data formats are:

I - Integer format
B - Binary format
X - Hexadecimal format
T - Octal format
F - Decimal format

I, B, X, and T are used to output numeric type data. F is used to output Quantity type data.

I, B, X, and T data formats must be followed by an Unsigned Integer which indicates the field width of the data to be output.

The F format is written in GOAL as Fx.y, where x represents the number of integer characters in the output and y represents the number of fractional characters to the right of the decimal point. If y is zero, the decimal point will not be a part of the output. If y is not specified, the format will be treated as if y had been specified as zero. If y is specified and there are no non-zero fractional digits in the actual output data, the output will contain zeroes in the fractional locations. The value of the y + 1 fractional digit (the most significant fractional digit not displayed) is used in rounding the formatted number. For example, when using an F1.1 format, "1.55" and "1.56" are displayed as "b1.6" and "1.54" is displayed as "b1.5". This also means that if y is not specified or is zero (F1.0 format), "1.5" is displayed as "b2".

NOTE: *The value of floating point representation for fractional numbers has a small degree of inaccuracy. For example "0.05" is represented as "0.0499999523". Therefore, when using an F1.1 format, "0.05" is displayed as "0.0".*

If the number cannot be output in the specified format, the entire output field will contain asterisks (*). If the number is

positive, the "+" sign will not be in the output field, but a blank will be reserved for it. Leading zeroes will be suppressed, and the number will be right justified in the output field.

The parameter NO UNITS may be selected to suppress output of engineering unit text information when outputting Internal Name or Function Designator Quantity type data.

The parameter NO UNITS may also be selected to suppress the output of 'ØB', 'ØX', or 'ØT' when outputting Internal Name numeric type data.

The parameter NO FD NAME may be selected to suppress a Function Designator name when displaying Function Designator data.

The parameter NO FD DESCRIPTOR may be selected to suppress the Function Designator 34-character descriptor when outputting Function Designator data.

If no format option is specified, data will be output according to the default formats given in Section 7.1.

In the following examples, the GOAL source statements and the display output will be shown. Column numbers are shown with the output. The "S" in column zero refers to the sign.

GOAL SOURCE STATEMENT EXAMPLES

GOAL Source Statements	Output
	0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 (1 1 1 1 1 1)
Record Numeric Name - I,B,T,X Formats	
RECORD (N10) FORMAT (I3) TO < PAGE-B > ;	1 0
RECORD (N10) FORMAT (B5) TO < PAGE-B > ;	B 0 1 0 1 0
RECORD (N10) FORMAT (T3) TO < PAGE-B > ;	T 0 1 2
RECORD (N10) FORMAT (X3) TO < PAGE-B > ;	X 0 0 A
RECORD (N10) FORMAT (X3, NO UNITS) TO < PAGE-B > ;	0 0 A
Record Quantity Name - F Format	
RECORD (Q) FORMAT (F2.2) TO < PAGE-B > ;	-5.35 AMP
RECORD (Q) FORMAT (F2.2, NO UNITS) TO < PAGE-B > ;	-5.35

Restrictions/limitations

Refer to the following for restrictions and limitations:

A. Format specifications may only be used with the RECORD DATA Statement.

B. NO FD NAME and NO FD DESCRIPTOR may only be specified when referencing a Function Designator.

C. Format specification information always applies to the data item which immediately precedes it. No other optional record data options may be imbedded between a format specification and its associated data.

D. Format specifications may not be specified for Text, State or Time Internal Name data, but may be used to suppress Function Designator name and descriptor when referencing State, Time or Text type Function Designators.

E. The maximum number of significant digits which should be specified are:

8 for F format

5 for I format

16 for B format

4 for X format

6 for T format

If more digits are specified, blanks will be filled in front of the number.

F. If a Format specification is used following TABLE FUNCTIONS, the format is applicable to all the row designators of the table. A default carriage return will not be supplied after each row.

7.2 MODIFY DISPLAY STATEMENT

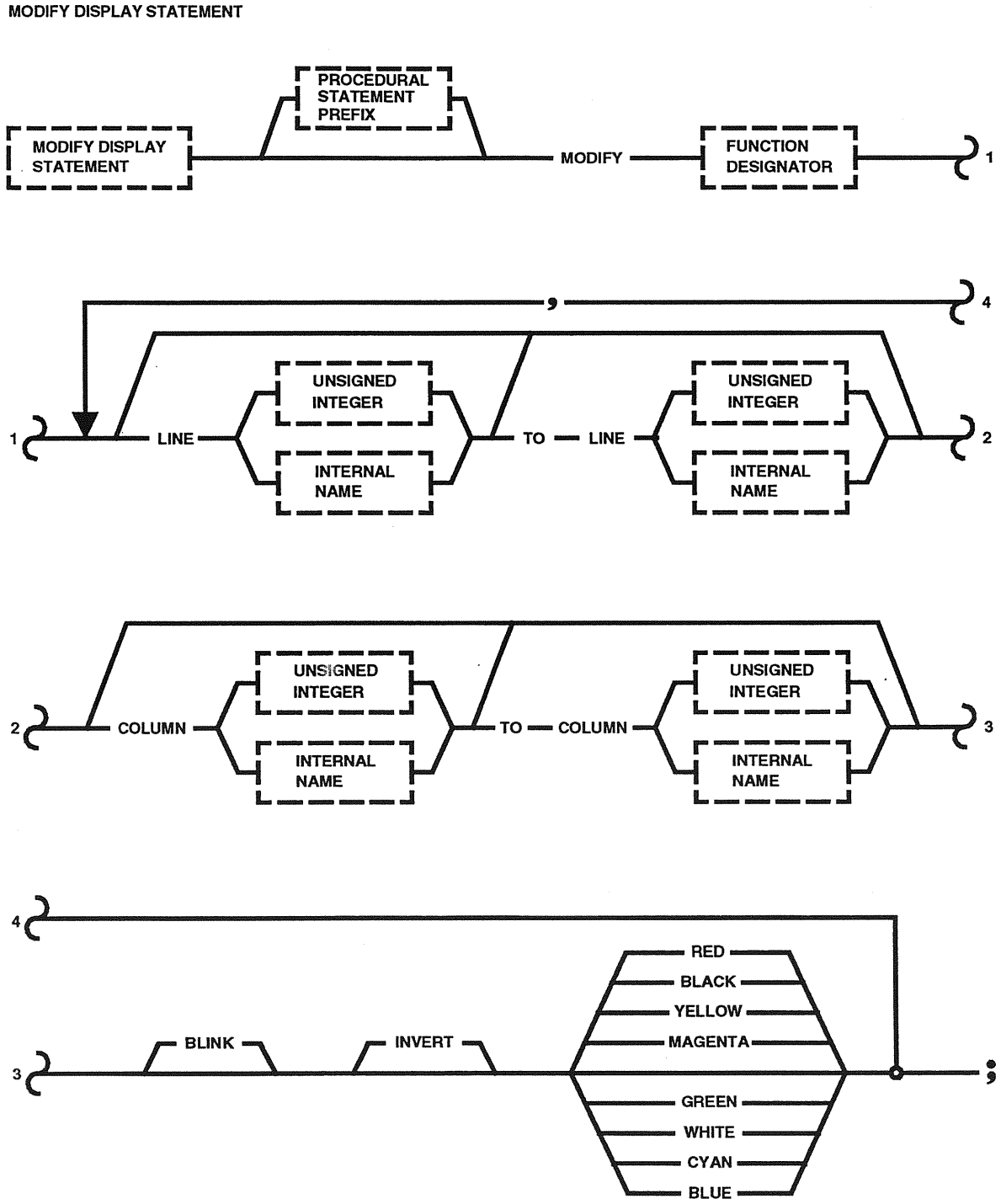


Figure 7-6 MODIFY DISPLAY Statement

Function

The MODIFY DISPLAY Statement is used to modify the BLINK, INVERT, and COLOR characteristics of data currently contained on a display page, or to cause LED's to blink.

Description

The MODIFY DISPLAY Statement may be used to modify specific lines and columns, parts of lines and columns, and sections of a display page with BLINK, INVERT, and COLOR characteristics.

A. LINE and TO LINE Options

The LINE and TO LINE options are used to select specific consecutive lines to be modified. If only the LINE is specified, a single line will be modified. If neither LINE nor TO LINE is specified, all lines on the Display Page are selected for modifying. A line number may be specified as an integer, constant or an Internal Name which specifies a single numeric value.

B. COLUMN and TO COLUMN Options

The COLUMN and TO COLUMN options are used to identify the portions of selected lines to be modified. If only COLUMN is specified, a single column will be modified. If neither COLUMN nor TO COLUMN is specified, all columns of selected lines will be modified. A column number may be specified as an integer constant or an Internal Name which specifies a single numeric value.

C. BLINK Option

The BLINK option may be selected to blink the current data at a rate determined by the output device. If the BLINK option is not specified, the current data to be modified will be set to a no-blink condition.

D. INVERT Option

The INVERT option may be selected to invert normal data. The inverted character will appear dark against a background field which is the same color as the normal data. If the data was already inverted, it is not affected. If the INVERT option is not specified, the current data to be modified will be set to a no-invert condition.

E. COLOR Option

The COLOR option may be selected to change the color of the current data. If the COLOR option is not specified, the current data to be modified will be set to cyan.

F. FEEDBACK Option

The options of this statement may be repeated, as indicated by the feedback arrow, to modify non-consecutive lines and/or columns.

MODIFY LINE and TO LINE Options

Examples

```
MODIFY < PAGE-B > LINE 5 RED;
MODIFY < PAGE-B > LINE (N5) RED;
MODIFY < PAGE-B > LINE 5 TO LINE 7 GREEN;
```

In the first example, Line 5 of PAGE-B will be modified so that the resulting display is red. In the second example, the Line specified by the Numeric Name, (N5), will be modified to red. In the third example, Line 5 to Line 7 will be modified to green.

MODIFY LINE TO MULTIPLE COLUMNS Option

Examples

```
MODIFY < PAGE-B > LINE 10 COLUMN 11 TO COLUMN 12,
    COLUMN 18 TO COLUMN 21;
MODIFY < PAGE-B > LINE 10 COLUMN 11 TO COLUMN 12,
    LINE 10 COLUMN 18 TO COLUMN 21;
```

In the first example, line 10 columns 11 to 12 and all lines of column 18 to 21 will be modified on PAGE-B. In the second example, line 10 columns 11 to 12 and line 10 columns 18 to 21 will be modified on PAGE-B.

MODIFY COLUMN and TO COLUMN Options

Examples

```
MODIFY < PAGE-A > COLUMN 8 WHITE;
MODIFY < PAGE-A > COLUMN (N8) WHITE;
MODIFY < PAGE-A > COLUMN 8 TO COLUMN 12 WHITE;
```

The first example modifies Column 8 on PAGE-A to be displayed in white. The second example modifies the Column specified by the Numeric Name, (N8), to be displayed in white. The third example changes the color of columns 8 to 12 to white.

MODIFY Lines and Columns Combined

Examples

```
MODIFY < PAGE-B > LINE 5 COLUMN 3 TO COLUMN 9 RED;
MODIFY < PAGE-B > LINE 5 TO LINE 8 COLUMN 3 TO COLUMN 9 RED;
```

The first example modifies PAGE-B Line 5 Columns 3 to 9 to be displayed in red. The second example modifies Lines 5 to 8 Columns 3 to 9 to be displayed in red.

MODIFY Using Feedback Option

Examples

```
MODIFY < PAGE-B > LINE 5 GREEN, LINE 7 YELLOW;
MODIFY < PAGE-B > LINE 30 COLUMN 5 TO COLUMN 10 BLUE,
    LINE 31 COLUMN 11 TO COLUMN 20 BLUE;
```

The first example modifies PAGE-B Line 5 to green and Line 7 to yellow. The second example modifies the following to be displayed in blue: (1) Line 30 Columns 5 to 10 and (2) Line 31 Columns 11 to 20.

MODIFY With BLINK and INVERT

Examples

```
MODIFY < PAGE-B > LINE 12 BLINK;
MODIFY < PAGE-B > LINE 12 BLINK RED;
MODIFY < PAGE-B > LINE 12 INVERT;
MODIFY < PAGE-B > LINE 12 BLINK INVERT MAGENTA;
MODIFY < LED1 > BLINK;
```

The first example modifies PAGE-B Line 12 to blink. The second example modifies PAGE-B Line 12 to be displayed in red and blinking. The third example inverts the characters on Line 12. The fourth example causes the characters on Line 12 to be displayed in magenta blinking and inverted. The last example modifies LED 1 to blink.

MODIFY Entire Page

Example

```
MODIFY < PAGE-B > YELLOW;
```

This example causes all of PAGE-B to be displayed in yellow.

Statement Execution

The GOAL Executor will request the Control and Display component of the Operating System to perform the display modifications specified on the MODIFY Statement. Requests to modify LED's will be sent to the PFP Interface processor. The GOAL Executor waits for successful completion of I/O before proceeding to the next statement in the procedure.

Unless otherwise specified, the MODIFY DISPLAY Statement will use the following default options:

OPTION	DEFAULT
Begin Line	4
Begin Column	0
End Line	33
End Column	71
Blink Option	No-Blink
Invert Option	No-Invert
Color Option	Cyan

Only BLINK may be specified for PFP-LED.

Error Processing

If Internal Names are used for line or column numbers, a run time error (Class IV) will result if their values are outside the limits of the display page.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. An LED may only be modified with BLINK or the no blink default.

B. If an integer constant is used, it must lie within the limits of the display page.

C. Inverting an inverted character has no effect.

D. The original attributes of a skeleton will not be refreshed unless it is reassigned to the modified page.

E. If consecutive lines or columns are specified, they must be indicated in order of increasing value.

F. When modifying data that was recorded using the SCRATCH option, the associated bulk memory location is modified as well as the CRT location. The page indicator will reflect the update.

Legal Function Designators

Application Pages (PAGE)
LED's (LED)

THIS PAGE INTENTIONALLY LEFT BLANK.

7.3 CLEAR DISPLAY STATEMENT

CLEAR DISPLAY STATEMENT

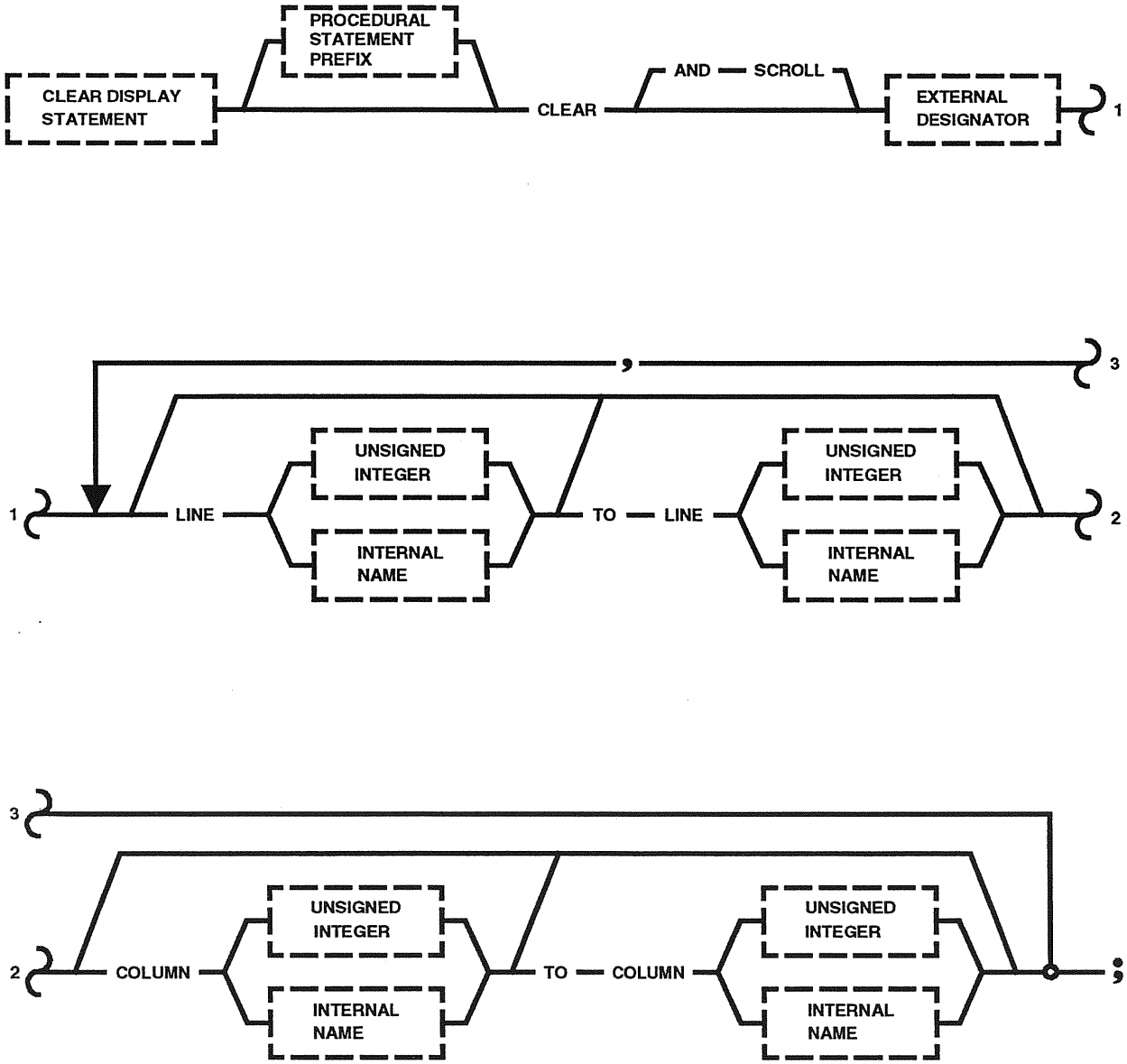


Figure 7-7 CLEAR DISPLAY Statement

Function

The CLEAR DISPLAY Statement is used to replace with blanks, data which is displayed to on-line CRT terminals or LED's.

Description

The CLEAR DISPLAY Statement is used to clear display pages, lines, columns, or portions of pages as well as LED's.

A. CLEAR LED Option

If a PFP-LED type Function Designator is specified, the LED half line is cleared. No Line or Column option may be specified.

B. LINE AND TO LINE Option

The LINE and TO LINE options are used to select specific consecutive lines to be cleared. If only the LINE is specified, a single line is cleared. If neither LINE nor TO LINE is specified, all lines on the Display Page are selected for clearing. A line number may be specified as an integer constant or an Internal Name which specifies a single Numeric value.

C. COLUMN AND TO COLUMN Option

The COLUMN and TO COLUMN options are used to identify the portions of selected lines to be cleared. If only COLUMN is specified, a single column is cleared. If neither COLUMN nor TO COLUMN is specified, all columns of selected lines are cleared. A column number may be specified as an integer constant or an Internal Name which specifies a single numeric value.

D. AND SCROLL Option

The AND SCROLL option is used to clear display data from specified lines or columns and move display data into the cleared area. Blank lines or columns appear at the bottom or far right of the display page.

The clearing procedure uses both line and column selection as described. First, the lines are selected. Next, the column positions are selected. Then the clearing procedure is performed. This procedure may be repeated, as indicated by the feedback arrow, to clear non-consecutive lines and/or columns.

A clear display page will clear both skeleton and live data from the page.

CLEAR Entire Page Option

Examples

```
CLEAR < PAGE-B >;
CLEAR < PAGE-1A >,
      < PAGE-1B >,
      < PAGE-2A >;
```

The first example clears PAGE-B. The second example clears Pages 1A, 1B, and 2A.

CLEAR LINE and TO LINE Option

Examples

```
CLEAR < PAGE-B > LINE 5;
CLEAR < PAGE-B > LINE (N5);
CLEAR < PAGE-B > LINE 8 TO LINE 16;
CLEAR AND SCROLL < PAGE-A > LINE 5;
CLEAR AND SCROLL < PAGE-A > LINE 5 TO LINE 10;
```

The first example clears PAGE-B Line 5. The second example clears the Line defined by the Name (N5). The third example clears Line 8 to 16.

The fourth example clears Line 5 and moves display data on Lines 6 through 33 upward on the display page, positioning it to be on Line 5. The fifth example clears Lines 5 through 10 and moves the display data on Lines 11 through 33 upward as described above.

CLEAR COLUMN and TO COLUMN Options

Examples

```
CLEAR < PAGE-A > COLUMN 8;
CLEAR < PAGE-A > COLUMN 8 TO COLUMN (N10);
CLEAR AND SCROLL < PAGE-A > COLUMN 10;
CLEAR AND SCROLL < PAGE-A > COLUMN 10 TO COLUMN 20;
```

The first example clears Column 8 of PAGE-A. The second example clears Column 8 to the Column defined by the Name (N10). The third example clears Column 10 and moves the display data in Columns 11 through 71 to the left on the display page, positioning it to begin in Column 10. The fourth example clears Columns 10

through 20 and moves the display data in Lines 21 through 71 to the left as described above.

CLEAR LINES and COLUMN Combined

Examples

```
CLEAR < PAGE-B > LINE 5 COLUMN 20 TO COLUMN 25;
CLEAR < PAGE-B > LINE 30 TO LINE 33,
    COLUMN 5 TO COLUMN 10;
```

The first example clears Columns 20 to 25 on Line 5 of PAGE-B. The second example clears the section of the page defined by Lines 30 to 33 and Columns 5 to 10.

CLEAR Using Feedback Option

Examples

```
CLEAR < PAGE-B > LINE 8, LINE 10, LINE 12;
CLEAR < PAGE-B > LINE 5 COLUMN 2 TO COLUMN 8,
    LINE 7 COLUMN 2 TO COLUMN 8;
CLEAR AND SCROLL < PAGE-B > LINE 5, COLUMN 10;
```

The first example clears Lines 8, 10, and 12 of PAGE-B. The second example clears the following: Line 5 Columns 2 to 8 and Line 7 Columns 2 to 8. The third example clears Line 5 and moves remaining display data upward, then clears Column 10 and moves remaining display data to the left, as described in individual examples.

CLEAR LED's

Examples

```
CLEAR < LED1 >;
CLEAR < LED1 >
      < LED2 >
      < LED3 >;
```

The first example clears LED 1. The second example clears LED's 1, 2, and 3.

Statement Execution

The GOAL Executor will request the Control and Display component of the Operating System to perform the display clears specified on the CLEAR DISPLAY Statement. Requests to clear the LED's will be sent to the PFP Interface Processor. The GOAL Executor waits for successful completion of I/O before proceeding to the next statement in the procedure.

Error Processing

If Internal Names are used for line or column numbers, a run time error (Class IV) will result if their values are outside the limits of the display page.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. If the LINE and/or COLUMN options are used, only one Function Designator may be referenced.

B. If a PFP-LED type Function Designator is specified, the LED half line is cleared. No LINE or COLUMN option may be specified.

C. If the AND SCROLL option is specified, only entire lines or columns may be cleared.

D. A clear display page will clear both skeleton and live data.

E. The system will not clear pages or LED's automatically.

F. If consecutive lines or columns are specified, they must be indicated in order of increasing value.

G. LED's and pages may not be referenced in the same statement.

Legal Function Designators

Application Pages (PAGE)
LED's (LED)

7.4 DISPLAY SKELETON STATEMENT

DISPLAY SKELETON STATEMENT

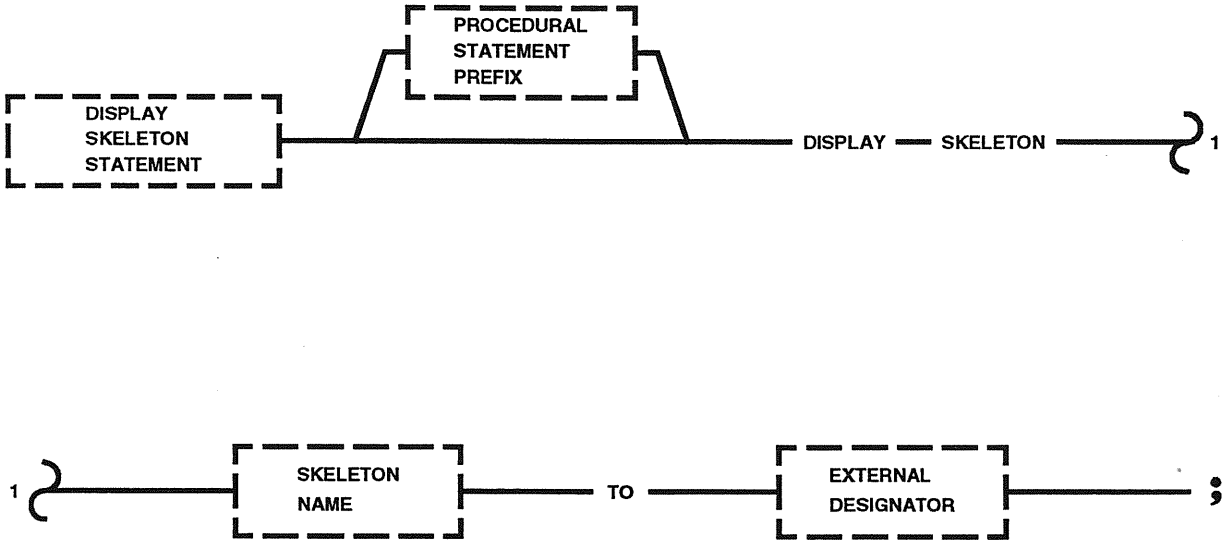


Figure 7-8 DISPLAY SKELETON Statement

Function

The DISPLAY SKELETON Statement is used to display predefined skeletons to CRT display pages.

Description

The DISPLAY SKELETON Statement is used to display a skeleton by specifying the Skeleton Name. Skeletons used in this statement must be defined in the Skeleton Build Library. The GOAL Language Processor will syntactically validate the SKELETON NAME, but it will not verify whether the display skeleton exists at compile time.

Examples

```
DISPLAY SKELETON (SKEL1) TO < PAGE-B >;
DISPLAY SKELETON (SKEL2) TO < PAGE-A >
                            < PAGE-B >;
```

The first example displays the skeleton (SKEL1) to PAGE-B. The second example displays the skeleton (SKEL2) to Pages A and B.

Statement Execution

The GOAL Executor retrieves the skeleton from console disk using Moving Head Disk IOS and displays it to the pages specified. The GOAL procedure does not advance to the next statement until the skeleton is retrieved from disk and displayed.

Error Processing

Errors encountered in retrieving and displaying a skeleton will be processed as Class IV errors.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Skeletons referenced must be defined in the Skeleton Build Library.

B. The GLP will not verify whether the display skeleton exists at compile time.

Legal Function Designators

Application Pages (PAGE)

THIS PAGE INTENTIONALLY LEFT BLANK.

7.5 OUTPUT EXCEPTION

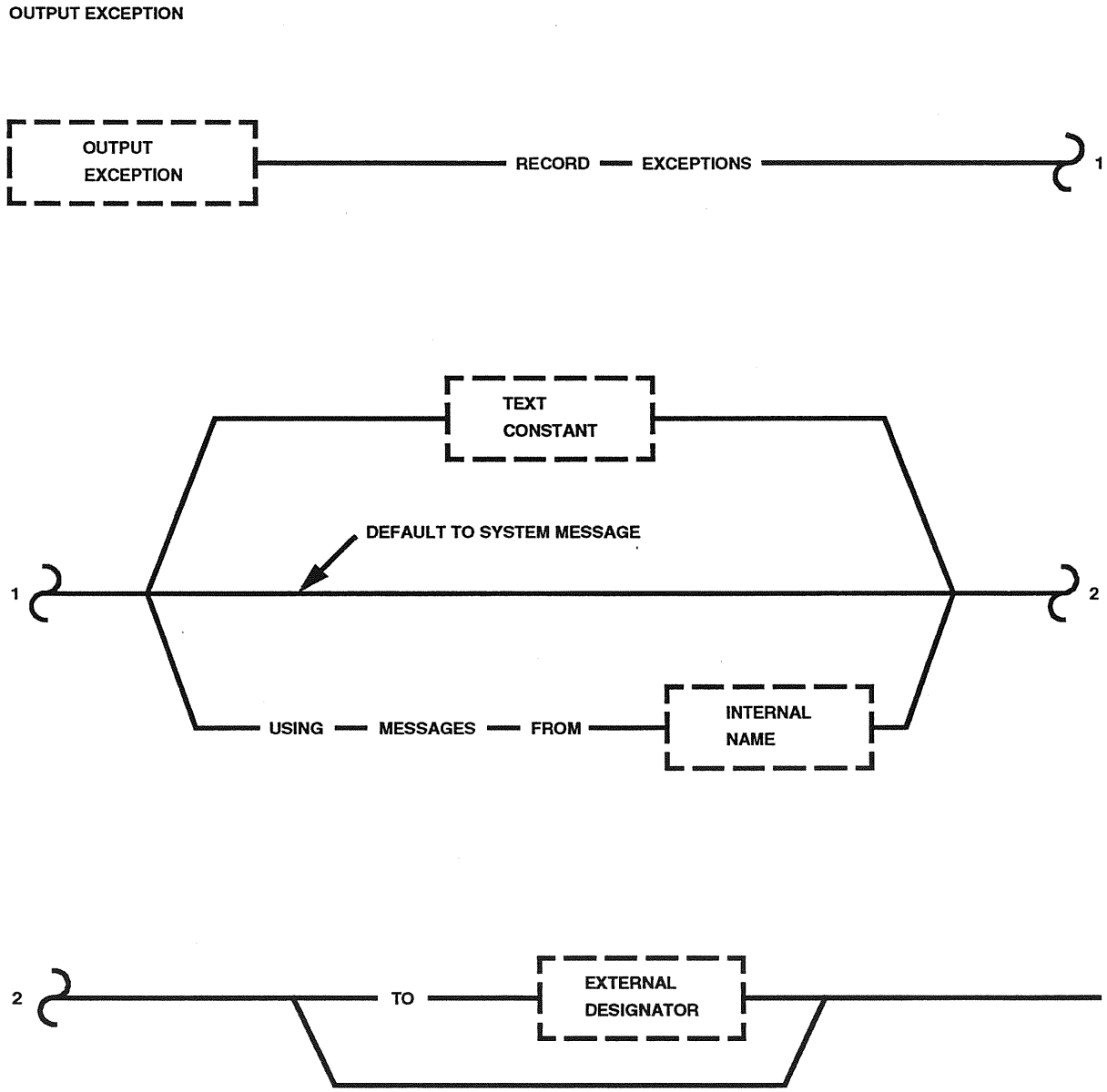


Figure 7-9 Output Exception

Function

The Output Exception element is used to output messages which describe discrepancies detected during a Comparison Test performed in the Verify Prefix.

Description

The Output Exception element is used with the Verify Prefix to output information about Comparison Test failures to such devices as display pages and printers. The exception messages may be optionally specified in the following ways:

A. Single user message:

A single message, defined by a Text Constant, is output when one or more failures occur in a Comparison Test.

B. System messages:

The GOAL Executor outputs messages for Comparison Test failures. The present value of each Function Designator which fails the Comparison Test (including the Function Designator name, 34-character descriptor, and present value) is output. Exceptions which are output to display pages are recorded in yellow.

C. List of user messages:

Messages defined in a Text List or Column of a Text Table are selected for output when a Comparison Test involving its corresponding Function Designator is not successful. The number of messages must equal the number of Function Designators being tested.

RECORD EXCEPTIONS - Single User Message Option

Examples

```
VERIFY < DM1 > IS ON ELSE RECORD EXCEPTIONS TEXT
    (POWER ON FAILURE) TO < PAGE-B > AND GO TO STEP 5;
VERIFY (STABLE1) FUNCTIONS ARE OFF ELSE RECORD EXCEPTIONS
    TEXT (POWER NOT OFF) AND STOP;
```

The first example records the message, (POWER ON FAILURE), to PAGE-B when the discrete measurement is not ON. The second example records the message, (POWER NOT OFF), when the discrete Function Designators in the Table (STABLE1) are not OFF.

RECORD EXCEPTIONS - System Message Option

Examples

```
VERIFY (QTABLE1) FUNCTIONS ARE LESS THAN 5 V ELSE RECORD
    EXCEPTIONS TO < PAGE-A > < CNSL-PP > AND GO TO STEP 5;
VERIFY (STABLE1) FUNCTIONS ARE ON;
```

In the first example, the name and value of each analog Function Designator which is not less than 5 volts will be recorded to PAGE-A and to the console printer/plotter. The second example uses the abbreviated form of the STOP Statement with the Verify Prefix. The Present Value of the Function Designators in (STABLE1) which are not ON will be displayed to PAGE-A (the default application page) and the procedure will stop if all Function Designators are not ON.

Examples

```
VERIFY < AI-01 > < AI-02 > ARE LESS THAN
    10V AND < AI-03 > < AI-04 > ARE LESS THAN
    15V ELSE RECORD EXCEPTIONS AND STOP;

VERIFY < AI-01 > < AI-02 > ARE LESS THAN
    10V OR < AI-03 > < AI-04 > ARE LESS THAN
    15V ELSE RECORD EXCEPTIONS AND STOP;
```

In the first example, exceptions will be recorded for each Function Designator in which an exception occurs. The name and value of each analog in that External Designator which is greater than or equal to the value to which it is compared will be recorded to PAGE-A. The procedure will stop after recording the exceptions.

In the second example, no exception messages will be recorded unless < AI-03 > or < AI-04 > (or both) is greater than or equal to 15 Volts. In these cases, the exceptions will be recorded as described for the first example and the procedure will stop.

NOTE: *For "or" type VERIFY ELSE RECORD EXCEPTIONS like the second example, exception messages will be recorded only for the final External Designator compared.*

RECORD EXCEPTIONS Using Messages from List Option

Examples

```
DECLARE STATE TABLE (STABLE5) WITH 3 ROWS AND 0 COLUMNS
    WITH ENTRIES
    < DM1 >,
    < DM2 >,
    < DM3 >;
DECLARE TEXT LIST (MESSAGE LIST) WITH 3 ENTRIES
    TEXT (POWER SUPPLY 1 NOT ON),
    TEXT (POWER SUPPLY 2 NOT ON),
    TEXT (POWER SUPPLY 3 NOT ON);
.....................................................
VERIFY (STABLE5) FUNCTIONS ARE ON ELSE RECORD EXCEPTIONS
    USING MESSAGES FROM (MESSAGE LIST) AND GO TO STEP 5;
```

In the example, Function Designators which are not ON will cause corresponding messages from (MESSAGE LIST) to be displayed to PAGE-A. For instance, if DM1 and DM3 are not ON, the following output will result:

```
POWER SUPPLY 1 NOT ON
POWER SUPPLY 3 NOT ON
```

Examples

```
DECLARE STATE TABLE (STABLE5) WITH 3 ROWS AND 0 COLUMNS
    WITH ENTRIES
    < DM1 >,
    < DM2 >,
    < DM3 >;
DECLARE TEXT TABLE (TTABLE3) WITH 3 ROWS AND 0 COLUMNS
    WITH ENTRIES
    < PAGE-A >,
    < PAGE-B >,
    < CNSL-PP >;
DECLARE TEXT LIST (MESSAGE LIST) WITH 3 ENTRIES
    TEXT (POWER SUPPLY 1 NOT ON),
    TEXT (POWER SUPPLY 2 NOT ON),
    TEXT (POWER SUPPLY 3 NOT ON);
.....................................................
VERIFY (STABLE5) FUNCTIONS ARE ON ELSE RECORD EXCEPTIONS USING
    MESSAGES FROM (MESSAGE LIST) TO (TTABLE3) FUNCTIONS AND
    STOP;
```

This example is the same as the previous example except that the exceptions are recorded to (TTABLE3) FUNCTIONS; i.e., to PAGE-A, PAGE-B, and the console printer/plotter.

Statement Execution

Requests to display exceptions are sent to the Control and Display component of the operating system. Requests to print exceptions are sent to Printer/Plotter IOS or to the SPA Subsystem depending upon whether the console printer/plotter or the SPA printer is specified. Requests to record exceptions to the CDS and PDR tape recorders are sent to the CDS and PDR Subsystem respectively.

Error Processing

Errors encountered in recording exceptions will be processed as Class IV errors.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The Letter "S" on the keyword EXCEPTIONS is optional.

B. If no output device is specified, the system will default to PAGE-A.

C. Output of messages from a list is not allowed when the Verify Prefix contains more than one condition test connected by the keywords AND or OR.

D. This statement may be used with the abbreviated ELSE option of the VERIFY statement (ELSE is the first keyword of the prefix used in VERIFY/IF-THEN-ELSE construct) only if the ELSE is associated with the VERIFY form of the VERIFY PREFIX.

Legal Function Designators

Application Pages (PAGE)
Console Printer/Plotter (CPP)
SPA Printer (PRTR)
PDR Recorder (PDRR)
CDS Recorder (PDRR)

8. COMMAND AND MONITOR STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The Command and Monitor statements which will be discussed in the indicated sections are as follows:

APPLY ANALOG STATEMENT (Section 8.1)

SET DISCRETE STATEMENT (Section 8.2)

TEST AND SET STATEMENT (Section 8.3)

ISSUE DIGITAL PATTERN STATEMENT (Section 8.4)

READ STATEMENT (Section 8.5)

READ FEP STATUS STATEMENT (Section 8.6)

The Command and Monitor statements provide the capability to send commands to and monitor responses from the system under test. Analog stimuli may be sent with the APPLY ANALOG Statement. Discrete stimuli may be set with the SET DISCRETE Statement. The TEST AND SET Statement is used to set the value of a pseudo Function Designator based on the current value of the pseudo Function Designator. Digital Pattern stimuli may be issued with the ISSUE DIGITAL PATTERN Statement. The READ Statement is used to monitor measurements in the system under test. The READ FEP STATUS Statement provides the means to obtain the value of various exception monitoring and console notification indicators for measurement Function Designators.

THIS PAGE INTENTIONALLY LEFT BLANK.

8.1 APPLY ANALOG STATEMENT

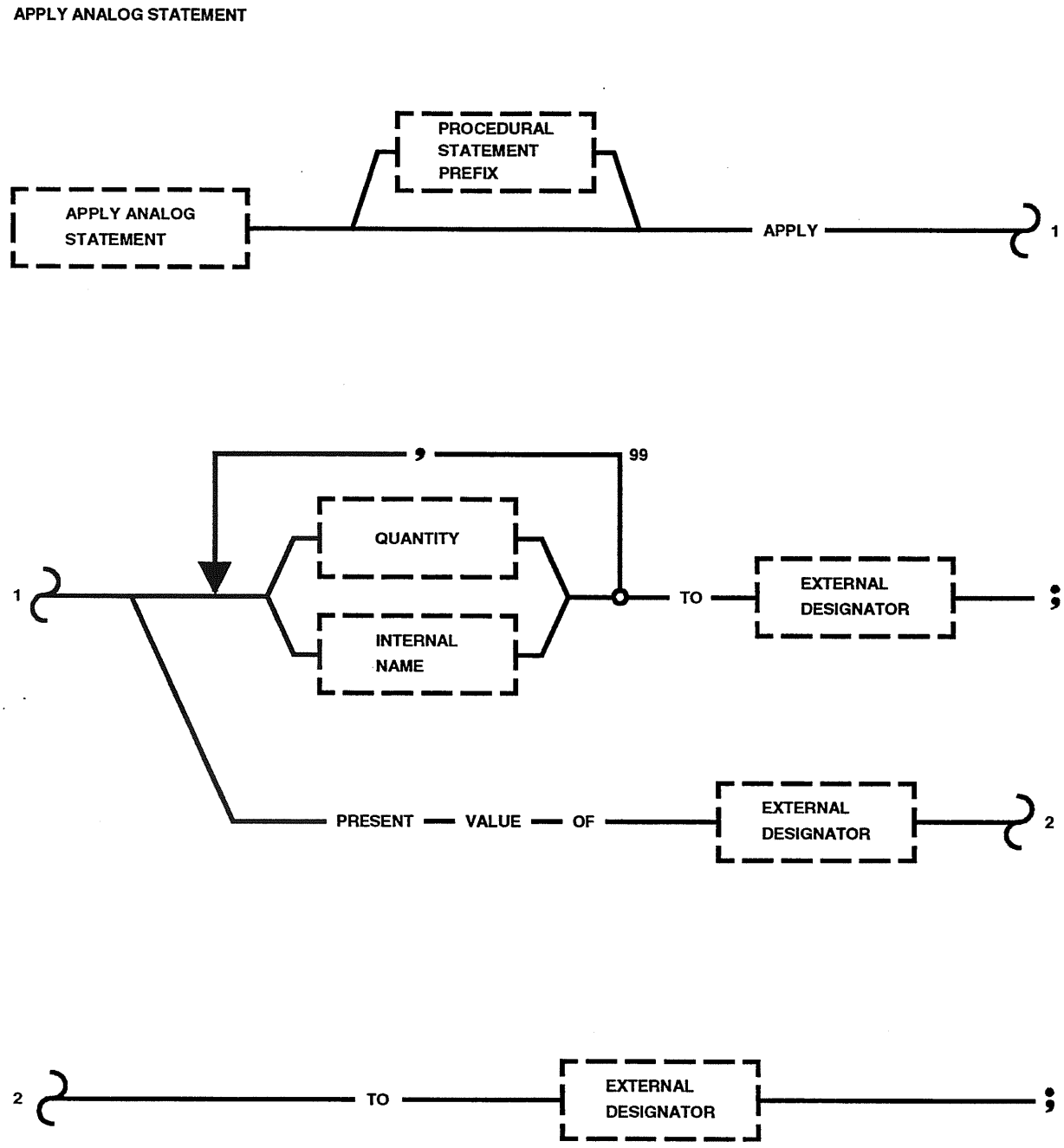


Figure 8-1 APPLY ANALOG Statement

Function

The APPLY ANALOG Statement issues analog commands to the system under test.

Description

The APPLY ANALOG Statement is used to issue quantity type data to the system under test. The quantity type data may be issued by referencing:

A. a quantity constant

B. an Internal Name

C. or the PRESENT VALUE OF an analog measurement or pseudo analog.

Refer to KSC-LPS-OP-033-04 for the on-board interface APPLY ANALOG Statement.

This statement may be used to apply one or more values to one or more Function Designators. Its operation is repeated until all data and Function Designators are processed. The following repetitive combinations are allowed:

One-to-One: A single value is applied to a single Function Designator.

One-to-Many: A single value is applied to many Function Designators.

Many-to-Many: Many values are applied to many Function Designators (first to first, second to second, etc.).

APPLY Quantity Option

A. One-to-One

Example

```
APPLY 5 AMP to < AS9 >;
```

This statement results in 5 Amperes (DC) being applied to the Function Designator < AS9 >.

B. One-to-Many

Example

```
APPLY 2V TO < AS1 >
            < AS2 >
            < AS3 >;
APPLY 1V TO (QTABLE2) FUNCTIONS;
```

C. Many-to-Many

Example

```
APPLY 1V, 1 AMP, 1V TO < AS1 >
                       < AS6 >
                       < AS3 >;
```

APPLY Internal Name Option

This option is the same as the APPLY QUANTITY option with the exception that Internal Names are used in place of Quantity constants.

A. APPLY Name to Function Designator

Example

```
APPLY (Q1) TO < AS1 >;
```

B. APPLY List or List Element to Function Designators

Examples

```
APPLY (QLIST1)2 TO < AS2 >;
APPLY (QLIST1)  TO < AS1 >
                   < AS2 >
                   < AS3 >;
```

C. APPLY Table Column or Table Element to Function Designators

Examples

```
APPLY (QTABLE1) ROW 1 COLUMN 2 TO < AS1 >;
APPLY (QTABLE1) COLUMN 1 TO  < AS1 >
                             < AS2 >
                             < AS3 >;
```

D. APPLY Many Internal Names to Many Function Designators

Example

```
APPLY (Q1), (QLIST1)3, (Q5) TO  < AS1 >
                                < AS2 >
                                < AS3 >;
```

APPLY PRESENT VALUE OF Option

In this option the present value of a Function Designator is read from the CDBFR by the GOAL Executor and applied to the analog stimulus referenced. If the PRESENT VALUE OF an LDB analog measurement is specified, the value will be read from the LDB using a TCS-1 READ operator.

Examples

```
APPLY PRESENT VALUE OF < AM2 > TO  < AS5 >;
APPLY PRESENT VALUE OF < AM1 > TO  < AS1 >
                                   < AS2 >
                                   < AS3 >;
APPLY PRESENT VALUE OF < AM1 > < AM2 > < AM3 >
                    TO < AS1 > < AS2 > < AS3 >;
```

Statement Execution

The APPLY ANALOG Statement is executed as follows:

A. The GOAL Executor initiates a request to apply the analog value to the appropriate FEP via the CDBFR.

B. The FEP applies the analog to the Function Designator and notifies the GOAL Executor if the operation was successful.

C. If the analog was successfully issued, the GOAL Executor will continue execution of the GOAL procedure. If an error response was received from the FEP, the GOAL Executor will process a Class III error. The GOAL Executor does not proceed to the next GOAL statement until the FEP response is received.

D. If repetitive operations are requested, each analog command is processed in the preceding manner. A successful response must be received from the FEP before the next analog command is issued. The GOAL Executor

does not proceed to the next statement until the last analog command in the statement is successfully issued.

E. If the APPLY ANALOG is executed for a command which has a prerequisite sequence, the request for the APPLY will be made through Control Logic.

F. If the External Designator specifies table functions, each row will be checked to insure it is active. If the row is inhibited, the operation will not be performed for the Function Designator in that row. This will not be considered an error and the operator will not be informed.

G. The value applied is scaled using the coefficients of the destination Function Designator.

H. If the value to be applied is lower than the low range, then the low range will be substituted for the value. If it is greater than the high range, then the high range will be substituted.

Error Processing

If an error response is received from the FEP on an APPLY ANALOG Statement, a Class III error will be processed.

If an error response is received from the FEP before the last repetitive operation has executed, a Class III error will be processed at that point. If procedure error override is active, the statement execution will continue. If procedure error override is inhibited, the operation of the procedure will be stopped.

A prerequisite sequence failure will be treated as a Class III error.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The data to be applied must be quantity type.

B. More than one value may not be applied to a single Function Designator.

C. In a Many-to-Many operation the number of applied values must equal the number of Function Designators.

D. The engineering units of the value being applied must be compatible with the engineering units of the Function Designator to which it is applied.

E. When a units mismatch occurs while using the Table Functions option, a compiler warning message will be output.

F. If a list or a table column is used to specify the data values, no additional data items may be specified.

G. In a One-to-Many operation, External Designators to which a single analog value is applied must be compatible types.

H. Each APPLY will be a separate request to the appropriate FEP.

I. For APPLY PVO of one table function to another, if any rows of the first table are inhibited, the corresponding rows of the second table must also be inhibited. Failure to adhere to this restriction will cause unpredictable execution results.

Legal Function Designators

Analog Stimulus (AS)
Pseudo Analog (PA)

Present Value Of option: The present value of an analog measurement or pseudo analog may be applied to an analog stimulus or pseudo analog.

THIS PAGE INTENTIONALLY LEFT BLANK.

8.2 SET DISCRETE STATEMENT

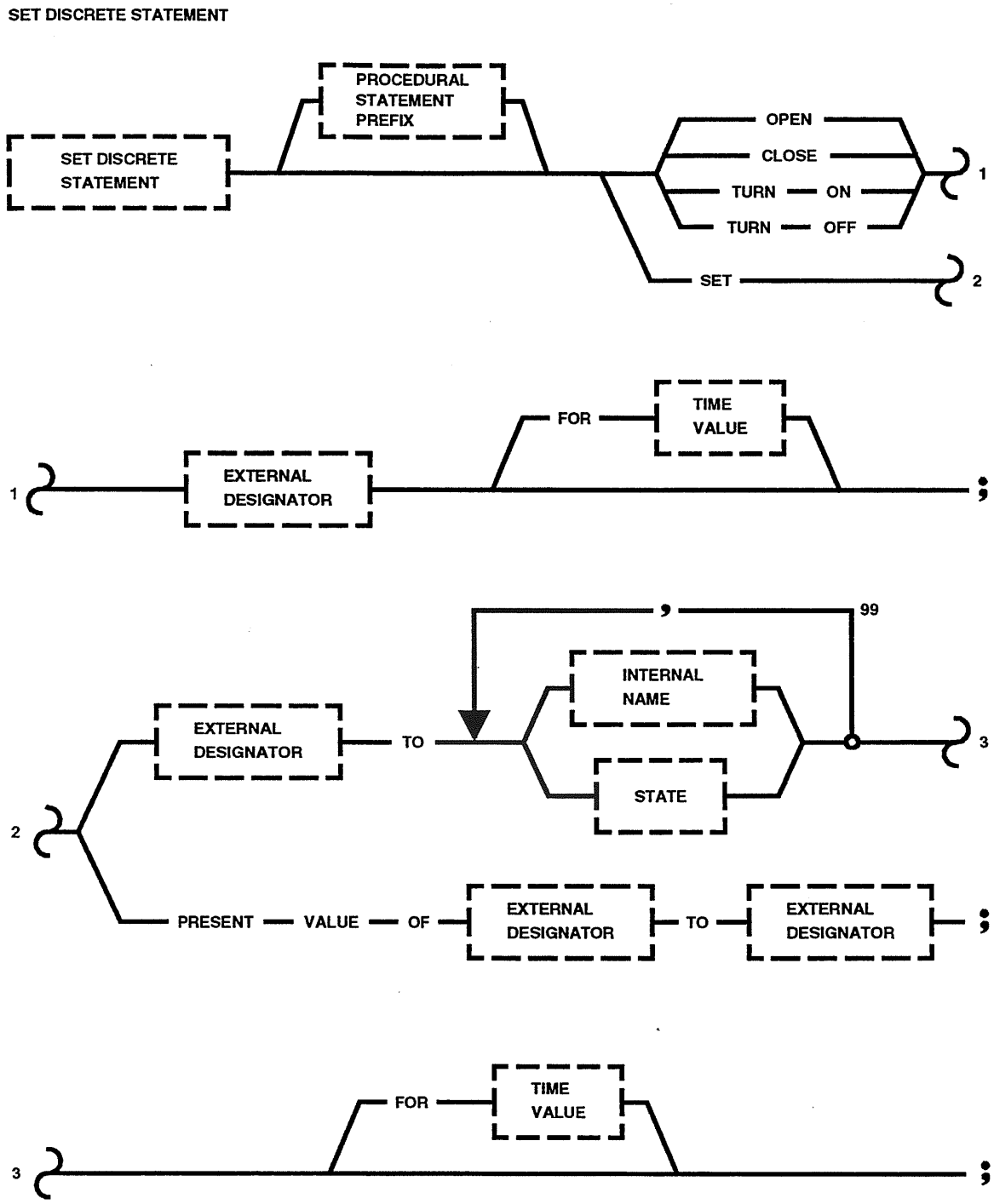


Figure 8-2 SET DISCRETE Statement

Function

The SET DISCRETE Statement issues discrete commands to the system under test and allows the procedure to control PFP lights, the audible alarms and pseudo discrete Function Designators.

Description

The SET DISCRETE Statement is used to issue state type data to the system under test. State type data includes discrete stimuli to the GSE or on-board system, PFP light status, audible alarm status and pseudo discretes. The state type data may be issued by referencing:

A. A state constant or literal

B. An Internal Name declared as a state data item

C. The present value of a discrete measurement, pseudo discrete or PFP light status

The statement may be used to set a discrete for a specified time value. When this option is used, the discrete will be set to the specified value. After the time value declared on the statement has expired, the state of the discrete will be reversed.

This statement may be used to set one or more Function Designators. Its operation is repeated until all Function Designators have been processed. The following repetitive operations are allowed:

One-to-One: A single value is issued to a single Function Designator.

Many-to-One: Many Function Designator set to one state.

Many-to-Many: Many values are applied to many Function Designators (first to first, second to second, etc.).

SET State Option

A. One-to-One

Examples

```
SET < DS1 > TO OFF;
TURN OFF < DS1 >;
```

Both examples set the discrete to OFF.

B. Many-to-One

Examples

```
SET < DS1 >
    < DS3 >
    < DS5 > TO ON;
CLOSE (STABLE1) FUNCTIONS;
TURN ON < DS3 > < DS2 >;
OPEN < DS6 > < DS7 > < DS9 >;
```

C. Many-to-Many

Example

```
SET < DS1 >
    < DS6 >
    < PD1 > TO ON, OFF, ON;
```

SET Internal Name Option

This option is the same as the SET State option with the exception that Internal Names are issued in place of State constants.

A. SET Function Designator to Name.

Example

```
SET < DS6 > TO (STATE7);
```

B. SET Function Designators to a list or an item of a list.

Example

```
SET < DS6 > TO (SLIST2)1;
SET < DS6 > < DS7 > < DS8 > TO (SLIST2);
```

C. SET Function Designator(s) to table column or table element.

Examples

```
SET < DS1 > TO (STABLE1) ROW1 COLUMN3;
SET < DS1 >
    < DS2 >
    < DS3 > TO (STABLE1) COLUMN2;
```

D. SET many Function Designators to many states using internal names.

Example

```
SET < DS1 > < DS6 > < DS11 > TO (STATE3), (STATE10),
(STATE20);
```

E. Set Table Functions to table column or table element.

Examples

```
SET (TABLE-A) FUNCTIONS TO (TABLE-B) COLUMN (X);
SET (TABLE-A) FUNCTIONS TO (TABLE-B) ROW 1 COLUMN 3;
```

NOTE: *In these two examples, Table B's inhibits will be accessed. If it is desired to access Table-A's inhibits instead, then use the following sequence of statements:*

```
ASSIGN (INT-NAME) = (TABLE-B) COLUMN (X);
SET (TABLE-A) FUNCTIONS TO (INT-NAME);
```

SET PRESENT VALUE OF Option

This option causes the present value of a discrete measurement to be obtained from the CDBFR or LDB FEP, and subsequently issued to the discrete stimulus referenced.

Examples

```
SET PRESENT VALUE OF < DM1 > TO < DS1 >;
SET PRESENT VALUE OF < PD1 > TO < DS1 >
                               < DS3 >
                               < DS5 >;
```

```
SET PRESENT VALUE OF < DM1 > < DM3 > < DM6 > TO
                     < DS1 > < DS2 > < DS6 >;
```

Time Value Option

This option causes the Function Designator to be set to the specified state. After the time value has elapsed, the Function Designator will be set to the opposite state.

```
SET < DS6 > TO OPEN FOR 5 MIN;
```

Statement Execution

The SET DISCRETE Statement is executed as follows:

A. The discrete state to be issued is obtained.

B. The GOAL Executor initiates a request to the responsible FEP, across the CDBFR, to set the discrete. If a prerequisite sequence is defined for the Function Designator, this request is accomplished through Control Logic.

C. The FEP issues the appropriate request to the hardware and notifies the GOAL Executor of the status of the operation.

D. If the discrete was successfully set, the GOAL Executor continues the execution of the procedure. If an error response was received from the FEP, the GOAL Executor will process a Class III error. The execution of the procedure does not continue until the response is received from the FEP.

E. If repetitive operations are requested, each discrete Function Designator is processed as above. A successful response must be received from the FEP before the next Function Designator is processed. The GOAL Executor does not proceed to the next statement until the last command in the SET Statement is issued.

F. If the Function Designator specified is a PFP light or audible alarm, no prerequisite checks are made. A request to the operating system is issued by the GOAL Executor to get the light or alarm turned ON or OFF. As with a discrete stimulus, processing will not continue until all the Function Designators have been processed.

G. If the Function Designator specified is a pseudo discrete, the GOAL Executor will request the appropriate discrete state to be written to the CDBFR. Processing will not continue to the next statement until all discrete Function Designators are processed. No prerequisite checks will be made before the pseudo discrete is set.

H. If a Time Value is specified on the SET Statement, processing will take place as above. After each Function Designator has been processed, the GOAL Executor will place the GOAL procedure in a hold state for the specified time. After that Time Value has elapsed, the Executor will complement the State issued, e.g., ON will become OFF, and issue the new State to the specified Function Designators. Processing will not continue until all discretes have been set, the Time Value exhausted and all the discretes reset.

I. If the External Designator is specified as table functions, each row will be checked to insure it is active. If a row is inhibited, the operation will not be performed for that row designator. This is not considered an error and the operator will not be informed.

Error Processing

If an error response is received from the FEP on a SET Statement, a Class III error will be processed.

If an error response is received from the FEP before the last repetitive operation has executed, a Class III error will be processed at that point. If procedure error override is active, the statement execution will continue. If procedure error override is inhibited, the operation of the procedure will be stopped.

A prerequisite sequence failure will be treated as a Class III error.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The data specified must be State type.

B. A single Function Designator may not be set to more than one State.

C. In a Many-to-Many operation, the number of State values must equal the number of Function Designators.

D. The State type of the value being set must be compatible with the State type of the Function Designator to which it is being issued.

E. If a list or a table column is used to specify the data values, no additional data items may be specified.

F. In a Many-to-One operation, Function Designators being set must be of compatible State types.

G. When a units mismatch occurs while using the Table Functions option, a compiler warning message will be output.

H. Each SET is processed as a separate output operation.

I. The time resolution of the SET Statement with a Time Value may be affected by the system processing load. This statement should not be used to generate pulses of critical duration.

J. When using the TIME VALUE option, the External Designator cannot be a mixture of CDBFR resident and non-CDBFR resident Function Designators.

K. For SET PVO of one table function to another, if any rows of the first table are inhibited, the corresponding rows of the second table must also be inhibited.

Acceptable Conditions

When a "TURN OFF <FD> FOR (TIME VALUE)" statement is specified with a console resident FD (i.e. PFPL), the GOAL Compiler generates a turn off instead of a turn on for the second issue command. The workaround for this problem is to code the "turn off", "delay", and "turn on" as separate statements.

If a statement is coded, setting more than one console resident (i.e. PFPL) or Pseudo FD to a mixed list of explicitly coded states and internal names for a time value, the GOAL Compiler will produce incorrect interpretive code. The workaround is to break up the statement.

Legal Function Designators

Discrete Stimulus (DS)
Pseudo Discretes (PD)
Programmable Function Panel Lights (PFPL)
Audible Alarms (ALRM)

Present Value Of Option: The present value of a discrete measurement, pseudo discrete, or PFP Light may be issued to a discrete type Function Designator.

THIS PAGE INTENTIONALLY LEFT BLANK.

8.3 TEST AND SET STATEMENT

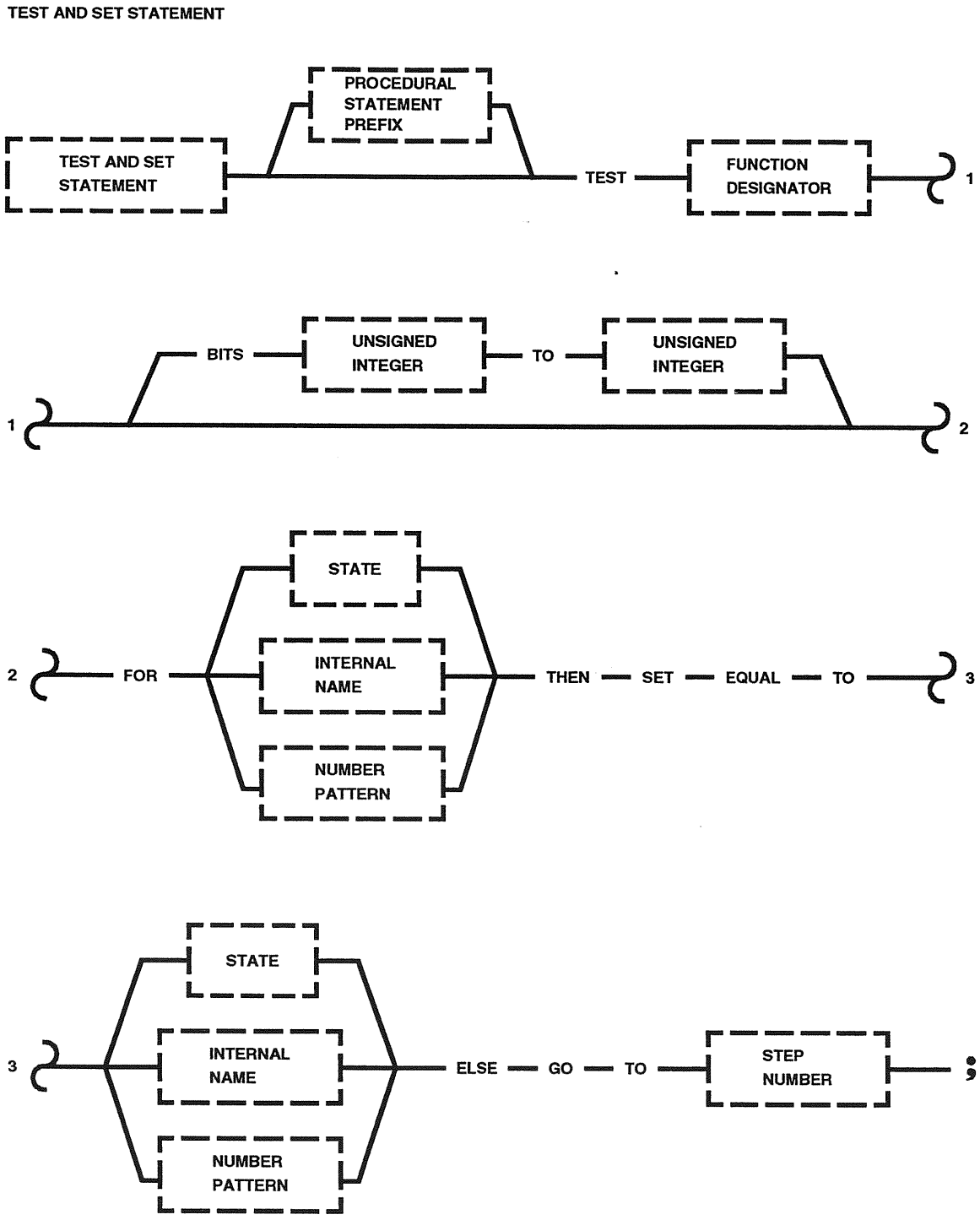


Figure 8-3 TEST AND SET Statement

Function

The TEST AND SET operation is used to conditionally set the value of a pseudo Function Designator based on the current value of the pseudo Function Designator or the value of a portion of the pseudo Function Designator as determined by the bit field option. This is a single uninterruptable operation.

Description

For State, Internal Name or Number Pattern options the pseudo Function Designator is verified equal to the value of a State, Internal Name or Number Pattern; then the same pseudo Function Designator is set to the value specified following the THEN SET EQUAL TO phrase. This is a single uninterruptable operation. If the pseudo Function Designator is not equal to the test value, the set operation will not occur and the pseudo Function Designator remains unchanged, and the ELSE option is executed.

Bit Field Option

The BIT FIELD option specifies which bit(s) within the pseudo digital pattern Function Designator are to be tested and set. This option is valid on pseudo digital pattern Function Designators only. The range of values is from 0 to 15, where 0 is the left and 15 is the right bit, the bit values being inclusive. If the value of the Internal Name or number pattern for either the test or set is larger than the bit field, then the rightmost bits up to the length of the bit field will be used.

Example

```
TEST <PD1> FOR OFF THEN SET EQUAL TO ON ELSE GO TO STEP 205;
```

In the previous example, the pseudo discrete Function Designator named <PD1> is tested for a status of OFF. If it is OFF, it is set to ON and the next statement will be processed. If it is not OFF, then processing will continue at step number 205.

Example

```
TEST <PDP1> BITS 13 TO 15 FOR 2 THEN SET EQUAL TO 4
    ELSE GO TO STEP 909;
```

In this example, bits 13 to 15 (inclusive) are tested for a value of 2. Bits 13 to 15 are set to a value of 4 if they are equal to 2. Processing then continues with the next statement. The

Function Designator will remain unchanged if bits 13 to 15 are not equal to 2, and processing will continue at step number 909.

Statement Execution

The TEST AND SET Statement tests a pseudo discrete, or pseudo digital pattern or a bit field in a pseudo digital pattern for equal to a value, and if equal to, sets the pseudo to a specified value. The TEST AND SET operation is performed, via CDBFR microcode, as one request so that no other concurrency or CPU can change the pseudo between the TEST AND SET operation. After the test the GO TO will be executed if the comparison fails.

Error Processing

A Class III error will occur if the required Function Designator cannot be read from the CDBFR.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Only one Function Designator can be specified.

B. The Function Designator type must be pseudo discrete or pseudo digital pattern.

C. The Internal Name must be a single numeric data item.

D. The step number referenced may not be within a repeat loop or a sequence unless the TEST AND SET Statement is located within the same loop or sequence.

E. The bit field option is only valid with pseudo digital pattern Function Designators.

Legal Function Designators

Pseudo Discrete (PD)
Pseudo Digital Pattern (PDP)

8.4 ISSUE DIGITAL PATTERN STATEMENT

ISSUE DIGITAL PATTERN STATEMENT

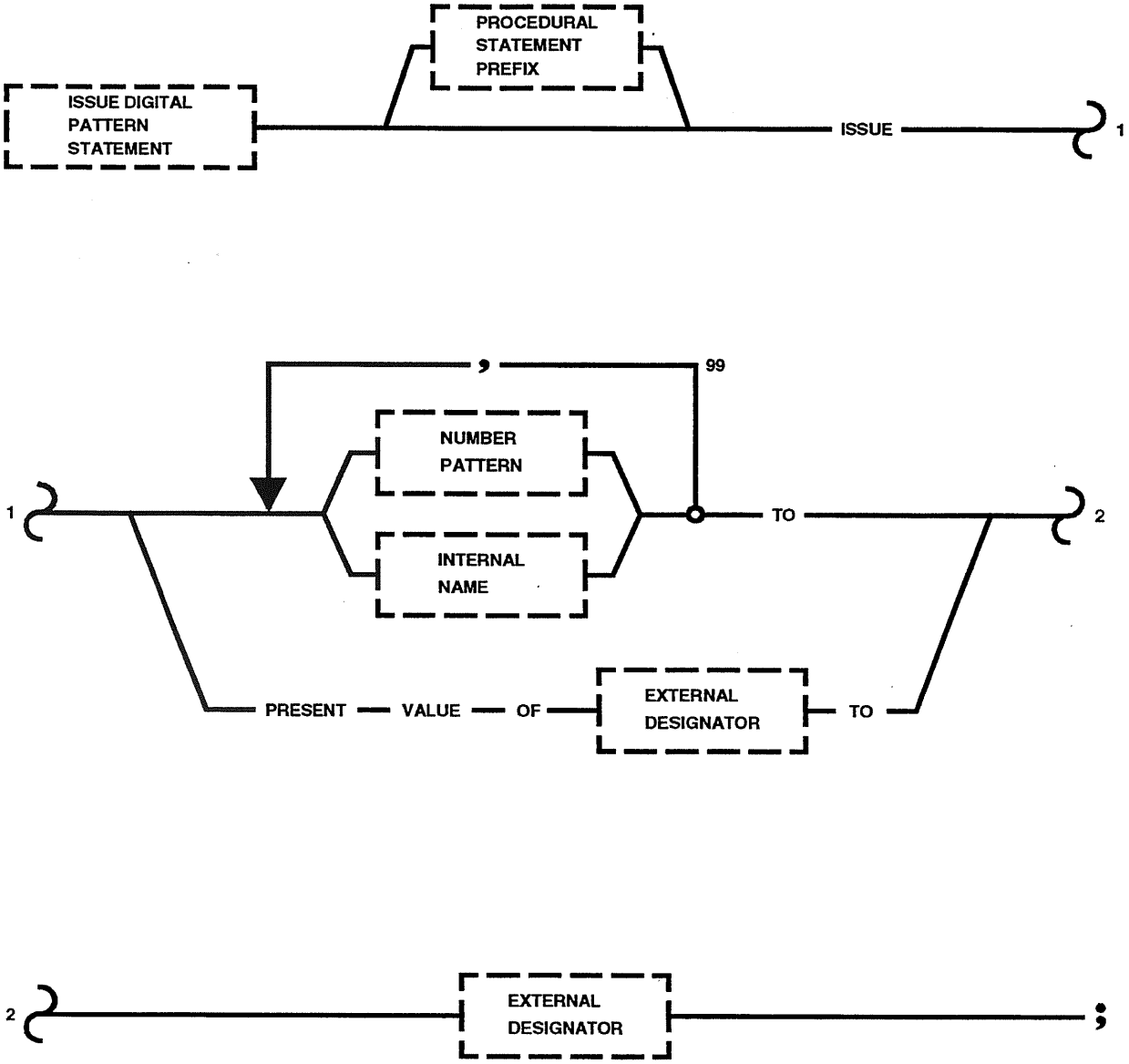


Figure 8-4 ISSUE DIGITAL PATTERN Statement

Function

The ISSUE DIGITAL PATTERN Statement issues digital pattern commands to the system being tested.

Description

The ISSUE DIGITAL PATTERN Statement is used to issue digital pattern data to the system under test. The digital pattern data may be issued by referencing:

A. A numeric constant

B. An Internal Name

C. The present value of a digital pattern measurement

Refer to KSC-LPS-OP-033-04 for the on-board interface ISSUE Statement.

This statement may be used to apply one or more values to one or more Function Designators. Its operation is repeated until all Function Designators are processed. The following repetitive operations are allowed:

One-to-One: A single value is issued to a single Function Designator.

One-to-Many: A single value is issued to many Function Designators.

Many-to-Many: Many values are applied to many Function Designators (first to first, second to second, etc.).

ISSUE Number Pattern Option

Examples

A. One-to-One

```
ISSUE T103 TO < DPS1 >;
```

This statement results in the octal value 103 being issued to the Function Designator < DPS1 >.

B. One-to-Many

```
ISSUE XFF TO < DPS1 > < DPS2 > < DPS3 >;
ISSUE T377 TO (NTABLE2) FUNCTIONS;
ISSUE B10001 TO < PDP1 > < PDP3 >;
```

C. Many-to-Many

```
ISSUE 103, 256, B101100 TO < DPS1 >
                           < DPS2 >
                           < DPS3 >;
```

ISSUE Internal Name Option

This option is the same as the ISSUE NUMBER PATTERN option with the exception that Internal Names are used instead of numeric constants.

Examples

A. ISSUE Name to Function Designator

```
ISSUE (T1) TO < DPS3 >;
```

B. ISSUE LIST or LIST Element to Function Designator

```
ISSUE (NLIST1) TO < DPS1 >
                  < DPS2 >
                  < DPS3 >;
ISSUE (NLIST2) 3 TO < DPS1 >;
```

C. ISSUE Table Column or Table element to Function Designators

```
ISSUE (NTABLE2) COLUMN 3 TO < DPS1 >
                            < DPS2 >
                            < DPS3 >;
ISSUE (NTABLE2) COLUMN 1 TO (NTABLE1) FUNCTIONS;
ISSUE (NTABLE1) ROW 3 COLUMN 2 TO < DPS3 >;
```

D. ISSUE many Internal Names to many Function Designators

```
ISSUE (B1), (NLIST3)2, (XABC) TO < DPS1 >
                                 < DPS2 >
                                 < DPS3 >;
```

PRESENT VALUE OF Option

This option will cause the value of the digital pattern measurement to be obtained and issued to the digital pattern stimuli.

Examples

```
ISSUE PRESENT VALUE OF < DPM1 > TO < DPS1 >;
ISSUE PRESENT VALUE OF < DPM1 > TO < DPS1 >
                                   < DPS2 >;
ISSUE PRESENT VALUE OF (NTABLE 1) FUNCTIONS TO < DPS1 >
    < DPS2 > < DPS3 >;
```

Statement Execution

The ISSUE DIGITAL PATTERN Statement is executed as follows:

A. The GOAL Executor will obtain the value of the Number Pattern to be issued, and initiate a request to the appropriate FEP via the CDBFR.

B. The FEP issues the Number Pattern to the digital pattern stimuli, and notifies the GOAL Executor of the success or failure status of the issue.

C. If the digital pattern was successfully issued, the GOAL Executor will continue execution of the GOAL procedure. If an error response was received from the FEP, the GOAL Executor will process a Class III error. The execution of the procedures does not continue until the response is received from the FEP.

D. If repetitive operations are requested, each digital command is processed in the preceding manner. A successful response must be received from the FEP before the next digital pattern command is issued.

E. If the ISSUE specifies digital pattern stimulus type Function Designators which have corresponding prerequisite sequence, the requests for the ISSUE are made through Control Logic. If the prerequisite sequence fails, Control Logic will notify the GOAL Executor and a Class III error will be processed.

F. If the External Designator specifies table functions, each row will be checked to insure it is active. If the row is inhibited, the operation will not be performed for

the Function Designator in that row. This will not be considered as an error, and the operator will not be informed.

G. If the Function Designator is a pseudo digital pattern, the GOAL Executor will write the number pattern in the appropriate CDBFR location rather than issuing the value via the FEP.

No prerequisite sequences are allowed for pseudo digital patterns. Otherwise, processing is the same as for other digital pattern stimuli.

Error Processing

If an error response is received from an FEP on an ISSUE Statement, a Class III error will be processed.

If an error response is received from Control Logic due to a prerequisite sequence failure, a Class III error will be processed.

If an error is encountered obtaining the present value of a digital pattern measurement, a Class III error will be processed.

If an error is encountered writing a pseudo digital pattern to the CDBFR, a Class III error will be processed.

If any of the above errors are encountered before the last repetitive operation has executed, a Class III error will be processed at that point. If procedure error override is active, the statement execution will continue. If procedure error override is inhibited, the operation of the procedure will be stopped.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The data to be issued must be numeric or keystroke type data.

B. More than one value may not be applied to a single Function Designator.

C. In a Many-to-Many operation, the number of values to be issued must equal the number of Function Designators.

D. If a List or Table column is used to specify the data values, no additional data items may be specified.

E. Each ISSUE will be a separate input/output request.

F. Digital Pattern stimuli for GSE may not exceed 8 bits.

G. For ISSUE PRESENT VALUE OF (PVO) of one table function to another, if any rows of the first table are inhibited, the corresponding rows of the second table must also be inhibited.

Legal Function Designators

Digital Pattern Stimulus (DPS)
Pseudo Digital Pattern (PDP)

Legal Function Designators for the PRESENT VALUE OF option are:

Digital Pattern Measurements (DPM)
Pseudo Digital Patterns (PDP)

THIS PAGE INTENTIONALLY LEFT BLANK.

8.5 READ STATEMENT

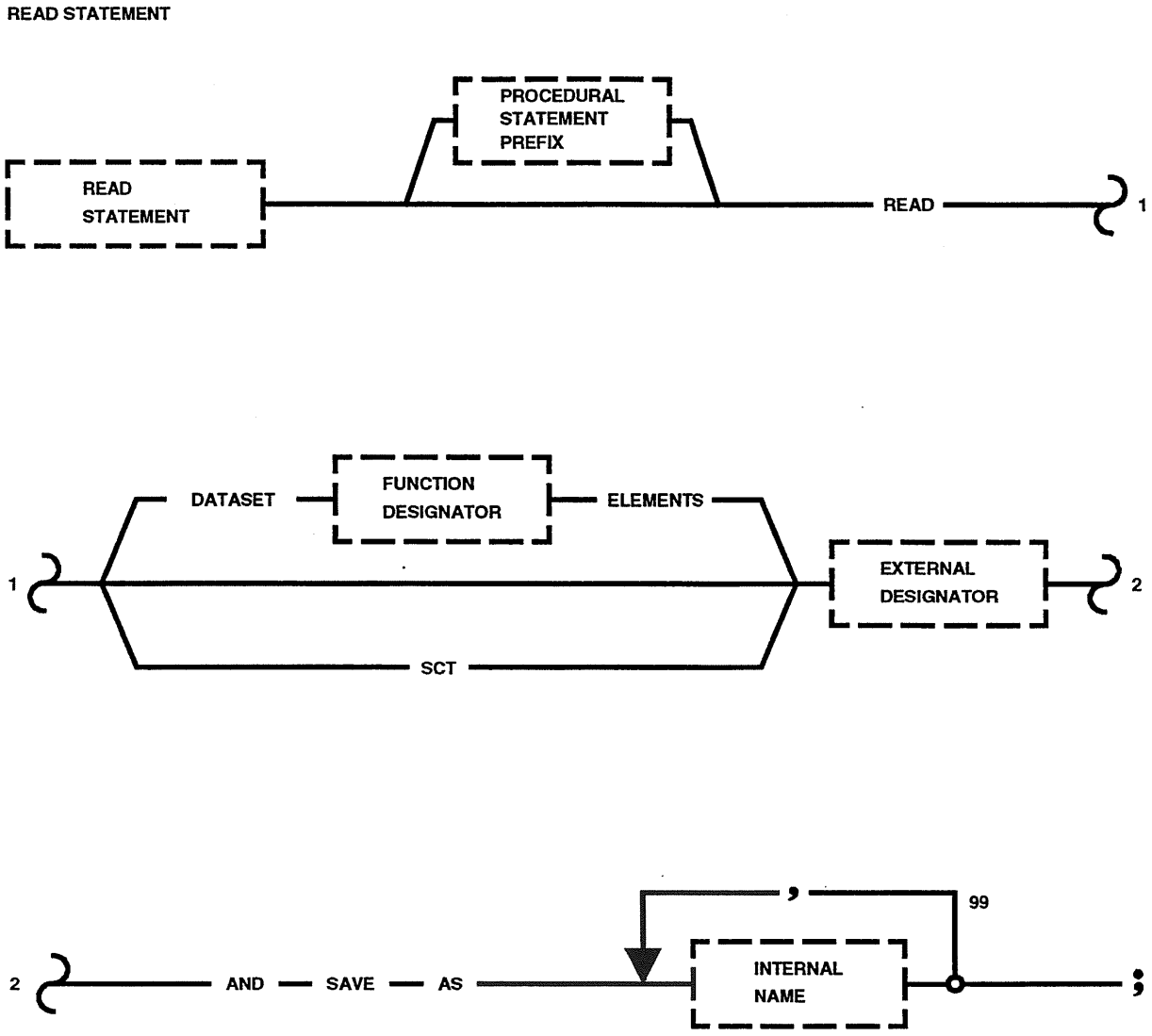


Figure 8-5 READ Statement

Function

The READ Statement is used to acquire data from external sources and store this data in specified internal variables.

Description

The READ Statement reads the values of Function Designators and stores these values in specified internal variables identified as Internal Names.

This statement may be used to read one or more Function Designators and store to one or more internal variables. Its operation is repeated until all Function Designators and internal variables are processed. The following repetitive combinations are allowed:

One-to-One: A single Function Designator is read and stored in a single internal variable.

One-to-Many: A single Function Designator is read and stored in many internal variables.

Many-to-Many: Many Function Designators are read and stored in many internal variables (first to first, second to second, etc.). The size of a list or table column may exceed the number of Function Designators to be read. The excess elements are unaffected.

Refer to KSC-LPS-OP-033-04 for the on-board interface READ Statements.

READ AND SAVE AS Option

Examples

```
READ < DM1 > AND SAVE AS (STATE1);
READ < AM1 >
     < AM2 >
     < AM3 > AND SAVE AS (QLIST1);
READ < DM1 > AND SAVE AS (STABLE1) COLUMN 1;
```

The first example reads the discrete measurement, < DM1 >, and saves its value in the Name, (STATE1). The second example reads three analog measurements, and saves them in the Quantity List (QLIST1). The third example reads a single discrete measurement, and saves its value in an entire table column.

DATASET Option

Example

```
READ DATASET < THDS1 > ELEMENTS < DM1 >,
    < AM1 > AND SAVE AS (STATE1), (Q1);
READ DATASET < THDS2 > ELEMENTS < DM1 >,
    < DM2 >, < DM3 > AND SAVE AS (SLIST2);
```

The DATASET option may be used when Function Designators have been defined as members of a dataset, and should be obtained as a time homogeneous group from the CDBFR. A single-time homogeneous Dataset Function Designator must be specified following the keyword DATASET, and the Function Designators which are to be read following the keyword ELEMENTS.

READ SCT Option

Examples

```
READ SCT <DM1> AND SAVE AS (NUM);
READ SCT <DM1> AND SAVE AS (NLIST);
READ SCT <DM1> <DM2> <DM3> AND SAVE AS (NLIST);
```

The first example reads the discrete measurement <DM1> and saves its value in the Name (NUM). The second example reads the same Function Designator but saves the value in the numeric list (NLIST). The third reads three discrete measurements and saves them in the numeric list (NLIST). The Function Designators must be of type SSA1 and subtype SCT.

Statement Execution

GSE and PCM Measurements:

GSE and PCM measurement status is read from the CDBFR using either the scatter/gather read or time homogeneous read depending upon the Function Designator subtype. The data obtained is converted to the proper format (e.g., analogs scaled) before it is saved.

GSE Stimuli:

The feedback from the last commanded status of GSE stimuli may be obtained from the CDBFR. The data obtained from the CDBFR will be converted to the proper format and saved.

LDB Measurements:

LDB measurements will be read from the LDB. The data obtained will be converted to the proper format before it is saved.

Multiword Data:

Multiword Data will be read from the CDBFR. No conversion will be done on this data before it is saved, except for GPC floating point which gets converted to Modcomp floating point value.

CDT, MET, GMT, and JTOY:

CDT, MET, GMT and JTOY will be obtained from Console memory.

Interval Timer:

The Interval Timer will be read from the GOAL Executor's memory. This timer must have been set by the procedure.

Date:

The date (month, day, and year) will be obtained from the CDBFR as text data (9 ASCII characters manually loaded by Master Console to CDBFR).

Pseudo Function Designators:

Pseudo Function Designators will be read from the CDBFR and be saved. No data manipulation is required, except for state data.

PFP Lights:

The status of PFP lights will be read via the Operating System PFP interface processor.

Error Processing

Failure of a READ request results in a Class III error.

Fixed Page 1:

The fixed Page 1 location will be obtained from the CDBFR as text data.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. No more than one Function Designator may be read and saved as a single internal variable.

B. In a Many-to-Many operation, the number of Function Designators must be equal to the number of internal variables. If a list or table column is used, the number of Function Designators must be equal to or less than the number of items in the list.

C. When doing MTU to Ground GMT Synchronization, the user should do multiple READ's of the GMT DELTA < FD >, since the possibility exists that variations may occur in the readings. The smallest value obtained in multiple READ's is the most accurate.

D. If a list or table column is used to specify the data values, no additional data items may be specified.

E. The Function Designator type being read must be compatible with the internal variable type in which it is saved.

F. When a units mismatch occurs while using the Table Functions option, a compiler warning message will be output.

G. All Function Designators specified with the DATASET option must be members of the time homogeneous dataset specified by the Function Designator following the keyword DATASET. This specification is validated by the GOAL Configurator.

H. A dataset type Function Designator which is a Pseudo Parameter may not be used in the READ Statement. Elements of a dataset may not be Pseudo Parameters.

I. Only one Multiword Digital Pattern Function Designator may be used in the READ statement.

J. If the External Designators are specified as table functions, a Row Designator must be active to have its value updated in the VDA.

K. A READ SCT Statement may only read a Function Designator of type SSA1 with subtype SCT. The value must be saved in a numeric type internal variable.

Function Designator Type	Internal Name Type
Analog Measurement (AM)	QUANTITY
Analog Stimulus (AS)	QUANTITY
Discrete Measurement (DM)	STATE
Discrete Stimulus (DS)	STATE
Digital Pattern Measurement (DPM)	NUMERIC
Digital Pattern Stimulus (DPS)	NUMERIC
Countdown Time (CDT)	QUANTITY
Greenwich Mean Time (GMT)	QUANTITY
Interval Timer (TIMR)	QUANTITY
Date (DATE)	TEXT
PFP Light (PFPL)	STATE
Analog Double Precision (AMDP)	QUANTITY
Floating Point (FP)	QUANTITY
Multiword Digital Pattern (MWDP)	NUMERIC LIST or COLUMN, or series of SINGLE DATA ITEMS
Time Homogeneous Dataset (THDS)	NONE
Pseudo Analog (PA)	QUANTITY
Pseudo Discrete (PD)	STATE
Pseudo Digital Pattern (PDP)	NUMERIC
Mission Elapsed Time (CDT)	QUANTITY
Julian Time of Year (CDT)	QUANTITY
System Status Discrete (SSA1)	STATE
System Status Discrete (w/READ SCT)	NUMERIC
System Status Digital Pattern (SSA2)	NUMERIC
GMT Delta (GMTD)	QUANTITY
Fixed Page 1 (FPG1)	TEXT

THIS PAGE INTENTIONALLY LEFT BLANK.

8.6 READ FEP STATUS STATEMENT

READ FEP STATUS STATEMENT

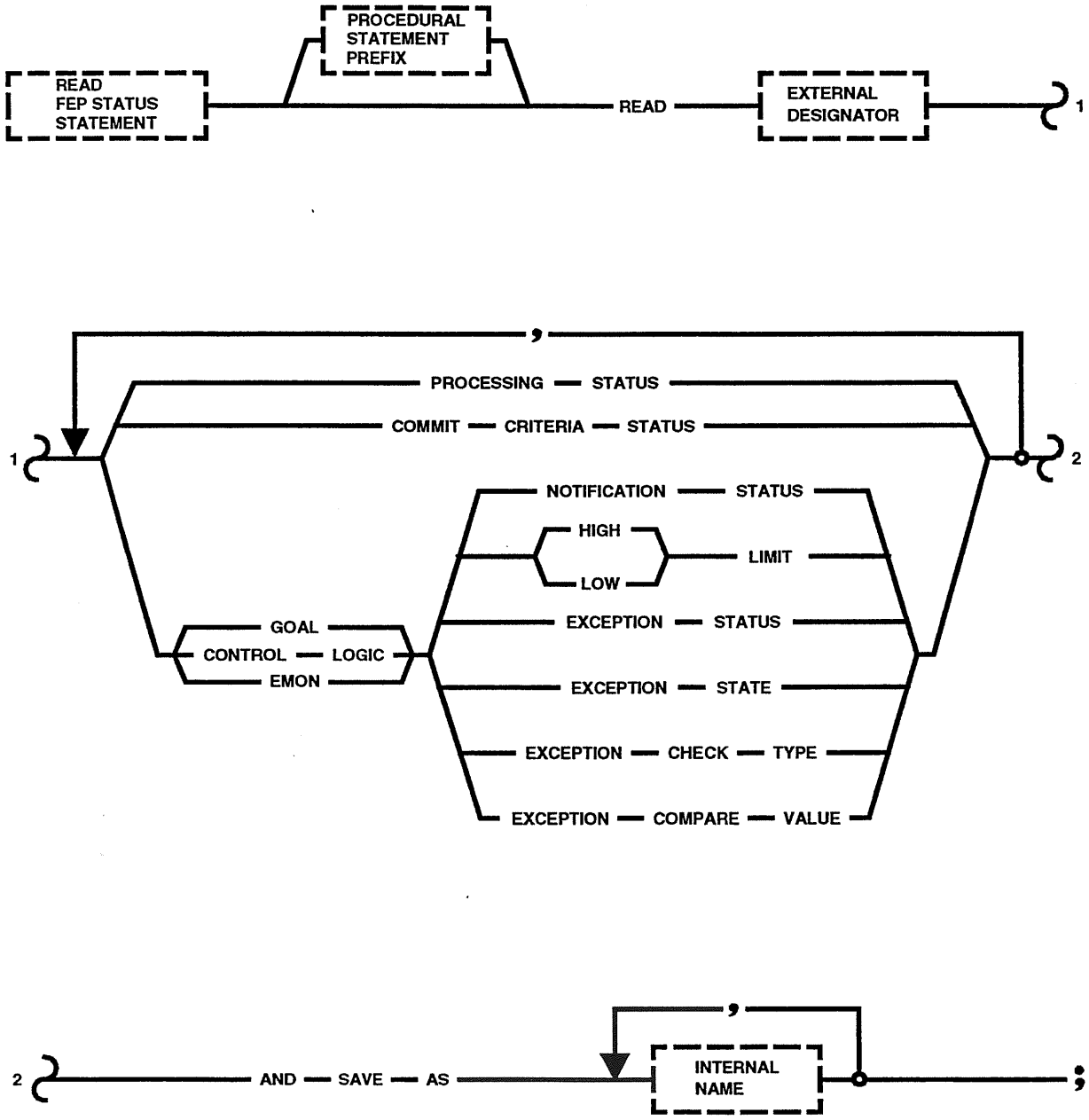


Figure 8-6 READ FEP STATUS Statement

Function

The READ FEP STATUS Statement is used to obtain, for one or more Function Designators, the value of various exception monitoring and console notification indicators.

Description

The following options are provided which return data for the Function Designator(s) as described:

A. PROCESSING STATUS

Returns the bit value (ON/OFF) indicating whether FEP processing (exception monitoring, significant change checking, linearization) is currently active or inhibited.

B. COMMIT CRITERIA STATUS

Returns the bit value (ON/OFF) indicating whether or not commit criteria console notification is currently active.

The following notification values and indicators may be read for GOAL, Control Logic, and Exception Monitoring:

A. NOTIFICATION STATUS

Returns the bit value (ON/OFF) indicating whether or not the specified system is to be notified when the measurements value is in an exception condition.

B. HIGH/LOW LIMIT

Returns the current high or low limit value of an analog Function Designator. These limits will be the same for CL and EMON.

C. EXCEPTION STATE

Returns the current state value defined as the exception value of a discrete Function Designator. These states will be the same for CL and EMON.

D. EXCEPTION CHECK TYPE

Returns the current type of exception checking (1 = equal to, 2 = not equal to, 0 = any change) for a digital pattern Function Designator. These checks will be the same for CL and EMON.

E. EXCEPTION COMPARE VALUE

Returns the current value for a digital pattern Function Designator against which exception checking is performed.

F. EXCEPTION STATUS

Returns a value showing whether or not the value of a Function Designator is currently in an exception condition (0 = IN, 1 = OUT or LOW, 2 = HIGH).

Examples

```
READ <DM1> EXCEPTION STATE AND SAVE AS (S1);
```

This example will cause the value currently defined as the exception state of <DM1> to be stored in the internal variable (S1).

```
READ <DPM1> EXCEPTION CHECK TYPE AND SAVE AS (N1);
```

This statement will result in value being stored in the internal variable (N1) which indicates whether the current type of exception checking for <DPM1> is "equal to", "not equal to", or "any change."

```
READ <AM1> COMMIT CRITERIA STATUS AND SAVE AS (S1);
```

This statement will return a discrete indicator showing whether or not commit criteria console notification is currently active for <AM1>.

Statement Execution

The GOAL Executor sends one FEP status request per Function Designator to the FEP. All status options chosen for the Function Designator are included. Upon receiving the request, the FEP sends back a "mini-status" response containing data for all possible status requests. The GOAL executor then chooses the data requested from the mini-status and stores it in the specified internal variables.

Error Processing

A Class III error will result if a request is made for analog limits or exception status of a GPC Floating Point which has no limits or if there is a communication failure with the FEP.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. For PROCESSING STATUS, COMMIT CRITERIA STATUS and NOTIFICATION STATUS options, the Internal Name(s) following AND SAVE AS must be state type with subtype ON/OFF.

B. For HIGH/LOW LIMIT options, the Internal Name(s) following AND SAVE AS must be quantity type and have engineering units compatible with the Function Designator(s).

C. When a units mismatch occurs while using the Table Functions option, a compiler warning message will be output.

D. For EXCEPTION STATUS, EXCEPTION CHECK TYPE and EXCEPTION COMPARE VALUE, the internal variable(s) following AND SAVE AS must be numeric.

E. For EXCEPTION STATE option, the internal variable(s) following AND SAVE AS must be state type and have subtype(s) compatible with the Function Designator(s).

F. The number of items represented by the Internal Name(s) following AND SAVE AS must equal the number of Function Designators specified.

G. If multiple options are specified in one statement, only one Function Designator may be specified.

H. HIGH/LOW LIMIT and EXCEPTION STATUS options should not be selected for GPC Floating Point Function Designators with no limits. This condition cannot be checked by the compiler and will result in an execution time error. The Executor will output a Class III error if this condition arises. If the procedure is resumed from the error stop or the stop is overridden, zeroes will be stored in the procedure's VDA and execution of the remainder of the statement will resume.

I. EXCEPTION STATUS is subject to change subsequent to a READ.

J. An option may not be specified more than once in the option list for a Function Designator.

Legal Function Designators

The following table represents legal Function Designator types and sources* for the options of the READ FEP STATUS Statement:

	FD TYPE					
STATEMENT OPTION	AM	DM	DPM	MWDP	FP	AMDP
PROCESSING STATUS	R,S, G,P	R,S, G,P	R,S, G,P	P	P	P
COMMIT CRITERIA STATUS	G,P	G,P	G,P	P	P	P
GOAL/CL/EM NOTIFICATION STATUS	R,S, G,P	R,S, G,P	R,S, G,P	P	P	P
GOAL/CL/EM HIGH/LOW LIMITS	R,S, G,P				P	P
GOAL/CL/EM EXCEPTION STATUS	R,S, G,P	R,S, G,P	R,S, P		P	P
GOAL/CL/EM EXCEPTION STATE		R,S, G,P				
GOAL/CL/EM EXCEPTION CHECK TYPE			R,S, P	P		
GOAL/CL/EM EXCEPTION COMPARE VALUE			R,S, P			

* Function Designator Sources:
R = RTU; S = SDB; G = GSE; P = PCM

9. PROCEDURE CONTROL STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The Procedure Control Statements are used to initiate and control GOAL programs and Non-GOAL programs. The Procedure Control Statements and related GOAL elements which will be discussed in the indicated sections are as follows:

CONCURRENT STATEMENT (Section 9.1)

PARAMETER/PSEUDO PARAMETER (Section 9.2)

DEFINE STATEMENT (Section 9.3)

RELEASE CONCURRENT STATEMENT (Section 9.4)

PERFORM PROGRAM STATEMENT (Section 9.5)

STOP STATEMENT (Section 9.6)

TERMINATE STATEMENT (Section 9.7)

THIS PAGE INTENTIONALLY LEFT BLANK.

9.1 CONCURRENT STATEMENT

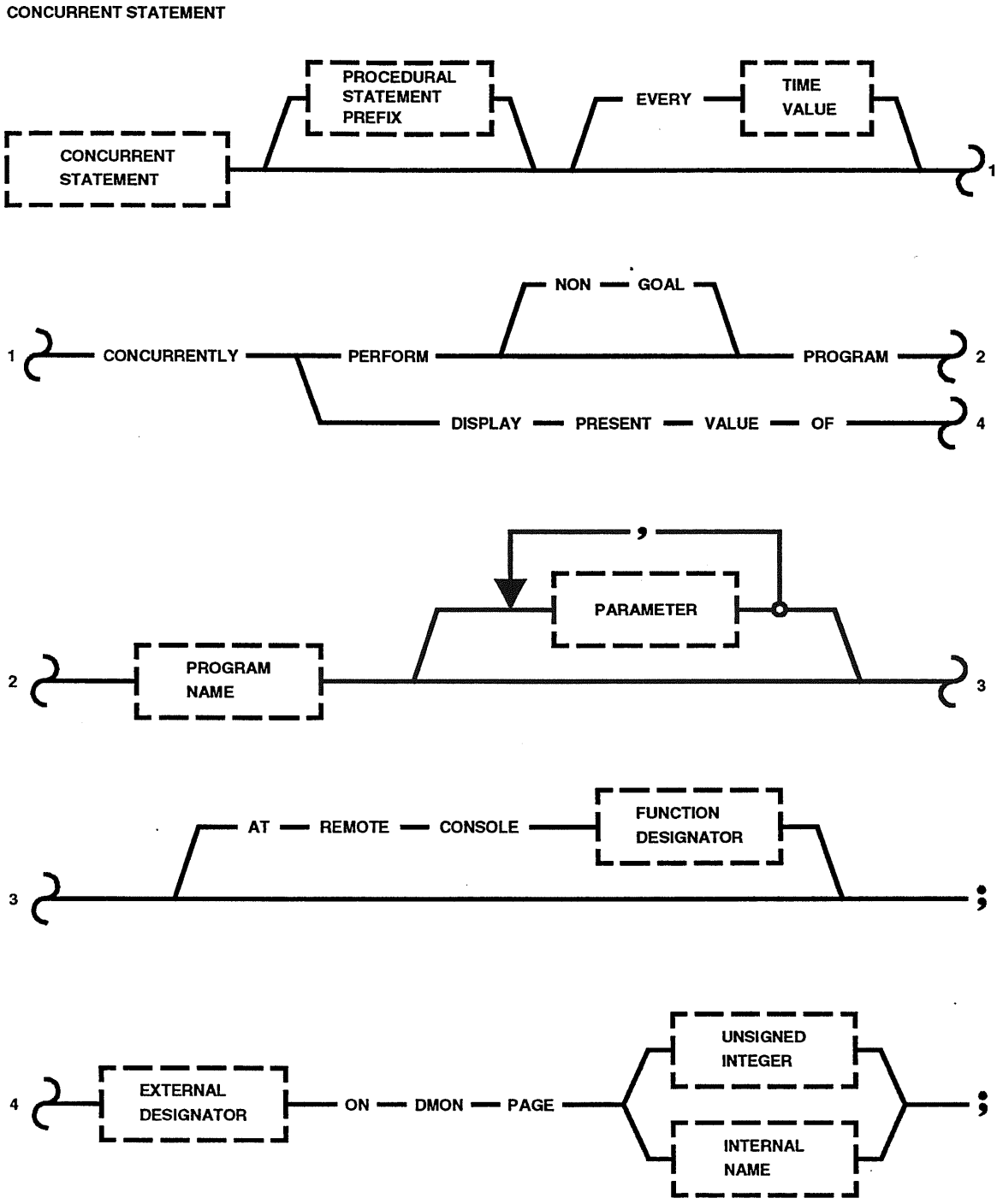


Figure 9-1 CONCURRENT Statement

Function

The CONCURRENT Statement is used to initiate the parallel execution of a program and pass any required parameters or to initiate monitoring of a specified measurement.

Description

A concurrent program is a program which executes in parallel with another program. When a program concurrently performs another program, the calling program continues execution. Concurrent programs may be executed only once, or they may be executed cyclically. Refer to KSC-LPS-OP-033-04 for the on-board interface CONCURRENT Statement.

If a Time Value is specified after the keyword EVERY, then the concurrent program will be restarted periodically at the specified time interval. If the time interval expires prior to completion of execution, the program will continue to completion and will then be restarted immediately. Internal Names used in cyclic concurrencies are not reinitialized at the beginning of successive cycles. This permits program logic and calculations to continue without interruption. A cyclically executing concurrent program may be discontinued by use of the RELEASE Statement. The CONCURRENT Statement may be used to concurrently perform GOAL or Non-GOAL programs.

Parameters may be passed to but not directly received from the program concurrently performed. Parameters passed to a cyclic task are passed only at the start of the first execution cycle.

The AT REMOTE CONSOLE option may be used to initiate a GOAL or Non-GOAL program at another console.

Monitoring of specified measurements is performed by concurrent operation of the Display Monitor (DMON) which reads the present value of specified Function Designators and displays them on the specified DMON page. Once activated, measurements are monitored continuously with the display being updated every nine seconds. Update occurs only while the specified page is selected for viewing and is terminated by execution of a RELEASE CONCURRENT Statement.

CONCURRENTLY PERFORM a GOAL Procedure

Examples

```
CONCURRENTLY PERFORM (TEST 1);
CONCURRENTLY PERFORM PROGRAM (TEST 2) (A), (B), < DM1 >;
CONCURRENTLY PERFORM PROGRAM (TEST 3)
    AT REMOTE CONSOLE < VAL1 >;
```

This first example concurrently performs the program (TEST 1). The second example concurrently performs the program (TEST 2) and passes parameters (A), (B), and < DM1 >. The last example concurrently performs the program (TEST 3) at a remote console.

Concurrent Cyclic Option

Examples

```
EVERY 30 SEC CONCURRENTLY PERFORM PROGRAM (TEST 4);
EVERY (TIME1) CONCURRENTLY PERFORM PROGRAM (TEST 5);
```

The first example concurrently performs the program (TEST 4) every 30 seconds. The second example concurrently performs the program (TEST 5) at the interval defined by the Name (TIME1).

CONCURRENTLY PERFORM Non-GOAL Programs

Examples

```
CONCURRENTLY PERFORM NON GOAL PROGRAM
    (TEST 6) (A), (B), (C), (D);
CONCURRENTLY PERFORM NON GOAL PROGRAM
    (TEST 7) AT REMOTE CONSOLE < VAC1 >;
```

The first example concurrently performs the Non-GOAL program (TEST 6) and passes the parameter variables (A), (B), (C), and (D). The second example concurrently performs a Non-GOAL program (TEST 7) in a remote console identified by < VAC1 >.

CONCURRENTLY DISPLAY PRESENT VALUE OF

Example

```
CONCURRENTLY DISPLAY PRESENT VALUE OF < AM9 > ON DMON PAGE 1;
```

This example concurrently performs the DMON program which displays the analog measurement present value to DMON page 1.

Statement Execution

To perform a task in the same console, the GOAL Executor will request one of the six application TCB's from the Operating System along with a 2.5K memory buffer. If the request is granted, the procedure will be obtained from the disk and will be initialized for execution. One disk access is required to complete this process. The GOAL Executor will also log the name of the task and the start time to the PDR.

To perform a task in a remote console, the GOAL Executor will issue a computer-to-computer communication to the GOAL Executor in the remote console. The GOAL Executor in the remote console then follows the procedure just described for performing a task. If parameters are required, they will be passed via a block transfer across the CDBFR.

To concurrently perform a Non-GOAL program, the GOAL Executor will request the Operating System to execute the program.

Error Processing

When concurrently performing a GOAL program, Class III errors will result if: No TCB's are available to execute the program, or the program being concurrently performed is already active at Level 1.

When concurrently performing a Non-GOAL program, a Class III error will be processed if the program cannot be obtained from disk for execution.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Only GOAL procedures in the same console may be cyclically performed. The Time Value specified must be an interval time; i.e., not GMT, CDT, MET, or JTOY.

B. The time interval for a cyclic perform must be in one second increments; therefore, any Time Values less than one second will be set to one second and any other Time Values will be truncated to whole second values.

C. The number of Parameters must be equal to the number of Pseudo Parameters specified on the BEGIN PROGRAM Statement of the called program.

D. The called program can terminate itself from cyclic operation by specification of TERMINATE SYSTEM.

E. No validation is performed on the DMON page number if it is an Internal Name. An invalid number will initiate a Class III run time error.

F. CYCLIC option is not allowed on a CONCURRENTLY DISPLAY.

G. A maximum of 30 DMON's can be requested for display on each of three DMON pages (includes those requested from keyboard). A Class III run time error will result if this maximum is exceeded.

H. Monitoring of LDB Function Designators is not supported by DMON.

Legal Function Designators

Only Console type Function Designators may be used with the REMOTE CONSOLE option.

Any Function Designator definable via the DEFINE Statement may be passed as a Parameter.

DMON

Analog Measurement (AM)
Pseudo Analog (PA)
Discrete Measurement (DM)
Pseudo Discrete (PD)
Digital Pattern Measurement (DPM)
Pseudo Digital Pattern (PDP)
Floating Point (FP)
Multiword Digital Pattern (MWDP)
Analog Measurement Double Precision (AMDP)

9.2 PARAMETER/PSEUDO PARAMETER

PARAMETER

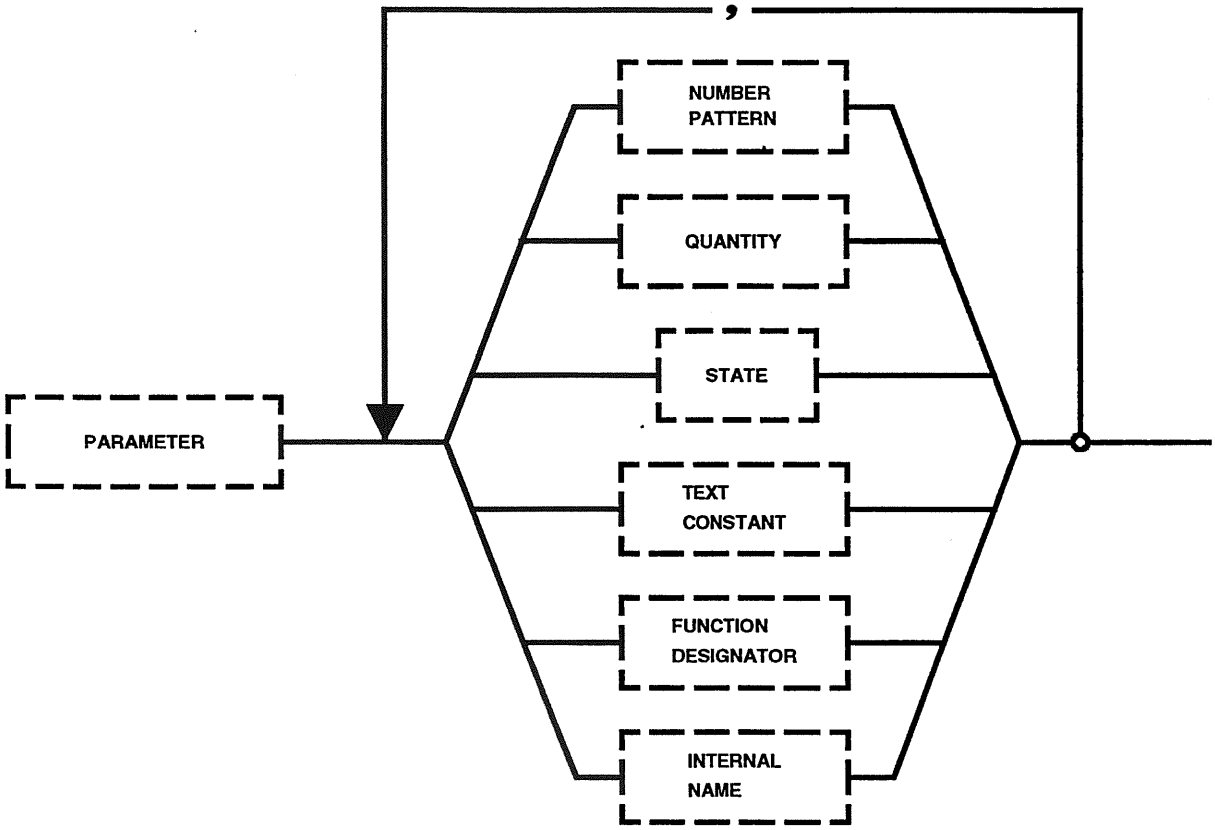


Figure 9-2 PARAMETER

PSUEDO PARAMETER

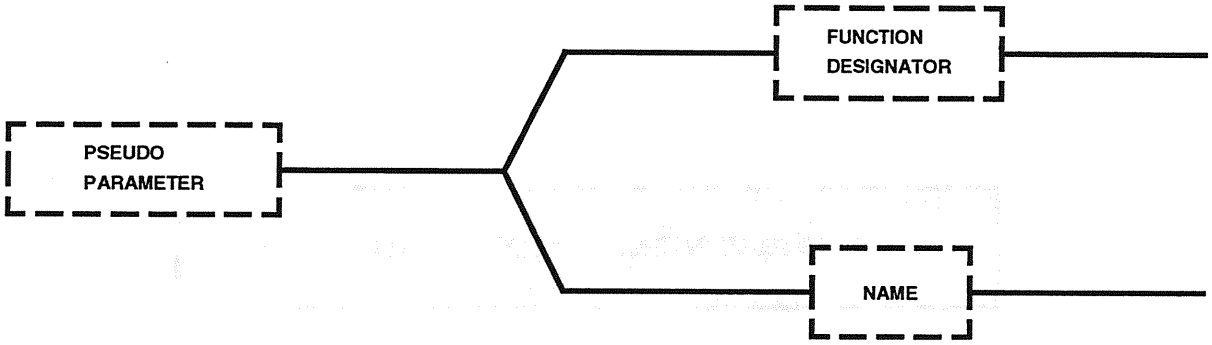


Figure 9-3 PSEUDO PARAMETER

THIS PAGE INTENTIONALLY LEFT BLANK.

Function

The Parameter and Pseudo Parameter elements are used to identify the data passed to a program.

Description

Parameters are specified on the CONCURRENT and the PERFORM Statements to identify the data to be passed. Pseudo Parameters are specified on the BEGIN PROGRAM Statement to identify the data to be received from a calling program.

Parameters may be specified as Number Pattern, Quantity, State, Text Constant, Internal Name, or Function Designator. Internal Names used as parameters may reference a Name, a List, or a Table Column. Entire Tables may not be used as parameters. Parameters must be compatible in type and size with the Pseudo Parameters which receive the data in the called program.

Pseudo Parameters may be specified as a Name or a Function Designator. Name refers to an Internal Name which may be a Name or a List Name. Table Names and Table Columns cannot be used as Pseudo Parameters. All Pseudo Parameters which are Names must be declared in a declaration statement. All Pseudo Parameters which are Function Designators must be defined in a DEFINE Statement. Pseudo Parameters may be used in a program according to the normal rules for Internal Names and Function Designators.

Pseudo Parameters must never be initialized. Their use is strictly to indicate the number of parameters to be received and data attributes to be syntactically checked at compile time.

It should be noted that both lists and table columns may be passed as parameters but that each is always received in a list in the receiving program.

Graphic characters are passed in the same manner as Text characters. Both require the declaration statement in the called program to specify a character count.

Parameter Passing

Example

```
PERFORM PROGRAM (EXAMPLE) 50, 5V, (STATE1), (QLIST1),
  < DM1 >, < AS1 >, TEXT(MESSAGE 1);
```

```
BEGIN PROGRAM (EXAMPLE) (PN1), (PQ1), (PSTATE1), (PQLIST1),
    < X >, < Y >, (PTEXT);
DECLARE NUMERIC (PN1) = I;
DECLARE QUANTITY (PQ1) = V;
DECLARE QUANTITY LIST (PQLIST1) WITH 100 ENTRIES V;
DECLARE TEXT (PTEXT) = 15 CHARACTERS;
DEFINE < X > AS SENSOR TYPE DISCRETE ON/OFF;
DEFINE < Y > AS LOAD TYPE ANALOG V;
APPLY (PQ1) to < Y >;
    .       .
    .       .
    .       .
```

The example illustrates parameter passing. Parameters are shown on the PERFORM Statement which performs the program (EXAMPLE). All of the data types which may be passed as parameters are included. The BEGIN PROGRAM Statement for the program (EXAMPLE) is shown. Note the Pseudo Parameters on the BEGIN Statement.

Declaration and Define Statements for the Pseudo Parameters are also included. Note the default subtypes in the declaration statements which do not initialize the Pseudo Parameters. After the parameters are passed, the APPLY Statements will APPLY 5 volts to < AS1 >.

Parameter Passing

To ensure compatibility between Parameters and Pseudo Parameters, the GOAL Configurator will perform parameter validation. The GOAL Language Processor will generate a Parameter Validation Table which contains information for each Parameter on a PERFORM Statement and each Pseudo Parameter on a BEGIN Statement. This information will be compared during procedure configuration, and any discrepancies between corresponding Parameters and Pseudo Parameters will result in the generation of a Configurator diagnostic message.

For Function Designators the information contained in the Parameter Validation Table is:

A. Type

B. Engineering units as state subtype

C. Function Designator precision

D. BTU classification (checked only if source is LDB)

E. Source

For Internal Names the information contained in the Parameter Validation Table is:

A. Type

B. Subtype

C. Internal Name length (number of entries in list or table column, or number of characters if text)

If an Internal Name parameter is obtained by the use of a variable index, only type checking will be performed.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. A maximum of 512 words are available for Parameter passing at execution time. The number of words required by each type of parameter follows:

1. Numeric - one word

2. Quantity - two words

3. State - one word

4. Text - two characters per memory word.

B. Pseudo Parameters must not be initialized in the declaration statements of the called program.

C. Entire tables may not be passed as parameters.

D. Table Names and Columns cannot be used as Pseudo Parameters, but a Table Column which is a Parameter may be passed to a List which is a Pseudo Parameter.

E. A dimensionless quantity literal parameter must specify either NULL or the decimal point; i.e., 10 NULL or 10.0.

Legal Function Designators

Any Function Designator describable via the DEFINE Statement may be passed as a Parameter.

9.3 DEFINE STATEMENT

DEFINE STATEMENT

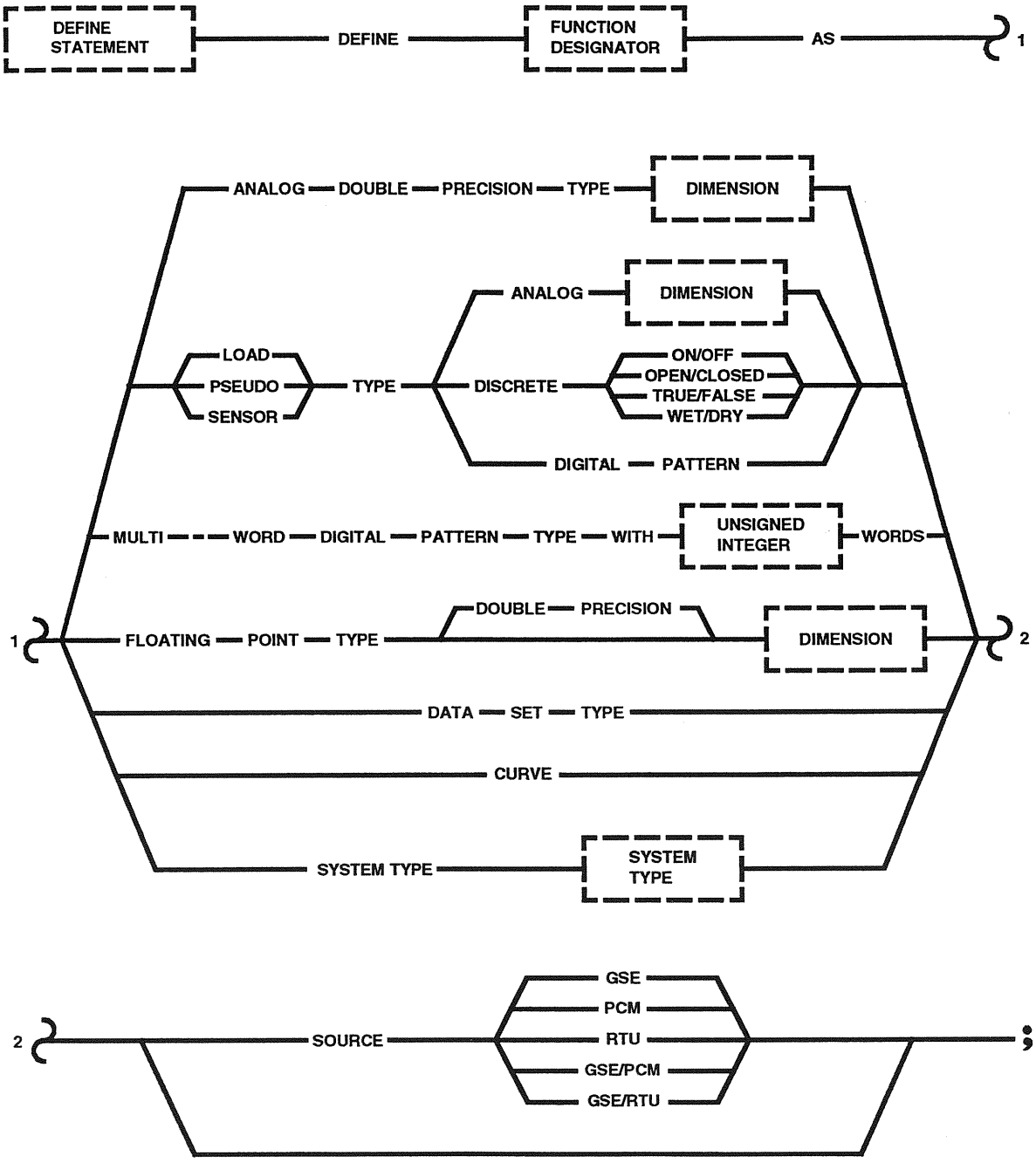


Figure 9-4 DEFINE Statement

Function

The DEFINE Statement is used to specify the characteristics of a Function Designator that is used as a Pseudo Parameter of a GOAL procedure.

Description

When parameters are passed to a program, the receiving procedure must have Pseudo Parameters which correspond to the parameters being passed. A Pseudo Parameter may be an Internal Name or a Function Designator. Pseudo Parameters which are Internal Names must be declared, and Pseudo Parameters which are FD's must be defined.

The DEFINE Statement is used to describe the characteristics of Pseudo Parameters which are Function Designators. The actual Function Designators which are being passed are described in the Data Bank. These must agree with the DEFINE Statement in order to ensure correct processing.

The SOURCE option is used to identify the source of the Function Designator. If the source is not specified, the default will be GSE/PCM/RTU.

To specify the characteristics of a Function Designator, used as a Pseudo Parameter of a GOAL Program and whose data is obtained via the LDB FEP or a PCM Uplink FEP, refer to the DEFINE Statement, GOAL On-Board Interface Language, KSC-LPS-OP-033-04.

EXAMPLE

```
BEGIN PROGRAM (A);

PERFORM PROGRAM (B) < AS1 >, < DM2 >;
```

```
BEGIN PROGRAM (B) < X >, < Y >;

DEFINE < X > AS LOAD TYPE ANALOG V;

DEFINE < Y > AS SENSOR TYPE DISCRETE ON/OFF;

VERIFY < X > IS LESS THAN 5 V;

VERIFY < Y > IS ON;
```

In the example, Program A calls Program B. Two Function Designators (< AS1 > and < DM2 >) are passed as parameters. In Program B, the Pseudo Parameters (< X > and < Y >) will be replaced at run time by the parameters which are passed from Program A. The DEFINE Statements must be used in Program B to define the Pseudo Parameters.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. When defining a multi-word digital pattern, the number of words required by the Function Designator must be specified.

B. The keyword MULTI-WORD must be hyphenated.

C. If LOAD is specified, the Function Designator must be a stimulus type. If SENSOR is specified, the Function Designator must be a measurement type. (Refer to Section 16.4, FUNCTION DESIGNATORS.)

D. If a SOURCE is specified, it must be a valid source for the specified Function Designator type. (Refer to KSC-LPS-OP-033-06, Part 1 for valid sources.) If a SOURCE is not specified, the default will be GSE/PCM/RTU.

Legal Function Designators

The DEFINE Statement allows the description of the following Function Designator types:

Analog Double Precision (AMDP)
Analog Measurement (AM)
Analog Stimulus (AS)
Pseudo Analog (PA)
Discrete Measurement (DM)
Discrete Stimulus (DS)
Pseudo Discrete (PD)
Digital Pattern Measurement (DPM)
Digital Pattern Stimulus (DPS)
Pseudo Digital Pattern (PDP)
Multiword Digital Pattern (MWDP)
Floating Point (FP)
Time Homogeneous Data Set (THDS)
Mission Elapsed Time (CDT)
Greenwich Mean Time (GMT)

Countdown Time (CDT)
Julian Time of Year (CDT)
Interval Time (TIMR)
System Interrupts (SI)
Application Pages (PAGE)
Disk File (FILE)
SPA Printer (PRTR)
Special DG Function Key (DGKY)
System Status Discrete (SSA1)
System Status Digital Pattern (SSA2)
PFP Function Key (PFPK)
PFP Light (Indicator) (PFPL)
Date (DATE)
Console (CONS)
Remote Communication (COM)
Console Alarm (ALRM)
Console Printer/Plotter (CPP)
File (PDRR)
DG Function Key (PFK)
LED's (LED)
FEP's (FEP)
Linearization Curves (LS1, LS4, LSS)

SYSTEM TYPE

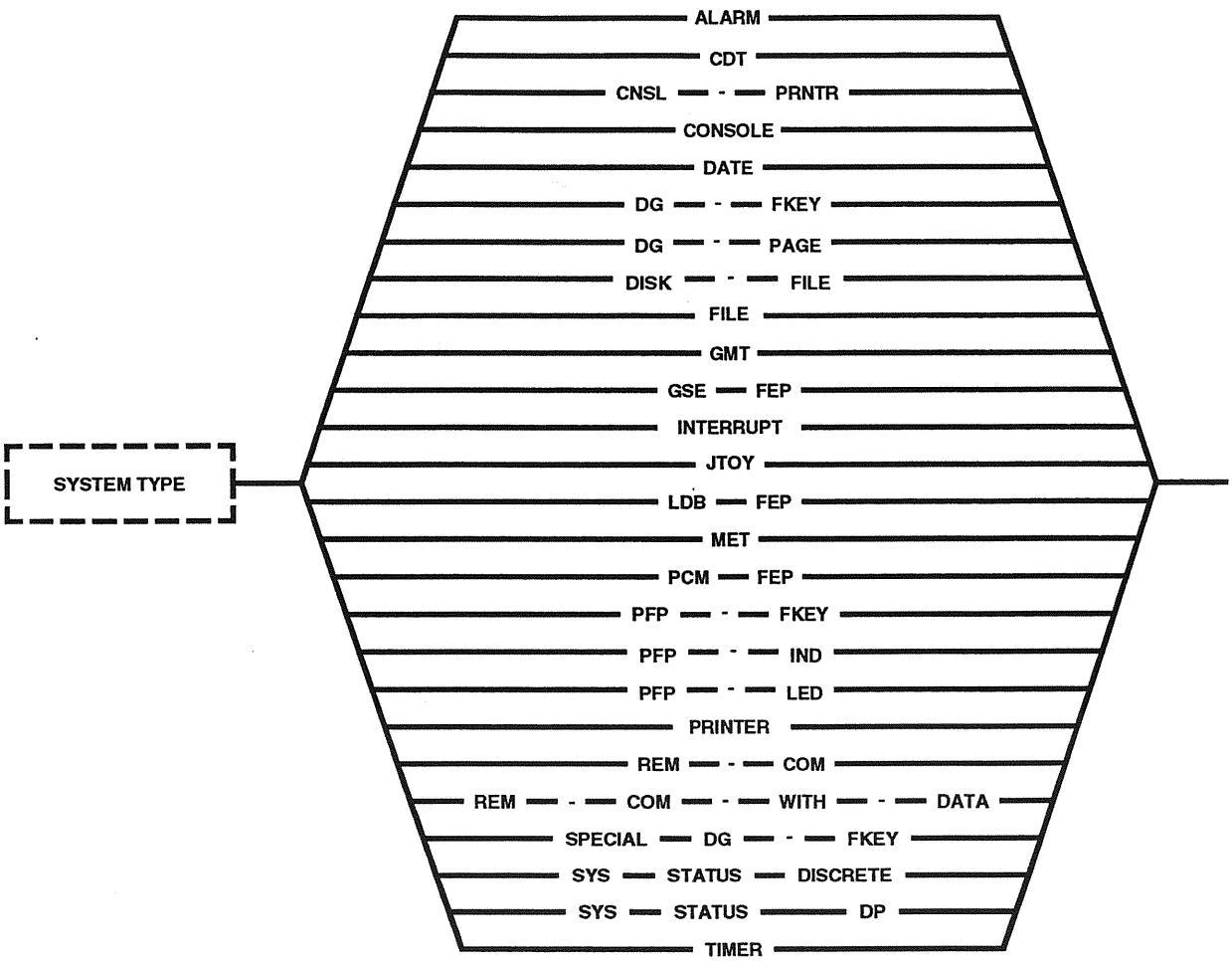


Figure 9-5 System Type

9.3.1 SYSTEM TYPE

Function

The SYSTEM TYPE option is used to specify the characteristics of a System Function Designator that is used as a Pseudo Parameter of a GOAL procedure.

Description

When System Type Function Designators are passed as parameters to a program, the receiving program must DEFINE Pseudo Parameters corresponding to the Function Designator parameters being passed. The System Type Function Designators being passed must agree with the System Type definition in order to ensure correct processing.

Example

```
BEGIN PROGRAM (A);
PERFORM PROGRAM (B) <PAGE-A>;
-----------------------------

  BEGIN PROGRAM (B) <PAGE>;
  DEFINE <PAGE> AS SYSTEM TYPE DG-PAGE;
  RECORD TEXT (MESSAGE) TO <PAGE>;
```

In the example, Program (A) calls Program (B). One Function Designator, <PAGE-A>, is passed as a parameter. In Program (B), the Pseudo Parameter <PAGE> will be replaced at run time by the parameter passed from Program (A).

Restrictions/Limitations

Refer to the following for restrictions and limitations:

The following words must be hyphenated:

DG-PAGE	REM-COM
DG-FKEY	DISK-FILE
PFP-LED	CNSL-PRNTR
PFP-FKEY	SPECIAL DG-FKEY
REM-COM-WITH-DATA	PFP-IND

Legal Function Designators

Mission Elapsed Time (CDT)
Greenwich Mean Time (GMT)
Countdown Time (CDT)
Julian Time of Year (CDT)
Interval Time (TIMR)
System Interrupts (SI)
Application Pages (PAGE)
SPA Printer (PRTR)
Disk File (FILE)
Special DG Function Key (DGKY)
System Status Digital Pattern (SSA2)
System Status Discrete (SSA1)
PFP Function Key (PFPK)
PFP Light (Indicator) (PFPL)
Date (DATE)
Console (CONS)
Remote Communication (COM)
Console Alarm (ALRM)
Console Printer/Plotter (CPP)
Disk File (FILE)
DG Function Key (PFK)
LED's (LED)
GSE FEP (FEP)
LDB FEP (FEP)
PCM FEP (FEP)
Remote Communication (COM)

9.4 RELEASE CONCURRENT STATEMENT

RELEASE CONCURRENT STATEMENT

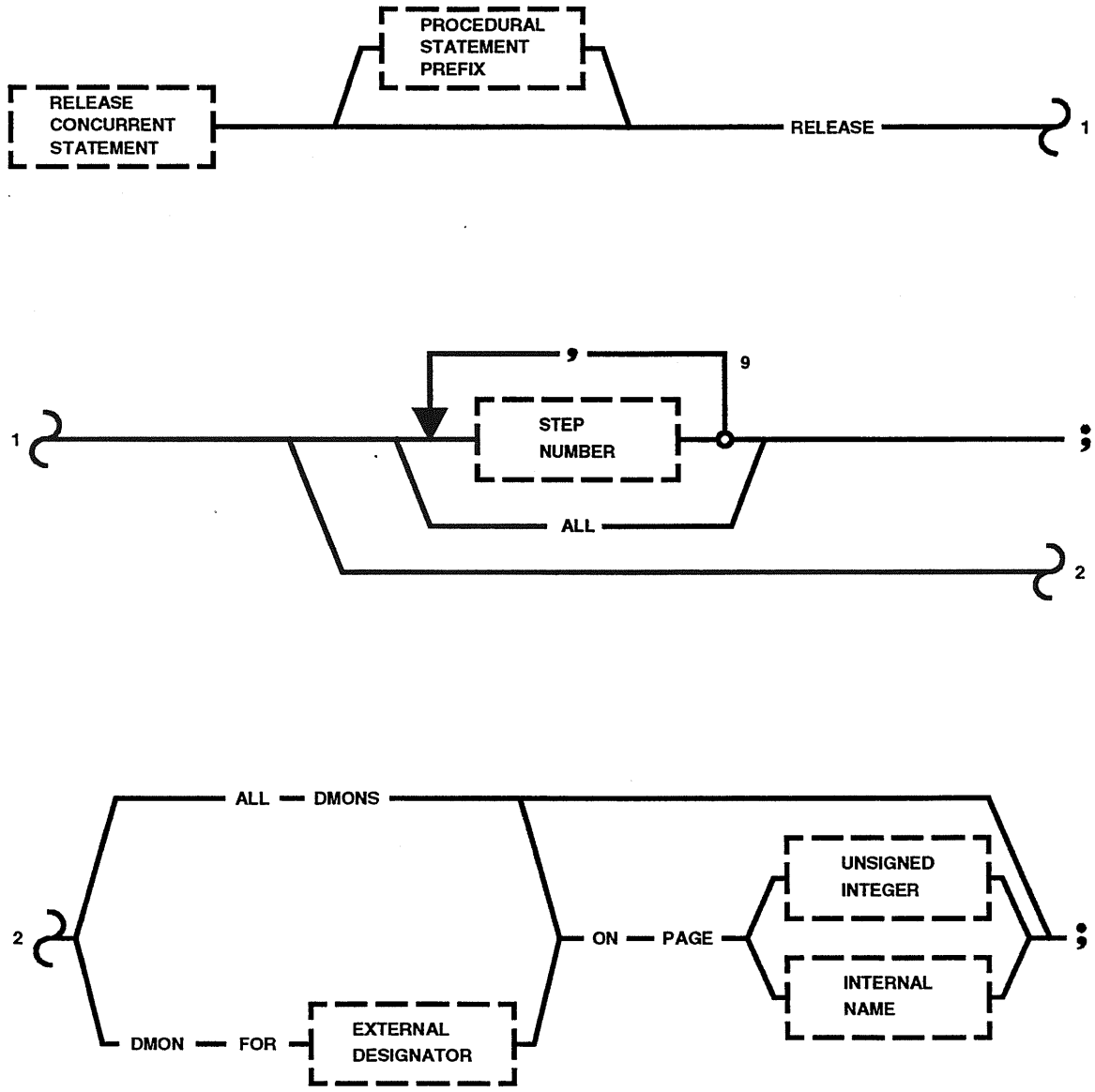


Figure 9-6 RELEASE CONCURRENT Statement

Function

The RELEASE CONCURRENT Statement is used to discontinue the cyclic execution of parallel operations initiated by a CONCURRENT Statement or to terminate DMON for specified measurements initiated by application software.

Description

The RELEASE CONCURRENT Statement is used to discontinue cyclic program tasks. If the task is executing when the RELEASE is requested, it will continue until its normal termination and will not be restarted. If the task is active but not executing, the automatic restart for the task will be discontinued. If a specified task is not active, the RELEASE Statement which references it is ignored.

Two program release options are available on the RELEASE CONCURRENT Statement. If the keyword ALL is specified, all concurrent programs initiated by the procedure are discontinued. If specific Step Numbers are given, only the concurrent programs performed at the Step Numbers referenced are affected.

Examples

```
RELEASE STEP 5, STEP 6, STEP 7;
RELEASE ALL;
```

The first example releases the tasks performed at Steps 5, 6, and 7. The second example releases all tasks initiated by the procedure.

Program Release Execution

The GOAL Executor will stop the specified cyclic tasks from being restarted when the RELEASE Statement is executed.

Description of DMON Release

The RELEASE CONCURRENT Statement is used to discontinue application program initiated DMON displays. When the RELEASE is requested, each of the specified Function Designators is no longer updated on the DMON page and the appropriate line on the page is cleared.

Three DMON release options are available:

A. Release of all active DMON's.

B. Release of all active DMON's on a specific page.

C. Release of DMON for specified External Designators active on a specified page.

If the keywords ALL DMONS only are specified, all active DMONS initiated by application programs are released for all three pages. The keywords ALL DMONS followed by ON PAGE release those DMONS active on the specified page number. The keywords DMON FOR allow release of only those specified External Designators active on the specified page number.

Example

Release of ALL Active DMON's

```
RELEASE ALL DMONS;
```

Release of ALL Active DMON's on Specified Page

```
RELEASE ALL DMONS ON PAGE 3;
```

Release of DMON's for External Designators

```
RELEASE DMON FOR < AM9 >, < DM6 > ON PAGE 2;
```

The first example releases all DMONS initiated by application programs on all three DMON pages. The second example releases all those on page 3 only, while the last example releases only the two specified Function Designators if they are active on page 2.

DMON Release Execution

The GOAL Executor notifies the DMON to stop update and display of the specified measurements.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

If a RELEASE Statement requests that a program which is not active be released, task execution is not affected.

Legal Function Designators

Measurement types for DMON.

THIS PAGE INTENTIONALLY LEFT BLANK.

9.5 PERFORM PROGRAM STATEMENT

PERFORM PROGRAM STATEMENT

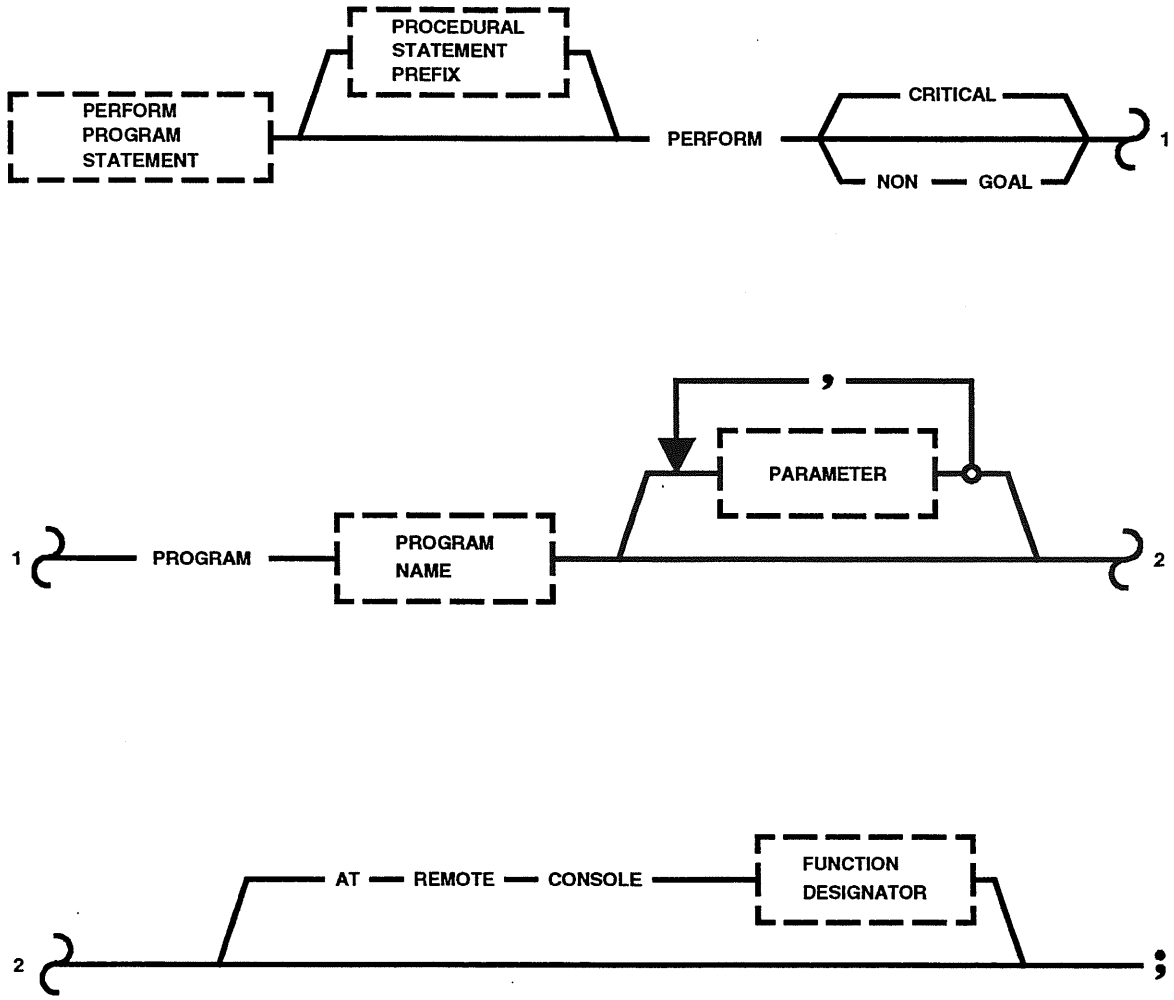


Figure 9-7 PERFORM PROGRAM Statement

Function

The PERFORM PROGRAM Statement is used to initiate execution of a program and pass any required parameters.

Description

The PERFORM PROGRAM Statement causes a program to be executed in series with the calling program. During the time the performed program is executing, execution of the performer (calling program) is suspended. Execution continues with the next sequential statement following the PERFORM Statement after termination of the performed program.

The PERFORM PROGRAM Statement may be used to initiate GOAL or Non-GOAL programs.

The CRITICAL option may be used to request that a GOAL program be placed on a "HIGH PRIORITY" dispatching queue. This means that the procedure gains control of the CPU resources until the critical mode enters an I/O wait or terminates. How often it would be allowed to run would depend upon how many other GOAL procedures were in critical mode and in normal mode. Generally, a critical program is allowed to run every other time a GOAL program is dispatched. At level termination, priority returns to the state prior to the perform. Up to six critical programs may execute concurrently.

The PERFORM PROGRAM Statement allows parameters to be passed to the performed program as well as back to the performer. This capability enables a program to be performed to accomplish testing or data analysis and to pass the results back to the performer.

The REMOTE CONSOLE option may be used to initiate a GOAL or Non-GOAL program at a remote console. Parameter data may be passed to a program performed at a remote console and back to the performer.

PERFORM a GOAL Program

Examples

```
PERFORM PROGRAM (TEST 1);
PERFORM PROGRAM (TEST 2) 5 V, 7, ON (Q3), < DM1 >;
PERFORM CRITICAL PROGRAM (TEST 3);
PERFORM PROGRAM (TEST 4) AT REMOTE CONSOLE < LH2 >;
PERFORM PROGRAM (TEST 2) ON, 25, (Q1), < DM1 >
    AT REMOTE CONSOLE < LH2 >;
```

The first example performs the GOAL program (TEST 1). The second example performs the GOAL program (TEST 2) and passes it parameters. The third example performs the critical program (TEST 3). The fourth example performs the program (TEST 4) at a remote console. The last example performs a program in a remote console with parameter passing.

PERFORM Non-GOAL Programs

Example

```
PERFORM NON GOAL PROGRAM (TEST 6);
PERFORM NON GOAL PROGRAM (TEST 7) (N1), 9V, < AM1 >;
PERFORM NON GOAL PROGRAM (TEST 8)
    AT REMOTE CONSOLE < LO2 >;
PERFORM NON GOAL PROGRAM (TEST 9) 25, (Q1), < DS1 >
    AT REMOTE CONSOLE < LO2 >;
```

The above examples illustrate the options for performing Non-GOAL programs.

Statement Execution

To perform a resident procedure, parameters to be passed will be collected from the performer's Variable Data Area (VDA), and the procedure will be located on the disk. The GOAL Executor will page the active procedure's VDA and the current interpretive code block to bulk memory, obtain the new level from disk and initialize it for execution. This process requires two disk accesses. The name of the procedure will be logged to the PDR along with the mainline name. If the procedure references the SPA printer, a request will be made to the SPA to initialize a spool for report data if a report has not already been started for the task.

If the GOAL procedure is to be performed in a remote console, a Computer-to-Computer (CTC) transaction will be formatted and sent to the GOAL Executor in the remote console. Any required parameters will be transferred via block transfer. The remote GOAL Executor will perform the program and respond via CTC to the initiating GOAL Executor when the program terminates. Any parameters passed to the program in the remote console will be passed back to the performer upon program termination.

To perform a non-GOAL program, parameters to be passed will be collected from the performer's variable data area, and the Executor will enqueue a request to the operating system task initiator. The Operating System obtains the program from disk and executes it. The Operating System will inform the Executor when the program terminates.

Error Processing

On a PERFORM PROGRAM Statement, if the requested program cannot be executed, a Class II error will be processed.

Restrictions/limitations

Refer to the following for restrictions and limitations:

A. The number of Parameters must be equal to the number of Pseudo Parameters specified on the BEGIN PROGRAM Statement of the performed program.

B. The parameters may not exceed 512 words of data

Legal Function Designators

Only Console type Function Designators may be used for remote performs.

Any Function Designator describable on a DEFINE Statement may be passed as a Parameter.

THIS PAGE INTENTIONALLY LEFT BLANK.

9.6 STOP STATEMENT

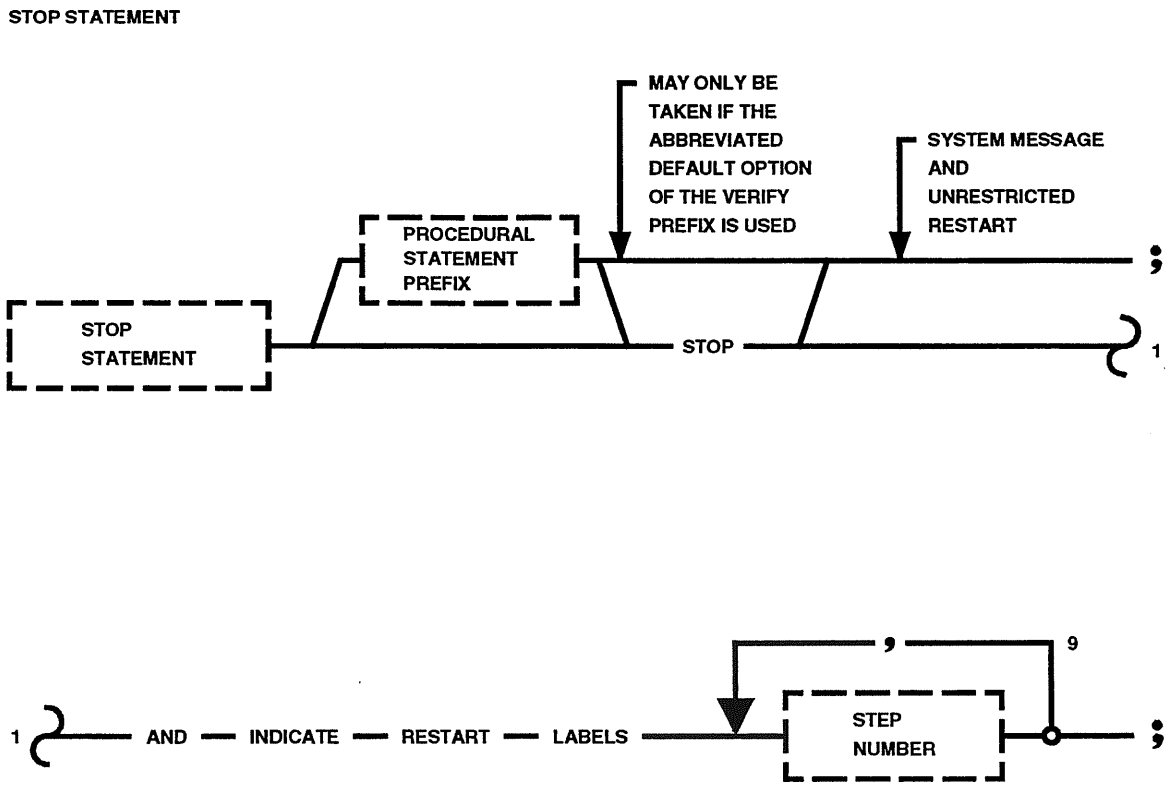


Figure 9-8 STOP Statement

Function

The STOP Statement is used to temporarily halt the execution of a GOAL procedure and enable manual selection of a restart point.

Description

The STOP Statement causes the execution of a procedure to stop, but it does not terminate the procedure. Manual restart options may be selected depending upon the statement option coded. The options of the STOP Statement are:

Unrestricted Restart Option

Restricted Restart Option

Abbreviated Stop Option

Refer to KSC-LPS-OP-033-04 for the on-board interface STOP Statement.

Unrestricted Restart Option

Example

```
STOP;
```

The UNRESTRICTED RESTART option causes the GOAL procedure to stop with no restrictions placed on where the procedure may be resumed. The procedure may be resumed from the point of the STOP by depressing the RESUME PROGRAM Function Key, or the GO TO Keyboard Command may be used to transfer control to any statement in the procedure. The use of the GO TO Keyboard Command with the STOP is not the recommended technique for branching in a procedure, since arbitrary branching could cause serious errors.

Restricted Restart Option

Example

```
STOP AND INDICATE RESTART LABELS STEP 1, STEP 2, STEP 3;
```

The RESTRICTED RESTART option causes the procedure to stop with restart allowed only at preselected points within the procedure. The example illustrates the use of this option. In the example, the procedure may be restarted at Step 1, 2 or 3 following the procedure stop. Upon execution of this statement, a tutorial message will be displayed to PAGE-A requesting the operator to

select a Step Number at which to resume the procedure. The operator must enter the desired Step Number and depress the TRANSMIT RESPONSE Function Key in order to restart the procedure.

Abbreviated Stop Option

Example

```
VERIFY < DM1 > IS ON;
```

The abbreviated STOP option is specified by using the Verify Prefix without the THEN or ELSE options. The effect is the same as if the keywords ELSE RECORD EXCEPTIONS AND STOP had been specified. In the example, if the VERIFY/COMPARISON TEST is unsuccessful, an exception will be recorded to PAGE-A and the procedure will be stopped. The same restart options are available as with the UNRESTRICTED RESTART option.

Statement Execution

Upon execution of any of the GOAL STOP options, the system will display a tutorial message and execution of the procedure will be stopped. The GOAL Executor will resume execution of the procedure based upon operator keyboard action. When restart labels are specified, those indicators will be output to PAGE-A via a prompt and one label must be entered to restart the procedure.

Error Processing

If a STOP AND INDICATE RESTART LABELS is executed when a prompt is already outstanding on PAGE-A, a Class II error will result. If the operator enters an incorrect label in response to the prompt, it will not be accepted. The prompt will be displayed again, and the operator will be allowed to select another restart point.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The GOAL STOP option is not allowed as the last statement in a REPEAT loop.

B. The operator must not branch into a REPEAT loop to resume execution following a STOP.

Legal Function Designators

None

9.7 TERMINATE STATEMENT

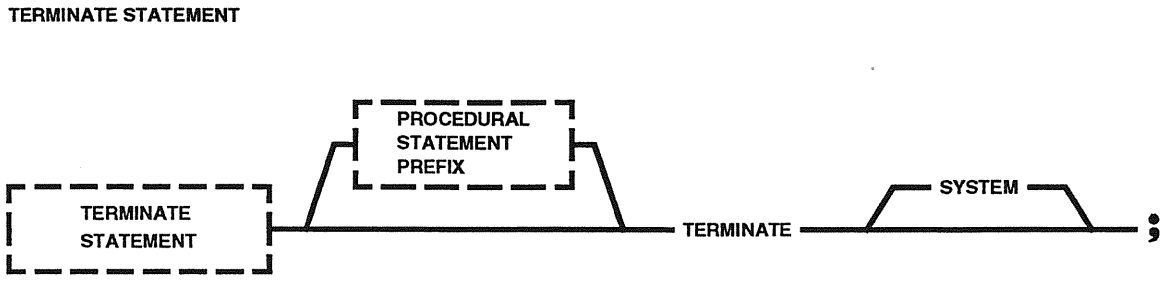


Figure 9-9 TERMINATE Statement

Function

The TERMINATE Statement is used to discontinue the execution of a GOAL procedure.

Description

The TERMINATE Statement will stop the execution of the program in which it occurs and purge it from the system. If the program being terminated is performed (not concurrent perform) by another GOAL program, control will be returned to the statement in the performer following the PERFORM Statement. If the program being terminated is concurrently performed using the cyclic option, it will be restarted at the beginning of its next cyclic interval.

When the TERMINATE SYSTEM option is used on any level, the task in which the TERMINATE SYSTEM occurs will be terminated. Any cyclic concurrencies started by this task will be released (refer to RELEASE CONCURRENT Statement for further explanation).

Example

```
TERMINATE;
TERMINATE SYSTEM;
```

Statement Execution

The steps taken to terminate a level of a GOAL Task are:

A. Purge all the interrupt block entries belonging only to this level by removing them from the Function Designator type lists.

B. Disable any CDT, MET, or TIMER WHEN INTERRUPT, by requesting timer services to cancel time delays.

C. Determine if any parameters are to be returned to the caller and collect them from the Variable Data Area (VDA).

D. Retrieve the calling procedure's VDA and interpretive code block from bulk memory.

E. Return control to the previous level as specified by the PERFORM (supplying parameters as required).

The steps taken to terminate a mainline (Level 1) procedure are:

A. Determine if this task is a cyclic task. If so, reinitialize the interpretive code address and request Buffer Management to obtain the first interpretive code blocks from disk. If the time between cyclic executions has elapsed, restart the procedure; otherwise, delay until the time interval has elapsed.

B. If report data is output to the SPA printer by this task, the SPA will be requested to terminate the spool and output the report.

C. The control block for this task will be housekept.

D. A request will be made to the task initiator/terminator to release the system resources.

If the SYSTEM option is specified on any level, any concurrencies started by this task will be released (as described in the RELEASE CONCURRENT Statement) and the task in which the TERMINATE SYSTEM occurs will be terminated.

To terminate the task, active level control blocks will be housekept. If a report is generated, the SPA will be requested to terminate the spool and print the report. The GOAL control block for this task will be housekept, and the system resources will be released.

Whenever a procedure or a task terminates, the termination will be logged to the CDBFR.

Error Processing

No errors are processed on the TERMINATE Statement.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

The task in which TERMINATE SYSTEM appears will be terminated, and all other tasks started by the task will be released.

Legal Function Designators

None

10. EXECUTION CONTROL STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The following Execution Control Statements will be discussed in the indicated sections:

A. DELAY STATEMENT (Section 10.1)

The DELAY Statement causes a delay in procedure execution for a specified time.

B. CONTINUE STATEMENT (Section 10.2)

The CONTINUE Statement provides a no-operation capability which results in continued execution of the procedure.

C. ACTIVATE/INHIBIT TABLE STATEMENT (Section 10.3)

This Statement is used to activate or inhibit the processing of Function Designators via table rows.

The Execution Control Statements are used to control the execution of the procedure in which they are executed.

THIS PAGE INTENTIONALLY LEFT BLANK.

10.1 DELAY STATEMENT

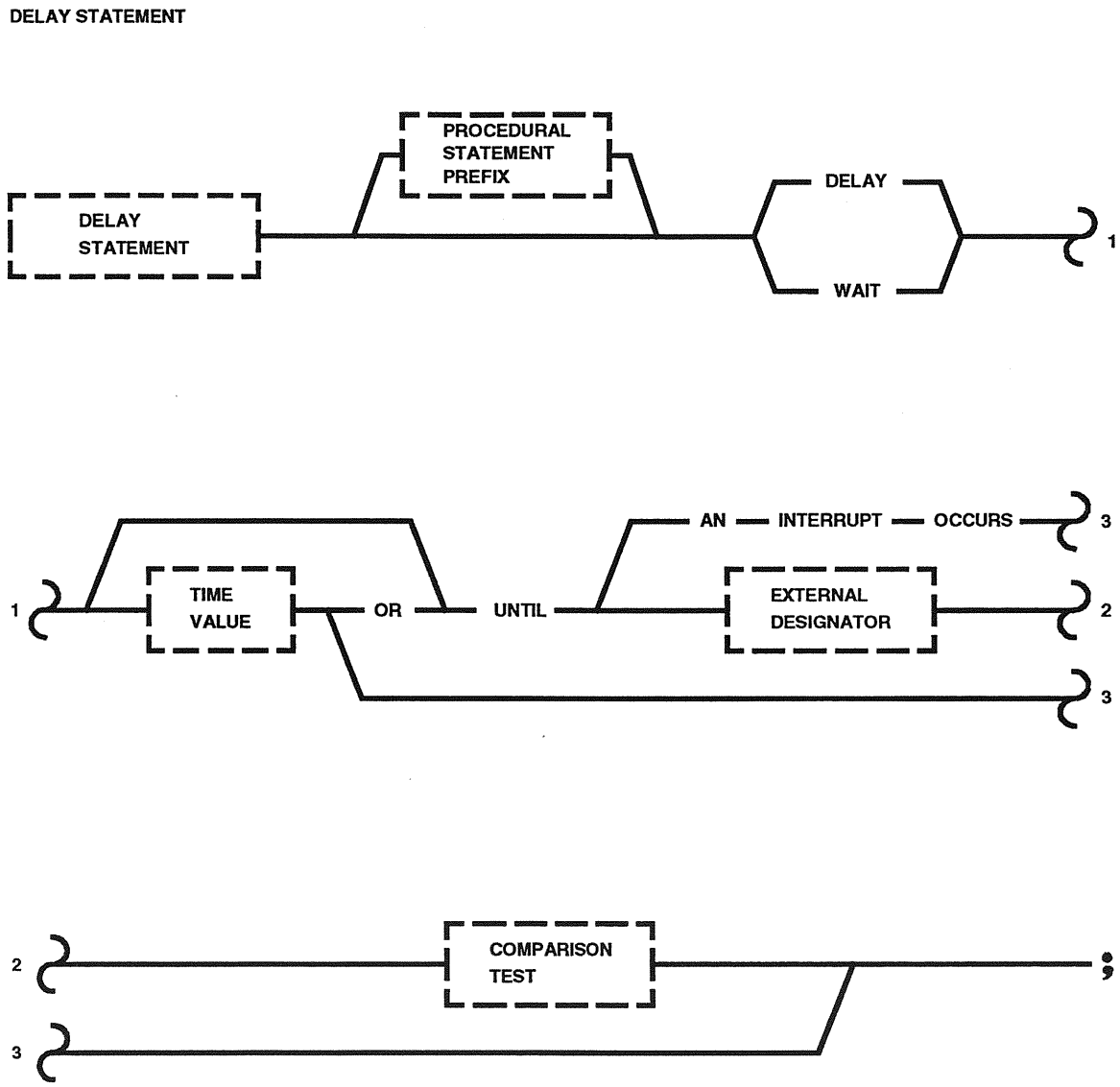


Figure 10-1 DELAY Statement

Function

The DELAY Statement is used to delay execution of a procedure for a time interval or until an external condition is satisfied or until an interrupt occurs.

Description

Use of a DELAY Statement will cause execution of a procedure to be temporarily halted until the options specified on the DELAY Statement are satisfied. During a delay, none of the statements in the procedure will be executed - including interrupts. The DELAY and the WAIT verbs are synonymous. The following options are available:

A. INTERVAL DELAY

An interval delay is specified using DELAY followed by a Time Value only. It causes execution of the procedure to be delayed until the Time Value specified has elapsed.

B. CONDITIONAL DELAY

A conditional delay is specified using the keyword, UNTIL, followed by an EXTERNAL DESIGNATOR/COMPARISON TEST. It causes execution of the procedure to be delayed until the conditions of the test are satisfied.

The name of the CDT count/hold status pseudo Function Designator and its values can be found in the INTERRUPT PROCESSING STATEMENTS section of this document.

C. INTERRUPT Option

An interruptable delay is specified using the keywords, UNTIL AN INTERRUPT OCCURS. It causes execution to be delayed until a currently active interrupt is received.

D. INTERVAL and CONDITIONAL DELAY

When both an interval and a conditional delay are specified, execution of the procedure will be delayed until the time interval has elapsed, or until the condition is satisfied, whichever occurs first.

E. INTERVAL and INTERRUPT Option

When both an interval and an interruptable delay are specified, execution of the procedure will be delayed until the time interval has elapsed, or until a currently active interrupt is received, whichever occurs first.

Interval Delay Option

Examples

```
DELAY 1 MIN 15 SEC;
DELAY (Time1);
```

Conditional Delay Option

Example

```
WAIT UNTIL < DM1 > IS ON;
```

Interrupt Delay Option

Example

```
DELAY UNTIL AN INTERRUPT OCCURS;
```

Interval and Conditional Delay Option

Examples

```
DELAY 5 SEC OR UNTIL  < DM1 > IS ON;
DELAY 500 MS OR UNTIL < AM1 >,
                      < AM2 > ARE LESS THAN 5V;
```

Interval and Interrupt Delay Option

Example

```
DELAY 10 SEC OR UNTIL AN INTERRUPT OCCURS;
```

Statement Execution

If a simple time delay is specified, the GOAL Executor will suspend the execution of the procedure until the Time Value is exhausted.

If only an UNTIL is specified, the GOAL Executor will perform an iterative check on the entire comparison until it is successful. After each check a task switch will be allowed.

If both an interval and a condition are specified, the processing will be as follows:

A. Check the condition, and if satisfied, go to the next statement.

B. Request the time delay.

C. Force a task switch to allow other active concurrencies to execute.

D. Check the conditions. If all are satisfied, cancel the time delay, and go to the next statement.

E. Check time delay, and if exhausted, go to the next statement.

F. Repeat steps 3, 4, and 5.

Error Processing

Errors encountered obtaining the value of the specified Function Designator for the DELAY UNTIL option will be handled as Class III errors.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. During a DELAY, none of the statements within the program in which the DELAY appears will be executed.

B. Interrupts will be processed during delays only if the interrupt option is used.

C. The time resolution of the DELAY Statement may be affected by operating system processing load.

D. Time intervals of GMT, CDT, JTOY and MET may not be specified on the DELAY Statement as TIME VALUE.

E. A delay which has been interrupted will not resume execution after an ACTIVE INTERRUPTS ON THIS LEVEL AND RETURN command is executed. Instead, the procedure will return to the statement following the DELAY which was interrupted.

Legal Function Designators

Analog Measurements (AM)
Analog Stimuli (AS)
Countdown Time (CDT)
Digital Pattern Measurements (DPM)
Digital Pattern Stimuli (DPS)
Discrete Measurements (DM)
Discrete Stimuli (DS)
Greenwich Mean Time (GMT)
Julian Time of Year (CDT)
Interval Timer (TIMR)
Mission Elapsed Time (CDT)
PFP Lights (PFPL)
Pseudo Analogs (PA)
Pseudo Digital Patterns (PDP)
Pseudo Discretes (PD)

10.2 CONTINUE STATEMENT

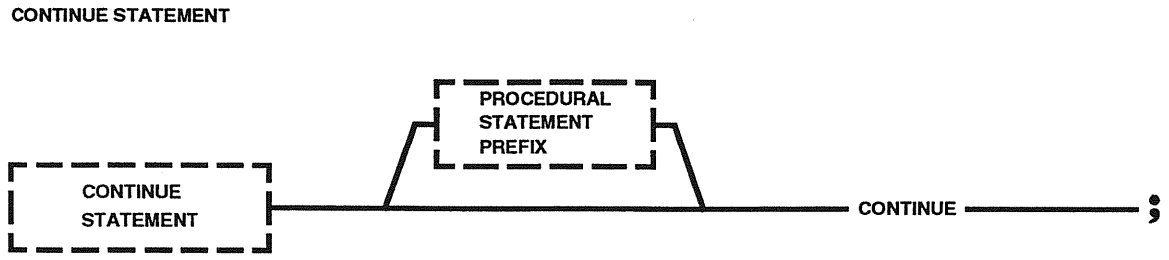


Figure 10-2 CONTINUE Statement

Function

The CONTINUE Statement provides the capability to perform a no operation.

Description

The CONTINUE Statement is a procedural statement which performs no operation. It may appear anywhere in the operating steps of a procedure to provide a convenient branch point. It may be used at the end of a repeat loop.

Example

```
CONTINUE;
AFTER < CDT > IS -20 MIN CDT, CONTINUE;
```

Statement Execution

The CONTINUE Statement is an executable statement that does nothing.

THIS PAGE INTENTIONALLY LEFT BLANK.

10.3 ACTIVATE/INHIBIT TABLE STATEMENTS

ACTIVATE TABLE STATEMENT

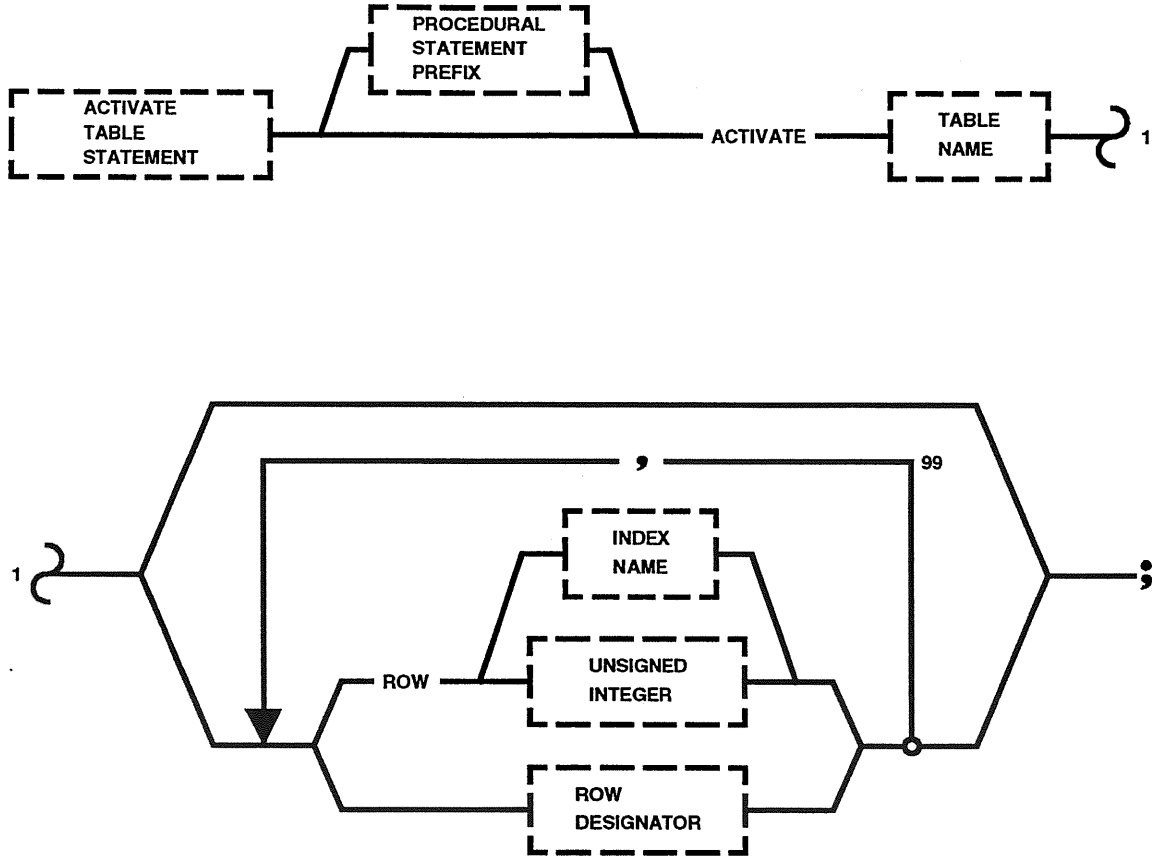


Figure 10-3 ACTIVATE TABLE Statement

INHIBIT TABLE STATEMENT

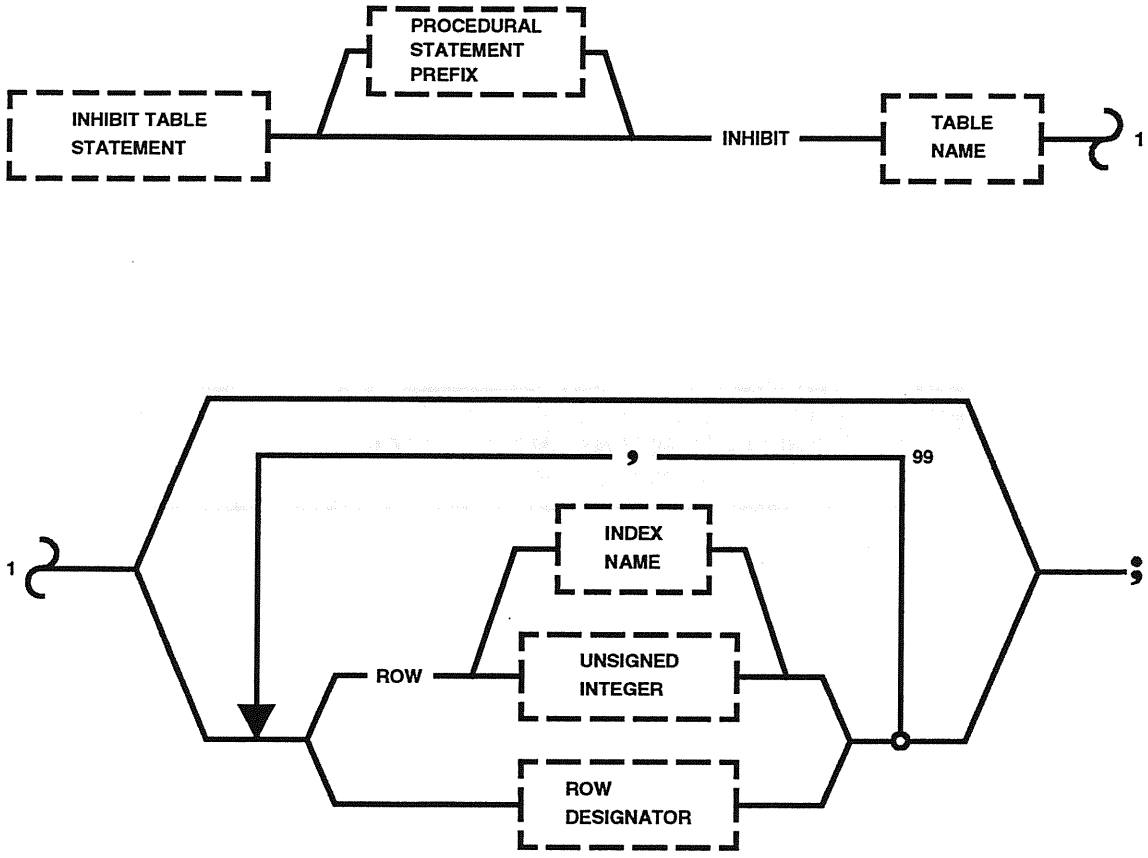


Figure 10-4 INHIBIT TABLE Statement

THIS PAGE INTENTIONALLY LEFT BLANK.

FUNCTION

The ACTIVATE and INHIBIT TABLE Statements are used to assign an active or inhibited status to all or selected Row Designators of a table only when referenced as Table Functions.

DESCRIPTION

Rows of tables may be assigned either an active or an inhibited status. The status of a row affects only the use of Row Designators referenced as Table Functions. It does not affect the use of data in the table. When Row Designators are referenced as Table Functions, the Row Designators for inhibited rows will not be used for input, output, or testing.

Individual rows of a table may be activated or inhibited by specifying Row Designators or by specifying the keyword ROW, followed by an Unsigned Integer or Index Name. If no individual Rows or Row Designators are specified following Table Name, the entire table is inhibited or activated.

ACTIVATE/INHIBIT Entire Table Option

Examples

```
ACTIVATE (STABLE1);
INHIBIT (QTABLE1);
```

ACTIVATE/INHIBIT Selected Rows Option

Examples

```
ACTIVATE (STABLE1) < DM1 >;
ACTIVATE (STABLE1) ROW 1, ROW 2, ROW 3;
INHIBIT (STABLE1) ROW (N1);
INHIBIT (STABLE1) < DM1 >,
                  < DM2 >,
                  < DM3 >;
```

Application

Example

```
DECLARE STATE TABLE (STABLE1) WITH 3 ROWS and 1 COLUMN WITH
    ENTRIES < DS1 >, ON,
            < DS2 >, OPEN
            < DS3 >, OFF;
```

```
INHIBIT (STABLE1) ROW 2;
SET (STABLE1) FUNCTIONS TO OFF;
```

Execution of the SET Statement in the example results in < DS1 > and < DS3 > being SET to OFF while < DS2 > is unaffected by the operation.

Statement Execution

A status bit (active or inhibited) is assigned in the VDA for each Row Designator in each table. The ACTIVATE and INHIBIT TABLE Statements merely change the value of the status bits. One word of VDA is utilized for each group of 16 Row Designators in the table.

Error Processing

If an Index Name used to specify a ROW number causes addressing beyond the boundaries of the table, a Class III error will result.

Operations referencing an inhibited row will not be executed. No error message will be generated and execution will continue.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. All rows of a table default to active when the procedure is initially performed.

B. A row may not be repeated in single ACTIVATE or INHIBIT TABLE Statements.

C. An attempt to ACTIVATE an active row or to INHIBIT an inhibited row has no effect.

Legal Function Designators

Only those Function Designators declared in the table referenced may be used.

11. CCMS CONTROL STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The following CCMS Control Statements will be discussed in the indicated sections:

A. TIMER CONTROL STATEMENT (Section 11.1)

The TIMER CONTROL Statement controls Count Down Time, Mission Elapsed Time, and the Interval Timer.

B. ACTIVATE SYSTEM STATEMENT (Section 11.2)

C. INHIBIT SYSTEM STATEMENT (Section 11.2)

The ACTIVATE and INHIBIT SYSTEM Statements control FEP measurement and command processing.

D. CHANGE STATEMENT (Section 11.3)

The CHANGE Statement is used to alter various FEP measurement processing parameters.

E. LOAD SAFING SEQUENCE STATEMENT (Section 11.4)

The LOAD SAFING SEQUENCE Statement is used to load a sequence of events to be executed in the event of a CDBFR failure.

F. ACTIVATE PLOT STATEMENT (Section 11.5)

G. INHIBIT PLOT STATEMENT (Section 11.6)

The ACTIVATE and INHIBIT PLOT Statements are used to start and stop plotting of data items on the system plotter.

H. CHANGE ANALOG PLOT STATEMENT (Section 11.7)

The CHANGE ANALOG PLOT Statement is used to change high and/or low limits for previously activated analog plots.

I. ACTIVATE CRITICAL MODE STATEMENT (Section 11.8)

J. INHIBIT CRITICAL MODE STATEMENT (Section 11.9)

The ACTIVATE and INHIBIT CRITICAL MODE Statements are used to enter and exit high priority (critical) execution, respectively.

The CCMS Control Statements are used to control the operation of various parts of CCMS.

THIS PAGE INTENTIONALLY LEFT BLANK.

11.1 TIMER CONTROL STATEMENT

TIMER CONTROL STATEMENT

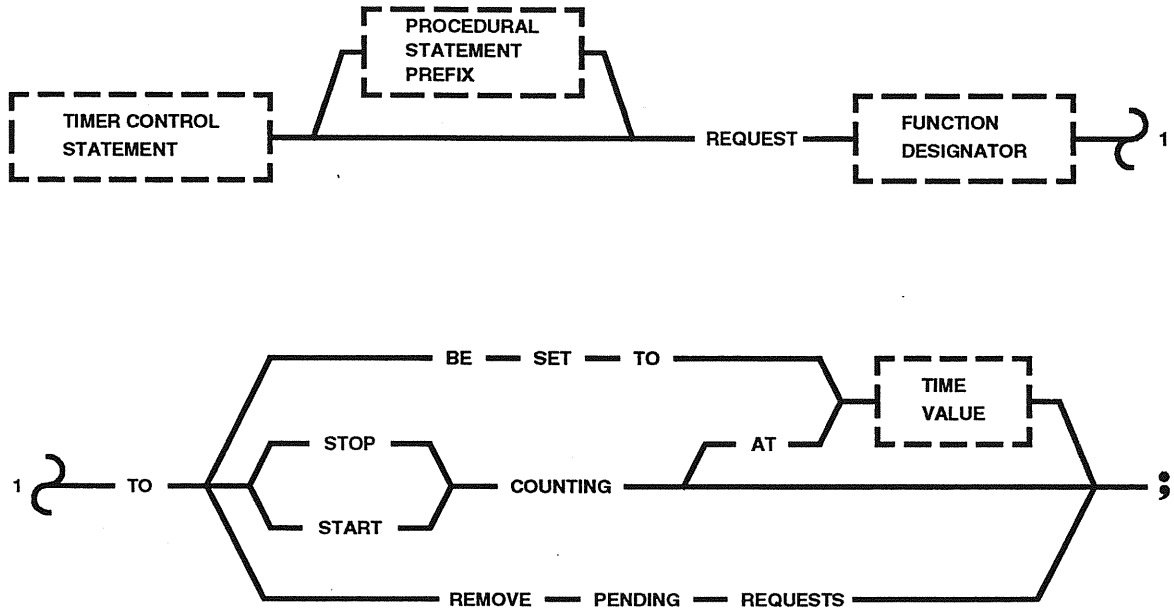


Figure 11-1 TIMER CONTROL Statement

Function

The TIMER CONTROL Statement is used to control the Count Down Time, Mission Elapsed Time and Interval Timer.

Description

The TIMER CONTROL Statement may be used to:

A. Start/Stop Count Down Time (CDT)

B. Start/Stop Mission Elapsed Time (MET)

C. Set Count Down Time, Mission Elapsed Time or the Interval Timer

D. Cancel pending requests to control Count Down Time or Mission Elapsed Time

The SET option may be used to set CDT, MET, or the Interval Timer. The Interval Timer is a clock maintained in console memory by the CCMS Operating System. Each task may have one timer. When the Interval Timer is set to a positive value, it is decremented once a second to zero. The Timer may be read or verified to control the sequencing of events. When the Timer counts to zero an interrupt may be generated to a GOAL procedure.

The START option is used to start Count Down Time and Mission Elapsed Time. These times may be started at a specified GMT by specifying a Time Value, or started immediately.

The STOP option is used to stop Count Down Time and Mission Elapsed Time. If a time value is specified, it must be CDT or MET. Count Down Time or Mission Elapsed Time will be stopped when it reaches the specified value; otherwise, it will be stopped immediately.

NOTE: *To ensure that MET stops/halts at a millisecond zero of the MET timing train, the specific time part of the statement should be used rather than the default STOP option; e.g.,*
Use: REQUEST < METIME > TO STOP COUNTING AT 5 HR 31 MIN 16 SEC MET;
Rather than: REQUEST < METIME > TO STOP COUNTING;

The DEFAULT option halts the MET immediately and will very rarely halt at the millisecond zero of MET. This option might cause problems later.

The option to REMOVE PENDING REQUESTS is used to cancel pending CDT start and stop requests. It is possible to have one pending start request and one pending stop request. If a start request is pending and another request to start CDT or MET at a GMT is executed, then the second request overrides the first. Stop requests may override pending stop requests in the same way.

Set Option

Examples

```
REQUEST < METIME > TO BE SET TO -30 SEC MET;
REQUEST < CDT > TO BE SET TO -20 SEC CDT;
REQUEST < CDT > TO BE SET TO (TIME1);
REQUEST < TIMER > TO BE SET TO 10 SEC;
```

START/STOP Options

Examples

```
REQUEST < CDT > TO START COUNTING;
REQUEST < CDT > TO START COUNTING AT 8 HR 30 MIN GMT;
REQUEST < CDT > TO STOP COUNTING;
REQUEST < METIME > TO START COUNTING;
```

REMOVE PENDING REQUESTS Option

Examples

```
REQUEST < CDT > TO REMOVE PENDING REQUESTS;
REQUEST < METIME > TO REMOVE PENDING REQUESTS;
```

Statement Execution

When the statement references the CDT or MET Function Designator, the GOAL Executor will request the Timing Generator PBIC, via the CDBFR, to perform the requested function.

If the Timer Function Designator is specified on the TIMER CONTROL Statement, the CCMS Operating System is requested to initialize the software timer and decrement it once a second.

Error Processing

If a negative Time Value is specified for the Time, or if the GOAL Executor cannot communicate with the Timing Generator PBIC, a Class III error will be processed.

If the Interval Timer is set to less than 1 second, a TIMER NOT ACTIVE run time error will be generated.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The Interval Timer may only be SET. The START and STOP options may not be used with the Timer Function Designator.

B. The Interval Timer must not be set to less than 1 second. If it is set to less than 1 second, a 'TIMER NOT ACTIVE' run time error will be generated.

C. Interval Timer Accuracy: The Interval Timer is decremented once each second. Therefore, fractions of a second should not be used when referencing the Interval Timer. If the system is busy, the timer may be decremented less often and be less accurate.

D. CDT and MET Accuracy: CDT and MET are incremented once each second. Therefore, fractions of a second should not be used when referencing CDT or MET. Also, CDT and MET are synchronized to millisecond zero of GMT. This means that if CDT or MET are requested to START immediately, it will not start until the millisecond fraction of GMT is 0, which may be almost 1 second later. In addition, requests to ZERO CDT or MET at a given GMT should not specify GMT values with millisecond fractions, since CDT or MET will not start until millisecond zero.

E. The maximum value which can be specified for Timer Function Designators is 32,767 seconds (9 HR, 6 MIN, 7 SEC).

F. CDT may only be stopped at a CDT Time Value and not a GMT value. MET may only be stopped at a MET Time Value and not a GMT value.

G. CDT may be started at a GMT Time Value but not a CDT value. MET may be started at a GMT Time Value but not a MET value.

H. CDT is immediately stopped when referenced in the SET option.

Legal Function Designators

Interval Timer (TIMR)
Countdown Time (CDT)
Mission Elapsed Time (CDT)

11.2 ACTIVATE/INHIBIT SYSTEM STATEMENTS

ACTIVATE SYSTEM STATEMENT

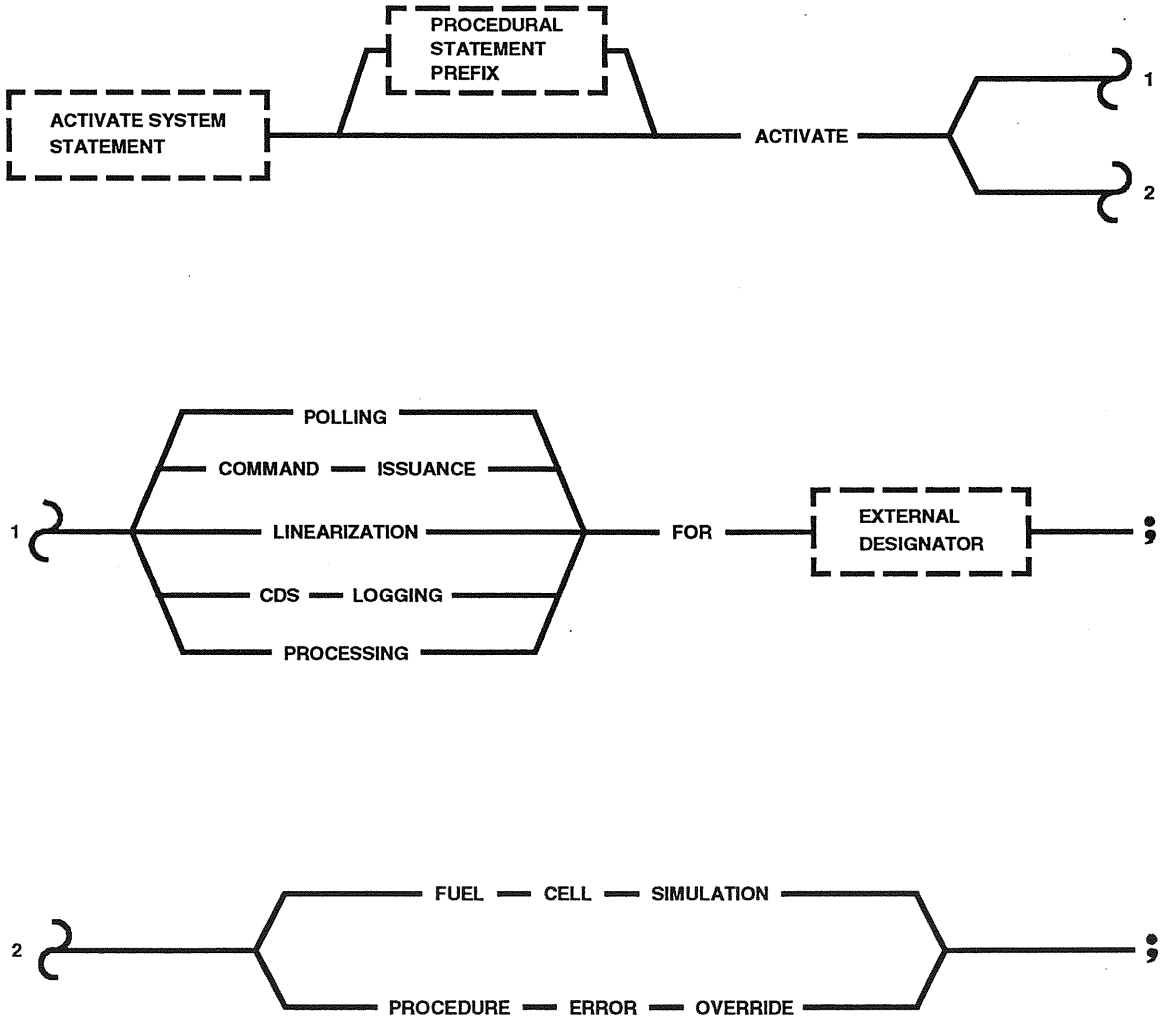


Figure 11-2 ACTIVATE SYSTEM Statement

INHIBIT SYSTEM STATEMENT

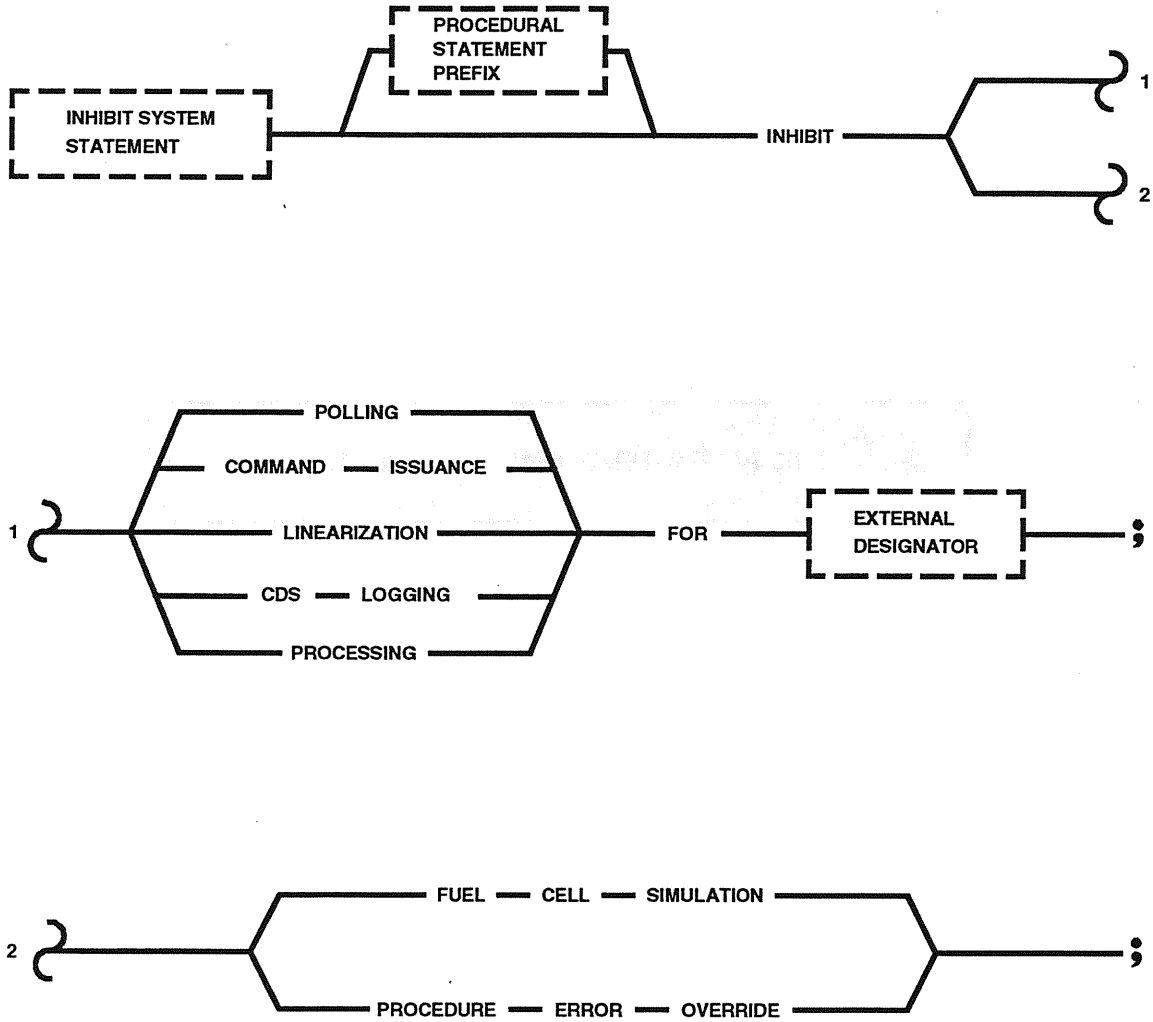


Figure 11-3 INHIBIT SYSTEM Statement

THIS PAGE INTENTIONALLY LEFT BLANK.

Function

The ACTIVATE and INHIBIT SYSTEM Statements are used to control measurement and command processing functions of the FEP's.

Description

The ACTIVATE and INHIBIT Statements allow the following measurement and command processing parameters in the FEP's to be activated or inhibited:

A. CDS LOGGING

The logging of stimuli and measurement changes and exceptions to CDS may be activated or inhibited.

B. COMMAND ISSUANCE

The issuance of commands may be activated or inhibited for individual stimulus type Function Designators. When a command is issued which is inhibited, the FEP will reject the command.

C. POLLING

When POLLING is activated, measurement data is periodically sampled from the HIM's in the case of the GSE FEP's or acquired from the PCM interface in the case of the downlink PCM FEP's. When POLLING is inhibited, new measurement data will not be obtained by the FEP; consequently, measurement values will not be updated in the CDBFR. Inhibiting POLLING also automatically inhibits processing.[1]

D. PROCESSING

When PROCESSING is activated, the FEP performs the following functions on the measurements acquired: significant change checking, exception monitoring, and linearization. When PROCESSING is inhibited, these functions are not performed. Inhibiting PROCESSING does not automatically inhibit POLLING.[1]

1. Refer to Appendix B for information about problems incurred for re-reporting GSE exception conditions.

E. PROCEDURE ERROR OVERRIDE

When PROCEDURE ERROR OVERRIDE is activated and a Class III error occurs, the appropriate error message is written to PAGE-A, and procedure execution continues with the next sequential statement. If PROCEDURE ERROR OVERRIDE is inhibited when a Class III error occurs, the error is processed normally; i.e., the error message is written to PAGE-A and execution of the procedure is stopped.

F. FUEL CELL SIMULATION

When FUEL CELL SIMULATION is activated, the CITE GSE FEP will operate the CITE Power Supply Assembly, such that to the cargo it appears as if a fuel cell is the source of its DC power.

G. LINEARIZATION

When LINEARIZATION is activated, the FEP performs linearization of the specified analog measurement(s) using the criteria in its Linearization Data Table. It then stores the resultant data in the CDBFR.

Examples

```
ACTIVATE POLLING FOR < AM1 >
                     < DM2 >;
INHIBIT COMMAND ISSUANCE FOR < DS1 >;
```

The first example causes the FEP to start polling < AM1 > and < DM2 >. The second example causes the FEP to reject all requests to issue the discrete stimulus, < DS1 >.

Statement Execution

The GOAL Executor transmits requests to the appropriate FEP to ACTIVATE or INHIBIT the specified parameters via the CDBFR. If many Function Designators are referenced on the statement, the requests will be transmitted one at a time. Execution will not continue to the next statement until all the Function Designators have been processed.

Error Processing

Rejection of a request by the FEP to update a processing Parameter results in a Class III error.

Failure to communicate across the CDBFR will result in a Class III error.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. ACTIVATE/INHIBIT POLLING is not valid for individual PCM FEP measurements.

B. Requests to ACTIVATE or INHIBIT POLLING in the ACTIVATE/INHIBIT SYSTEM Statements are only valid from the Master or Integration Console.

C. The ACTIVATE/INHIBIT SYSTEM Statements (with the exception of COMMAND ISSUANCE and CDS LOGGING) are not valid for Function Designators which use the LDB.

D. Analog Measurement Function Designators must be used with the Linearization option.

E. Function Designators which are pseudo parameters may not be used with the LINEARIZATION option.

F. The External Designator specified with the LINEARIZATION option must have an associated curve.

TABLE 11-1 LEGAL FUNCTION DESIGNATORS ACTIVATE/INHIBIT SYSTEM STATEMENTS (1 OF 2)

STATEMENT	TYPE	SOURCE	CLASS
POLLING	ANALOG MEASUREMENT (AM), DISCRETE MEASUREMENT (DM) DIGITAL PATTERN MEASUREMENT (DPM)	GSE, RTU	N/A
COMMAND ISSUANCE	ANALOG STIMULUS (AS), DISCRETE STIMULUS (DS) DIGITAL PATTERN STIMULUS (DPS), DIGITAL PATTERN STIMULUS W/DTA (DPSD) BUS TERMINAL UNIT (BTU), SIO ANALOG STIMULUS (ASS), SIO DISCRETE STIMULUS (DSS), SIO DIGITAL PATTERN STIMULUS (DPSS),	GSE, RTU. UPL	N/A
		LDB	MDM MEC EIU PCM PDI FLX SCA
LINEARIZATION	ANALOG MEASUREMENT (AM)	GSE, PCM, RTU	N/A
PROCESSING	ANALOG MEASUREMENT (AM), DISCRETE MEASUREMENT (DM) DIGITAL PATTERN MEASUREMENT (DPM), ANALOG MEASUREMENT DOUBLE PRECISION (AMDP), MULTI-WORD DIGITAL PATTERN (MWDP), FLOATING POINT (FP)	GSE, PCM RTU,	N/A

TABLE 11-1 LEGAL FUNCTION DESIGNATORS ACTIVATE/INHIBIT SYSTEM STATEMENTS (2 OF 2)

STATEMENT	TYPE	SOURCE	CLASS
CDS LOGGING	ANALOG MEASUREMENT (AM), ANALOG STIMULUS (AS), DISCRETE MEASUREMENT (DM), DISCRETE STIMULUS (DS), DIGITAL PATTERN MEASUREMENT (DPM), DIGITAL PATTERN STIMULUS (DPS), ANALOG MEASUREMENT DOUBLE PRECISION (AMDP), DIGITAL PATTERN STIMULUS W/DATA (DPSD), MULTI-WORD DIGITTAL PATTERN (MWDP), FLOATING POINT (FP), DATA SET (DSFD), BUS TERMINAL UNIT (BTU), SIO ANALOG MEASUREMENT (AMS), SIO ANALOG STIMULUS (ASS), SIO DISCRETE MEASUREMENT (DMS), SIO DISCRETE STIMULUS (DSS), SIO DIGITAL PATTERN MEASUREMENT (DPMS), SIO DIGITAL PATTERN STIMULUS (DPSS)	NONE, GSE, PCM RTU, UPL	N/A
		LDB	MDM, MEC, EIU, PCM, PDI, FLX, SCA

THIS PAGE INTENTIONALLY LEFT BLANK.

11.3 CHANGE STATEMENT

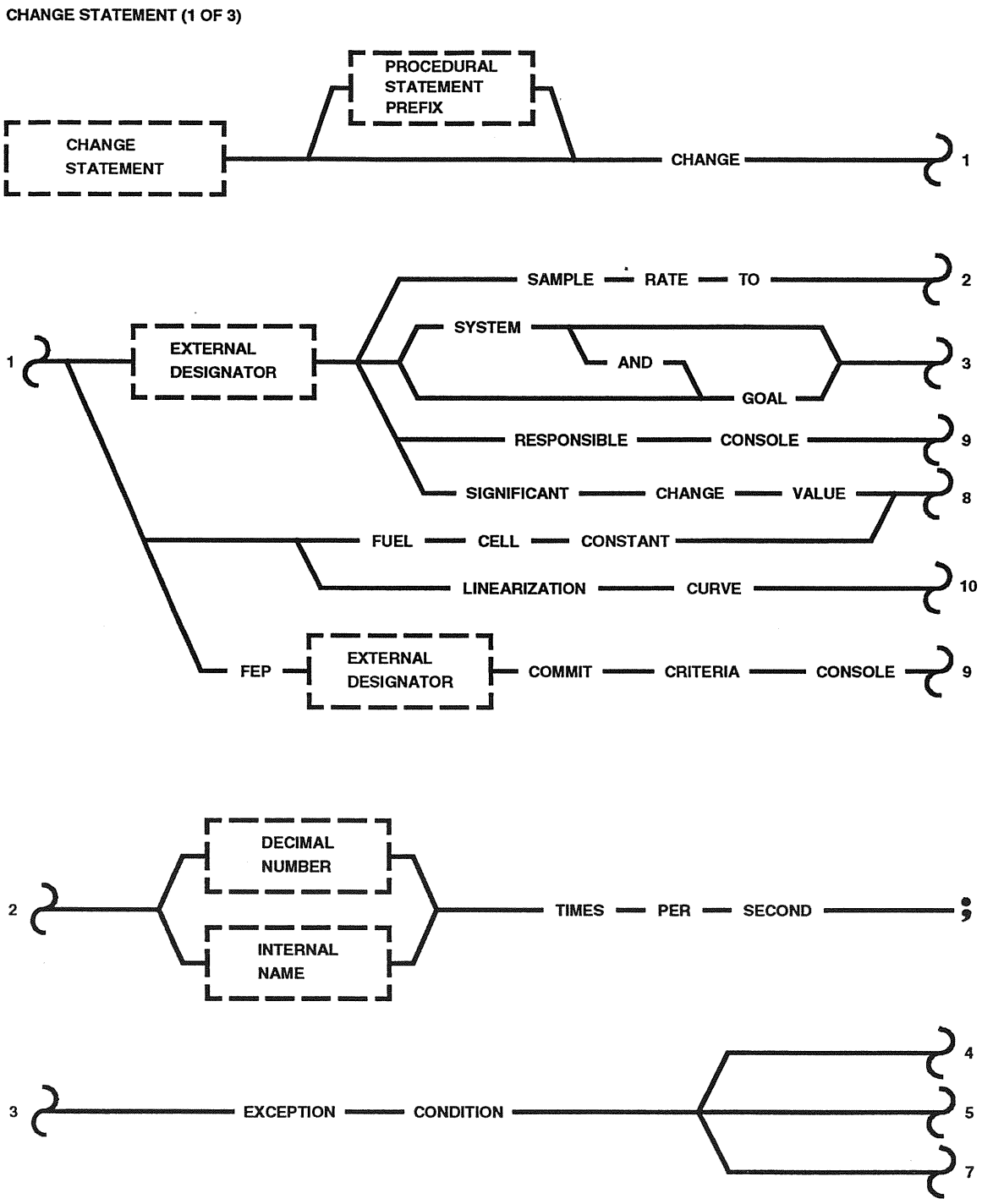


Figure 11-4 CHANGE Statement (1 of 3)

CHANGE STATEMENT (2 OF 3)

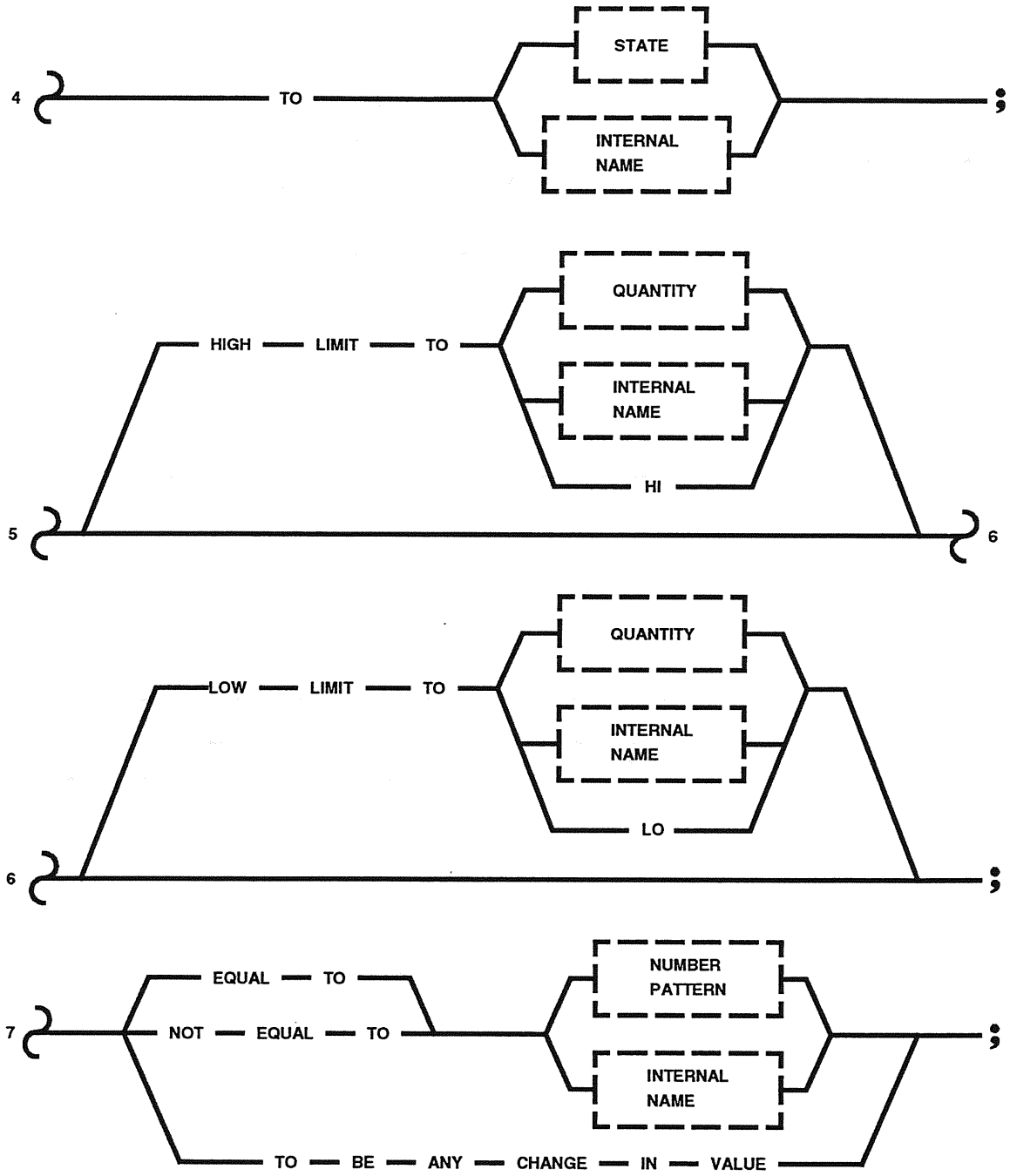


Figure 11-4 CHANGE Statement (2 of 3)

CHANGE STATEMENT (3 OF 3)

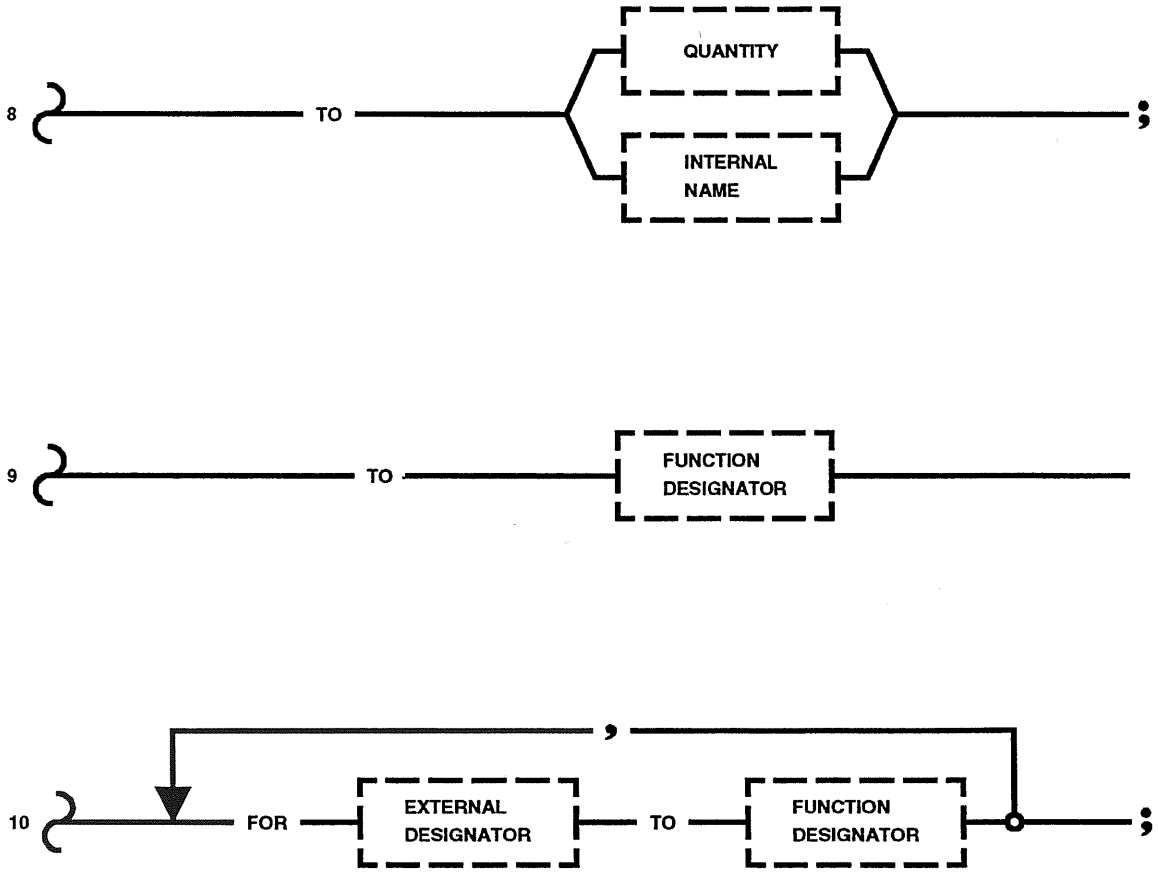


Figure 11-4 CHANGE Statement (3 of 3)

Function

The CHANGE Statement is used to change the following measurement processing parameters in a FEP: sample rate, system exception condition, GOAL exception condition, significant change value, responsible console, fuel cell constant, commit criteria console, and linearization curve.

Description

The following statement options are available:

A. CHANGE SAMPLE RATE

Requests the GSE FEP to change the sample rate of a given Function Designator to a specified value.

B. CHANGE SYSTEM AND GOAL EXCEPTION CONDITION

System exception conditions refer to Exception Monitor and Control Logic exception criteria. GOAL exception criteria may be defined separately from system exception criteria. Options are as follows:

1. Request the FEP to change the exception state for a given discrete measurement to a specified state.

2. Request the FEP to change the system (Exception Monitor/Control Logic) or GOAL exception limits for a given analog measurement to a specified value. Use of the keyword LO will set the low limit to a value lower than the measurements low range thus prohibiting exception reporting on the low end. Similarly, use of the keyword HI will set the high limit to a value higher than the measurements high range thus prohibiting exception reporting on the high end. The HI and LO options are not valid for GPC Floating Point type FD's.

3. Request the FEP to change the exception condition for a given digital pattern measurement to: equal to a value, not equal to a value, or any change in bit pattern from one sample to the next (not valid for GSE FEP Function Designators).

C. CHANGE SIGNIFICANT CHANGE VALUE

Requests the FEP to change the significant change value for a given analog measurement to a specified value.

D. CHANGE FUEL CELL CONSTANT

Requests CITE GSE FEP to change the value for the fuel cell aging constant.

E. CHANGE RESPONSIBLE CONSOLE

Requests the FEP to change the responsible console for the given Function Designator.

F. CHANGE LINEARIZATION CURVE

Requests the FEP to change the linearization curve for the specified Function Designators.

G. COMMIT CRITERIA CONSOLE (CCC)

Requests the FEP(s) to change the CCC used for dual notification to the console specified by Function Designator. The list of Function Designators (system type-FEP represented by EXTERNAL DESIGNATOR) will have exception information sent to their RSYS console and the new commit criteria console. The default console receiving the limits violation information is the BACKUP console. When CCC notification is in effect, commit criteria console may be changed from BACKUP to either the MASTER or INTEGRATION console.

The following repetitive operations are allowed:

One-to-One: The sample rate, exception criteria, or significant change value for a single Function Designator is changed to the value of a single internal variable (or high and low limits for analog measurements).

Many-to-One: The sample rate, exception criteria, or significant change value for many Function Designators are changed to the value of a single internal variable (or high and low limits for analog measurements).

CHANGE SAMPLE RATE Option

Examples

```
CHANGE < DM1 > SAMPLE RATE TO 10 TIMES PER SECOND;
CHANGE < AM1 > SAMPLE RATE TO (N100) TIMES PER SECOND;
```

CHANGE COMMIT CRITERIA CONSOLE Option

Example

```
CHANGE FEP <GS1A>, <GS3>, <GPCA> COMMIT CRITERIA
    CONSOLE TO <INTEG>;
```

CHANGE COMMIT CRITERIA Option

Example

```
CHANGE < DM1 > SYSTEM EXCEPTION CONDITION TO ON;
CHANGE < AM1 > GOAL EXCEPTION CONDITION HIGH LIMIT TO
    5V LOW LIMIT TO -5.2V;
CHANGE < AM2 > SYSTEM EXCEPTION CONDITION HIGH LIMIT TO
    (Q5) LOW LIMIT TO (Q2);
CHANGE < DPM1 > GOAL EXCEPTION CONDITION EQUAL TO X FF;
```

CHANGE SIGNIFICANT CHANGE VALUE Option

Examples

```
CHANGE < AM1 > SIGNIFICANT CHANGE VALUE TO .2V;
CHANGE < AM2 > SIGNIFICANT CHANGE VALUE TO (Q2);
```

CHANGE FUEL CELL CONSTANT Option

Example

```
CHANGE FUEL CELL CONSTANT TO 32.5;
```

CHANGE RESPONSIBLE CONSOLE Option

Example

```
CHANGE < DM1 > RESPONSIBLE CONSOLE TO < ARMS >;
```

CHANGE LINEARIZATION CURVE Option

Example

```
CHANGE LINEARIZATION CURVE FOR < AM1 > TO < CURVE 1 >;
CHANGE LINEARIZATION CURVE FOR < AM1 >, < AM27 > TO < CURVE 2 >;
CHANGE LINEARIZATION CURVE FOR < AM1 > < AM2 > TO < CURVE 1 >,
    FOR < AM3 > TO < CURVE 2 >, FOR < AM4 > < AM5 > < AM6 > TO
    < CURVE 3 >;
```

Statement Execution

The GOAL Executor sends a request to the appropriate FEP to change the parameters specified on the CHANGE Statement via the CDBFR. If the statement references more than one Function Designator, multiple requests will be generated by the GOAL Executor.

Execution will not continue to the next statement until all of the Function Designators have been processed.

For 'Change Limit' option, if the specified limit value is lower than the low range, then the low range will be substituted. If it is higher than the high range, then the high range will be substituted.

WARNING

When reactive control logic sequences are used to verify limit exceptions, quantity-to-count conversion techniques are different from those used by GOAL. Reference KSC-LPS-OP-033-08, Procedural Statement Prefixes section.

Error Processing

Errors encountered on requests to the FEP's will result in Class III errors.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Change GOAL/System exception conditions for digital patterns Function Designators are not valid for Function Designators assigned to GSE FEP's.

B. Sample rates may only be changed to the following rates: 100, 10 and 1 samples per second. Sample rates may not be changed to a value which is less than the original sample rate of the measurement. This option is only valid for GSE Function Designators.

C. FUEL CELL CONSTANT must be expressed in Volts and range from 32.32 to 32.96.

D. The EQUAL TO, NOT EQUAL To or TO BE ANY CHANGE IN VALUE option must be specified when the exception options for digital pattern Function Designators are selected.

E. The STATE option must be specified when the exception options for discrete Function Designators are selected.

F. A HIGH and/or LOW LIMIT option must be specified when the EXCEPTION options for analog Function Designators are selected.

G. The CHANGE Statement may not be used to reference LDB or Uplink Command FEP Function Designators.

H. Entire lists or table columns may not be used in the CHANGE Statement.

I. When using the LINEARIZATION CURVE option, the Function Designator(s) specified after the keyword FOR must be analog measurement type, and the Function Designator following the keyword TO must be a legal curve name defined in the Data Bank.

J. Function Designators which are Pseudo Parameters are illegal for the LINEARIZATION CURVE option.

K. CHANGE RESPONSIBLE CONSOLE

1. The new responsible console must have the command's prerequisite sequence if one exists.

2. The new responsible console must have the measurement's reactive sequence if it is Control Logic dependent.

3. A command to change responsible console is valid only from the current responsible console or Type II consoles.

L. Function Designators which are Pseudo Parameters are illegal for the console Function Designator on the CHANGE COMMIT CRITERIA option. Valid consoles are MASTER, INTEGRATION, or BACKUP (TYPE II).

M. Function Designators specified with HI and LO options of the HIGH LIMIT and LOW LIMIT branches must have the same type and subtype (unipolar/bipolar).

N. When a units mismatch occurs while using the Table Functions option, a compiler warning message will be output.

O. SYSTEM and/or GOAL EXCEPTION conditions may not be changed for Discrete Measurement Function Designators when notification is active.

Acceptable Conditions

High and low limits specified as literal numbers are range checked by the GOAL Configurator using counts. High and low limits specified as Internal Names are range checked by the GOAL Executor using quantity values.

Any token that follows a non-keyword token and precedes the semicolon will be ignored.

Responsible Console Legal Function Designators

Analog Stimulus (AS)
Discrete Stimulus (DS)
Digital Pattern Stimulus (DPS)
Time Homogeneous Data Sets (THDS)
Analog Measurement Double Precision (AMDP)
Multiword Digital Pattern Measurements (MWDP)
Digital Pattern Stimulus with Data (DPSD)

Legal Function Designators for Change Statement

Analog Measurements (AM)
Discrete Measurements (DM)
Digital Pattern Measurements (DPM)
Console Type (for Responsible Console) (CONS)
Linearization Curve Type (Change Linearization Curve) (LS1, LS4, LSS)
FEP Type (for Commit Criteria) (FEP)

11.4 LOAD SAFING SEQUENCE STATEMENT

LOAD SAFING SEQUENCE STATEMENT

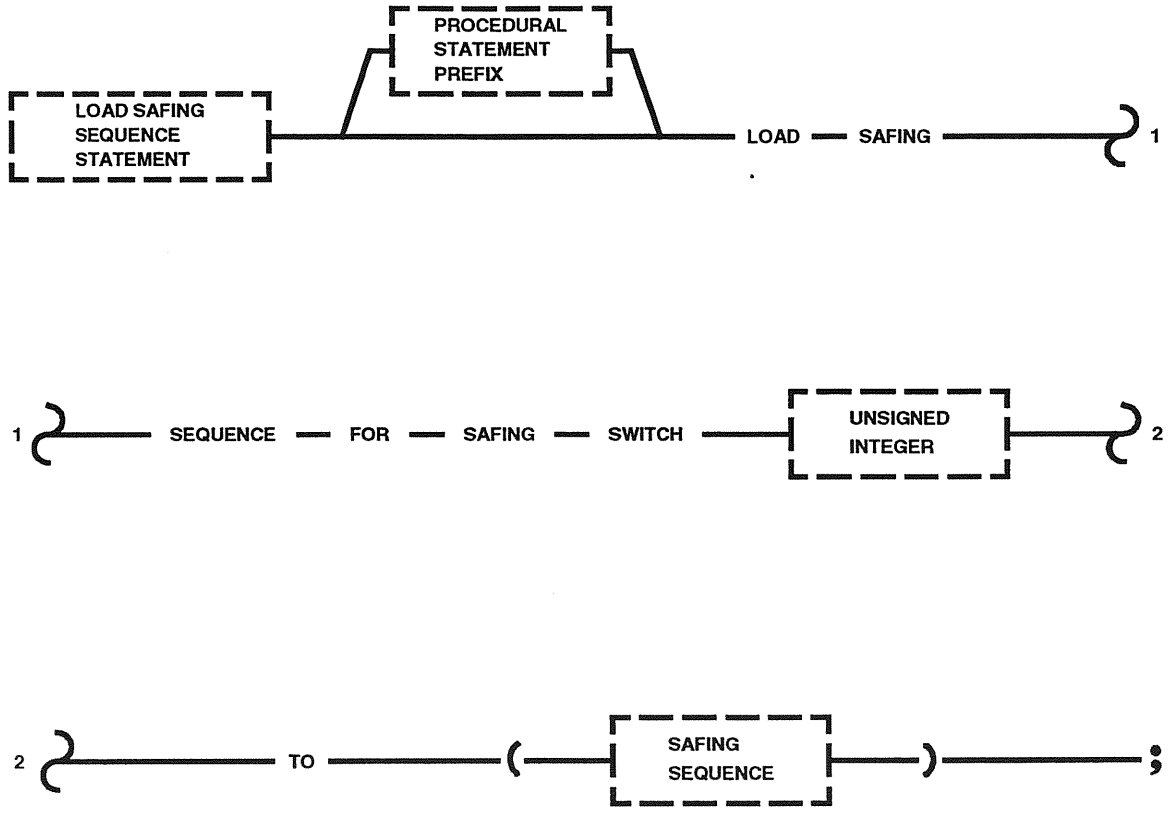


Figure 11-5 LOAD SAFING SEQUENCE Statement

Function

The LOAD SAFING SEQUENCE Statement allows the test engineer to load a sequence of events to be executed via manual action in the event of a CDBFR failure.

Description

The LOAD SAFING SEQUENCE Statement allows the procedure writer to cause a sequence to be transmitted to the LDB FEP, but it does not execute the procedure. The execution of the procedure must be initiated by manually throwing a switch on the console safing panel. The options of the SAFING SEQUENCE Statements are:

A. SET Option

B. ISSUE Option

C. COMMAND Option

D. SRB MDM Option

LOAD SAFING SEQUENCE

Example:

```
LOAD SAFING SEQUENCE FOR SAFING SWITCH 2 TO (SET
    < V76K1903XL > TO OFF, COMMAND LS RECYCLE);
```

SET Option

Examples

```
SET < V72K1321XL > TO OFF;
SET < V72K1321XL > TO OFF FOR 1 SEC;
```

In the first example, the SET option is used to issue a single discrete value to a single Function Designator. In the second example, it is used with a pulse time delay of one second to turn off the specified Function Designator for 1 second; then turn it back on. In execution of the second example, the Function Designator is set to OFF. The GPC waits for 1 second to elapse, complements the state, then issues the complement.

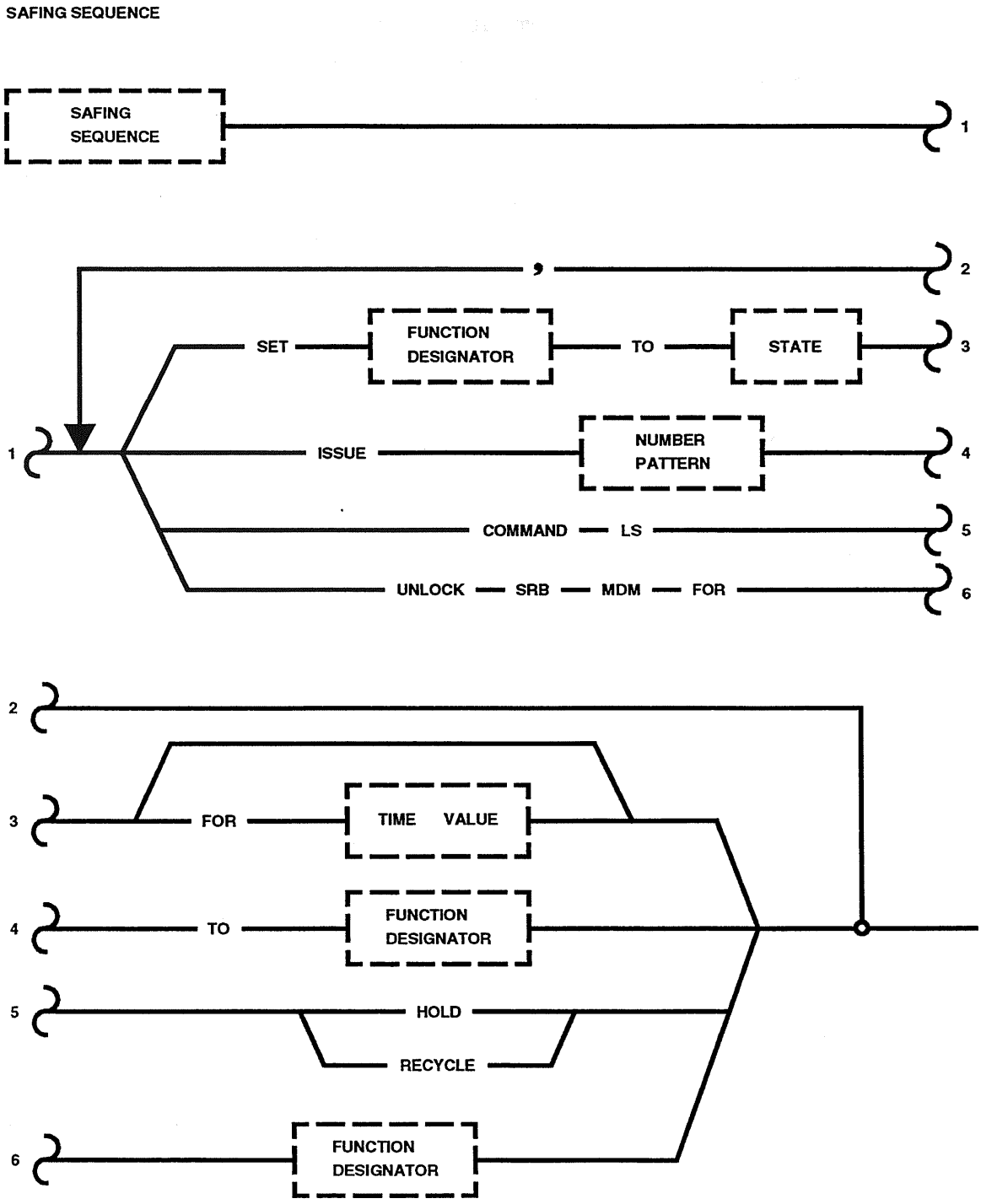


Figure 11-6 Safing Sequence

Issue Option

Example

```
ISSUE 117 TO < V76K1371BL >;
```

The Issue option is used to issue a numeric constant to a single Function Designator.

Command Option

Example

```
COMMAND LS HOLD;
```

The Command option may command Launch Sequence to:

A. HOLD

B. RECYCLE

SRB MDM Option

Example

```
UNLOCK SRB MDM FOR < B55K1300XL >;
```

The SRB MDM option results in two SACS ISSUE operators being sent to the MDM within 700 ms and with a data value of zero (0). This action results in the UNLOCK operation.

Statement Execution

The GOAL Language Processor processes each safing sequence request, and the resulting data is built as a contiguous block in the GOAL Procedure's work area.

The GOAL Executor obtains the block of data and issues it to the LDB FEP via the CDBFR.

If the data was successfully issued, it will be saved in the LDB FEP for possible future execution, and the GOAL Executor will continue execution of the GOAL procedure. If an error response was received from the LDB FEP, the GOAL Executor will process it as a Class III error.

Error Processing

If an error response is received from a FEP on a SAFING SEQUENCE Statement, a Class III error will be processed.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. A maximum of 250 words is allowed in the sequence:

OPTION	NUMBER OF WORDS
Set	5
SRB MDM	4
Issue	3
Command	1

B. The safing switch number must range from 0 to 31.

C. Once loaded, the same switch cannot be loaded again unless the LDB FEP is reloaded.

D. The Time Value on the SET option must be a literal value.

E. Time Value may not exceed 1.26 seconds.

Legal Function Designators

SET Option

Discrete Stimulus MDM/BTU type (must be LDB if Time Value is used)

ISSUE Option

Digital Pattern Stimulus MDM/BTU type

SRB MDM Option

LDB Discrete Stimulus MDM/BTU type

11.5 ACTIVATE PLOT STATEMENT

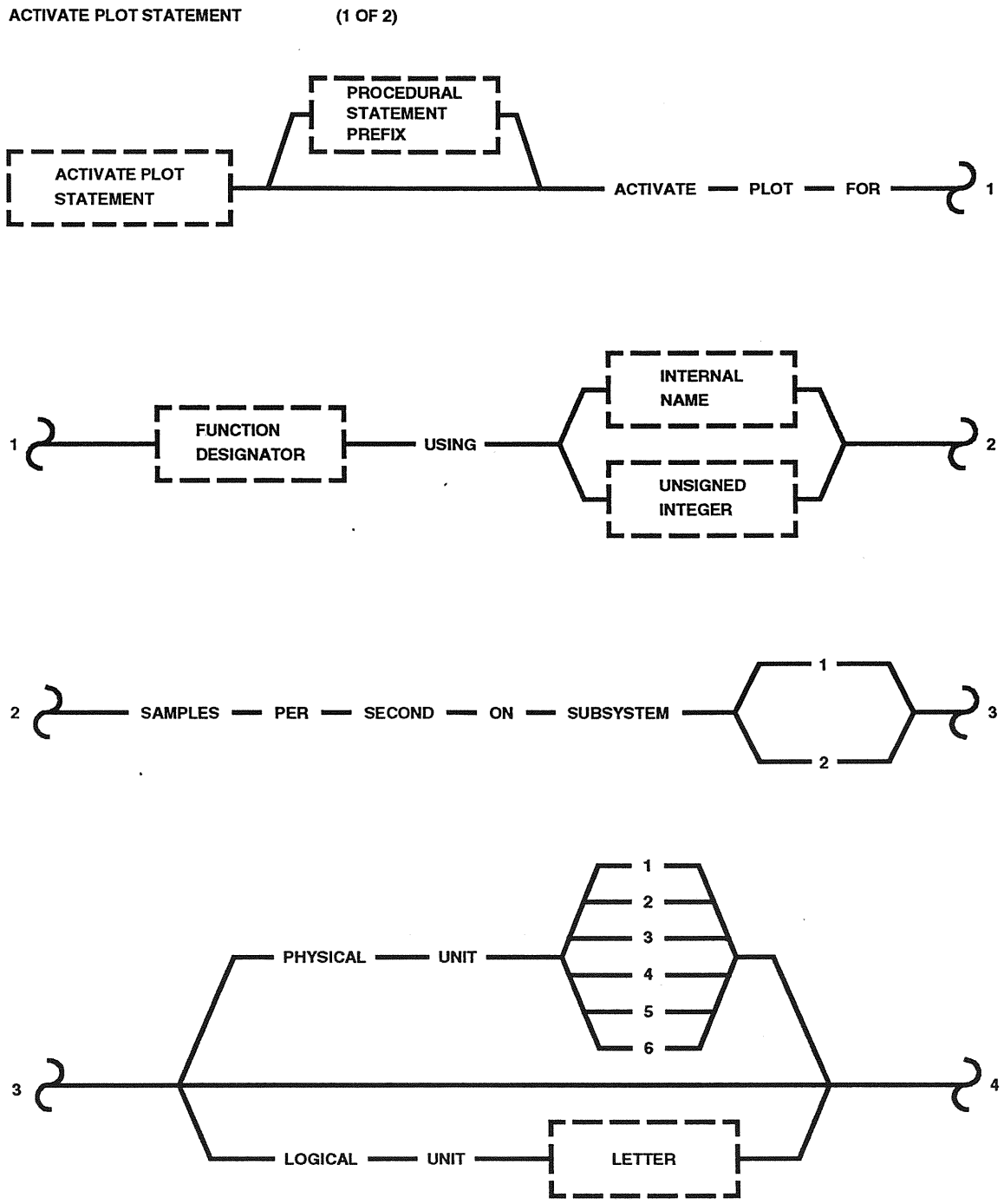


Figure 11-7 ACTIVATE PLOT Statement (1 of 2)

ACTIVATE PLOT STATEMENT (2 OF 2)

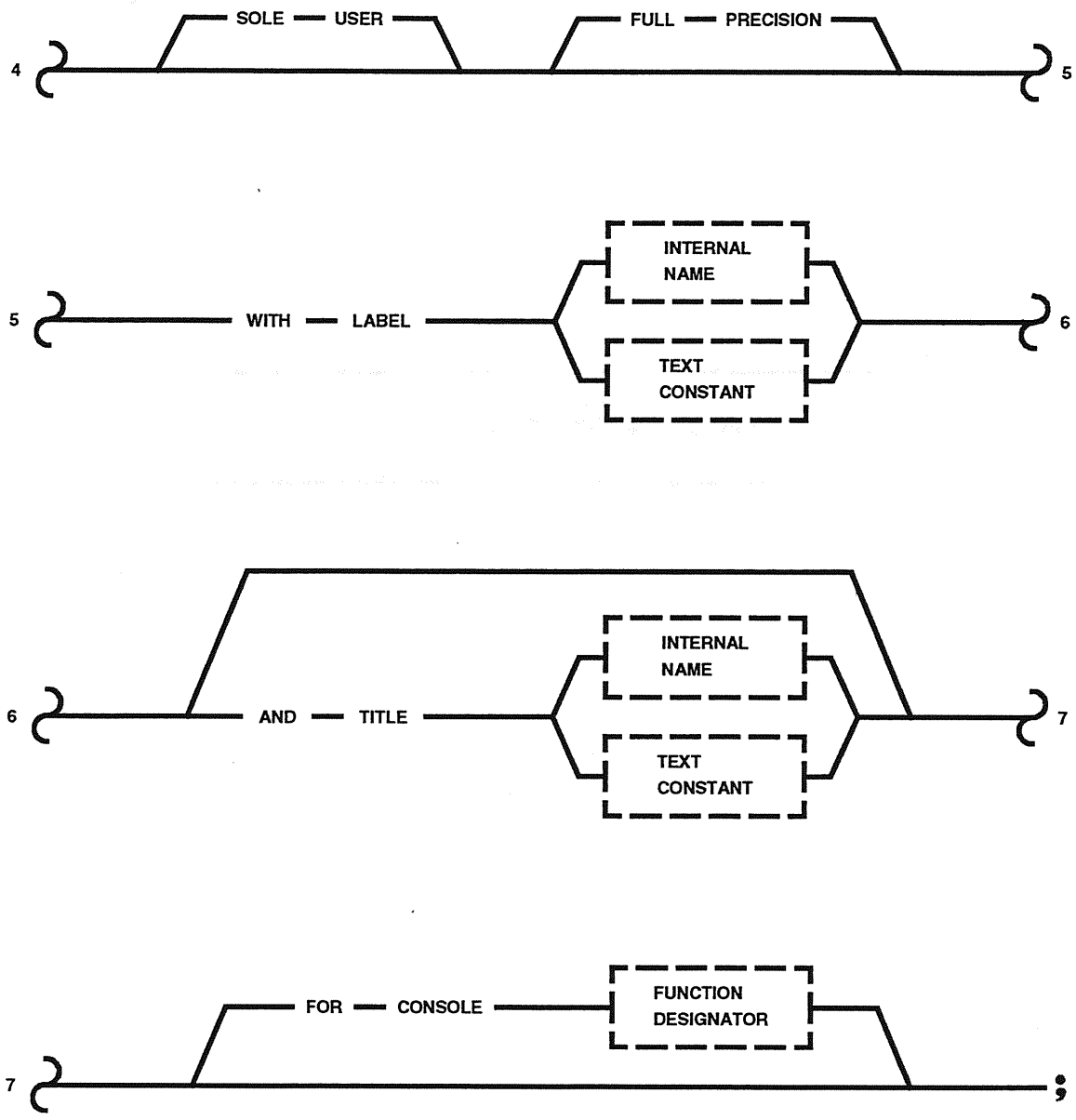


Figure 11-7 ACTIVATE PLOT Statement (2 of 2)

THIS PAGE INTENTIONALLY LEFT BLANK.

Function

The ACTIVATE PLOT Statement is used to initiate plotting of a data item from the CDBFR to the system plotter.

Description

The ACTIVATE PLOT Statement allows the test engineer to initiate plotting for a single Function Designator. The test engineer supplies the sample rate at which the data is to be plotted and a label.

Statement Options

PHYSICAL UNIT

Plot data is directed to one of six physical units on a given subsystem.

LOGICAL UNIT

Plot data is directed to a logical unit by specifying a single letter which identifies the logical unit.

SOLE USER

This option is specified if the unit will be dedicated to a single user.

FULL PRECISION

This option is specified to generate a plot with an eight bit field width format. If this system is not specified, a seven bit field width format will be used.

CONSOLE

This option is specified to assign a set of data to a specific console explicitly.

TITLE

A 60-character header can be specified for the top of the plot by using this option.

ACTIVATE PHYSICAL UNIT Option

Example

```
ACTIVATE PLOT FOR < DM1 > USING 25 SAMPLES PER SECOND ON
    SUBSYSTEM 1 PHYSICAL UNIT 6 WITH LABEL TEXT (DM1);
```

ACTIVATE LOGICAL UNIT Option

Example

```
ACTIVATE PLOT FOR < AM1 > USING 10 SAMPLES PER SECOND ON
    SUBSYSTEM 1 LOGICAL UNIT B WITH LABEL (N);
```

ACTIVATE SOLE USER/FULL PRECISION/TITLE Options

Example

```
ACTIVATE PLOT FOR < DM1 > USING 25 SAMPLES PER SECOND ON
    SUBSYSTEM 1 PHYSICAL UNIT 6 SOLE USER FULL PRECISION WITH
    LABEL TEXT (DM1bbbbb) AND TITLE TEXT (THIS IS A PLOT OF
    DM1) FOR CONSOLE < ARMS >;
```

Statement Execution

The GOAL Executor sends a request to the printer/plotter subsystem to initiate the specified plot.

Error Processing

A failure to send the request to the printer/plotter subsystem results in a Class III error.

If the request is rejected by the printer/plotter subsystems a Class V error will result.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Sample rates must be between 1 and 50.

B. A title has a maximum of 60 characters.

C. A label is exactly 8 characters.

D. Plotted data must be buffer resident.

TABLE 11-2 LEGAL FUNCTION DESIGNATORS ACTIVATE PLOT STATEMENT

TYPES	SOURCE	CLASS
ANALOG MEASUREMENTS (AM) ANALOG STIMULUS (AS) DISCRETE MEASUREMENTS (DM) DISCRETE STIMULUS (DS) PSEUDO DISCRETES (PD) ANALOG MEASUREMENT DOUBLE PRECISION (AMDP) FLOATING POINT (FP) SYSTEM STATUS AREA 1 (SSA1)	GSE, PCM, RTU, NONE	N/A

THIS PAGE INTENTIONALLY LEFT BLANK.

11.6 INHIBIT PLOT STATEMENT

INHIBIT PLOT STATEMENT

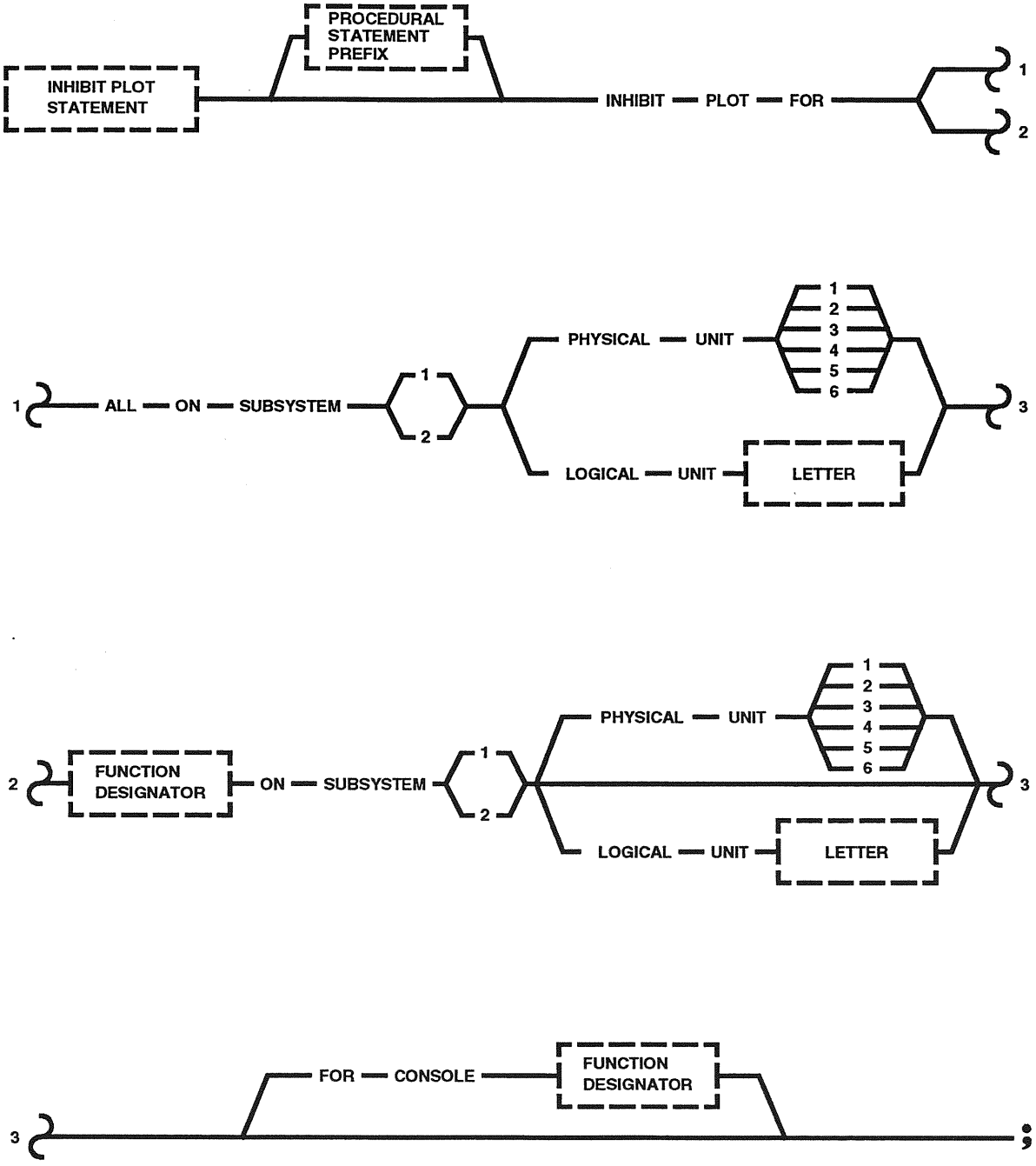


Figure 11-8 INHIBIT PLOT Statement

Function

The INHIBIT PLOT Statement is used to stop a plot that was initiated by the ACTIVATE PLOT Statement.

Description

The INHIBIT PLOT Statement allows the test engineer to terminate the plotting for one or all Function Designators. The plot must have been previously activated by ACTIVATE PLOT Statements.

Statement Options

ALL

All plots on a given physical or logical unit will be terminated.

PHYSICAL UNIT

A physical unit may be specified on which the plot or plots are currently executing.

LOGICAL UNIT

A logical unit may be specified on which the plot or plots are currently executing.

CONSOLE

This option may be selected to terminate a plot which was initiated by another console.

INHIBIT PLOT for One Function Designator

Example

```
INHIBIT PLOT FOR < AM1 > ON SUBSYSTEM 1;
```

INHIBIT ALL/PHYSICAL UNIT Options

Example

```
INHIBIT PLOT FOR ALL ON SUBSYSTEM 2 PHYSICAL UNIT 4;
```

INHIBIT LOGICAL UNIT/CONSOLE Options

Example

```
INHIBIT PLOT FOR < AM2 > ON SUBSYSTEM 1 LOGICAL UNIT B FOR
CONSOLE
    < ARMS >;
```

Statement Execution

The GOAL Executor sends a request to the printer/plotter subsystem to terminate the specified plot or plots.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The CONSOLE option may only be used by procedures residing in the Master or Integration Console.

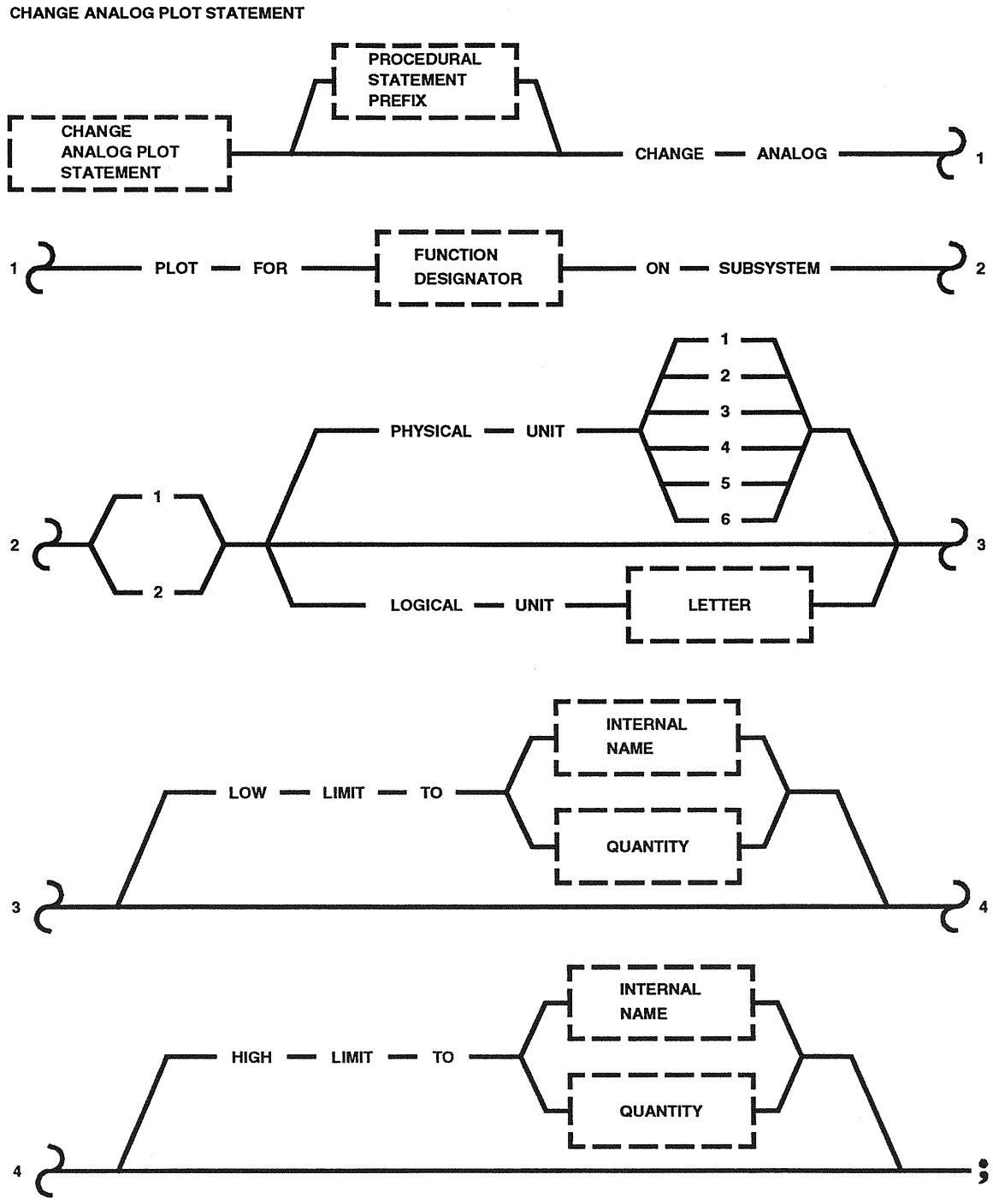
B. A physical or logical unit must be specified when the ALL option is selected.

C. Plotted FD's must be buffer resident.

Legal Function Designators (Buffer Resident Only)

Analog Stimulus (AS)
Analog Measurements (AM)
Discrete Measurements (DM)
Discrete Stimulus (DS)
Pseudo Discretes (PD)
Analog Measurement Double Precision (AMDP)
Floating Point (FP)
System Status Area 1 (SSA1)

11.7 CHANGE ANALOG PLOT STATEMENT



NOTE: EITHER LOW LIMIT AND/OR HIGH LIMIT MUST BE SPECIFIED.

Figure 11-9 CHANGE ANALOG PLOT Statement

Function

The CHANGE ANALOG PLOT Statement is used to change high and/or low limits for previously activated analog plots.

Description

The CHANGE ANALOG PLOT Statement allows the test engineer to change plot limits of a currently active plot for a single Function Designator. The test engineer supplies the new high and/or low limits. A return status from the subsystem will indicate whether the change request was successful or failed.

Statement Options

PHYSICAL UNIT

A physical unit may be specified on which the plot is currently executing.

LOGICAL UNIT

A logical unit may be specified on which the plot is currently executing.

LOW LIMIT

This option is specified to assign a new low limit for a plot that is currently active.

HIGH LIMIT

This option is specified to assign a new high limit for a plot that is currently active.

Statement Execution

The GLP will generate interpretive code for the Executor to issue a C-to-C for the non-GOAL program to perform $PPCHTR at the SPA. The non-GOAL program will request the change analog plot for the specified subsystem. The parameters passed to the non-GOAL program will be put on the Top Stack, and the Executor will enqueue a request to the Operating System (OS) task initiator. The OS obtains the program from disk, and executes it. The amount of time required to perform the Change Plot Request depends upon the time it takes for the enqueued request to be executed.

The OS will inform the Executor when the program terminates. The return status left on the Top Stack by the non-GOAL program will be tested by the Executor for a bad return status.

CHANGE PLOT PHYSICAL UNIT Option

Example

```
CHANGE ANALOG PLOT FOR <AM1> ON SUBSYSTEM 1 PHYSICAL UNIT 2 LOW
    LIMIT TO (QUAN1) HIGH LIMIT TO (QUAN2);
```

CHANGE PLOT LOGICAL UNIT Option

Example

```
CHANGE ANALOG PLOT FOR <AM3> ON SUBSYSTEM 2 LOGICAL UNIT F HIGH
    LIMIT TO 5V;
```

CHANGE PLOT ALL UNITS Option

Example

```
CHANGE ANALOG PLOT FOR <AM5> ON SUBSYSTEM 1 LOW LIMIT TO (QUAN1)
    HIGH LIMIT TO 5V;
```

Error Processing

If the non-GOAL program $PPCHTR cannot be executed, a Class II error will be processed. The Executor will issue a Class III error for an invalid return status.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. FD must be buffer-resident

B. Either High Limit and/or Low Limit must be specified.

Legal Function Designators

Analog Measurements (AM)
Analog Stimulus (AS)
Analog Measurements Double Precision (AMDP)
Pseudo Analog (PA)

11.8 ACTIVATE CRITICAL MODE STATEMENT

ACTIVATE CRITICAL MODE STATEMENT

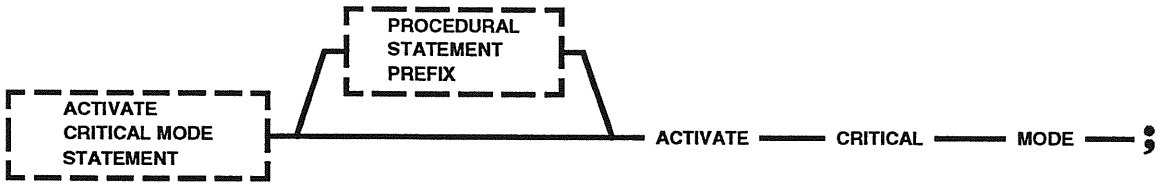


Figure 11-10 ACTIVATE CRITICAL MODE Statement

Function

The ACTIVATE CRITICAL MODE Statement is used to allow a procedure to enter a critical mode of execution (higher priority mode).

Description

Upon execution of the ACTIVATE CRITICAL MODE Statement, the GOAL procedure is executed in critical mode (receives a higher priority) until an INHIBIT CRITICAL MODE Statement is encountered.

If a procedure is already in critical mode when an ACTIVATE CRITICAL MODE Statement is encountered, the statement will have no effect.

A Level 1 procedure always begins execution with critical mode inhibited.

The initial mode of any procedures executed via the PERFORM PROGRAM Statement is the mode currently in effect in the calling program.

A procedure executed via the CONCURRENTLY PERFORM Statement will be entered with critical mode inhibited.

When control is returned to the calling procedure following a PERFORM PROGRAM Statement, execution will resume in the mode of execution in effect prior to the PERFORM PROGRAM Statement.

Statement Execution

The ACTIVATE CRITICAL MODE Statement causes the GOAL procedure to be placed on a "high priority" dispatching queue. This means that the procedure will gain control of the CPU resources until the critical mode enters an I/O wait or terminates. How often it would be allowed to run would be dependent upon how many other GOAL procedures were in critical mode and in normal mode. Generally, a critical program is allowed to run every other time a GOAL program is dispatched. After a critical GOAL task terminates, its execution will no longer have an effect on the dispatching of other GOAL programs.

THIS PAGE INTENTIONALLY LEFT BLANK.

11.9 INHIBIT CRITICAL MODE STATEMENT

INHIBIT CRITICAL MODE STATEMENT

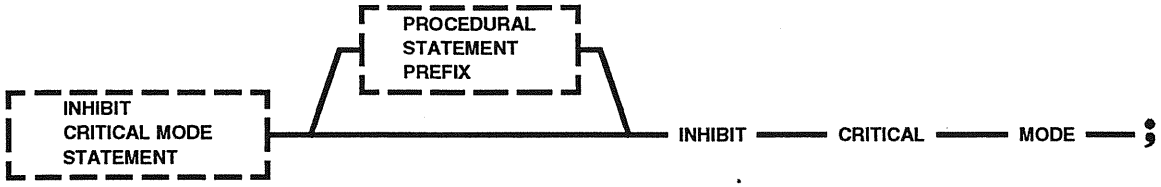


Figure 11-11 INHIBIT CRITICAL MODE Statement

Function

The INHIBIT CRITICAL MODE Statement is used to allow a program to exit a critical (higher priority) mode of execution.

Description

Upon execution of the INHIBIT CRITICAL MODE Statement, the GOAL procedure will cease to execute in a critical (high priority) mode.

If a procedure is not executing in critical mode when this statement is encountered, the INHIBIT CRITICAL MODE Statement will have no effect.

A Level 1 procedure always begins execution with critical mode inhibited.

The initial mode of any procedure executed via the PERFORM PROGRAM Statement is the mode currently in effect in the calling program.

A procedure executed via the CONCURRENTLY PERFORM Statement will be entered with critical mode inhibited.

When control is returned to the calling procedure following a PERFORM PROGRAM Statement, execution will resume in the mode of execution in effect prior to the PERFORM PROGRAM Statement.

Statement Execution

The INHIBIT CRITICAL MODE Statement causes the GOAL procedure to be placed on the normal priority dispatching queue. This means that the procedure will gain control of the CPU resources on an equal basis with other Goal procedures.

THIS PAGE INTENTIONALLY LEFT BLANK.

12. INTERRUPT PROCESSING STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The Interrupt Processing Statements which will be discussed in the indicated sections are as follows:

A. SPECIFY INTERRUPT STATEMENT (Section 12.4)

The SPECIFY INTERRUPT Statement is used to define the external event for which a GOAL procedure needs to be notified and to specify the Step Number to which to transfer control when the interrupt occurs. The SPECIFY INTERRUPT Statement can also specify a numeric list that will receive 7 words of data when a remote communication interrupt is processed. The data will consist of 4 words of user data followed by 3 words of system data. The GOAL program must not alter the system data.

B. ACTIVATE NOTIFICATION STATEMENT (Section 12.5)

The ACTIVATE NOTIFICATION Statement is used to activate Interrupt Processing in a GOAL procedure or in a FEP. If Interrupt Processing is activated in a procedure, interrupts will be passed to the procedure so that appropriate action may be taken. If Interrupt is activated in an FEP, the FEP will notify the Responsible Console when a measurement reaches an exception condition so that the GOAL procedure may be given the interrupt.

C. INHIBIT NOTIFICATION STATEMENT (Section 12.6)

The INHIBIT NOTIFICATION Statement is used to inhibit Interrupt Processing in a GOAL procedure or in an FEP. When Interrupt Processing is inhibited in a GOAL procedure, the procedure will not be notified when an interrupt occurs. When Interrupt Processing is inhibited in a FEP, the FEP will not notify the Responsible Console when a measurement reaches an exception condition.

D. WHEN INTERRUPT STATEMENT (Section 12.7)

The WHEN INTERRUPT Statement is used to define an interrupt condition based on Count Down Time, Mission Elapsed Time or the Interval Timer and to specify a Step Number to which to transfer control when the interrupt occurs. Interrupts may be requested for Count Down Time or Mission Elapsed Time equal to a specified value or for the Time Value of the Interval Timer reaching 0.

E. SEND INTERRUPT STATEMENT (Section 12.8.1.1)

The SEND INTERRUPT Statement is used to send an interrupt to another GOAL procedure at the same console or at a remote console. It may also be used to pass up to 4 user defined numeric data words with the interrupt.

F. RELAY INTERRUPT STATEMENT (Section 12.8.1.2)

The RELAY INTERRUPT Statement is used to relay data received with a remote communication interrupt to other consoles or other concurrencies within the same console.

G. RETURN INTERRUPT STATEMENT (Section 12.8.1.3)

The RETURN INTERRUPT Statement is used to return response data back to the console that originally sent a remote communication interrupt with data via the SEND INTERRUPT Statement. This response data could serve as an acknowledgment that the request specified by the original interrupt's data was completed successfully, unsuccessfully, etc.

The Interrupt Processing Statements are the means by which a GOAL procedure receives notification of external events.

12.1 TYPES OF INTERRUPTS

Interrupts will be allowed on the following Function Designator types:

A. MEASUREMENT INTERRUPTS

Measurement interrupts may occur under the following conditions:

1. Discrete

The specified discrete changes to a state defined as the exception state.

2. Analog

The specified analog goes out of limits.

3. Digital Pattern

The specified digital pattern goes to an exception value which is one of the following:

(A) equal to a value

(B) not equal to a value

(C) any change in value

B. DG FUNCTION KEY INTERRUPTS and PFP INTERRUPTS

Interrupts for DG Function Keys and PFP Function Keys occur when the specified key is depressed.

C. CDT INTERRUPTS

An interrupt on CDT may be requested when CDT reaches a specified value.

D. INTERVAL TIMER INTERRUPTS

An interrupt on the Interval Timer occurs when the time reaches zero.

E. REMOTE COMMUNICATION INTERRUPTS

An interrupt on a Remote Communication Function Designator occurs when another GOAL procedure sends an interrupt to the procedure which has specified an interrupt on the Remote Communication Function Designator. Up to 4 words of user defined numeric data may also be passed with the interrupt. After receiving an interrupt with data, a GOAL procedure may use a RELAY INTERRUPT Statement to relay the interrupt to another console. The procedure may also use a RETURN INTERRUPT Statement to return an acknowledging interrupt back to the original interrupt-sending procedure.

F. SYSTEM INTERRUPTS

System interrupts may be requested on the following conditions:

1. Measurement validity changes

2. CPU status changes

3. Class III errors

4. Interrupt event occurrence on a higher level

5. CDT status changes

   To determine what the current CDT status is, check the value of the PDP pseudo FD <NCDTSTAT>:

Bit Value(s)	Description
Bits 0-11	spare
Bit 12	1 = hold at pending
Bit 13	1 = count at pending
Bit 14	1 = counting 0 = holding
Bit 15	1 = set

Examples:

Numeric Value	Status
0	CDT not set
1	holding
3	counting
5	holding with count at pending
7	counting with count at pending
9	holding with hold at pending
11	counting with hold at pending

6. MET status changes

G. MET INTERRUPTS

An interrupt on MET may be requested when MET reaches a specified value.

12.2 HOW TO REQUEST AN INTERRUPT

The basic requirements for requesting an interrupt are:

A. SPECIFY the Interrupt.

B. Activate Interrupt Processing.

C. Activate FEP Interrupt Check for Measurement Type Function Designator.

The SPECIFY INTERRUPT Statement must be coded before any procedural statement to define the interrupt condition and the Step Number to GO TO when the interrupt occurs. The ACTIVATE INTERRUPT Statement must be used to activate Interrupt Processing prior to receiving an interrupt since Interrupt Processing is initially inhibited when a procedure is performed. Measurement Type interrupts require procedure activation of FEP Interrupt Check to ensure event processing.

12.2.1 HOW TO REQUEST A MEASUREMENT INTERRUPT

In order to request an interrupt on a measurement, a SPECIFY INTERRUPT must first be used to identify the measurement and the Step to GO TO when the interrupt occurs.

Example

```
SPECIFY INTERRUPT < DM1 >
    AND ON OCCURRENCE GO TO STEP 7;
. . . . . .
ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL;
ACTIVATE FEP INTERRUPT CHECK FOR < DM1 >;
```

In the example, when the discrete measurement < DM1 > goes to the exception state, control will be transferred to STEP 7. The second requirement for receiving an interrupt is to activate Interrupt Processing. This is accomplished by the second statement in the example, which causes all interrupts specified in the procedure (i.e., ON THIS LEVEL) to be activated. The last requirement for receiving a measurement interrupt is to request notification from the FEP when the measurement goes to the exception state. This request is accomplished by the third statement in the example.

12.2.2 HOW TO REQUEST A DG FUNCTION KEY INTERRUPT

Interrupts may be received from the following DG Function Keys:

A. Transmit Cursor

B. Execute Command

C. Disarm Command

D. Programmable Function Key

The first three (Transmit Cursor, Execute Command, and Disarm Command) are associated with cursor position. When the cursor is

positioned to the appropriate coordinates on the color CRT and the specified key is depressed, an interrupt will be transferred to the GOAL procedure. Interrupts from the Programmable Function Keys, unlike the other DG Function Keys, are not tied to cursor position but are associated with the page which is being viewed. When the specified page is being viewed and the specified Programmable Function Key is depressed, an interrupt is transmitted to the GOAL procedure.

Examples

```
SPECIFY INTERRUPT < XMIT-PA > LINE 8 COLUMN 12
    AND ON OCCURRENCE GO TO STEP 7;
SPECIFY INTERRUPT < PFK5-PB >
    AND ON OCCURRENCE GO TO STEP 20;
. . . . . .
ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL;
```

The first SPECIFY in the example defines an interrupt for a Transmit Cursor Function Key. When the cursor is positioned to Line 8 Column 12 PAGE-A and the Transmit Cursor Function Key is depressed, control will be transferred to STEP 7. The second SPECIFY in the example defines an interrupt for Programmable Function Key 5 on PAGE-B. When PAGE-B is being viewed and PFK 5 is depressed, control will be transferred to STEP 20. The last statement in the example activates Interrupt Processing for the GOAL procedure (i.e., the LEVEL). No interrupts will be received by the procedure unless Interrupt Processing is activated.

12.2.3 HOW TO REQUEST A PFP FUNCTION KEY INTERRUPT

Requesting a PFP Function Key interrupt is similar to requesting a DG Programmable Function Key interrupt. Both a SPECIFY INTERRUPT Statement and an ACTIVATE INTERRUPT Statement must be used.

Examples

```
SPECIFY INTERRUPT < PFPK2 >
    AND ON OCCURRENCE GO TO STEP 7;
. . . . . .
ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL;
```

In the example, when PFP Function Key 2 is depressed when viewing the default PFP Presentation, an interrupt will be transmitted to the GOAL procedure and control will be transferred to STEP 7.

12.2.4 HOW TO REQUEST A SYSTEM STATUS INTERRUPT

In order to request an interrupt on a System Status Function Designator, both a SPECIFY INTERRUPT and an ACTIVATE INTERRUPT Statement must be used.

Example

```
SPECIFY INTERRUPT <SGS1ASTAT> AND ON OCCURRENCE GO TO STEP 10;
. . . . . .
ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL;
```

In the example, when the status of the GS1A FEP changes, an interrupt will occur, and control will be transferred to STEP 10.

12.2.5 HOW TO REQUEST A CDT, MET OR A TIMER INTERRUPT

Requesting a CDT, MET or a Timer interrupt requires use of the WHEN INTERRUPT Statement. The SPECIFY INTERRUPT Statement is not used.

Examples

```
ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL;
     .                .
     .                .
     .                .
WHEN INTERRUPT < CDT >  EQUAL TO -20 MIN CDT OCCURS
    GO TO STEP 8;
WHEN INTERRUPT < TIMER > OCCURS
    GO TO STEP 30;
WHEN INTERRUPT < METIME > EQUAL TO 5 MIN MET OCCURS
    GO TO STEP 10;
```

Note that the first requirement for receiving an interrupt on CDT, MET, or the Interval Timer is to activate Interrupt Processing. This activation is accomplished by the first statement in the example. Next, a WHEN INTERRUPT Statement may be used as a procedural statement to define the requested interrupt. The second statement in the example will cause control to be transferred to STEP 8 when CDT is equal to -20 MIN CDT. The third statement in the example will cause control to be transferred to STEP 30 when the Interval Timer reaches zero. The fourth statement will cause control to be transferred to STEP 10 when MET is equal to 5 MIN MET.

12.3 HOW TO PROCESS AN INTERRUPT

When an interrupt is transmitted to a GOAL procedure, two things occur: Interrupt Processing is inhibited for the procedure (i.e., the level) and control is passed to the step specified on the SPECIFY INTERRUPT or WHEN INTERRUPT Statement. Interrupt Processing is inhibited automatically by the system so that the appropriate statements can be executed in response to the interrupt without having another interrupt occur and transfer control to another step. After the GOAL procedure completes the required action in response to the interrupt, an activate Interrupt Processing should be used to allow the procedure to receive additional interrupts.

An option is available on the ACTIVATE INTERRUPT Statement to transfer control back to the point in the procedure where the execution path was altered by the interrupt. This transfer is accomplished by specifying:

```
ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL AND RETURN.
```

Also, if a measurement interrupt occurs, no additional interrupt notifications for the measurement involved will be sent from the FEP until another ACTIVATE FEP INTERRUPT CHECK FOR is executed.

Example

```
        $ INTERRUPT PROCESSING ROUTINE $
STEP 7  SET < DSI > TO OFF;
          .                .
          .                .
          .                .
        ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL;
```

In the example, assume that an interrupt occurred which caused a transfer to STEP 7. After the required processing is accomplished, an ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL is executed to enable the procedure to receive additional interrupts if they occur.

12.4 SPECIFY INTERRUPT STATEMENT

SPECIFY INTERRUPT STATEMENT

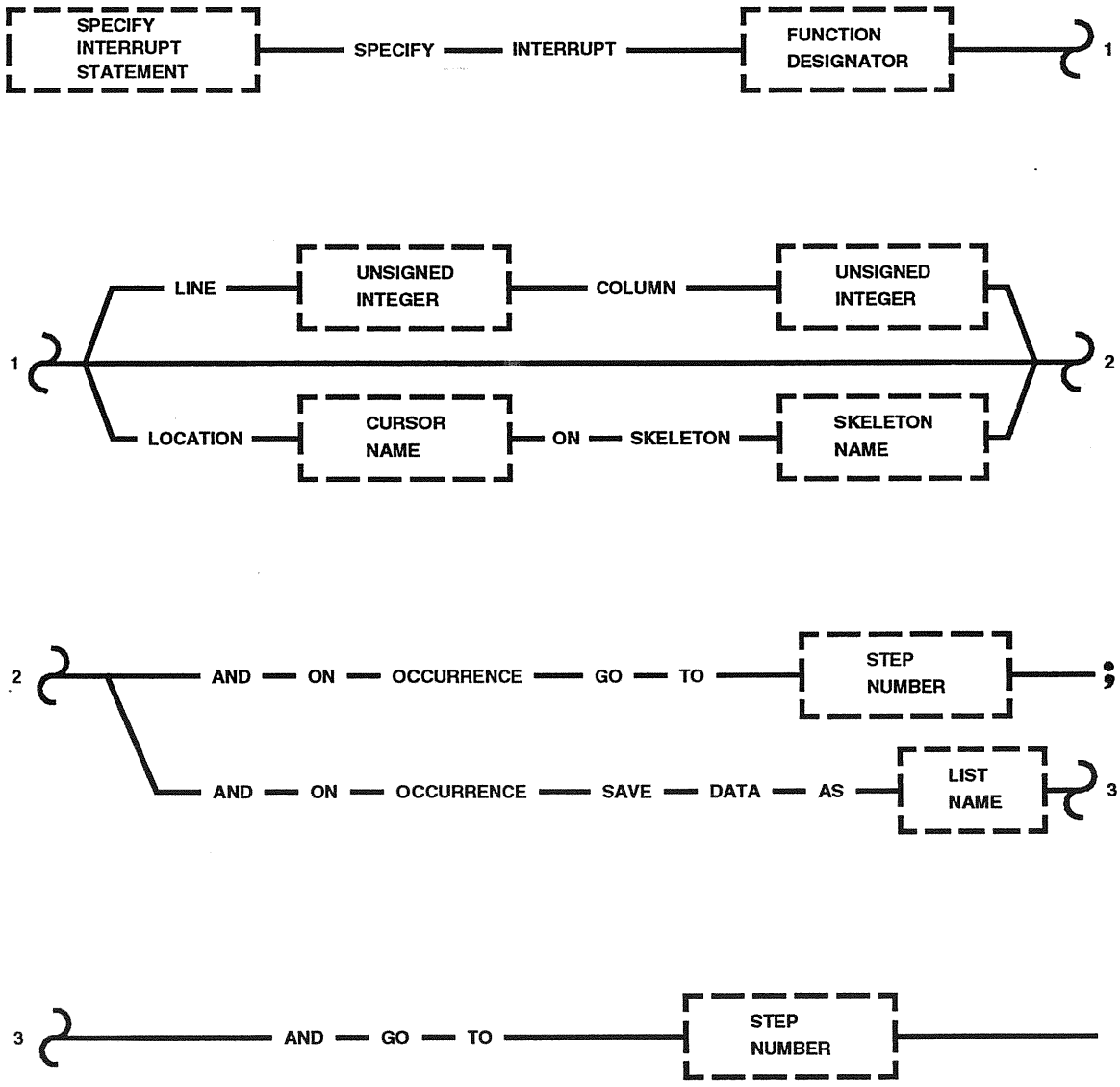


Figure 12-1 SPECIFY INTERRUPT Statement

Function

The SPECIFY INTERRUPT Statement is used to define the external events for which a GOAL procedure needs notification in real time.

Description

This statement accomplishes three purposes: (1) it defines an external condition to be used to interrupt a GOAL procedure, (2) it specifies a Step Number to GO TO when the interrupt occurs, and (3) specifies the list for saving the data words of a REMCOM with data interrupt. The list must be a numeric list at least 7 words long. The first 4 words will receive the passed data. Entries 5, 6, and 7 will receive the system data. The 3 words of system data consist of the LPORT of the sender, the acknowledging Function Designator's REMCOM number, and the system data's checksum. SPECIFY INTERRUPT Statements are non-procedural statements and must appear before any GOAL procedural statement.

Example

```
SPECIFY INTERRUPT < DM1 > AND ON OCCURRENCE GO TO STEP 5;
```

In the example, when the discrete measurement < DM1> goes to the exception state, a branch to STEP 5 will occur in the GOAL procedure.

The Line and Column option and the Location On Skeleton option are used with the following DG Function Keys: Transmit Cursor, Execute Command, and Disarm Command. When the cursor is positioned to the specified coordinates on the display page and the specified DG Function key is depressed, an interrupt will be transmitted to a GOAL procedure.

NOTE: *The Transmit Cursor, Execute Command, and Disarm Command Function Keys are the same functionally. They generate interrupts to a GOAL procedure. The procedure must be written to accomplish the required actions as a result of the interrupt, e.g., executing a command or disarming a command.*

The types of interrupts which may be specified include: Measurement Exception, DG Function Keys, PFP Function Keys, Remote Communications, and System Interrupts.

SPECIFY Measurement

Example

```
SPECIFY INTERRUPT < AM1 > AND ON OCCURRENCE GO TO STEP 5;
```

In the example, when the value of < AM1 > is out of limits, normal execution of the procedure will be interrupted, and control will be transferred to STEP 5.

SPECIFY DG Programmable Function Key

Example

```
SPECIFY INTERRUPT < PFK1-P2A >
    AND ON OCCURRENCE GO TO STEP 40;
```

If the operator depresses DG Programmable Function Key 1 while viewing PAGE-2A, normal execution of the procedure will be interrupted and control will be transferred to STEP 40.

SPECIFY DG Function Keys with Cursor Position

Examples

```
SPECIFY INTERRUPT < XMIT-PA > LINE 10 COLUMN 35
    AND ON OCCURRENCE GO TO STEP 15;
SPECIFY INTERRUPT < EXEC-P1A > LINE 5 COLUMN 5
    AND ON OCCURRENCE GO TO STEP 5;
SPECIFY INTERRUPT < DISARM-PA > LINE 5 COLUMN 5
    AND ON OCCURRENCE GO TO STEP 5;
```

In the first example, when the cursor is positioned to line 10 column 35 on PAGE-A and the Transmit Cursor Function Key is depressed, normal execution of the procedure will be interrupted, and control will be transferred to STEP 15. The second and third examples are similar. In the second example, the interrupt is generated by positioning the cursor to line 5 column 5 while viewing PAGE-1A and depressing the Execute Command Function Key. In the third example, the interrupt is generated by positioning the cursor to line 5 column 5 while viewing PAGE-A and depressing the Disarm Command Function Key.

SPECIFY DG Function Keys with Symbolic Cursor Position

Examples

```
SPECIFY INTERRUPT < XMIT-PA > LOCATION (LOC1) ON SKELETON
    (DSKEL1) AND ON OCCURRENCE GO TO STEP 5;
SPECIFY INTERRUPT < EXEC-P1A > LOCATION (LOC2) ON SKELETON
    (DSKEL2) AND ON OCCURRENCE GO TO STEP 15;
SPECIFY INTERRUPT < DISARM-PA > LOCATION (LOC3) ON SKELETON
    (DSKEL3) AND ON OCCURRENCE GO TO STEP 5;
```

In the first example, when the cursor is positioned on the symbolic location identified by (LOC1) on Display Skeleton (DSKEL1) on PAGE-A and the Transmit Cursor Function Key is depressed, normal execution of the procedure will be interrupted, and control will be transferred to STEP 5. The second and third examples are similar. In the second example, the interrupt is generated by positioning the cursor on symbolic location identified by (LOC2) on Display Skeleton (DSKEL2) while viewing PAGE-1A and depressing the Execute Command Function Key. In the third example, the interrupt is generated by positioning the cursor to symbolic location (LOC3) on Display Skeleton (DSKEL3) while viewing PAGE-A and depressing the Disarm Command Function Key.

SPECIFY PFP Function Key

Examples

```
SPECIFY INTERRUPT < PFPK3 >
    AND ON OCCURRENCE GO TO STEP 3;
SPECIFY INTERRUPT < PFPK1-P6 >
    AND ON OCCURRENCE GO TO STEP 5;
```

In the first example, when the operator depresses PFP Function Key 3 on the default PFP Presentation, the procedure will be interrupted, and control will be transferred to STEP 3. In the second example, when the operator depresses PFP Function Key 1 while viewing Presentation 6, the procedure will be interrupted, and execution will continue at STEP 5.

SPECIFY Interrupt Occurrence in a Higher Level

Example

```
SPECIFY INTERRUPT < MEAS-INT > AND
    ON OCCURRENCE GO TO STEP 50;
```

The interrupt in the example will occur when a measurement exception occurs which a procedure in a higher level has specified. For instance, assume a level 1 procedure specifies an interrupt on a measurement and then performs a level 2 procedure which contains the SPECIFY Statement in the example. When the measurement goes to the exception condition, the interrupt for a pending interrupt in a higher level (i.e., < MEAS-INT >) will occur in level 2. Control will be transferred to STEP 50 so that the level 2 procedure can terminate. Execution will transfer to the level 1 procedure so that the measurement exception interrupt will occur and be processed.

SPECIFY REMCOM with Data

Example

```
SPECIFY INTERRUPT <RCDATA> AND ON OCCURRENCE SAVE DATA
    AS (NUMLIST7) AND GO TO STEP 25;
```

The interrupt in the example will occur when a REMCOM with data interrupt is sent, returned, or relayed from another concurrency. Normal execution of the procedure will be interrupted, the 7 data words (4 passed data words plus 3 system words) will be saved in (NUMLIST7), and control will be transferred to Step 25.

Statement Execution

The actual execution of the SPECIFY INTERRUPT Statement does not take place until an interrupt event occurs. At perform time a 512 word table containing all the I/C addresses of the program's interrupt routines is written to bulk memory. When an interrupt occurs, statement processing is flagged and the I/C address of the interrupt routine is read from bulk memory. Once the I/C address is obtained an I/C refresh may be required to obtain the I/C for the interrupt routine. Thus, the next GOAL Statement executed will be the statement which was specified to process the interrupt.

Interrupts are prioritized by type and within this type they are prioritized by order of specification. If more than one interrupt

occurs before the first one can be completed (this can happen if a burst of interrupts occur or if interrupt processing was procedurally inhibited for a time), the fact that the interrupt occurred will be saved, but the order in which they occurred will not be saved. Priority for handling these "stacked" interrupts will be first by type. All measurement interrupts will be processed first, followed by DG keys, PFP keys, remote communication, system CDT/MET and timer.

Priority for handling these "stacked" interrupts within a single interrupt type (such as measurements), will be by order of the SPECIFY Statements in the program listing. This order may be perturbated, however, by level performs. The order of SPECIFY INTERRUPT Statements in a level 4 program will not define the priority of interrupts for the level if any of the events have been specified on a previous level. If this were the case, the priority of any events specified on a previous level as well as level 4 would have priority over the others defined on level 4. The priority of the events specified on the previous levels could be determined with a similar analysis of level 3, level 2, and then level 1.

When an interrupt event occurs, the executor normally flags the interrupt as being pending on all levels interested in it. A level is interested if the interrupt is specified and enabled on that level. The interrupt is processed only once however, and is normally processed by the lowest level program which is interested in it. The exception to this rule occurs if the lowest level is keyboard terminated before the executor has the chance to process the interrupt. In this case, the interrupt will be processed by the next lowest level program which is interested in it.

The specification of "above me" or "class" interrupts (<MEAS-INT>, <DG-INT>, <PFP-INT>, <REMCOM-INT> and <SYS-INT>), causes special processing to be done by the executor when an interrupt occurs which has been specified on a previous (higher) level. "Class" interrupts are generated by the executor when a normal interrupt event occurs for which the current active level is not interested, but some previous (higher) level is. If the current level is interested in the "class" interrupt corresponding to the interrupt type of the occurring interrupt, then the "class" interrupt will be made pending on the current level. The "class" interrupts are also generated or canceled by the executor when transferring to a new level, which is interested in them, via the PERFORM LEVEL PROGRAM or LEVEL TERMINATE

Statements. A "class" interrupt will be generated when transferring to a new level if a specific interrupt corresponding to the class is pending for some higher level, but not for the new level. A "class" interrupt will be cancelled when transferring to a new level if no specific interrupt corresponding to the class is pending for some higher level and also not pending for the new level. Figure 12-2 illustrates how this process works on a few specific cases.

NOTE: *In Figure 12-2 , Case #5, <AI-01> was made pending on levels 1 & 4. However, it will get processed only once (on level 4, if the program is not level terminated first). If level 4 is terminated before the processing of <AI-01> occurs, then <MEAS-INT> will be made pending when execution returns to level 2 since <AI-01> will still be pending on level 1.*

	CASE #1	CASE #2	CASE #3	CASE #4	CASE #5
LEVEL 1	<AI-01> INH	INH	ENAB	ENAB	ENAB
2	<AI-01> ENAB <MEAS-INT> ENAB	NOT SPEC NOT SPEC	NOT SPEC NOT SPEC	INH ENAB	INH ENAB
3	<AI-01> INH <MEAS-INT> ENAB	INH ENAB	INH ENAB	INH NOT SPEC	INH NOT SPEC
4	<AI-01> INH <MEAS-INT> ENAB	INH ENAB	INH ENAB	INH ENAB	ENAB ENAB
INTERRUPTS MADE PENDING IN EACH CASE IF EXCEPTION OCCURS FOR <AI-01> WHILE PROGRAM ON LEVEL 4	<AI-01> LEVEL 2 <MEAS-INT> ON LEVEL 4	NEITHER <AI-01> OR <MEAS-INT> ANYWHERE	<AI-01> ON LEVEL 1 <MEAS-INT> ON LEVEL 4	<AI-01> ON LEVEL 1 <MEAS-INT> ON LEVEL 4	<AI-01> ON LEVELS 1 AND 4 SET PENDING BOTH PLACES, BUT WILL GET PROCESSED ONLY ONCE (LEVEL 4, IF PROGRAM NOT LEVEL TERMINATED FIRST)

IN ABOVE CHART:

1. INH = INTERRUPT SPECIFIED, BUT INHIBITED.
   ENAB = INTERRUPT SPECIFIED AND ENABLED.
2. ASSUME EXCEPTION FOR <AI-01> OCCURS WHILE PROGRAM IS ON LEVEL 4.

Figure 12-2 Stacked Interrupt Processing Examples

Error Processing

If the GOAL Executor encounters an Undefined FD while processing the WHEN INTERRUPT BLOCKS, it sets all level control flag bits OFF so that no interrupts can match the Undefined FD.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Interrupts are only passed to the procedure between the execution of successive statements. Interrupts will not be processed until statement execution is complete. For example, if a GOAL procedure is executing a DELAY Statement such as:

```
DELAY UNTIL < DM3 > IS OFF:
```

No interrupts will be processed and given to the procedure until <DM3> is OFF.

B. Interrupts are processed only when a GOAL program is actively executing procedural statements. If an interrupt occurs while this is not the case, its processing will be deferred until active execution resumes. Interrupts are not processed during the following conditions:

1. While executing another program which was called using the PERFORM Statement.
2. During a delay caused by the AFTER/WHEN prefix.
3. While a STOP Statement is being processed.
4. When a keyed-in entry is requested on the RECORD Statement.
5. During execution of a DELAY Statement, the When Interrupt option is not utilized.
6. While a GOAL program is manually stopped (HOLD).
7. While interrupt processing on this level is inhibited.

C. An interrupt event is processed only by the lowest program level that has a SPECIFY INTERRUPT for that event. If a task is executing a procedure at level 3 and level 1, 2 and 3 all want interrupt notification on the same event, then level 3 will be interrupted. Level 1 and level 2 will not be given interrupt notification for that event. Class ("above me") interrupts are exceptions to this rule.

D. ALL SPECIFY INTERRUPT Statements must appear before any GOAL procedural statement.

E. A Pseudo Parameter FD must appear in a DEFINE Statement before an interrupt may be specified on it.

F. The LINE and COLUMN or the LOCATION ON SKELETON option must be used to define the DG cursor position associated with Transmit Cursor, Execute Command, and Disarm Command Function Designator events. These options may not be used for any other Function Designator events. Line numbers must be in the range 4-33 and column numbers in the range 0-71.

G. Multiple interrupts may be specified for the special DG Function Keys (XMIT, EXEC, DISARM) provided they are for different cursor positions. All other Function Designators may only be on one SPECIFY INTERRUPT Statement.

H. The order of the SPECIFY Statement within a group dictates the priority of processing when two or more are pending simultaneously.

I. A SPECIFY INTERRUPT Statement for a REMCOM With Data Function Designator must specify a numeric list with at least 7 entries in which to save the REMCOM data words. The first 4 words are available for user numeric data and entries 5, 6 and 7 are reserved for system data. The GOAL program must not alter entries 5, 6 and 7. The list should be initialized to zero.

TABLE 12-1 LEGAL FUNCTION DESIGNATORS SPECIFY INTERRUPT STATEMENTS

STATEMENT	TYPE	SOURCE	CLASS
SPECIFY INTERRUPT <FD>	ANALOG MEASUREMENT (AM), DISCRETE MEASUREMENT (DM), DIGITAL PATTERN MEASUREMENT (DPM), ANALOG MEASUREMENT DOUBLE PRECISION (AMDP), MULTI-WORD DIGITAL PATTERN (MWDP), FLOATING POINT (FP), DATA SET (DSFD) -SUBTYPE 3 SYSTEM STATUS AREA 1 (SSA1), SYSTEM STATUS AREA 2 (SSA2), SPECIAL DG FUNCTION KEY (DGKY), REMOTE COMMUNICATION (COM), PROGRAMMABLE FUNCTION KEY (PFK), SYSTEM INTERRUPT (SI), PFP FUNCTION (PFPK)	NONE, [a]GSE, PCM RTU	N/A

a. DPM FD's with a source of GSE are invalid with this statement.

12.5 ACTIVATE NOTIFICATION STATEMENT

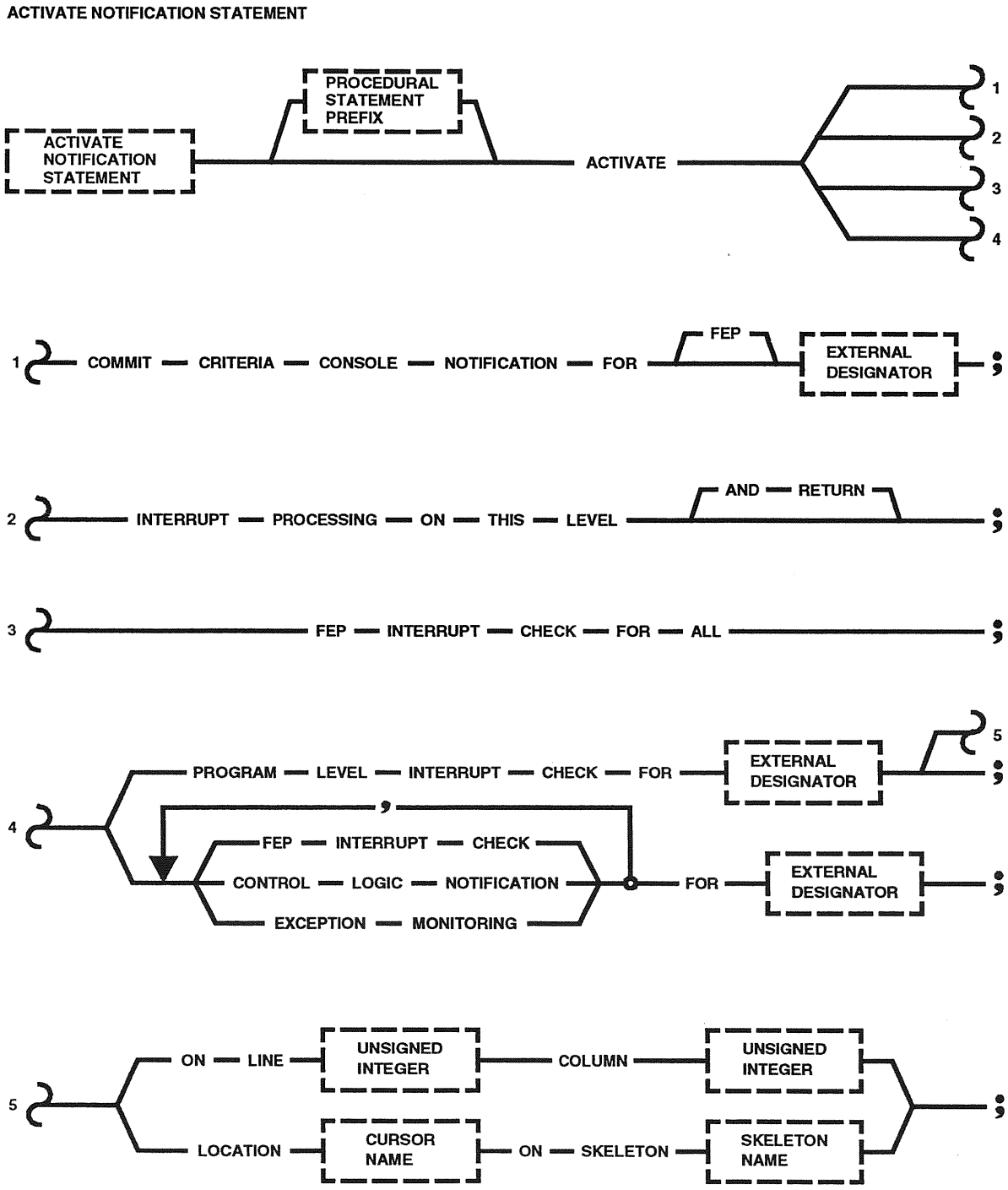


Figure 12-3 ACTIVATE NOTIFICATION Statement

Function

The ACTIVATE NOTIFICATION Statement, in conjunction with the INHIBIT Statement, is used to control the status of interrupt processing in a GOAL procedure and in the CCMS.

Description

The ACTIVATE NOTIFICATION Statement controls the active status of the interrupt events defined by the SPECIFY INTERRUPT Statements or by processed WHEN INTERRUPT Statements.

When a GOAL procedure is performed, Interrupt Processing for the procedure is inhibited. When Interrupt Processing for a procedure (or level) is inhibited, all processing of interrupt events is deferred. Deferring processing means that if an interrupt occurs, the interrupt event will be remembered, but the GOAL procedure will not receive the interrupt until Interrupt Processing is activated. Interrupt Processing is activated by means of the ACTIVATE NOTIFICATION Statement.

The following options may be used to activate interrupt processing:

A. ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL

This option is used to activate interrupt processing for an entire procedure (or level). All interrupts defined by SPECIFY and WHEN INTERRUPT Statements will be activated.

B. ACTIVATE PROGRAM LEVEL INTERRUPT CHECK FOR

This option is used to activate interrupt processing for specific events that must have previously been defined with SPECIFY Statements. This option has no effect unless the INHIBIT PROGRAM LEVEL INTERRUPT CHECK FOR option has been previously executed.

Also, in order for a GOAL procedure to ensure receiving an interrupt on a measurement, the FEP must be requested to send notification of the measurement exception event to GOAL. This request is accomplished by means of the ACTIVATE FEP INTERRUPT CHECK FOR option. When exception events occur for the Function Designators referenced in this statement, the FEP will send notification to the console regarding the exceptions. This notification affects all program tasks within the console that

have SPECIFY Statements for the exception events. Once FEP notification has been sent, no further notification will be given for reported exceptions until the FEP has again been activated.

A. ALL Option:

The ALL option may be used with ACTIVATE FEP INTERRUPT CHECK FOR to request FEP notification for all measurements defined by SPECIFY Statements within a procedure.

B. AND RETURN Option:

The AND RETURN option is used with ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL as the last statement in an interrupt processing routine. When an interrupt is received by a GOAL procedure, further interrupt processing for the level is automatically inhibited. This permits an interrupt to be processed without receiving another interrupt before processing is complete on the first. The last statement in an interrupt processing routine should therefore be a request to activate interrupt processing so that additional interrupts may be received. If the ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL AND RETURN option is used, execution of the procedure will branch to and continue at the point it received the original notification of the interrupt. This option allows normal execution of the interrupted procedure to continue.

C. LINE and COLUMN Option and LOCATION ON SKELETON Option:

These options are used to define the DG cursor position associated with the Transmit Cursor, Execute Command, and Disarm Command Function Keys. These options may not be used for any other Function Designators.

D. COMMIT CRITERIA CONSOLE NOTIFICATION:

This option activates dual notification of specified Function Designators. If COMMIT CRITERIA CONSOLE (CCC) NOTIFICATION is active on both the Function Designator and on the FEP, system exception notification will be reported to the responsible system console. Control Logic exception notification will be reported only to the Commit Criteria Console. GOAL exception notification will be reported to both. If either the Function

Designator or the FEP has a CCC status of inactive, all exceptions will be reported to the Responsible Console. Control Logic sequences will execute only in the console to which the CL notification was reported.

If CCC notification is active on both the Function Designator and on the FEP all GOAL requests for that Function Designator will be processed by the Commit Criteria Console only. These GOAL requests are issued via the ACTIVATE/INHIBIT SYSTEM statement, the CHANGE statement, and the ACTIVATE/INHIBIT NOTIFICATION statement. (Refer to Sections 11.2, 11.3, 12.5, and 12.6.) If CCC notification is inactive on either the Function Designator or on the FEP, all GOAL requests for that Function Designator will be handled by the Responsible Console.

E. CONTROL LOGIC NOTIFICATION:

When CONTROL LOGIC NOTIFICATION is activated, Control Logic will be notified by the FEP of measurement exception conditions. This notification may result in initiation of a reactive sequence.

F. EXCEPTION MONITORING:

When EXCEPTION MONITORING (EMON) is activated, the system exception monitor in the console will be notified of measurement exception conditions, and the exceptions will be displayed to the Exception Monitor Pages.

G. FEP INTERRUPT CHECK (GOAL LIMITS CHECK):

When a GOAL limits violation occurs, exception information is routed to the measurement's responsible console. If CCM notification is active for the measurement and the FEP GLOBAL CCM indicator is on, notification of limits violation for the measurement is also sent to the Measurements Commit Criteria console.

SYSTEM LIMITS CHECK

When a system limits violation occurs, exception information is routed to the measurement's responsible console only for the following conditions:

A. EMON notification is active.

B. Control Logic notification is active, and the CCM indicator for the measurement or the FEP GLOBAL CCM indicator is inhibited.

For dual notification of system limits violations, EMON notification, and CONTROL LOGIC notification, the CCM indicator for the measurement and the FEP GLOBAL CCM indicator must be active.

The responsible console will receive EMON and GOAL notification. The Commit Criteria console will receive GOAL and Control Logic notification.

INTERRUPT PROCESSING Option

Example

```
ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL;
```

This option activates interrupt processing for the level in which it is executed.

Example

```
ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL AND RETURN;
```

This option is used to complete the processing of an interrupt event. The AND RETURN option will cause execution of the interrupted GOAL procedure to continue at the point it received notification of the interrupt. This allows normal execution of the interrupted GOAL procedure to continue.

PROGRAM LEVEL INTERRUPT CHECK FOR Option

Examples

```
ACTIVATE PROGRAM LEVEL INTERRUPT CHECK FOR < DM1 >;
ACTIVATE PROGRAM LEVEL INTERRUPT CHECK FOR < DM1 >
    < DM2 > < DPM3 > < AM1 >;
ACTIVATE PROGRAM LEVEL INTERRUPT CHECK FOR (STABLE1) FUNCTIONS;
ACTIVATE PROGRAM LEVEL CHECK FOR < XMIT-P1A > ON
    LINE 8 COLUMN 37;
```

These examples activate Interrupt Processing for only the individual Function Designators specified. The first three examples activate Interrupt Processing for measurement Function Designators. The third example activates Interrupt Processing using Table Functions. Row Designators for rows which are inhibited will not be affected. The last example activates Interrupt Processing for the XMIT CURSOR Function Key. To generate the interrupt, the cursor must be positioned to Line 8 Column 37 on PAGE-1A and the XMIT Cursor Function Key depressed.

NOTE: *Program Level Interrupt Check is normally active, therefore, this statement has no effect unless the INHIBIT PROGRAM LEVEL INTERRUPT CHECK FOR OPTION has been previously executed.*

FEP INTERRUPT CHECK FOR Option

Examples

```
ACTIVATE FEP INTERRUPT CHECK FOR  < DM1 >;
ACTIVATE FEP INTERRUPT CHECK FOR  < AM5 >
                                  < AM1 >
                                  < AM4 >
                                  < DPM1 >
                                  < DM5 >;
ACTIVATE FEP INTERRUPT CHECK FOR (QTABLE1) FUNCTIONS;
```

These examples request notification from the FEP when measurement exceptions occur for the Function Designators specified.

Example

```
ACTIVATE FEP INTERRUPT CHECK FOR ALL;
```

This statement requests GOAL notification of exceptions to be activated for all measurement Function Designators referenced in SPECIFY INTERRUPT Statements in this procedure.

Example

```
ACTIVATE FEP INTERRUPT CHECK, CONTROL LOGIC NOTIFICATION,
    EXCEPTION MONITORING FOR <DM3>;
```

This statement requests that the FEP activate notification of all three types of exceptions (GOAL, Control Logic, Exception Monitoring) to the console.

COMMIT CRITERIA CONSOLE NOTIFICATION Option

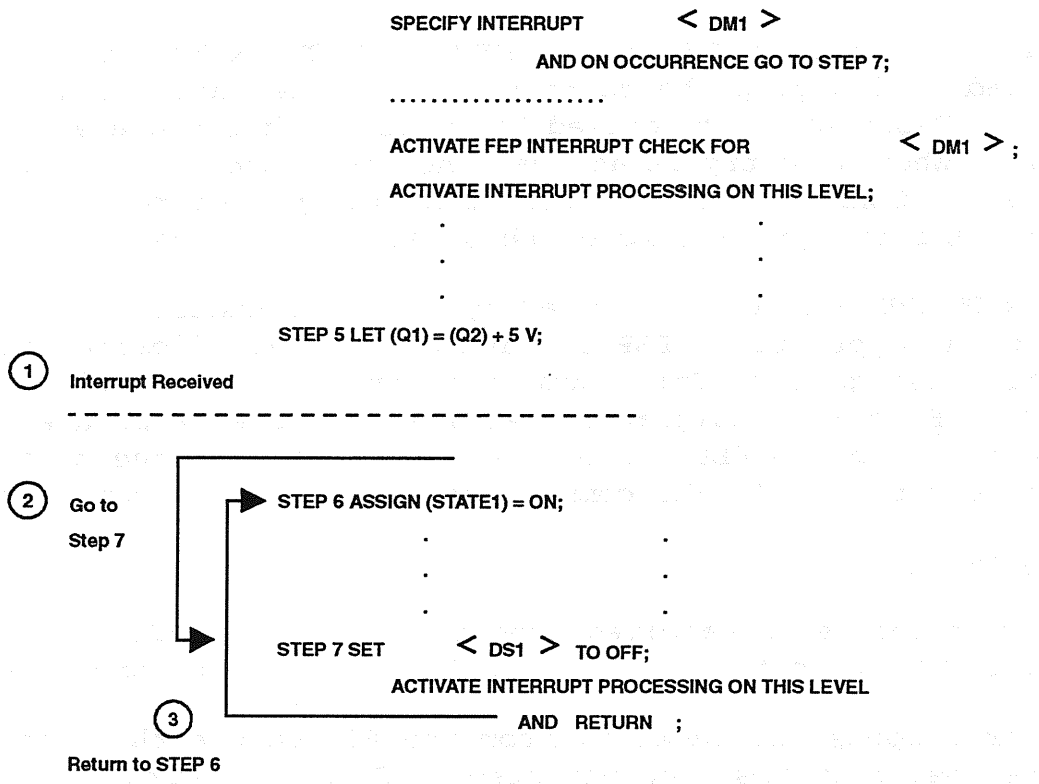
Example

```
ACTIVATE COMMIT CRITERIA CONSOLE NOTIFICATION FOR
    <AM1> <DM1>;
```

This example will result in notification to the Commit Criteria Console being activated. When the specified Function Designators enter their exception condition, the Commit Criteria Console as well as the console responsible for the measurements will be notified.

Application

Example



The example illustrates an application of the ACTIVATE NOTIFICATION Statement. The required SPECIFY and ACTIVATE Statements for receiving an interrupt on < DM1 > are shown. The interrupt for< DM1 > occurs between STEP 5 and STEP 6, and control is transferred to STEP 7. After processing the interrupt, interrupt processing is activated using the AND RETURN option. Control is transferred back to STEP 6, the point of the interrupt.

Statement Execution

The execution of the INTERRUPT PROCESSING ON THIS LEVEL option is accomplished by clearing the level interrupt flag for this procedure. Any pending or new interrupts will now be processed.

The execution of the PROGRAM LEVEL INTERRUPT CHECK FOR option is accomplished by flagging the entries in the interrupt blocks for the Function Designators specified to indicate that these entries are active. When an entry is active, the occurrence of the event will cause the GOAL Executor to interrupt the procedure, providing interrupt processing on this level is active.

FEP INTERRUPT CHECK FOR is executed by the GOAL Executor by requesting the appropriate FEP to turn on GOAL notification for the Function Designator. This communication takes place via the CDBFR. Each Function Designator specified requires a separate communication with the FEP. Execution will not continue to the next statement until all FEP communications have been completed.

Error Processing

If an error response is received from the FEP on the ACTIVATE FEP INTERRUPT CHECK FOR option, a Class III error will be processed.

If an error response is received from the FEP before the last repetitive operation has been executed, a Class III error will be processed at that point. If procedure error override is active, the statement execution will continue. If procedure error override is inhibited, the operation of the procedure will be stopped.

If the procedure attempts to execute an AND RETURN option and the GOAL executor does not have a return address because the statement was not executed as the result of an interrupt, a Class II error will be processed. The execution of the procedure will be stopped with an appropriate error message output to PAGE-A. The operator must restart the procedure at a desired location.

If the GOAL Executor encounters an Undefined FD while executing the PROGRAM LEVEL INTERRUPT CHECK FOR option, it will output a Class III error and skip to the next entry.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Interrupt processing for the level will be automatically inhibited when interrupt notification is passed to the procedure. This means an INHIBIT INTERRUPT PROCESSING ON THIS LEVEL is executed automatically when any interrupt notification is passed to the procedure. Only one level of interrupt processing should be programmed at one time. Therefore, the procedure writer should not ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL in response to an interrupt until all processing required to react to the interrupt has been completed.

B. The Display Generator cursor position must be specified as either ON LINE and COLUMN or LOCATION ON SKELETON if the Transmit Cursor, Execute Command and Disarm Command Display Generator Function Keys are referenced.

C. Whenever a program is performed, whether it is requested via the keyboard, a PERFORM Statement, or a CONCURRENT Statement, interrupt processing for the procedure is initially inhibited, and any interrupt events which occur will be saved. Before any interrupts are sent to the procedure, INTERRUPT PROCESSING must be activated.

D. The FEP option may only refer to CDBFR resident measurement type Function Designators.

E. The External Designator specified with the PROGRAM LEVEL INTERRUPT CHECK option must have been referred to on a SPECIFY INTERRUPT Statement or by a processed WHEN INTERRUPT Statement for CDT, MET or TIMER interrupts.

F. The FEP option affects all program tasks within the console that have SPECIFY INTERRUPT Statements on measurements processed by the FEP.

G. If the FEP option specifies a measurement which is in an exception condition when the FEP receives the activate GOAL notification request, an exception notification will immediately be sent to the console.

H. GOAL notification is sent to the console only once when the Function Designator violates the defined exception conditions. Exception checking for GOAL notification is

then discontinued by the FEP and may only be restarted by the execution of another ACTIVATE FEP INTERRUPT CHECK request.

I. The statement following an ACTIVATE...AND RETURN Statement should have a Step Number.

J. A Floating Point Function Designator must have a valid set of limits in the hardware record to be specified with the EXCEPTION MONITORING, PROGRAM LEVEL INTERRUPT CHECK, FEP INTERRUPT CHECK, or CONTROL LOGIC NOTIFICATION options. The error returned for the FEP will be, 'RQST FAILED - FEP/REQUESTED FUNCTION NOT VALID,' when an invalid Floating Point Function Designator is specified.

K. An ACTIVATE...AND RETURN Statement is not allowed within a Repeat Group.

L. ACTIVATE CONTROL LOGIC and EXCEPTION MONITORING are not valid for GSE Digital Patterns.

M. The ACTIVATE PROGRAM CHECK option has no effect unless the INHIBIT PROGRAM LEVEL CHECK option has been executed.

N. The ACTIVATE COMMIT CRITERIA option used with a THDS FD, will work only with Type III THDS FD's or members of a Type II Time Homogeneous Dataset. Specifying a Type II THDS FD or a member of a Type III THDS will cause a class III error to be processed from the FEP.

TABLE 12-2 LEGAL FUNCTION DESIGNATORS ACTIVATE NOTIFICATION STATEMENTS (1 OF 2)

STATEMENT	TYPE	SOURCE	CLASS
COMMIT CRITERIA CONSOLE NOTIFICATION FOR FEP <FD>	[a]FRONT END PROCESSOR (FEP)	NONE	N/A
COMMIT CRITERIA CONSOLE NOTIFICATION FOR <FD> CONTROL LOGIC NOTIFICATION FOR <FD> EXCEPTION MONITORING FOR <FD> FEP INTERRUPT CHECK FOR <FD>	ANALOG MEASUREMENT (AM) [b]DISCRETE MEASUREMENT (DM) DIGITAL PATTERN MEASUREMENT (DPM) ANALOG MEASUREMENT DOUBLE PRECISION (AMDP) MULTI-WORD DIGITAL PATTERN (MWDP) FLOATING POINT (FP) DATA SET (DSFD) -SUBTYPE 3	NONE **GSE PCM RTU	

a. The following FEP FD's are not valid for the COMMIT CRITERIA CONSOLE NOTIFICATION FOR FEP <FD> statement: LDBA, LDBD, LDBS, UPLK, HOSR, and HOSC.

b. DPM FD's with a source of GSE are invalid with these statements.

TABLE 12-2 LEGAL FUNCTION DESIGNATORS ACTIVATE NOTIFICATION STATEMENTS (2 OF 2)

STATEMENT	TYPE	SOURCE	CLASS
PROGRAM LEVEL INTERRUPT CHECK FOR <FD>	ANALOG MEASUREMENT (AM) DISCRETE MEASUREMENT (DM) **DIGITAL PATTERN MEASUREMENT (DPM) ANALOG MEASUREMENT DOUBLE PRECISION (AMDP) MULTI-WORD DIGITAL PATTERN (MWDP) FLOATING POINT (FP) DATA SET (DSFD) -SUBTYE 3 SYSTEM STATUS AREA 1 (SSA1) SYSTEM STATUS AREA 2 (SSA2) SPECIAL DG FUNCTION KEY (DGKY) INTERNAL TIMER - GMT - CDT - MET - JTOY (TIME) REMOTE COMMUNICATION (COM) PROGRAMMABLE FUNCTION KEY (PFK) SYSTEM INTERRUPT (SI) PFP FUNCTION KEY (PFPK)	NONE **GSE, PCM RTU	N/A

** DPM FD's with a source of GSE are invalid with these statements.

12.6 INHIBIT NOTIFICATION STATEMENT

INHIBIT NOTIFICATION STATEMENT

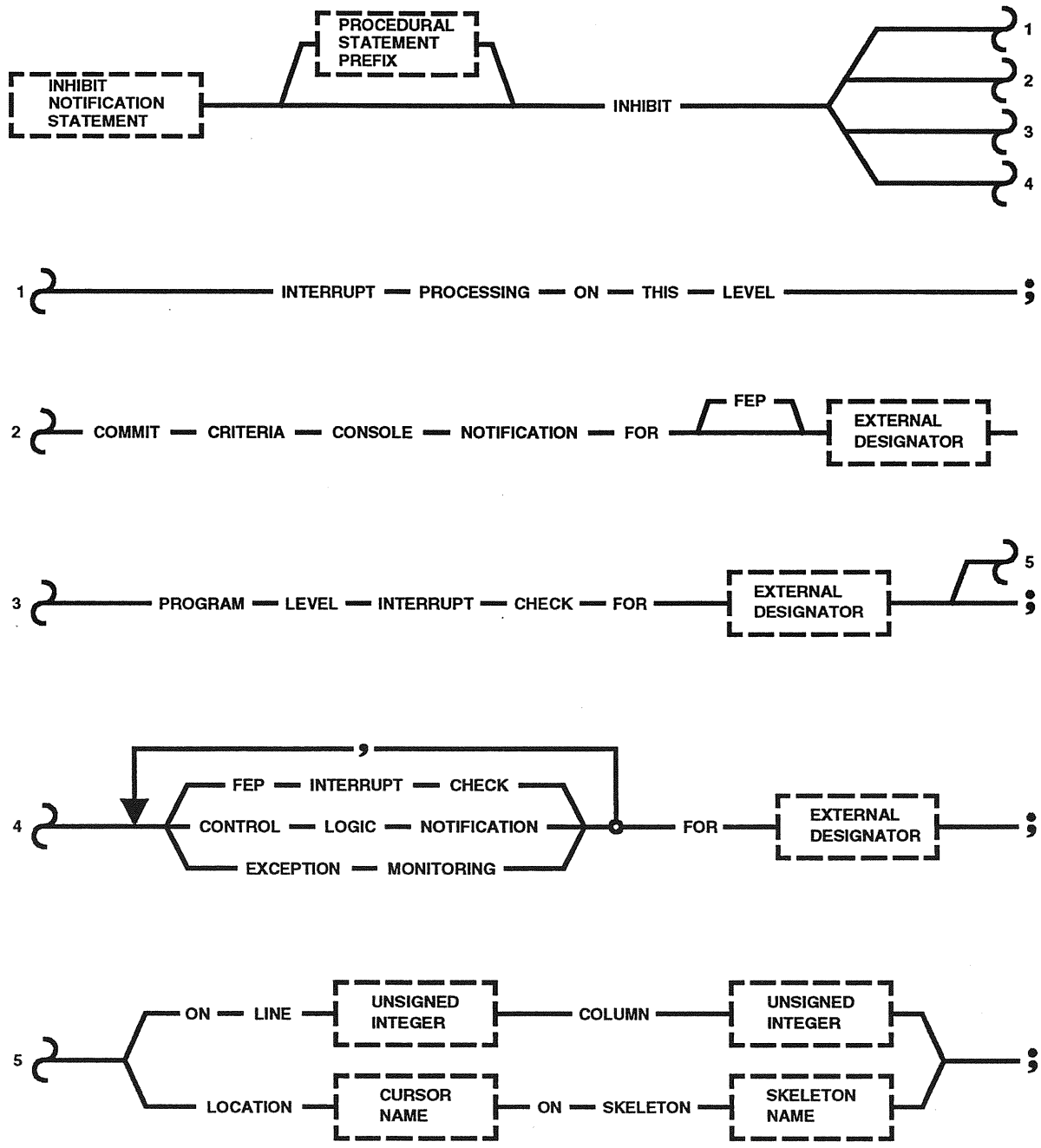


Figure 12-4 INHIBIT NOTIFICATION Statement

Function

The INHIBIT NOTIFICATION Statement is used to cancel or temporarily defer the request for Interrupt Processing defined by SPECIFY and WHEN INTERRUPT Statements.

Description

The INHIBIT NOTIFICATION Statement may be used to:

A. Disallow Interrupt Processing between statements.

B. Cancel requested Interrupt Processing for events previously defined by SPECIFY and WHEN INTERRUPT Statements.

C. Cancel requested responsible console, Control Logic, Exception Monitor, or Commit Criteria Console notification by a FEP for measurement limit or state violations.

The following options are available:

A. INHIBIT INTERRUPT PROCESSING ON THIS LEVEL Option:

This option prevents the program statement execution sequence from being altered by an interrupt occurrence defined by SPECIFY and WHEN INTERRUPT Statements. Events which occur during this time are deferred until an ACTIVATE INTERRUPT PROCESSING Statement is executed. This option has no effect on the active or inhibited Interrupt Processing status of individual events.

B. INHIBIT PROGRAM LEVEL INTERRUPT CHECK FOR Option:

This option is used to cancel Interrupt Processing requests previously defined by SPECIFY and WHEN INTERRUPT statements. Events are specifically inhibited using this option by referencing the Function Designator for each event. Events which occur that have been inhibited by this option are not deferred. They are ignored and the GOAL procedure will never receive them. Interrupt processing for the inhibited events is canceled for this program and level only and does not affect other levels or tasks. Inhibited events may be reactivated by the ACTIVATE INTERRUPT Statement.

C. INHIBIT FEP INTERRUPT CHECK FOR Option:

This option is used to cancel requested GOAL notification of a measurement exception. This option affects all program tasks within the console since the check for exception condition is discontinued by the FEP, and no program will receive notification for that measurement. This option will only be accepted by the FEP from the Responsible, Master or Integration Consoles.

D. LINE AND COLUMN OPTION AND LOCATION ON SKELETON Option:

These options are used to define the DG cursor position associated with the Transmit Cursor, Execute Command and Disarm Command Function Keys.

E. CONTROL LOGIC NOTIFICATION Option:

When CONTROL LOGIC NOTIFICATION is inhibited, Control Logic will not be notified of exception conditions.

F. EXCEPTION MONITORING Option:

When EXCEPTION MONITORING is inhibited, the system exception monitor will not be notified of measurement exceptions.

G. COMMIT CRITERIA CONSOLE NOTIFICATION Option:

If CCC notification is inhibited, dual notification to the Commit Criteria Console will not be performed by a FEP if the Function Designator specified is a system FEP type. If the Function Designators specified are measurement type, dual notification will not be performed on the list of specified Function Designators.

INTERRUPT PROCESSING Option

Example

```
INHIBIT INTERRUPT PROCESSING ON THIS LEVEL;
```

This statement inhibits Interrupt Processing for the level in which it is executed. Interrupts will be defined (queued up) following execution of this statement.

Program Level Interrupt Check For Option

Examples

```
INHIBIT PROGRAM LEVEL INTERRUPT CHECK FOR < DM1 >;
INHIBIT PROGRAM LEVEL INTERRUPT CHECK FOR < DM1 > < DM2 >
    < DPM3 > < AM1 >;
INHIBIT PROGRAM LEVEL INTERRUPT CHECK FOR (STABLE1) FUNCTIONS;
INHIBIT PROGRAM LEVEL INTERRUPT CHECK FOR < XMIT-P1A > ON
    LINE 8 COLUMN 37;
```

The first three examples inhibit Interrupt Processing for the individual measurement Function Designators specified. The last example inhibits Interrupt Processing for the Transmit Cursor Function Key for PAGE-1A, Line 8, Column 37.

FEP Interrupt Check For Option

Examples

```
INHIBIT FEP INTERRUPT CHECK FOR < DM1 >;
INHIBIT FEP INTERRUPT CHECK FOR < AM5 >
                                < AM1 >
                                < AM8 >
                                < DPM1 >
                                < DM5 >;
INHIBIT FEP INTERRUPT CHECK FOR (QTABLE1) FUNCTIONS;
```

These examples cause the FEP to stop GOAL notification of measurement exceptions for the Function Designators specified.

Example

```
INHIBIT FEP INTERRUPT CHECK, CONTROL LOGIC NOTIFICATION,
    EXCEPTION MONITORING FOR <AM1>;
```

This statement requests that the FEP stop notification of all three types of exceptions to the console for the specified measurement.

Commit Criteria Console Notification Option

Example

```
INHIBIT COMMIT CRITERIA CONSOLE NOTIFICATION FOR <DM3>;
```

This example requests that the FEP stop exception notification for <DM3> to the Commit Criteria Console, but does not affect notification to themeasurement's responsible console.

Statement Execution

The execution of the INTERRUPT PROCESSING ON THIS LEVEL option is accomplished by setting the level interrupt flag for this procedure. Interrupts are saved if "inhibit" on this level is used. If an interrupt occurs while the entry is inhibited, it will be saved and the procedure notified when the processing on the level is reactivated.

The execution of the PROGRAM LEVEL INTERRUPT CHECK FOR option is accomplished by flagging the entries in the interrupt blocks for the Function Designators specified to indicate that these entries are inhibited. Interrupts are not saved if the INHIBIT PROGRAM LEVEL INTERRUPT CHECK FOR option is used. When a specific entry is inhibited, the occurrence of the event will be ignored by the GOAL Executor, and the procedure will not be notified of the event.

FEP INTERRUPT CHECK FOR is executed by the GOAL Executor by requesting the appropriate FEP to turn off GOAL notification for the Function Designator. This communication takes place via the CDBFR. Each Function Designator specified requires a separate communication with the FEP. Execution will not continue to the next statement until all FEP communications have been completed.

Error Processing

If an error response is received from the FEP on the INHIBIT FEP INTERRUPT CHECK FOR Statement, a Class III error will be processed.

If an error response is received from the FEP before the last repetitive operation has been executed, a Class III error will be processed at that point. If procedure error override is active, the statement execution will continue. If procedure error override is inhibited, the operation of the procedure will be stopped.

If the GOAL Executor encounters an Undefined FD while executing the Program Level Interrupt Check For option, it will output a Class III error and skip to the next entry.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Interrupt processing for the level will be automatically inhibited when interrupt notification is passed to the procedure. An INHIBIT INTERRUPT PROCESSING ON THIS LEVEL is executed automatically when any interrupt notification is passed to the procedure.

B. The Display Generator cursor position must be specified as either ON LINE and COLUMN or LOCATION ON SKELETON if the Transmit Cursor, Execute Command and Disarm Command DG Function Keys are referenced.

C. The FEP INTERRUPT CHECK option may only refer to measurement type Function Designators.

D. All External Designators on the INHIBIT NOTIFICATION Statement, except those used with Commit Criteria Console, Control Logic, or Exception Monitoring options, must have been referred to on a SPECIFY INTERRUPT Statement or WHEN INTERRUPT Statement.

E. The FEP option may be used to cancel requested GOAL notification of a measurement exception. This option affects all program tasks within the console since the check for the exception condition is discontinued by the FEP, and no program will receive notification for that measurement.

F. When terminating a level and returning to a calling program, pending interrupt events common to both programs which are at that time being deferred by INHIBIT INTERRUPT PROCESSING will be transferred to the calling procedure. When calling a program, the pending interrupts will be transferred to the called sequence.

G. When terminating a level, the pending interrupts which are being deferred and are not common to the program that called this level are discarded.

H. INHIBIT CONTROL LOGIC NOTIFICATION and EXCEPTION MONITORING are not valid for GSE Digital Patterns.

I. The INHIBIT COMMIT CRITERIA option used with a thds fd will work only with type III THDS FD's or members of a Type II Time Homogeneous Dataset. Specifying a Type II THDS FD or a member of a Type III THDS will cause a Class III error to be processed from the FEP.

Legal Function Designators

Analog Measurements (AM)
Countdown Time (CDT)
Digital Pattern Measurements (DPM)
Display Generator Function Keys (PFK)
Discrete Measurements (DM)
Interval Timer (TIMR)
Mission Elapsed Time (CDT)
Programmable Function Panel Function Keys (PFPK)
Remote Communication Function Designators (COM)
System Status Discrete (SSA1)
System Status Digital Pattern (SSA2)
Front End Processor (FEP) - for FEP branch of COMMIT CRITERIA option
Time Homogeneous Dataset (THDS)

System Interrupts (SI) to include:

Measurement validity changes
CPU status changes
Class III error
Interrupt occurrence at a higher level
CDT status changes
MET status changes

12.7 WHEN INTERRUPT STATEMENT

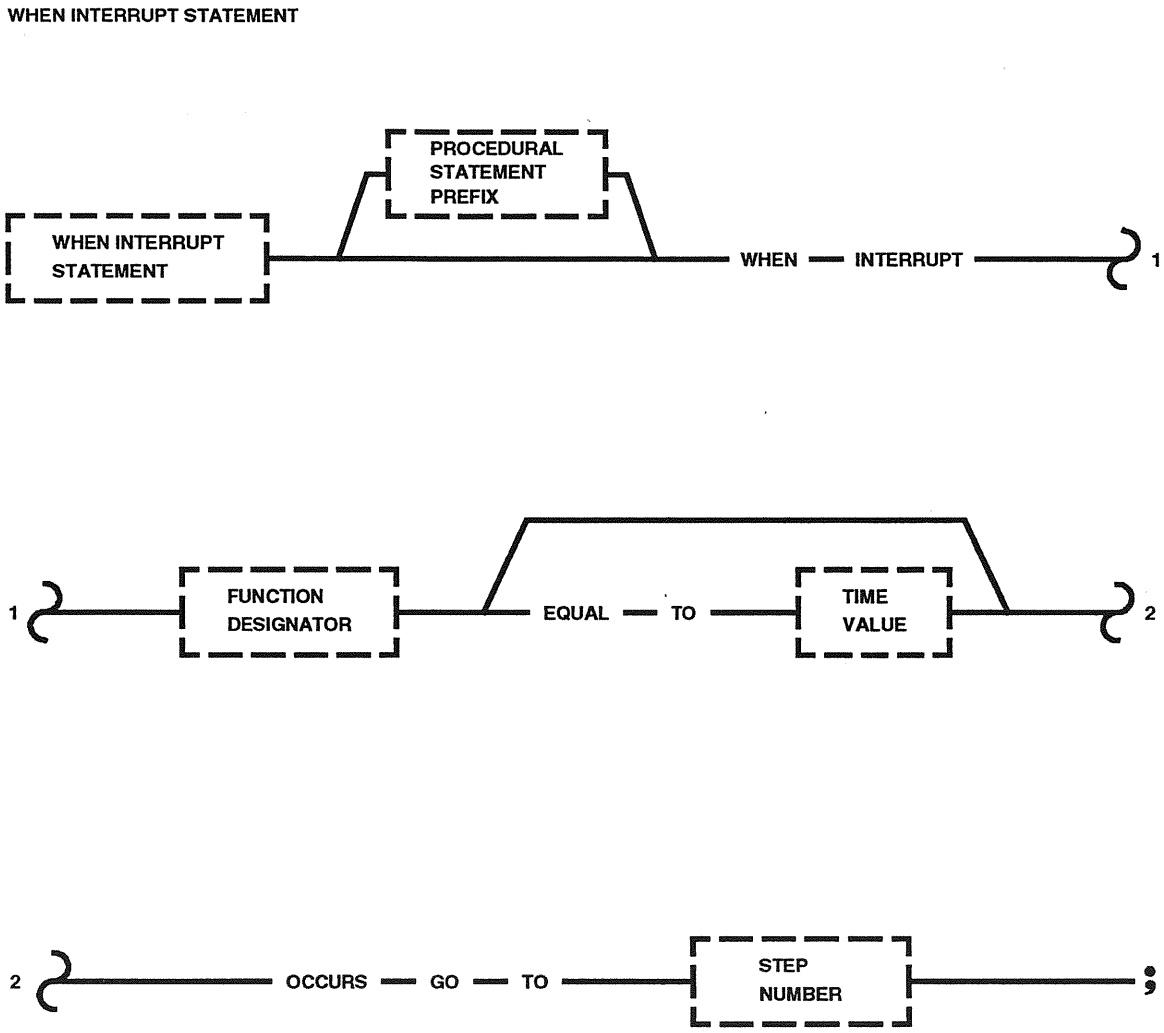


Figure 12-5 WHEN INTERRUPT Statement

Function

The WHEN INTERRUPT Statement specifies the processing to be performed when the defined time condition occurs.

Description

The WHEN INTERRUPT Statement is used to specify interrupts and their corresponding conditions for Count Down Time, Mission Elapsed Time and the Interval Timer. Using this statement, the GOAL procedure can request an interrupt at specified Count Down Time or Mission Elapsed Time, or request an interrupt after a specified interval of time has elapsed.

WHEN INTERRUPT On CDT

EXAMPLE

```
WHEN INTERRUPT < CDT > EQUAL TO -3 HR 10 MIN CDT
    OCCURS GO TO STEP 70;
```

This statement will cause the GOAL procedure to be interrupted when the Count Down Time is equal to -3 hours and 10 minutes. Control will be transferred to STEP 70.

WHEN INTERRUPT On Interval Timer

Examples

```
WHEN INTERRUPT < TIMER > EQUAL TO 45 SEC OCCURS
    GO TO STEP 80;
```

This statement will cause the GOAL procedure to be interrupted when 45 seconds elapse from the time the WHEN INTERRUPT Statement is executed.

NOTE: *This statement initializes the timer to 45 seconds (specified time), and starts timer countdown. At T = 0 the timer initializes an interrupt which causes control to be transferred to Step 80.*

Example

```
WHEN INTERRUPT < TIMER > OCCURS GO TO STEP 90;
```

This statement will cause the GOAL procedure to be interrupted when the Interval Timer set up by the TIMER CONTROL Statement

reaches zero. Control will be transferred to STEP 90. Processing of this statement requires that the timer be preset via the TIMER CONTROL Statement, otherwise a Class III error will result at run time.

WHEN INTERRUPT On MET

Example

```
WHEN INTERRUPT < METIME > EQUAL TO 1 HR MET OCCURS GO TO STEP
   75;
```

This statement will cause the GOAL procedure to be interrupted when the Mission Elapsed Time is equal to 1 hour. Control will be transferred to STEP 75.

Statement Execution

When the Count Down Time or Mission Elapsed Time Function Designator is referenced, the GOAL Executor requests the CCMS operating system to inform it when the specified CDT or MET is reached. When this notification is received, the GOAL procedure's normal execution will be interrupted and control transferred to the specified step.

When the Timer Function Designator is used with a time value specified, the GOAL Executor requests the CCMS Operating System to set an Interval Timer, decrement it once a second, and notify the Executor when the time value is exhausted. When this notification is received the GOAL Procedure's normal execution will be interrupted and control transferred to the specified step.

When the Timer Function Designator is used with no time value, the notification will be sent to the procedure when the Interval Timer reaches 0 (timer must be preset), and control will be transferred to the specified step.

There is only one Interval Timer per task (i.e. for all levels). An interrupt on < TIMER > will be routed to the lowest level in the task interested in that interrupt. Any WHEN INTERRUPT on < TIMER > will override both Time and Step Number of an outstanding WHEN INTERRUPT on < TIMER > for the same level but will override only Time of an outstanding WHEN INTERRUPT on < TIMER > which was started on a previous level.

Each level in a task can have a CDT interrupt outstanding. When a specified countdown time occurs, notification will be sent to all levels which have requested an interrupt at that time.

Error Processing

If a negative Time Value is specified in conjunction with the Interval Timer, a Class III error will be processed.

If no time value is specified in conjunction with the Interval Timer and the Timer has not been set up using a TIMER CONTROL Statement, a Class III error will be processed.

If the procedure error override is active, the execution of the procedure will continue. If procedure error override is inhibited, the operation of the procedure will be stopped.

If the Time Value specified in conjunction with CDT or MET has already been passed, a Class IV error will be processed.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Interrupts are processed only when a GOAL PROGRAM is actively executing procedural statements. If an interrupt occurs while this is not the case, its processing will be deferred until active execution resumes.

B. A WHEN INTERRUPT Statement specifying CDT must specify the Time Value.

C. If a WHEN INTERRUPT references a Function Designator of the same type which had already been used on a previously executed WHEN INTERRUPT Statement, the previous options will be deleted, and processing will be performed according to the most recently executed WHEN INTERRUPT Statement.

D. If INTERRUPT PROCESSING has been inhibited (INHIBIT INTERRUPT PROCESSING ON THIS LEVEL) and a previous WHEN INTERRUPT time value has expired (TIMER) or the time value has been met (MET or CDT) causing the interrupt to be pending, an INHIBIT PROGRAM LEVEL INTERRUPT CHECK FOR the Function Designator must be executed prior to executing another WHEN INTERRUPT (on the same Function Designator), or there will be an immediate branch to the

new WHEN's Interrupt Routine when an ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL Statement is executed. This will happen even though the timer has not expired or the time value for MET or CDT has not been met in the WHEN INTERRUPT Statement.

E. Interrupt processing for the level will be automatically inhibited when interrupt notification is passed to the procedure.

F. The TIME VALUE option must be used for TIMER if a TIMER CONTROL Statement has not been executed to initialize the time.

G. To remove a previously executed INHIBIT INTERRUPT Statement referencing CDT, MET, or TIMER, a WHEN INTERRUPT Statement must be executed.

H. All programs are performed with interrupt processing deferred. To receive any interrupt, the GOAL procedure must execute an ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL Statement.

I. When a GOAL procedure receives an interrupt due to the <TIMER> running out, the interrupt enabled indication is not reset. If the <TIMER> is used again without the WHEN INTERRUPT statement, the previously defined interrupt will occur when the <TIMER> runs out.

Legal Function Designator

Countdown Time (CDT)
Interval Time (TIMR)
Mission Elapsed Time (MET)

12.8 COMMUNICATION INTERRUPT PROCESSING STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

12.8.1 INTRODUCTION

The SEND, RELAY, and RETURN INTERRUPT Statements provide a capability for GOAL programs in the same or different consoles to communicate with one another for data communication or control purposes. A communication between programs is initiated by one of the programs via the use of the SEND INTERRUPT Statement. This statement defines the receivers of the communication through specification of a destination console Function Designator and a user defined Remote Communication serial number Function Designator (REM-COM FD). To complete the communication link, GOAL programs in the destination console which wish to receive the communication must have the same REM-COM FD specified in a SPECIFY INTERRUPT Statement as was used by the sending program on the SEND Statement. Two types of REM-COM FD's exist, REM-COM and REM-COM-WITH-DATA. As their names suggest, one is used for sending an interrupt without data while the other sends an interrupt with data.

The sending of a REM-COM interrupt without data causes the GOAL Executor to route its occurrence in the destination console to all interested GOAL programs, synonymous with the routing of other types of interrupts. An interested program is one which has the interrupt specified and enabled on some program level.

REM-COM-WITH-DATA interrupts permit up to four user-defined numeric data items to accompany the interrupt. Values of zero will be supplied automatically for missing data items if less than four are specified by the SEND INTERRUPT Statement. Unlike REM-COM interrupts, REM-COM-WITH-DATA interrupts can be received by only one program in the destination console. The program which receives the interrupt will be the first one which is interested in it and is not currently "BUSY" with a previous occurrence of the same interrupt. A program will be considered "BUSY" for a specific interrupt from the time of its occurrence until the GOAL Executor has caused the control transfer to the GOAL program's interrupt routine (Step Number) corresponding to the interrupt. A destination console will be considered "BUSY" for a particular interrupt if no program at the console was able to receive the interrupt (the interrupt was not specified and enabled, or it was "BUSY" for all programs).

In addition to the four user supplied data words, two system-supplied communication control words will be sent with a REM-COM-WITH-DATA interrupt. These two words specify the console from which the interrupt is being sent and the REM-COM-WITH-DATA FD specified by the ACKNOWLEDGE BY path of the SEND INTERRUPT

Statement. This information will be used by the RETURN INTERRUPT Statement. In the receiving console, a third system word is added to the data before it is moved to the receiving GOAL program's list variable, which was specified on the SPECIFY INTERRUPT Statement. This variable is a checksum of the other control words. The system control words must not be changed by the GOAL program or keyboard operator if continued communication via the RELAY and RETURN INTERRUPT Statements is to be completed successfully.

The RELAY INTERRUPT Statement allows data which was sent via the SEND INTERRUPT Statement to be relayed to other destination consoles or concurrencies. The system data is not altered by this statement. However, the user data may have been altered by the GOAL program sending the relay interrupt. The RETURN INTERRUPT Statement allows the GOAL program which processes a REM-COM-WITH-DATA interrupt to communicate with the original sender. The system data is used in determining the destination of the return interrupt. The destination console will be the console in which the SEND INTERRUPT Statement was executed, and the return REM-COM-WITH-DATA FD will become the one specified by the ACKNOWLEDGE BY path of the SEND Statement. If this path was bypassed on the SEND Statement, the RETURN Statement is illegal and a Class III runtime error will occur if it is attempted.

The SEND WITH DATA, RELAY, and RETURN Statements are automatically retried up to three times at 250 ms intervals if the destination console is "BUSY". If the destination is "BUSY" on the third retry, GOAL program execution will transfer to the Step Number specified by the AND ON BUSY GO TO path of the syntax diagrams. The following example illustrates the use of these statements.

Example

```
BEGIN PROGRAM (A);
  .
  .
  .
DECLARE NUMERIC LIST (NUMLIST) WITH 7 ENTRIES X;
  .
  .
  .
SPECIFY INTERRUPT <COMDATA3> AND ON OCCURRENCE SAVE
    DATA AS (NUMLIST) AND GO TO STEP 100;
  .
  .
```

```
          .
STEP 50 DELAY 1 SEC;
        $SEND REMCOM WITH DATA$

          SEND INTERRUPT <COMDATA1> WITH DATA (NUM1),5,(NUM3)
              TO CONSOLE <CNSL1>
              ACKNOWLEDGE BY <COMDATA3>
              AND ON BUSY GO TO STEP 50;
            .
            .
            .
STEP 100 CONTINUE;
          $RESPOND TO ACKNOWLEDGEMENT$
            .
            .
            .
          ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL AND RETURN;
          END PROGRAM;
          BEGIN PROGRAM (B);
            .
            .
            .
          DECLARE NUMERIC LIST (NUMLIST) WITH 100 ENTRIES X;
            .
            .
            .
          SPECIFY INTERRUPT <COMDATA1> AND ON OCCURRENCE SAVE
              DATA AS (NUMLIST) AND GO TO STEP 100;
            .
            .
            .
STEP 50   DELAY 1 SEC;
          $RELAY INTERRUPT DATA$
STEP 100 RELAY INTERRUPT <COMDATA2> WITH DATA (NUMLIST) TO
             LOCAL CONSOLE AND ON BUSY GO TO STEP 50;
          ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL AND RETURN;
          END PROGRAM;
          BEGIN PROGRAM (C);
            .
            .
            .
          DECLARE NUMERIC LIST (NUMLIST) WITH 10 ENTRIES X;
            .
            .
            .
          SPECIFY INTERRUPT <COMDATA2> AND ON OCCURRENCE SAVE
```

```
                DATA AS (NUMLIST) AND GO TO STEP 100;
              .
              .
              .
            $ACKNOWLEDGE INTERRUPT$
STEP 50     DELAY 1 SEC;
STEP 100    LET (NUMLIST)1 = 0; $ SUCCESSFUL STATUS$
            RETURN INTERRUPT (NUMLIST) AND ON BUSY GO TO STEP 50;
            ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL AND
            RETURN; END PROGRAM;
```

In this example, program A sends a REM-COM-WITH-DATA interrupt to program B in another console. Program B routes the data to program C in the same console via another interrupt. Program C processes the interrupt and returns an acknowledge to the original sender, program A.

THIS PAGE INTENTIONALLY LEFT BLANK.

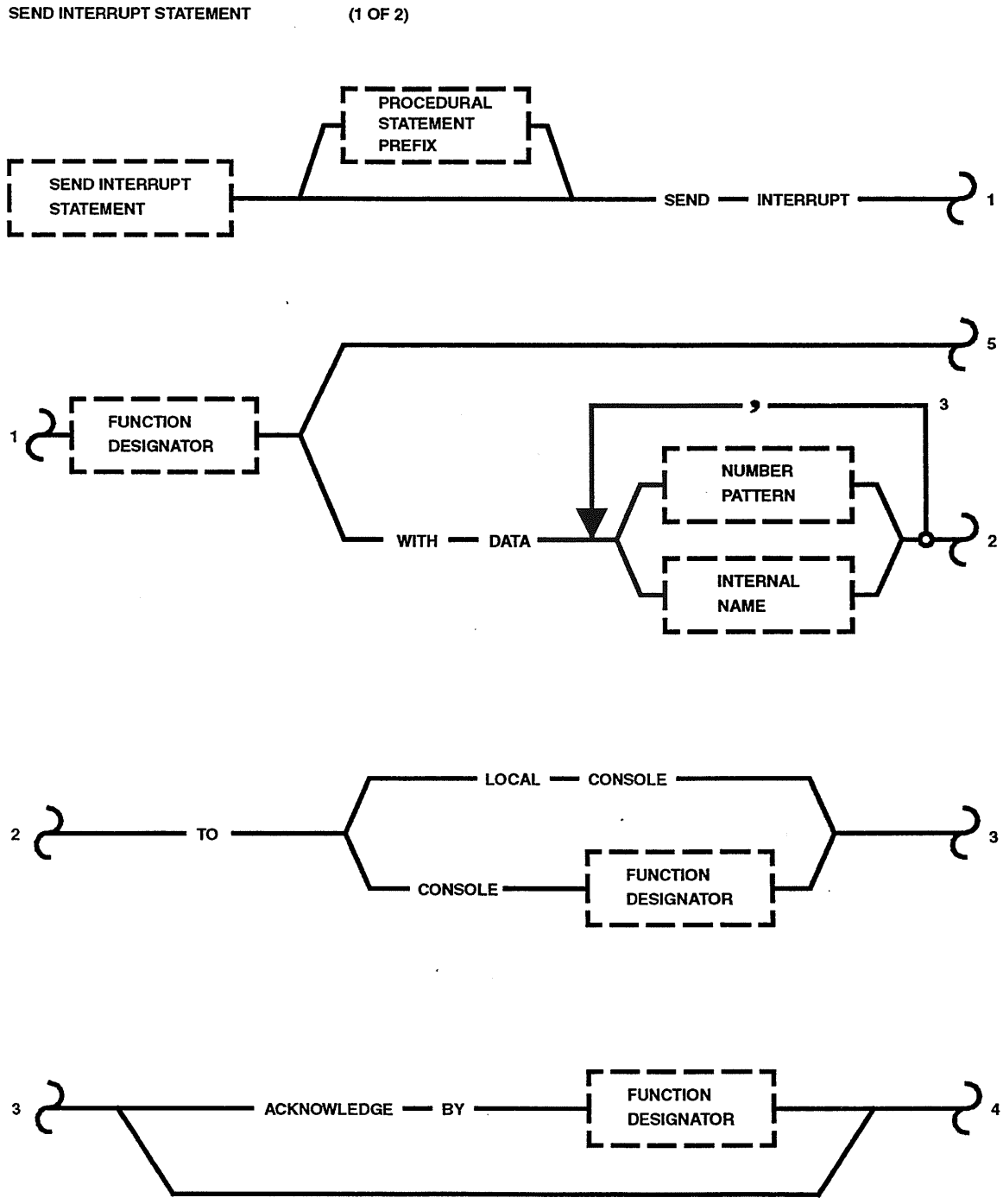


Figure 12-6 SEND INTERRUPT Statement (1 of 2)

SEND INTERRUPT STATEMENT (2 OF 2)

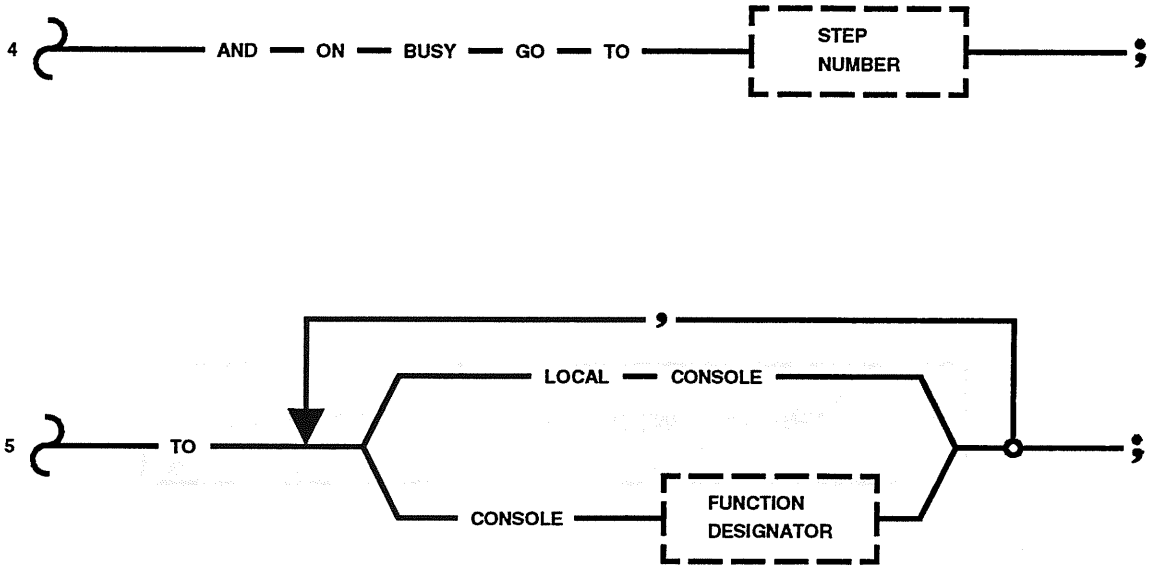


Figure 12-6 SEND INTERRUPT Statement (2 of 2)

THIS PAGE INTENTIONALLY LEFT BLANK.

12.8.1.1 SEND INTERRUPT STATEMENT

Function

The SEND INTERRUPT Statement provides communication between GOAL procedures running in the same console, or in a different console via an interrupt with or without passed data.

Description

The SEND INTERRUPT Statement is used to interrupt another GOAL procedure; e.g., to request a task to be performed or to notify the other procedure of an event in which it is interested.

The SEND INTERRUPT Statement can also be used to pass four data words of numeric data to another GOAL program in the same or a different console. If less than four data words are specified, the remaining words will be defaulted to zero. For this option, a communication Function Designator with subtype "with data" may be specified after the keyword ACKNOWLEDGE BY, to provide a communication link for acknowledgment. An acknowledgment interrupt can be sent back via the RETURN INTERRUPT Statement in the acknowledging procedure. The Step Number specifies where to branch to if the destination console is BUSY.

The LOCAL CONSOLE option is used to specify that the console at which the GOAL procedure is executing is the destination console.

Example

```
SEND INTERRUPT < REMCOM4 > TO CONSOLE < FLTCTL >;
```

This statement will cause any active procedures in the Flight Control Console which have a SPECIFY INTERRUPT Statement referencing the REMCOM4 Function Designator to be interrupted.

```
SEND INTERRUPT < REMCOM7 > TO
       CONSOLE < HYFUEL >,
       CONSOLE < HYHEGN >,
       CONSOLE < HYOXID >;
```

This statement causes the interrupt to be sent to the specified destinations.

```
SEND INTERRUPT <COMDATA> WITH DATA (N1),(N2),(N3)
    TO LOCAL CONSOLE
    ACKNOWLEDGE BY <COMDATA2>
    AND ON BUSY GO TO STEP 100;
```

This statement causes the interrupt with the four data words (N1, N2, N3, 0) to be sent to the local console. A receiving GOAL procedure may send back an acknowledgment of the interrupt via <COMDATA2>.

Statement Execution

If the interrupt is being sent to the same console in which the statement is executed, the GOAL Executor routes the interrupt to the specified destination.

If the interrupt is being sent to a console other than the one in which the statement is executed, the GOAL Executor will communicate with the GOAL Executor in the other console(s) to get the interrupt routed to all procedures interested in the event.

A CTC is required, except for interrupts being sent to the same console or Monitor consoles.

Only local requests are honored at Monitor consoles.

If more than one destination is specified for an interrupt without data, they will be handled one at a time until all the destinations have been processed. Execution will not continue to the next statement until all of the specified destinations have been set up to interrupt the appropriate programs.

For interrupts with data, the "ON BUSY" branch will be taken if the destination console did not receive and accept the interrupt. The interrupt request will be retried three times at 250 ms intervals. If the destination is BUSY on the third retry, GOAL Program execution will transfer to ON BUSY step number, otherwise execution will continue to the next statement.

Error Processing

If an error is encountered during this statement's execution, a Class III error will be processed. If a Class III error is encountered in a statement with repetitive operations, a RESUME will allow the remaining elements of the statement to be processed. If Class III error override is active, statement execution will continue and the remaining elements of the statement will be processed.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The SEND INTERRUPT Statement may only reference a single Remote Communication type Function Designator following the verb SEND INTERRUPT. If data is sent, the Function Designator must have subtype "with data".

B. The keyword ACKNOWLEDGE BY may be followed by only a single Remote Communication Function Designator with subtype "with data".

C. When sending data, only one destination console may be specified in the statement.

D. For communication without data, the SEND INTERRUPT Statement has no effect if the Remote Communication Function Designator has not been referenced by a SPECIFY INTERRUPT Statement at the time it is executed. The operator will not be informed, and no error processing will occur.

E. If the interrupt is sent with data, the ON BUSY branch will be taken if the destination console did not receive and accept the interrupt. For a detailed explanation of the BUSY condition, refer to the beginning of this section.

F. The Local Console option may be specified only once in the statement.

G. The numeric data to be passed may be literal values, single data items, or a list.

Legal Function Designators

The verb SEND INTERRUPT must be followed by a single Remote Communication Function Designator. When sending data, the Function Designator must also have the subtype "with data."

The keyword ACKNOWLEDGE BY must be followed by a single Remote Communication Function Designator with subtype "with data."

The keyword CONSOLE must be followed by a Console Function Designator.

RELAY INTERRUPT STATEMENT

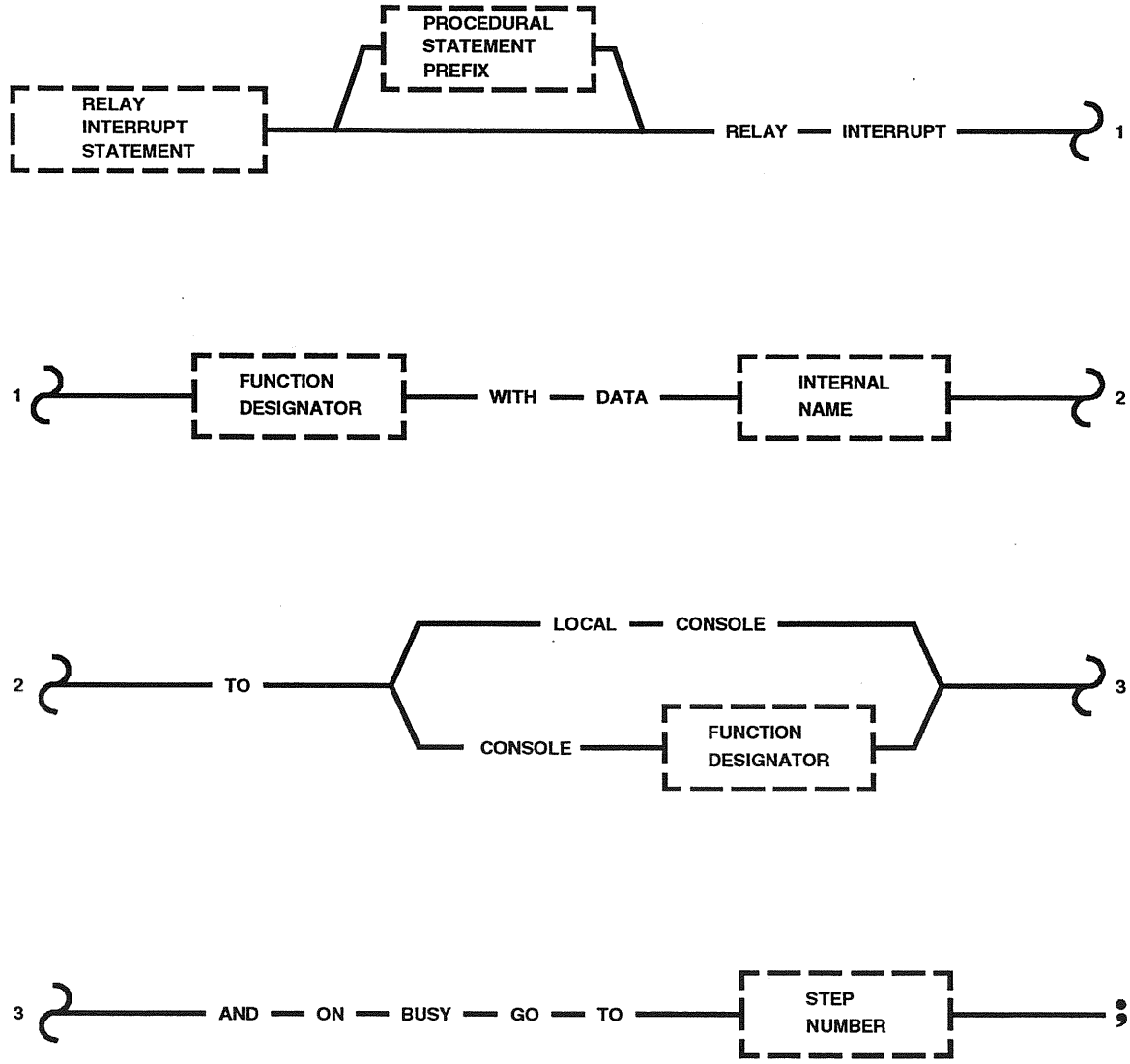


Figure 12-7 RELAY INTERRUPT Statement

12.8.1.2 RELAY INTERRUPT STATEMENT

Function

The RELAY INTERRUPT Statement provides a mechanism to route interrupts with data through a central location for dispersal to other CCMS consoles or concurrencies.

Description

The RELAY INTERRUPT Statement is used to relay a received interrupt with data to other consoles or concurrencies.

The Internal Name specified must be the name of the numeric list received by the SPECIFY INTERRUPT Statement.

The LOCAL CONSOLE option is used to specify that the console at which the GOAL procedure is executing is the destination console.

The Step Number specifies the branch to be taken if the destination console is "BUSY".

Example

```
RELAY INTERRUPT <COMDATA3> WITH DATA (NUMLIST7) TO <FLTCTL>
    AND ON BUSY GO TO STEP 50;
```

This statement will cause the first active procedure in the Flight Control Console which has a SPECIFY INTERRUPT Statement referencing the COMDATA3 Function Designator to be interrupted, and to receive the data in NUMLIST7.

Statement Execution

If the interrupt is being relayed to the same console in which the RELAY Statement is executed, the GOAL Executor routes the interrupt and its data to the specified destination.

If the interrupt is being relayed to a console other than the one in which the RELAY Statement is executed, the GOAL Executor will communicate with the GOAL Executor in the other console to get the interrupt and its data routed to the first procedure interested in the event.

A CTC is required, except for interrupts being sent to the same console or Monitor consoles.

Only local requests are honored at Monitor consoles.

The interrupt request will be retried three times at 250 ms intervals if the destination console is BUSY for the interrupt being sent.

Error Processing

If an error is encountered during this statement's execution, a Class III error will be processed. If Class III error override is active, procedure execution will continue.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The RELAY INTERRUPT Statement may only reference a single Remote Communication type Function Designator with subtype "with data" following the verb RELAY INTERRUPT.

B. The numeric list specified must be the numeric list received by the SPECIFY INTERRUPT Statement. It must be at least 7 words in length.

C. Only one destination console may be specified in the statement.

D. The ON BUSY branch will be taken if the destination console did not receive and accept the interrupt. For a detailed explanation of the BUSY condition refer to the beginning of this section.

Legal Function Designators

The verb RELAY INTERRUPT must be followed by a single Remote Communication Function Designator with subtype "with data".

The keyword CONSOLE must be followed by a Console Function Designator.

THIS PAGE INTENTIONALLY LEFT BLANK.

RETURN INTERRUPT STATEMENT

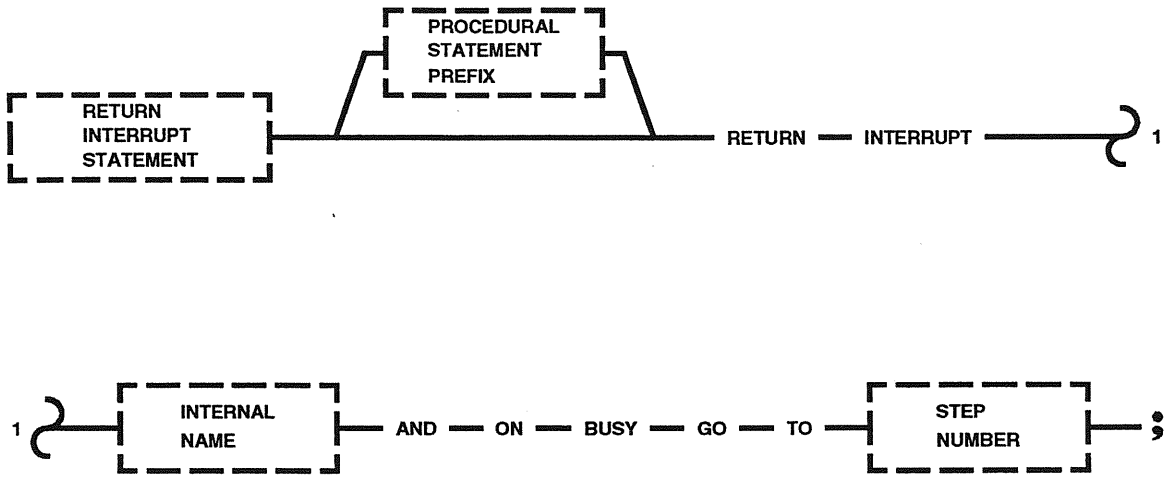


Figure 12-8 RETURN INTERRUPT Statement

12.8.1.3 RETURN INTERRUPT STATEMENT

Function

The RETURN INTERRUPT Statement provides an acknowledgment mechanism for communication interrupts with passed data.

Description

The RETURN INTERRUPT Statement is used to acknowledge the receipt of a REM-COM interrupt with data. The acknowledgment is a REMCOM "with data" interrupt on the Function Designator that was specified after the ACKNOWLEDGE BY keyword in the SEND INTERRUPT Statement in the original interrupt-sending procedure.

The Internal Name specified must be the name of the numeric list received by the SPECIFY INTERRUPT Statement.

This statement will send an acknowledgment interrupt and the data received to only the original interrupt sender. It will not acknowledge any locations that relayed the interrupt.

The step number specifies the branch to be taken if the original sender console is BUSY.

Example

```
RETURN INTERRUPT (NUMLIST) AND ON BUSY GO TO STEP 50;
```

This statement will send an interrupt with the data in NUMLIST back to the procedure that sent the original interrupt. To receive the returned interrupt, the procedure that sent the original interrupt must contain a SPECIFY INTERRUPT Statement with the same Remote Communication Function Designator that is specified after the ACKNOWLEDGE BY keyword in the original SEND INTERRUPT Statement.

Statement Execution

If the acknowledgment interrupt is being sent to the same console in which the RETURN Statement is executed, the GOAL Executor routes the interrupt and the data list to the original interrupt-sending procedure.

If the acknowledgment interrupt is being sent to a console other than the one in which the RETURN Statement is executed, the GOAL Executor will communicate with the GOAL Executor in the other

console to get the interrupt and the data list routed to the original interrupt-sending procedure.

A CTC is required, except for interrupts being sent to the same console or Monitor consoles.

Only local requests are honored at Monitor consoles.

The interrupt request will be retried three times at 250 ms intervals if the destination console is BUSY for the interrupt being sent.

Error Processing

If the RETURN INTERRUPT Statement is used and no interrupt has been received, an error condition will occur.

If an error is encountered during this statement's execution, a Class III error will be processed. If Class III error override is active, procedure execution will continue.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The numeric list specified must be the numeric list received by the SPECIFY INTERRUPT Statement. It must be at least 7 words in length.

B. The "ON BUSY" branch will be taken if the acknowledged console did not receive and accept the interrupt. For a detailed explanation of the "BUSY" condition refer to the beginning of this section.

13. DISK FILES

THIS PAGE INTENTIONALLY LEFT BLANK.

The following statements will be discussed in this section:

There are two types of disk files in GOAL. One disk file type is a one to seven sector data file located on the console moving head disk; it will be referenced as the SECTOR option. There are six of these files per console. The file is referenced as a disk file type Function Designator. These disk files may be accessed only by the READ DISK FILE and RECORD DISK FILE Statements, using the SECTOR option. One to four sectors of this disk file type may be read or written at a time.

The other disk file type is built on CDS, and maintained in the GOAL procedure library; it will be referenced as the FILE option. The maximum size for this disk file type is 64K words. On CCMS this type is located on the console moving head disk. This disk file type is built on CDS by the GOAL Language Processor and may be initialized at build time. This disk file type is defined using the DISK FILE DEFINITION Statements, the DECLARE DATA Statements (refer to Section 4.1), the DECLARE LIST Statements (refer to Section 4.2), and the DISK FILE RECORD INITIALIZATION Statements. All records in the disk file must be in the format of this definition. This disk file type will be accessed in a GOAL procedure using the READ DISK FILE Statement, RECORD DISK FILE Statement, OPEN and CLOSE DISK FILE Statements, and the ALLOCATE and DEALLOCATE DISK FILE Statements.

These disk files are accessed one record at a time. The DISK FILE DEFINITION will allow for specification of subfile size (the number of words of a disk file to be put in bulk memory).

The FILE option type disk file must be built and compiled on CDS and configured into the TCID PARTITION in order to be used in a GOAL procedure. This disk file must be defined using the DISK

FILE RECORD STRUCTURE DEFINITION Statements, and the DECLARE Statements in each GOAL procedure that will READ FROM or WRITE TO a disk file. An ALTERABLE AT SPA option may be specified on the BEGIN DISK FILE DEFINITION Statement. This option will allow the disk file to be copied using the Copy GOAL Disk File program, from a console to a disk file resident on the TCID bulk disk at the SPA. The disk file definitions in the GOAL disk file and in the GOAL procedure must be compatible.

On CCMS 10,112 words of bulk memory at each console are reserved and used as a cache storage medium for GOAL disk files actively being accessed (open files). This bulk memory is shared by a maximum of six "systems" simultaneously. A system is a set of GOAL procedures which may request that any portion of the 10,112 words be allocated to it for disk file usage.

Prior to being able to read data from and write data to a disk file, bulk memory must be allocated for that file's use. Bulk memory allocation is a two-step process. First, a block of bulk memory must be allocated at a system level via the ALLOCATE Statement. This allocation reserves a bulk memory area (called a "system area") for a particular system's use. The second step is to reserve a portion of the system area for a particular file. This step is done via the OPEN Statement. The ALLOCATE Statement also specifies the maximum number of files which a particular system will have open at any given time. The total of the number of files specified on the ALLOCATE Statements of all systems allocated bulk memory may not exceed eighteen. The user is responsible for controlling disk file bulk memory usage, so that enough space exists for system allocation and file opening. Run-time errors will occur if not enough bulk memory is available, or if the maximum possible number of open files exceeds eighteen.

After a file has been opened into a system area, data may be read from and written to it. Data within a disk file is organized into logical records and subfiles. A subfile is a group of records. The format of a record and number of records per subfile is determined at compile time for a disk file definition. All disk file I/O's involve the reading and writing of these records, and are accomplished by using the READ DISK FILE and RECORD DISK FILE Statements. If a record referenced on a READ or RECORD Statement resides in bulk memory, no disk I/O will be needed to access that record. If a referenced record is not bulk memory resident the GOAL Executor automatically pages disk file subfiles between bulk memory and disk, so that the record being accessed is bulk memory resident. This process may cause several disk accesses. After

the first record access, the GOAL Executor keeps one subfile in bulk memory for all open files. To prepare for a series of time-critical events, a GOAL procedure may read a disk file record to cause a particular subfile to be bulk memory resident. This action will guarantee that continued access of records in that subfile will not cause disk I/O.

The CLOSE Statement will write any changed data in the subfile currently in bulk memory out to disk, and frees bulk memory space reserved for that file for use by other files in that system area. The DEALLOCATE Statement frees bulk memory reserved for a system by the ALLOCATE Statement. Much "allocating and deallocating" or "opening and closing" may cause the 10112 word bulk memory area and the system areas to become fragmented. When deallocating bulk memory (because of a DEALLOCATE or CLOSE Statement), the GOAL Executor combines free space fragments adjacent to the area being deallocated into one large free area. However, no attempt is made to combine non-adjacent fragments by moving any disk file data.

The following is an example of this type of disk file usage.

Example (Disk File Definition Procedure):

```
BEGIN DISK FILE DEFINITION FOR FILE (EXAMPLE1) WITH
    100 RECORDS BLOCKED AT 10 RECORDS;
DECLARE QUANTITY (Q1) = V;
DECLARE STATE (S1) = ON/OFF;
DECLARE NUMBER (N1) = I;
INITIALIZE RECORD 1 AS 10V, ON, 35;
END DISK FILE DEFINITION;
```

The above example is a disk file procedure with 100 records, blocked into 10 subfiles containing 10 records each. Each record has the format of one quantity data type of volts, followed by one state data type of ON/OFF, followed by one number data type of integer. The first record is initialized as quantity 10 volts, followed by state ON, followed by number 35 integer. Records 2 through 100 will not be initialized.

Example (Disk File Usage in a Goal Procedure):

```
BEGIN PROGRAM (A);
BEGIN RECORD FORMAT FOR FILE (EXAMPLE1);
DECLARE QUANTITY (VOLTAGE) = V;
DECLARE STATE (STATE) = ON/OFF;
DECLARE NUMBER (NUM) = I;
END RECORD FORMAT;
```

```
    .
    .
    .
    .
ALLOCATE 2000 WORDS FOR AREA (LOX) FOR 5 FILES;
OPEN FILE (EXAMPLE1) INTO AREA (LOX);
$ TO READ FIRST RECORD $
READ FILE (EXAMPLE1) IN AREA (LOX) RECORD 1 AND SAVE AS
    (QTY1), (STATE1), (NUM1);
    .
    .
    .
$ TO WRITE TO THE TENTH RECORD $

RECORD (QTY1), (STATE1), (NUM1) TO FILE (EXAMPLE1)
    IN AREA (LOX) RECORD 10;
    .
    .
    .
CLOSE FILE (EXAMPLE1) IN AREA (LOX);
DEALLOCATE AREA (LOX);
TERMINATE;
END PROGRAM;
```

The above example is a GOAL procedure using a GOAL disk file. The procedure first declares the record format for disk file EXAMPLE1 the same as it was declared in GOAL disk file EXAMPLE1. Next, 2000 words of bulk memory are reserved for use by 5 disk files for system LOX. Disk file EXAMPLE1 is then opened into the previously reserved bulk memory for LOX. Record 1 of disk file EXAMPLE1 is read and the contents saved into the specified internal variables. This operation causes the first subfile of EXAMPLE1 to be read from the disk and written to bulk memory. In this example, there are 10 subfiles which contain 10 records each. Next, the contents of the specified internal variables are written to Record 10 of disk file EXAMPLE1 using the RECORD DISK FILE Statement. Finally, disk file EXAMPLE1 is closed and the bulk memory allocated for system LOX is deallocated.

THIS PAGE INTENTIONALLY LEFT BLANK.

BEGIN DISK FILE DEFINITION STATEMENT

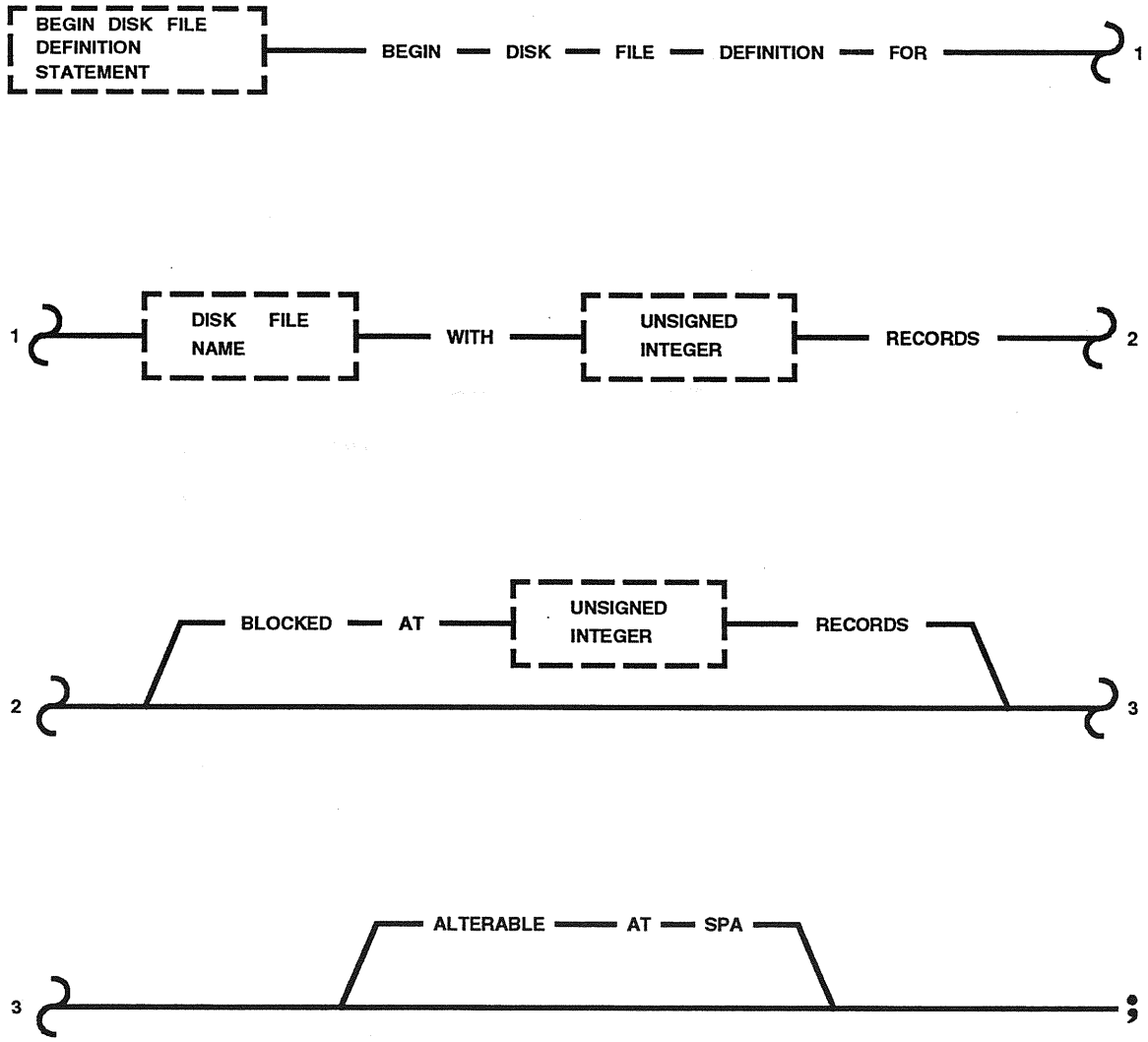


Figure 13-1 BEGIN DISK FILE DEFINITION Statement

13.1 BEGIN DISK FILE DEFINITION STATEMENT

Function

The BEGIN DISK FILE DEFINITION Statement is used to indicate the start of a GOAL disk file, specify its name, number of records, blocking of records, and specify whether this disk file may be copied to at the SPA.

Description

The BEGIN DISK FILE DEFINITION must be the first statement in a GOAL disk file and may appear only once. The disk file name follows the standard format of name and may be up to 8 characters in length. All references to the disk file must use this name.

The number of records in a subfile is determined at compile time. Each record may contain any mixture of variables of types (numeric, quantity, state, time, text) and structures (single item of table and tables with 1 column lists). The only limitation is that the total record size be less than or equal to 384 words.

The BLOCKED AT option will group the specified number of records into a subfile, which is the number of records read into bulk memory by the GOAL Executor when a READ/RECORD Statement references the disk file. The default block size is the entire file. Refer to Section 18.4 for additional information about the blocking of records.

The Alterable At SPA option will allow the disk file to be copied using the Copy GOAL Disk File program, from a console to a disk file with the same name resident on the TCID bulk disk at the SPA.

Examples

```
BEGIN DISK FILE DEFINITION FOR FILE (EXAMPLE1) WITH 10 RECORDS;
```

This example will be the first statement in the GOAL disk file EXAMPLE1 and indicates the start of the disk file.

```
BEGIN DISK FILE DEFINITION FOR FILE (EXAMPLE2) WITH 100 RECORDS
    BLOCKED AT 10 RECORDS;
```

This example indicates the beginning of the GOAL disk file EXAMPLE2 and specifies subfiles to be blocked at 10 records.

```
BEGIN DISK FILE DEFINITION FOR FILE (TEST1) WITH 25 RECORDS
    ALTERABLE AT SPA;
```

This example indicates the beginning of the GOAL disk file TEST1 and allows the disk file to be copied from a console to a disk file with the same name on the TCID bulk disk at the SPA.

Statement Execution

The GOAL Executor uses sizing information and bits set in the program control block indicating a disk file and whether or not the disk file is alterable at the SPA.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

The BEGIN DISK FILE DEFINITION Statement cannot be used in a GOAL procedure. It is used only in GOAL disk files.

THIS PAGE INTENTIONALLY LEFT BLANK.

END DISK FILE DEFINITION STATEMENT

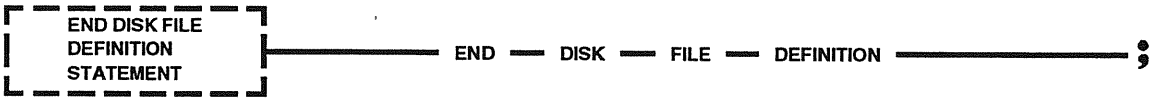


Figure 13-2 END DISK FILE DEFINITION Statement

13.2 END DISK FILE DEFINITION STATEMENT

Function

The END DISK FILE DEFINITION Statement is used to indicate the end of a GOAL disk file.

Description

The END DISK FILE DEFINITION Statement is a required statement for all GOAL disk files and must appear as the last statement.

Any data following the END DISK FILE DEFINITION Statement is flagged as an error.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

The END DISK FILE DEFINITION Statement cannot be used in a GOAL prodedure. It is used only in GOAL disk files.

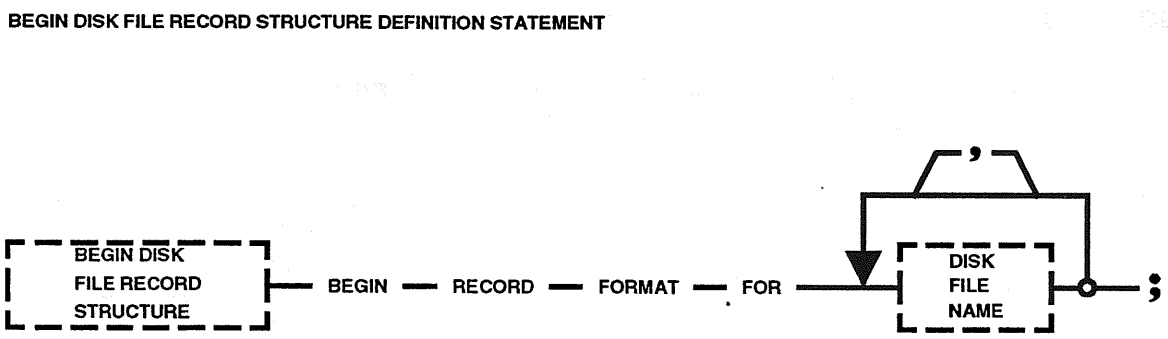


Figure 13-3 BEGIN DISK FILE RECORD STRUCTURE DEFINITION Statement

13.3 BEGIN DISK FILE RECORD STRUCTURE DEFINITION STATEMENT

Function

The BEGIN DISK FILE RECORD STRUCTURE DEFINITION Statement is used in a GOAL procedure to indicate the start of a GOAL disk record format description, and specify its name.

Description

The BEGIN DISK FILE RECORD STRUCTURE DEFINITION Statement must precede a set of DECLARE DATA Statements and/or DECLARE LIST Statements that will describe the disk file record format. A disk file record format description is required for every disk file referenced in a GOAL procedure via the READ or RECORD Statement.

The Disk File Name must either be a valid disk file external reference or the Internal Name in the BEGIN PROGRAM Statement for a passed disk file.

The number of records must match the number of records specified in the GOAL disk file definition.

Example

```
BEGIN RECORD FORMAT FOR FILE (FILEX);

BEGIN RECORD FORMAT FOR FILE (FILEX), FILE (FILEZ);
```

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The BEGIN DISK FILE RECORD STRUCTURE DEFINITION Statement must be embedded in a GOAL procedure.

B. DISK FILE RECORD STRUCTURE DEFINITIONs must precede the first procedural statement.

END DISK FILE RECORD STRUCTURE DEFINITION STATEMENT

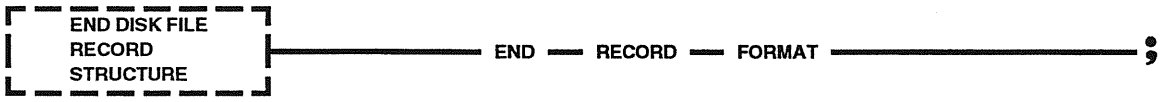


Figure 13-4 END DISK FILE RECORD STRUCTURE DEFINITION Statement

13.4 END DISK FILE RECORD STRUCTURE DEFINITION STATEMENT

Function

The END DISK FILE RECORD STRUCTURE DEFINITION Statement is used to indicate the end of a disk file record format definition within a GOAL procedure.

Description

The END DISK FILE RECORD STRUCTURE DEFINITION Statement may appear only once in a record format definition.

The END DISK FILE RECORD STRUCTURE DEFINITION Statement is a required statement for all record format definitions and must appear at the end of the record format definition.

DISK FILE RECORD INITIALIZATION STATEMENT

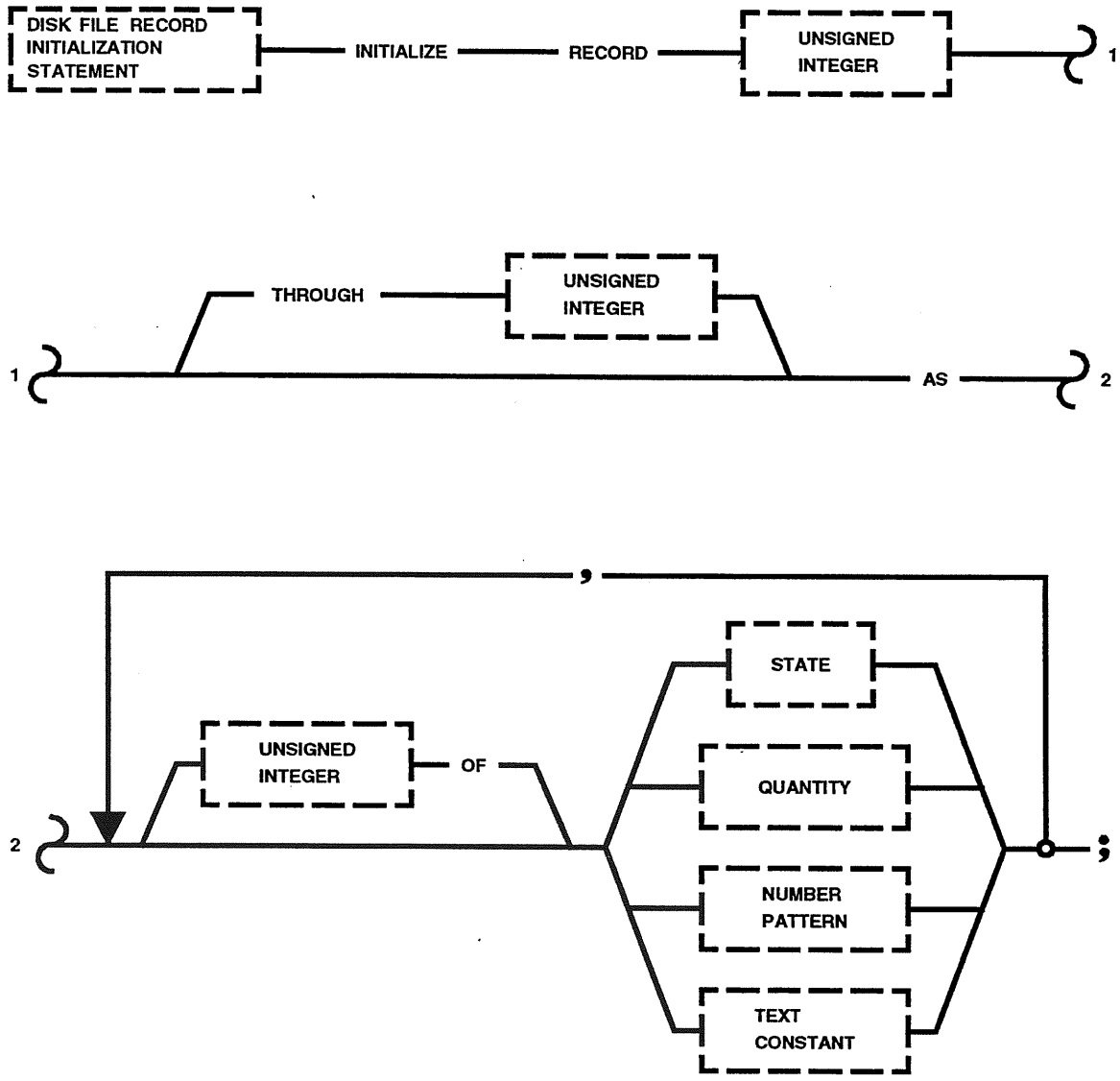


Figure 13-5 DISK FILE RECORD INITIALIZATION Statement

13.5 DISK FILE RECORD INITIALIZATION STATEMENT

Function

The DISK FILE RECORD INITIALIZATION Statement is optionally used to initialize disk file data in GOAL disk files either with the same data in all records or unique data in each record.

Description

The DISK FILE RECORD INITIALIZATION Statement, if used, must appear after the DISK FILE RECORD DECLARATION Statements and before the END DISK FILE DEFINITION Statement. The data types specified in the record initialization must be the same by type and order as those declared in the RECORD STRUCTURE DEFINITION Statements.

All records or selective records may be initialized. All record numbers not specified for initialization will be initialized to zero for quantity and number data types, uninitialized for state data types, and blanks for text data types.

Examples

```
INITIALIZE RECORD 1 AS 10 OF ON, 10V, 5V;
```

This example will initialize only the first record as ten State values of ON followed by a Quantity of 10 volts followed by a Quantity of 5 volts.

```
INITIALIZE RECORD 1 THROUGH 20 AS 5 OF ON, 5 OF OFF, 2 OF 10V;
```

This example will initialize twenty records as five State values of ON followed by five State values of OFF followed by two Quantity values of 10V.

Statement Execution

The DISK FILE RECORD INITIALIZATION Statement is transparent to the GOAL Executor. It is used by the GOAL Compiler to initialize the object file records to the specified values for specified records.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Data types specified in the record initialization must be the same as those declared in the record structure definition.

B. In the initialization of a Text Constant, blanks and commas are significant.

C. Any ASCII character may be used in the initialization of a Text Constant with the exception of the parenthesis and the ampersand.

D. The DISK FILE RECORD INITIALIZATION Statement cannot be used in a GOAL procedure. It can only be used in GOAL disk files.

E. If the number of characters specified in the initialization of a Text Constant is less than the number of characters declared in the record structure definition, the balance of the entry will be padded with blanks.

THIS PAGE INTENTIONALLY LEFT BLANK.

ALLOCATE DISK FILE STATEMENT

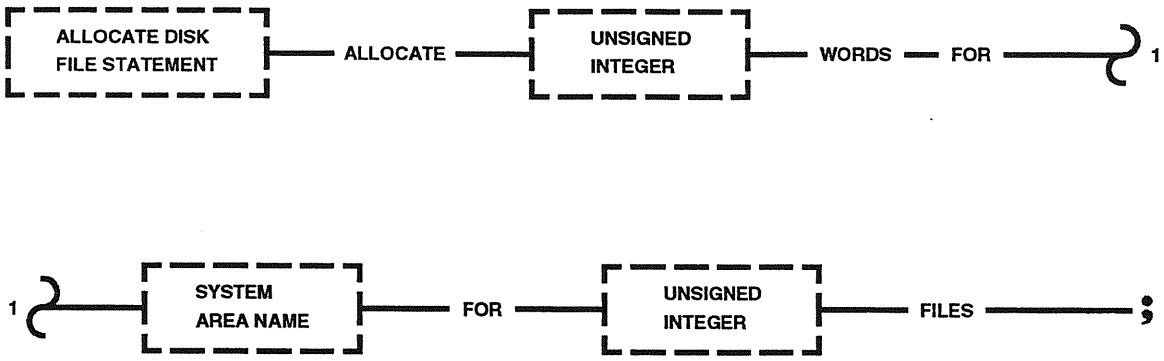


Figure 13-6 ALLOCATE DISK FILE Statement

13.6 ALLOCATE DISK FILE STATEMENT

Function

The ALLOCATE DISK FILE Statement is used in a GOAL procedure to specify the amount of bulk memory to be reserved for the specified number of files for the specified System Area Name.

Description

The ALLOCATE DISK FILE Statement is used in a GOAL procedure, prior to opening any disk file in that system area, to allocate bulk memory needed for disk file use for a specified system area. It is the user's responsibility to allocate enough bulk memory to accommodate one subfile for each of the maximum number of files to be opened at one time in the specified System Area Name. Refer to Section 18.3 for information concerning the amount of bulk memory needed by a subfile. The maximum amount of bulk memory specified to be allocated for a system area (also for system areas combined) is 10,112 words. The maximum number of files specified for a system area (also for system areas combined) is 18.

Example

```
ALLOCATE 1000 WORDS FOR AREA (LOX) FOR 5 FILES;
```

This example will reserve 1,000 words of bulk memory for use by system area LOX; and up to 5 files may be opened into that area of bulk memory.

Statement Execution

The GOAL Executor will reserve the specified number of words for the specified number of files for use by a system. As a file is opened into an area, the number of words needed for one subfile is set aside from the allocated bulk memory for use by that file. If not enough bulk memory is allocated to open the specified number of files, or an attempt is made to open more than the specified number of files, a run time error will occur.

Error Processing

The following run time errors can occur during the allocation process:

A. Invalid number of bulk memory words specified (less than 1 or greater than 10,112) - Class III.

B. Invalid number of files specified (less than 1 or greater than 18) - Class III.

C. Too many files requested. The number of files specified would have caused the total number of possible open files to exceed the limit of 18 - Class III.

D. The system specified has already been allocated an area of bulk memory - Class III.

E. Maximum number of systems exceeded. The limit is 6 - Class III.

F. Bulk memory read error while reading the free space table - Class II.

G. Bulk memory write error while writing the free space table - Class II.

H. Not enough bulk memory free space available (requested amount not available) - Class III.

I. Allocates inhibited (a bulk memory error has previously occurred while writing the free space table) - Class II.

Reference KSC-LPS-0P-033-07 (System Message Catalog, GOAL Message Inserts - Part 3) for more detailed descriptions of these errors.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. No more than 10,112 words of bulk memory may be allocated to a System Area Name.

B. No more than 10,112 words of bulk memory may be allocated to all system areas combined in a console.

C. The maximum number of files that can be specified for an allocated system area name is 18.

D. The maximum number of files that can be specified for all allocated system areas combined is 18 in a console.

THIS PAGE INTENTIONALLY LEFT BLANK.

DEALLOCATE DISK FILE STATEMENT

Figure 13-7 DEALLOCATE DISK FILE Statement

13.7 DEALLOCATE DISK FILE STATEMENT

Function

The DEALLOCATE DISK FILE Statement is used in a GOAL procedure to free previously allocated bulk memory for a specified System Area Name.

Description

The DEALLOCATE DISK FILE Statement is used in a GOAL procedure to free previously allocated bulk memory after all disk files opened into that portion of bulk memory have been closed. That bulk memory will then be available for allocation to another system or to the same system for use with other disk files.

Example

```
DEALLOCATE AREA (LOX);
```

This example will free the bulk memory reserved for LOX by the ALLOCATE DISK FILE Statement.

Statement Execution

The Goal Executor will verify that all files opened into the specified system area have been closed. A Class III error will result if this is not the case.

The bulk memory space of the system area specified will be deallocated and available to be used on subsequent ALLOCATE statements. Bulk memory free segments contiguous with the area being deallocated will be combined during the deallocation process, but fragmentation will occur for non-contiguous segments. No automatic deallocation of bulk memory will be done when programs are terminated.

Error Processing

The following run time errors can occur during the deallocation process:

A. Invalid system name - Class III.

B. All files not closed (not all files belonging to the system have been closed) - Class III.

C. Bulk memory read error while reading the free space table - Class II.

D. Bulk memory write error while writing the free space table - Class II.

E. Disk file system incongruity (overlap exists between bulk memory space being deallocated and bulk memory specified in the free space table) - Class II.

Refer to KSC-LPS-0P-033-07, Part 3 (System Message Catalog, GOAL Message Inserts) for more detailed descriptions of these errors.

THIS PAGE INTENTIONALLY LEFT BLANK.

OPEN DISK FILE STATEMENT

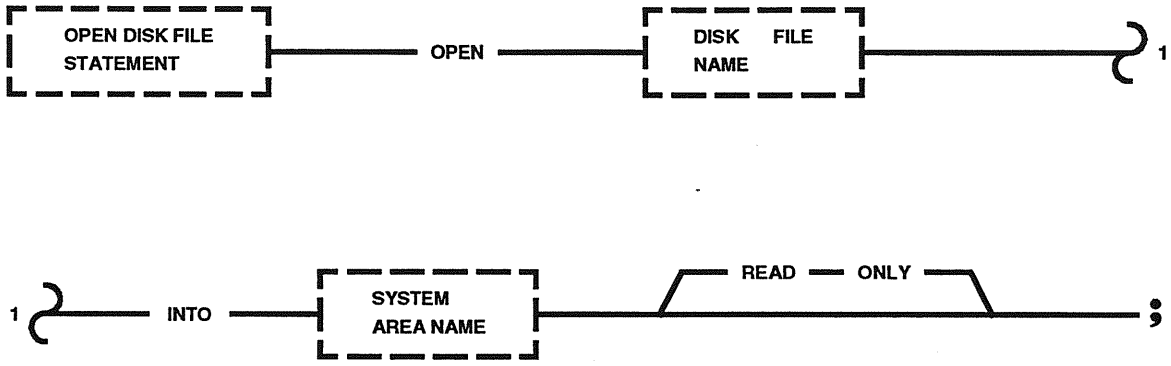


Figure 13-8 OPEN DISK FILE Statement

13.8 OPEN DISK FILE STATEMENT

Function

The OPEN DISK FILE Statement allows the disk file specified to be accessed by a GOAL procedure.

Description

The OPEN DISK FILE Statement allows the disk file specified in the statement to be read or written to by a GOAL procedure until the CLOSE DISK FILE Statement is reached.

Examples

```
OPEN FILE (DSKFILE1) INTO (GLS);
OPEN FILE (SAM) INTO (A) READ ONLY;
```

The first example opens the disk file (DSKFILE1) into the system area (GLS). The second example opens (SAM) into system area (A), but the data in the disk file can be read and not written.

Statement Execution

The OPEN Statement will instruct the GOAL Executor to reserve bulk memory space within the specified system area's allocated space for the file being opened. The amount of bulk memory reserved is equal to the file's subfile size, which is determined when the disk file is created and compiled. Refer to Section 18.4 for information about the amount of bulk memory required by a subfile. An error will occur if not enough contiguous bulk memory space is available. It is also an error condition if this open statement will cause more files to be open for the system area than was specified on the ALLOCATE Statement. The OPEN Statement will not read any disk file data from disk into bulk memory. However, a disk access is required to obtain the file's characteristics (e.g., size).

Error Processing

The following run time errors can occur during the opening process:

A. File name specified does not exist - Class III.

B. File copy in progress - Class III.

C. File update in progress - Class III.

D. Invalid system name specified - Class III.

E. File specified is already open - Class III.

F. Maximum number of files exceeded. The limit is the number specified on the ALLOCATE Statement - Class III.

G. Error reading the GOAL high speed directory entry from bulk memory - Class II.

H. Disk error reading the disk file's PCB (Program Control Block) - Class III.

I. PCB checksum error - Class III.

J. Opens inhibited (a bulk memory error has previously occurred while writing the file space table) - Class II.

K. Bulk memory read error while reading the free space table - Class II.

L. Bulk memory write error while writing the free space table - Class II.

M. Not enough bulk memory free space available within the specified system's area - Class III.

Reference KSC-LPS-OP-033-07, Part 3 (System Message Catalog, GOAL Message Inserts) for more detailed descriptions of these errors.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The system area specified may open only as many files as was specified on the ALLOCATE Statement.

B. A maximum of 18 files may be open simultaneously for all system areas combined in a console.

C. The system area specified must have enough bulk memory space available to accommodate one of the specified file's subfiles.

D. A GOAL disk file can be opened by more than one system, but cannot be opened by the same system more than once.

THIS PAGE INTENTIONALLY LEFT BLANK.

CLOSE DISK FILE STATEMENT

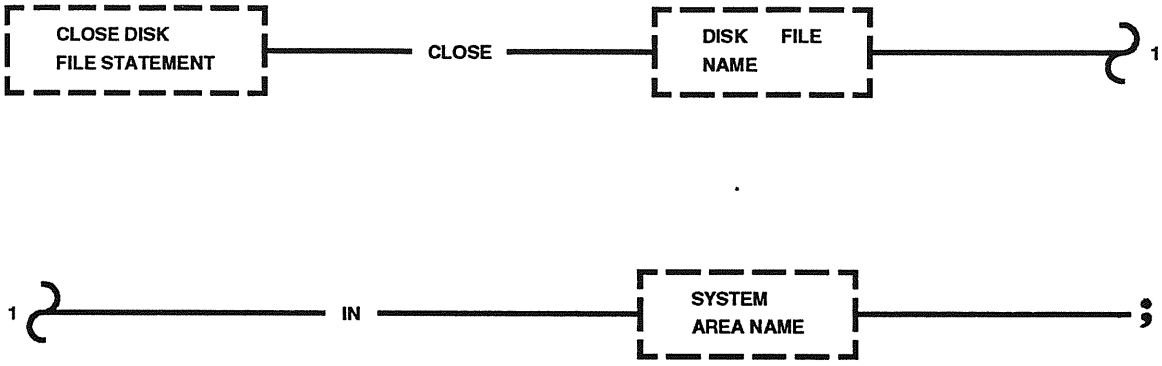


Figure 13-9 CLOSE DISK FILE Statement

13.9 CLOSE DISK FILE STATEMENT

Function

The CLOSE DISK FILE Statement closes the disk file specified to access by a GOAL procedure.

Description

The CLOSE DISK FILE Statement closes the disk file so that access by a GOAL procedure will be denied.

Example

```
CLOSE FILE (DSKFILE1) IN (GLS);
```

The above example closes the disk file (DSKFILE1) used in system area (GLS).

Statement Execution

The GOAL Executor will deallocate the bulk memory reserved for the file when it was opened, enabling it to be reused when another file is opened into the specified system area. If the data on bulk memory for the file being closed has changed, then the current resident subfile is written back to the disk.

Error Processing

The following run time errors can occur during the closing process:

A. Invalid system name specified - Class III.

B. File specified is not open - Class III.

C. File record read/write is in progress - Class III.

D. Bulk memory read error while paging subfile data or while reading the free space table - Class II.

E. Bulk memory write error while writing the free space table - Class II.

F. Disk write error while paging subfile data - Class II.

G. Disk file system incongruity (overlap exists between bulk memory space being deallocated and bulk memory space in the free space table) - Class II.

Refe to KSC-LPS-OP-033-07, Part 3 (System Message Catalog, GOAL Message Inserts) for more detailed descriptions of these errors.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

The disk file specified must be open under the System Area Name before the GOAL Executor can close it. This condition is not verifiable by the compiler.

THIS PAGE INTENTIONALLY LEFT BLANK.

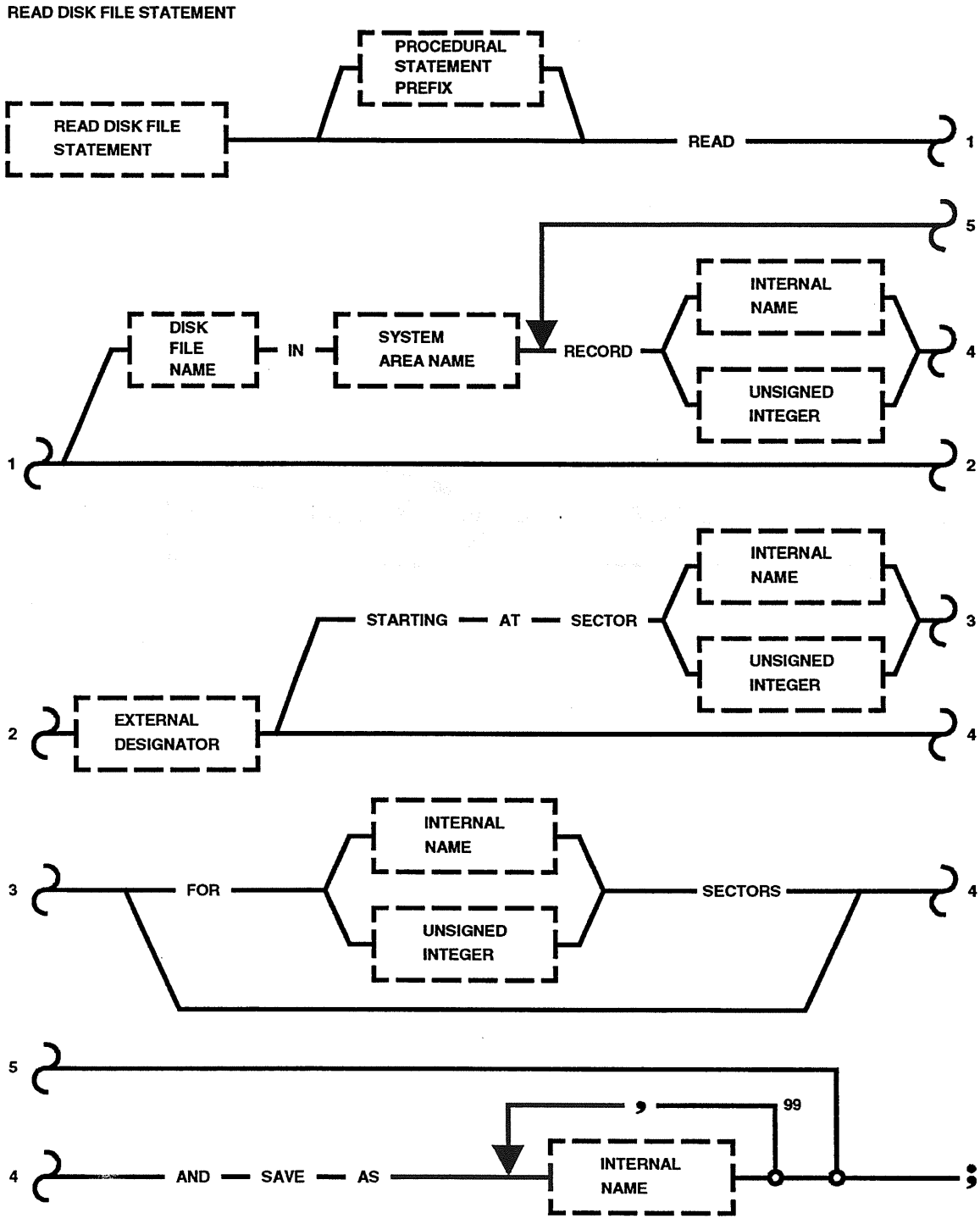


Figure 13-10 READ DISK FILE Statement

13.10 READ DISK FILE STATEMENT

Function

The READ DISK FILE Statement is used to acquire data from files on a console moving head disk, and store that data in specified internal variables.

Description

Two options are available for reading disk files:

The SECTOR option allows data to be read from disk files which are represented by Function Designators (<DISK-F1>, <DISK-F2>, etc.). Up to seven 128-word sectors of data may be read and stored using this option (four sectors at a time).

The FILE option provides the means to read data from a disk file which has been defined via a disk file definition. One or more records of the file, which will have been allocated and opened in the console bulk memory area specified, may be read, and the data saved in one or more series of internal variables conforming to the defined format of records for that file.

SECTOR Option

Examples

```
READ < DISK-F1 > STARTING AT SECTOR 1
    AND SAVE AS (NLIST1), (NLIST2);
READ < DISK-F2 > STARTING AT SECTOR 2 FOR (N1) SECTORS
    AND SAVE AS (NLIST1), (QLIST1);
```

The first example will read the first 128 words of the specified disk file. The second example will begin reading data at sector 2, ignoring the first 128 words of data on the disk. The number of sectors to be read is determined by Internal Name (N1).

FILE Option

Example

```
READ FILE (SAM) IN AREA (LOX) RECORD 10 AND SAVE AS (NUM1),
    (STATE1), (STATE2), (NUMLIST1), RECORD 30 AND SAVE AS
    (NUM2), (STATE3), (STATE4), (NUMLIST2);
```

This statement will result in a read of records 10 and 30 of disk file SAM, which has been opened in the console bulk memory area called LOX. The data will be stored in the specified lists of variables. Note that the variable lists have the same data format, since all records of the disk file must follow the same structure.

Statement Execution

SECTOR Option

A disk file will be read from console moving head disk. This file should have been written by a procedure. It is the responsibility of the procedures involved to ensure the data is in the proper format. Specified sectors within a file may be read by using the STARTING AT SECTOR option. One sector from the specified start sector will be read from the file unless the number of the sectors to be read is stated. There are seven sectors per file from which data may be read. They are numbered 1 through 7 with each sector containing 128 words of data.

FILE Option

Upon receiving a request to read a disk file, the GOAL Executor will verify that the requested file has been allocated and opened in the area specified in the READ Statement. If not, a Class III error will result. After verifying the legality of the request, the Executor will determine whether the requested record is contained in the subfile currently resident in bulk memory. If the requested record is not bulk memory resident, the current active subfile is read from bulk memory and written to the moving head disk if necessary. This action is only necessary if the data in this subfile has changed. The subfile containing the requested record is then read from the disk and written to bulk memory. Once bulk memory resident, the requested record is read into main memory and moved to the internal variables specified on the READ Statement. If the record is initially bulk memory resident, the automatic paging of subfiles is not done, and the record is read from bulk memory immediately.

Error Processing

SECTOR Option

Failure of a read disk file request will result in a Class III error.

FILE Option

The following is a list of errors which can occur while reading a disk file record.

Invalid system name - Class III.

File name specified is not open - Class III.

Bulk memory read error while paging subfile data - Class II.

Bulk memory write error while paging subfile data - Class II.

Disk read error while paging subfile data - Class III.

Disk write error while paging subfile data - Class II.

Invalid record number specified - Class III.

No work area available for GOAL Executor to use as a paging buffer between disk and bulk memory - Class II.

Checksum error in data when paging a subfile from disk to bulk memory - Class III.

Bulk memory parity and hardware errors while reading a record - Class III.

Refer to KSC-LPS-OP-033-07, Part 3 (System Message Catalog, GOAL Message Inserts) for more detailed descriptions of these errors.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. SECTOR Option

1. Sector number may be from 1 to 7.

2. Number of sectors to be read may be from 1 to 4.

3. If an internal variable is used to specify sector number or number of sectors, the variable must be a single numeric item.

4. If the STARTING AT SECTOR path is not taken, the number of sectors specified will be read, starting at sector 1.

5. If the starting sector is specified, but number of sectors is not, one sector will be read starting at the requested starting sector.

B. FILE Option

1. The disk file specified must have been opened in the console bulk memory area specified. This is not verifiable by the compiler. Failure to meet this condition will result in a Class III error at execution time.

2. The record number(s) must be within the legal limits of the specified file.

3. Internal variables used to specify record number must be numeric and may not be entire lists, table columns, or items of a list or table with variable indices.

4. The Internal Name(s) following 'AND SAVE AS' must conform in order, data type and in the cases of quantity and state data, subtype, to the record format defined for the disk file being read.

Legal Function Designators

DISK FILE (FILE) - for SECTOR option only

THIS PAGE INTENTIONALLY LEFT BLANK.

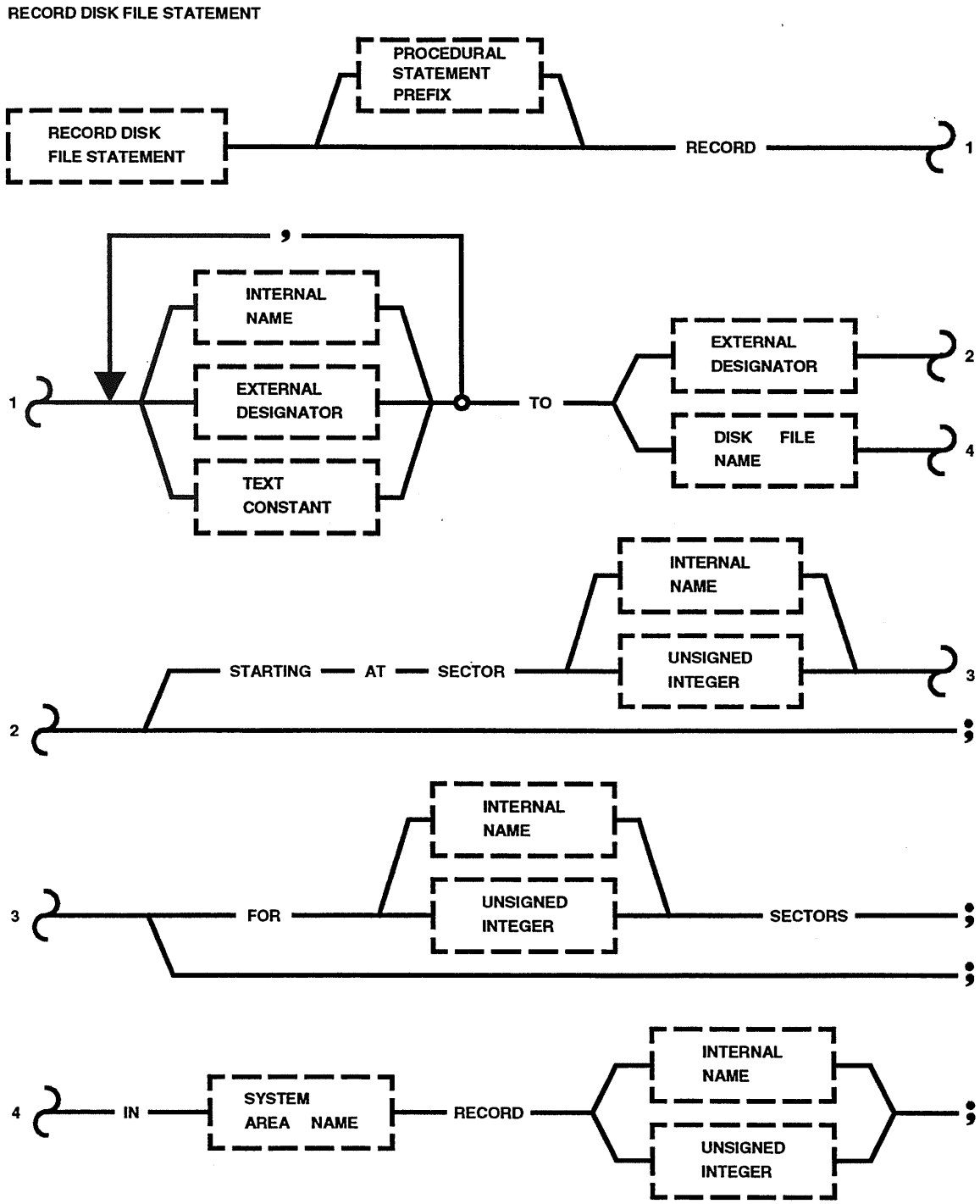


Figure 13-11 RECORD DISK FILE Statement

13.11 RECORD DISK FILE STATEMENT

Function

The RECORD DISK FILE Statement is used to write data into a disk file for storage purposes.

Description

The RECORD DISK FILE Statement is used to specify output to data files located on the console moving head disk. The output data is written in the form of binary records. There are two options available for writing to disk files:

The SECTOR option allows data to be written to a disk file which is represented by a Function Designator (<DISK-F1>, <DISK-F2>). Up to seven 128-word sectors may be written out to the disk (four sectors at a time).

The FILE option provides the means to write a record to a disk file which has been defined by a disk file definition. Data conforming to the defined format for a record may be written to a file that has been allocated and opened in the console bulk memory area specified.

SECTOR Option

Examples

```
RECORD (SLIST3) TO < DISK-F1 >;
RECORD (NLIST1) TO < DISK-F2 > STARTING AT SECTOR 3;
```

The first example writes the State List (SLIST3) to Disk File 1. The second example writes the Numeric List (NLIST1) to the third sector of Disk File 1.

FILE Option

Examples

```
RECORD (SLIST3) TO FILE (DSKFILE1) IN AREA (GLS) RECORD 5;
RECORD (N1), (N2), (TEXT1), (TEXT2), (ST1) TO FILE (DSKFILE2) IN
    AREA (XXXX) RECORD (NUM);
```

The first example writes the State List (SLIST3) to Disk File 1 specified in area (GLS) into the fifth record of the disk file. The second example specifies five Internal Names to be written into the disk file in the record number specified by (NUM).

Statement Execution

A. SECTOR Option

Requests to record data to disk files are sent to Moving Head Disk IOS, which handles the disk write operation. The GOAL procedure does not advance to the next statement until the disk write is complete.

B. FILE Option

Upon receiving a request to write a record to a disk file, the GOAL Executor will verify that the requested file has been allocated and opened in the area specified in the RECORD Statement. If not, a Class III error will result. After verifying the legality of the request, the Executor will determine whether the requested record is contained in the subfile currently resident in bulk memory. If the requested record is not bulk memory resident, the current active subfile is read from bulk memory and written to the moving head disk if necessary. This action is only necessary if the data in this subfile has changed. The subfile containing the requested record is then read from the disk and written to bulk memory. Once the record is bulk memory resident, the new data specified by the internal variables on the RECORD Statement is written to the record in bulk memory. If the record is initially bulk memory resident, the automatic paging of subfiles is not done and the new record is written to memory immediately.

Error Processing

SECTOR Option

Errors encountered on write requests will be processed as Class III errors for disk files.

FILE Option

The following is a list of errors which can occur while writing a disk file record.

Invalid system name - Class III.

File name specified is not open - Class III.

Bulk memory read error while paging subfile file data - Class II.

Bulk memory write error while paging subfile data - Class II.

Disk read error while paging subfile data - Class III.

Disk write error while paging subfile data - Class II.

Invalid record number specified - Class III.

No work area available for GOAL Executor to use as a paging buffer between disk and bulk memory - Class II.

Checksum error in data when paging a subfile from disk to bulk memory - Class III.

Write was attempted on a read only file - Class III.

Bulk memory parity and hardware errors while writing a record - Class III.

Refer to KSC-LPS-OP-033-07, Part 3 (System Message Catalog, GOAL Message Inserts) for more detailed descriptions of these errors.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. SECTOR Option

1. Sector number may be from 1 to 7.

2. Number of sectors to be written to may be from 1 to 4 sectors.

3. If an internal variable is used to specify sector number or number of sectors, the variable must be a single numeric item.

4. If the STARTING AT SECTOR path is not taken, one sector will be written starting at sector 1.

5. If the starting sector is specified but the number of sectors is not, one sector will be written starting at the requested sector.

6. No compatibility checks are made by the language processor or the GOAL Executor on data written and subsequently read from disk.

7. If the External Designator following the keyword TO is specified in the form of table functions, output will not be performed for any Row Designators which are inhibited.

8. A scratch area is used by GOAL procedures when these procedures write data to disk files. This data may be destroyed by other CCMS programs. Possible data overwrite will occur only after termination of the GOAL procedure writing the data to disk.

B. FILE Option

1. The disk file specified must have been opened without using the READ ONLY option in the console bulk memory area specified. This is not verifiable by the compiler. Failure to meet these conditions will result in a Class III error at execution time.

2. The record numbers must be within the legal limits of the specified file.

3. Internal variables used to specify record numbers must be numeric and may not be entire lists or table columns. They may also not be items of lists or tables with variable indices.

4. No automatic read of a disk file record from bulk memory will be done prior to the RECORD.

14. IN-LINE SUBROUTINES

THIS PAGE INTENTIONALLY LEFT BLANK.

The following statements will be discussed in this section:

Section	Title
14.1	BEGIN SUBROUTINE Statement
14.2	END SUBROUTINE Statement
14.3	TERMINATE SUBROUTINE Statement
14.4	INCLUDE SUBROUTINE Statement
14.5	PERFORM SUBROUTINE Statement
14.6	LOCK SUBROUTINE Statement
14.7	UNLOCK SUBROUTINES Statement

A subroutine is a sequence of GOAL statements which performs a particular function. A subroutine must start with a BEGIN SUBROUTINE Statement and end with an END SUBROUTINE Statement. A subroutine's code appears within a GOAL procedure. Subroutines may be executed by calling them from the GOAL procedure within which they are contained (parent GOAL procedure), or from other subroutines which are within the same GOAL program. The PERFORM SUBROUTINE Statement is used to accomplish this function. Subroutines may receive and return passed parameters and may also declare and define their own local variables. System subroutines from APLM may also be called if they are included in the GOAL procedure by using the INCLUDE SUBROUTINE Statement. Subroutines may be kept in memory even when they are not being executed, by using the LOCK and UNLOCK SUBROUTINE Statements. For frequently called subroutines the LOCK SUBROUTINE Statement can often be used to improve the performance of the GOAL procedure. If subroutines are called less frequently, the LOCK SUBROUTINE Statement could decrease performance. The effects of the LOCK SUBROUTINE Statement on performance are discussed in Section 14.6. The TERMINATE SUBROUTINE Statement is used to stop execution of a subroutine and return to its caller.

The following is an example of a GOAL procedure containing subroutines:

```
BEGIN PROGRAM (GOALPROC);
  .
  .
  .
LOCK SUBROUTINE (A), (B);
  .
  .
  .
```

```
PERFORM SUBROUTINE (SYST1);
  .
  .
  .
PERFORM SUBROUTINE (A) (N1),3,(Q2);
PERFORM SUBROUTINE (B);
  .
  .
  .
UNLOCK SUBROUTINES;
  .
  .
  .
PERFORM SUBROUTINE (C);
  .
  .
  .
TERMINATE;
$ SUBROUTINES SECTION $
INCLUDE SUBROUTINE (SYST1) (3);
BEGIN SUBROUTINE (A) (N11),(N2),(Q22);
DECLARE NUMBER (N11)=I,(N2)=I; DECLARE QUANTITY (Q22)=AMP;
  .
  .
END SUBROUTINE;
BEGIN SUBROUTINE (B);
  .
  .
END SUBROUTINE;
*BLOCK;
BEGIN SUBROUTINE (C);
  .
  .
END SUBROUTINE;
END PROGRAM;
```

Subroutine Special Considerations and Restrictions

A. Subroutines may not contain Macro definitions.

B. There must not be any branching into or out of a subroutine using the GO TO Statement. The only way to initiate execution of a subroutine is through the PERFORM SUBROUTINE Statement.

C. Data may be passed to a called subroutine and returned back to the caller via the use of parameters (as in a GOAL level perform).

D. All parameters received by a subroutine will be considered to be local to the subroutine and must be declared or defined within the subroutine. To say that a variable is a local variable means that it can only be referenced within the subroutine in which it is defined.

E. All variables declared or defined within a parent GOAL procedure are considered to be global, which means that they may be referenced by the procedure's subroutines without being passed as parameters. All variables declared or defined within a subroutine are considered to be local to that subroutine only. These subroutine variables must have names that are no greater than 24 characters in length. The compiler will add the subroutine name as a prefix to all local variables to distinguish them from global variables or variables that are local to other subroutines. For example, if subroutine GPROG contains a variable called TEMP, the compiler will change the variable name and all references to it to GPROGTEMP. Storage requirements for local variables will be satisfied by the inclusion of them in the parent procedure's Variable Data Area.

F. Local and global variable example:

```
BEGIN PROGRAM (GOAL1);
DECLARE NUMBER (A)=I;
DECLARE NUMBER (B)=I;
 .
 .
LET (A)=1;
 .
 .
PERFORM SUBROUTINE (SUB1);
PERFORM SUBROUTINE (SUB2) (B);
 .
 .
BEGIN SUBROUTINE (SUB1);
DECLARE NUMBER (A)=I;
 .
 .
LET (A)=2;
 .
```

```
.
END SUBROUTINE;
BEGIN SUBROUTINE (SUB2) (C);
DECLARE NUMBER (C)=I;
.
.
LET (C)=3;
LET (A)=4;
.
.
END SUBROUTINE;
END PROGRAM;
```

The variable (A) that is referenced within subroutine SUB1 is also declared within SUB1 and therefore is local to SUB1. Its name will be changed by the compiler to (SUB1A) to distinguish it from the variable (A) that is declared at the beginning of the GOAL procedure. Variable (C) is declared within subroutine SUB2 and therefore is local to SUB2. Its name will be changed by the compiler to (SUB2C). The variable (A) that is referenced within SUB2 is not defined within SUB2, and therefore it is the global variable (A) that is declared at the beginning of the GOAL procedure. Its name will not be changed.

G. All subroutines that are called from a GOAL program must be defined or included within the subroutines section of the GOAL program prior to any BEGIN SUBROUTINE Statement. The subroutines section must appear immediately before the END PROGRAM Statement. Comments and compiler directives are the only statements that may appear in the subroutines section between the subroutine definitions and INCLUDE SUBROUTINE Statements. If any of the subroutines that are called by the parent GOAL program subsequently call other subroutines, these other subroutines must also be included or defined within the subroutines section.

H. Up to 3 levels of nesting are allowed for subroutine performs.

I. If a LOCK SUBROUTINE Statement specifies more than one subroutine and the subroutines are not contiguous, then all intervening subroutines will also be locked into memory.

J. The combined length of subroutines locked into memory at any one time cannot be greater than 384 words nor span more than 3 sectors. To determine the size of a subroutine or group of subroutines, reference the expanded source for the hexadecimal addresses of the beginning and end of the subroutine(s). The *BLOCK directive may be used as needed to cause a subroutine to begin on a sector boundary.

K. Subroutine step numbers will not appear in the GOAL program's step number table. Therefore, to use a keyboard GOTO command to branch to a particular step number, the ISN of the step number must be used.

L. A GOAL program plus all its defined and included subroutines must not exceed 65,536 words in size.

M. The TERMINATE and TERMINATE SYSTEM Statements may not appear within a subroutine.

N. Subroutines may not contain SPECIFY INTERRUPT and ACTIVATE INTERRUPT PROCESSING ON THIS LEVEL AND RETURN Statements.

O. Fifteen words of variable data area space will automatically be reserved by the compiler for use by the GOAL Executor for controlling subroutine execution in procedures which define or "include" subroutines. Twelve additional words will be reserved if the procedure contains any interrupt specifications.

BEGIN SUBROUTINE STATEMENT

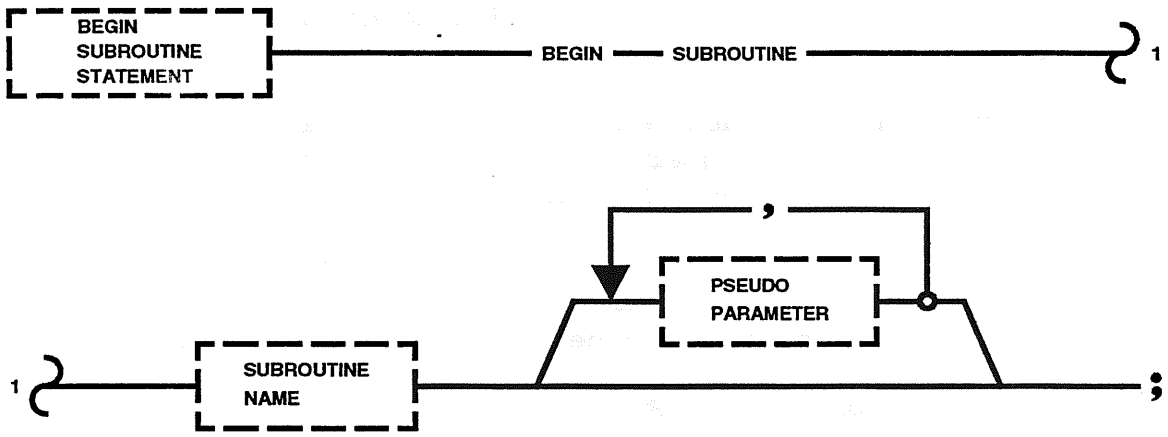


Figure 14-1 BEGIN SUBROUTINE Statement

14.1 BEGIN SUBROUTINE STATEMENT

Function

The BEGIN SUBROUTINE Statement is used to indicate the start of a subroutine, specify its name, and identify any required input parameters.

Description

The BEGIN SUBROUTINE Statement must be the first statement in a GOAL subroutine and may appear only once within a subroutine. The subroutine name follows the standard format of Name and may be up to eight characters in length. All references to the subroutine must use this Name. Pseudo parameters may optionally be provided in this statement to identify data which is provided by the calling GOAL program or calling subroutine at execution time. All pseudo parameters must be declared or defined within the subroutine. All variables declared or defined within a subroutine will be considered to be local variables to that subroutine.

All variables declared or defined within the parent GOAL program are considered to be global and may be referenced within a subroutine without being passed as parameters.

Refer to the PERFORM SUBROUTINE Statement for additional information on parameter passing.

Examples

```
BEGIN SUBROUTINE (SUBR1);
```

This example will be the first statement in the in-line subroutine SUBR1 and indicates the start of the subroutine.

```
BEGIN SUBROUTINE (SUBR2) (PARM1),<A>,<B>;
```

This example indicates the beginning of the subroutine SUBR2 and identifies the input pseudo parameters PARM1, A and B.

Statement Execution

The BEGIN SUBROUTINE Statement is transparent to the GOAL Executor if no parameters are expected by the subroutine. If parameters are expected, the Executor will obtain the data from the task work area and store it in the variable data area via store operations.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. If pseudo parameters are used, they must be declared (if Names) or defined (if Function Designators) in the subroutine.

B. Pseudo parameter names are restricted to 24 characters.

C. A maximum of 512 words may be received by a subroutine via pseudo parameters.

Legal Function Designators

All Function Designators are legal for use by this statement.

THIS PAGE INTENTIONALLY LEFT BLANK.

END SUBROUTINE STATEMENT

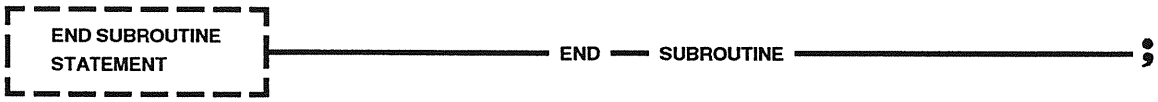


Figure 14-2 END SUBROUTINE Statement

14.2 END SUBROUTINE STATEMENT

Function

The END SUBROUTINE Statement is used to indicate to the Language Processor the end of a GOAL subroutine.

Description

The END SUBROUTINE Statement is a required statement for all subroutines and must appear as the last statement of a subroutine. The END SUBROUTINE Statement must be immediately preceded by an unconditional GOTO Statement, TERMINATE SUBROUTINE Statement, or restrictive STOP Statement. None of the above may have a Verify Prefix.

TERMINATE SUBROUTINE STATEMENT

Figure 14-3 TERMINATE SUBROUTINE Statement

14.3 TERMINATE SUBROUTINE STATEMENT

Function

The TERMINATE SUBROUTINE Statement is used to discontinue the execution of a subroutine and return to the caller.

Description

The TERMINATE SUBROUTINE Statement will stop the execution of the subroutine in which it occurs. Control is returned to the calling parent program or calling subroutine at the statement following the PERFORM SUBROUTINE Statement.

Statement Execution

The TERMINATE SUBROUTINE Statement causes a control transfer back to the caller (returning parameters if necessary).

Returning from a subroutine which was being executed because of a keyboard GOTO command is not permitted and will result in a Class II error.

Upon successful completion of a TERMINATE SUBROUTINE Statement, the GOAL program's execution priority (critical/non-critical) will return to its state prior to the subroutine perform.

Example

```
            .
            .
            .
           PERFORM SUBROUTINE (A);
  STEP 100 CONTINUE;
            .
            .
            .
            .
           BEGIN SUBROUTINE (A);
            .
            .
           TERMINATE SUBROUTINE;
            .
            .
           END SUBROUTINE;
```

Upon execution of the TERMINATE SUBROUTINE Statement, control will be returned to the parent GOAL program at the statement "STEP 100 CONTINUE".

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The TERMINATE SUBROUTINE Statement may only appear within GOAL subroutines.

B. A subroutine must contain at least one TERMINATE SUBROUTINE Statement.

THIS PAGE INTENTIONALLY LEFT BLANK.

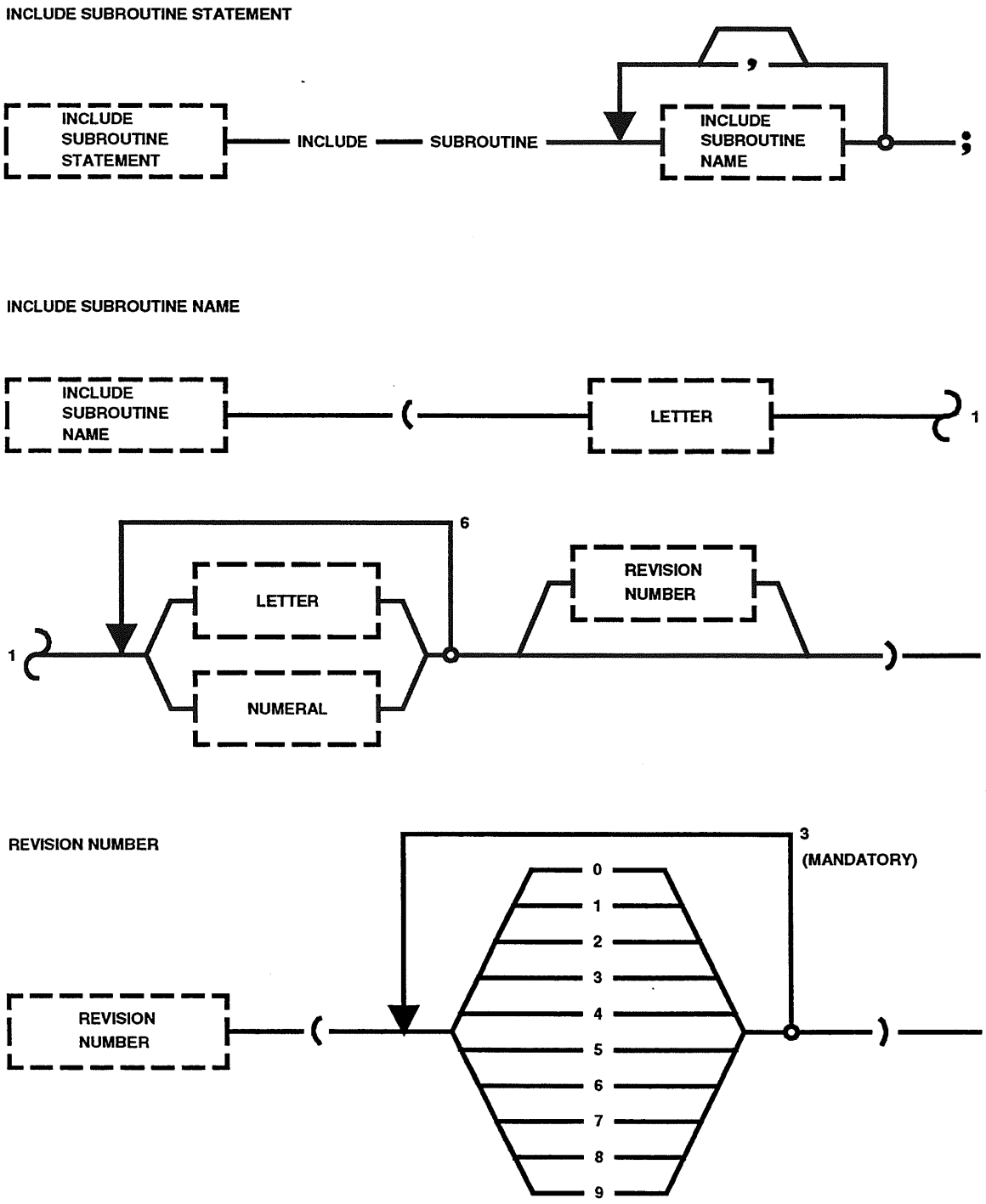


Figure 14-4 INCLUDE SUBROUTINE Statement

14.4 INCLUDE SUBROUTINE STATEMENT

Function

The INCLUDE SUBROUTINE Statement is used to indicate to the Language Processor where the specified system subroutine(s) are to be included in the subroutines section of a GOAL program.

Description

The INCLUDE SUBROUTINE Statement indicates where specified system subbroutine(s) will be included in a GOAL program. The system subroutines are located in APLM. If no revision number is specified, the highest revision of the subroutine will be used. The source code for included system subroutines will appear on the GOAL procedure's expanded source listing in place of the INCLUDE SUBROUTINE Statement.

Example

```
INCLUDE SUBROUTINE (SYSTEM1);
BEGIN SUBROUTINE (A);
  .
  .
  .
END SUBROUTINE;
BEGIN SUBROUTINE (B);
  .
  .
  .
END SUBROUTINE;
INCLUDE SUBROUTINE (SYSTEM2(0003));
```

This example indicates that the highest revision of system subroutine SYSTEM1 will be included in the GOAL program. The system subroutine SYSTEM2 will be included after subroutine B, and revision 0003 will be used.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The INCLUDE SUBROUTINE Statement may only appear in the subroutines section of a GOAL program.

B. Only one revision of a particular GOAL system subroutine may be included in any one GOAL program.

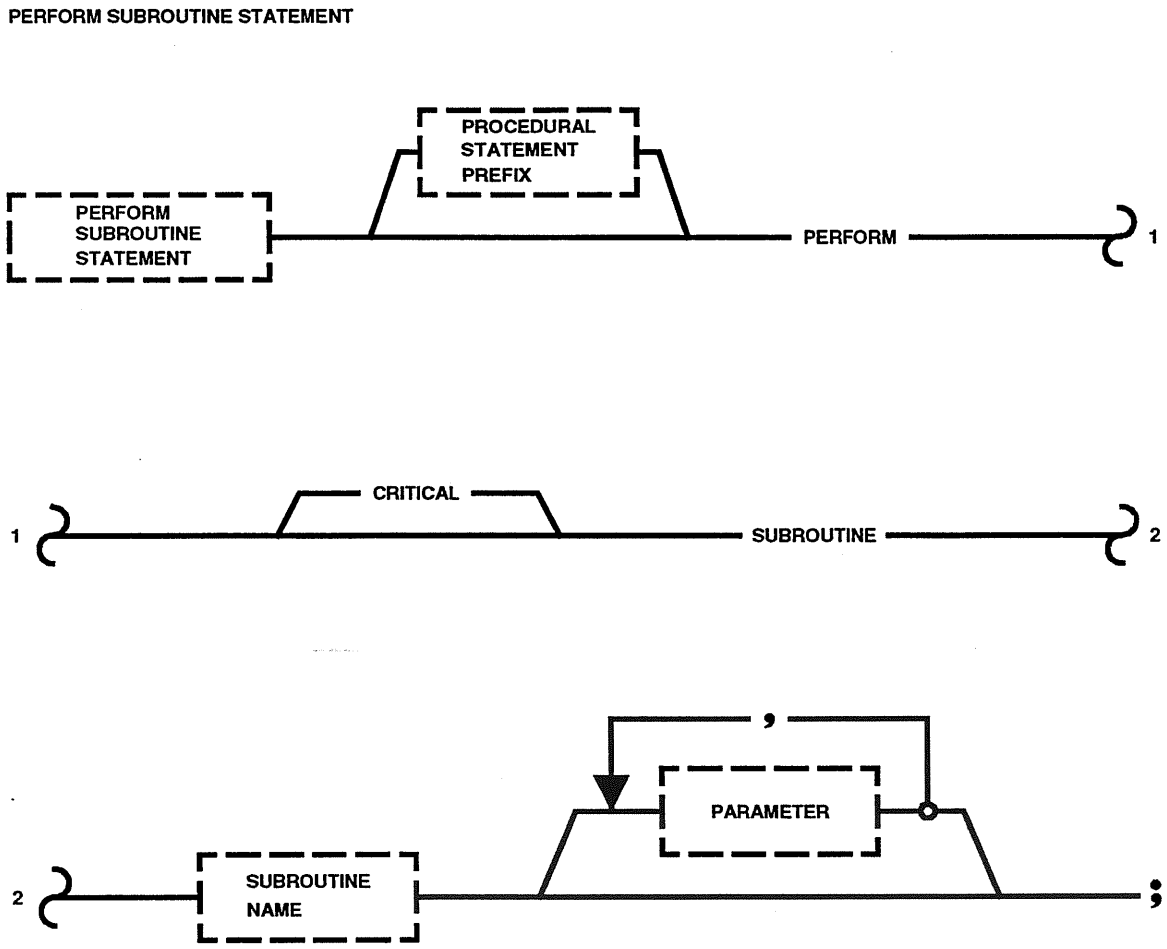


Figure 14-5 PERFORM SUBROUTINE Statement

14.5 PERFORM SUBROUTINE STATEMENT

Function

The PERFORM SUBROUTINE Statement is used to initiate execution of a specified subroutine and pass any required parameters.

Description

The PERFORM SUBROUTINE Statement causes a subroutine to be executed in series with the calling program or calling subroutine. During the time the subroutine is executing, execution of the calling program or subroutine is suspended. After termination of the performed subroutine, execution continues with the next sequential statement following the PERFORM SUBROUTINE Statement. Up to three levels of nesting are allowed for performing subroutines.

The CRITICAL Option may be used to request that a subroutine be executed with higher priority than other concurrencies being executed.

The PERFORM SUBROUTINE Statement allows parameters to be passed to the performed subroutine as well as back to the performer. This ability enables a subroutine to be performed to accomplish testing or data analysis and to pass the results back to the calling program or subroutine. However, all variables declared or defined within the parent GOAL program are considered to be global and may be referenced within a subroutine without being passed as parameters. All variables declared or defined within a subroutine will be considered to be local variables to that subroutine. If any parameter mismatches occur, the compiler will generate an error.

Examples

```
PERFORM SUBROUTINE (SUB1);
PERFORM SUBROUTINE (SUB2) (N1),<FD1>,(Q1);
PERFORM CRITICAL SUBROUTINE (SUB3);
```

The first example performs the Subroutine SUB1. The second example performs the Subroutine SUB2 and passes parameters to it. The last example performs the critical Subroutine SUB3.

Statement Execution

A subroutine is performed by the Executor via transfer of control to the first executable statement of the subroutine. This

transfer will require a disk access if the interpretive code (I/C) of this statement is not currently memory resident. Upon return from the subroutine, a control transfer to the statement following the PERFORM Statement will occur. This transfer may also require a refresh of I/C from the disk. The LOCK SUBROUTINE Statement may be used to keep a subroutine of less than 385 words memory resident, to prevent thrashing of the disk by frequent calls.

If the execution of the perform subroutine request would cause the nesting level to exceed the limit of three performs, the request will be aborted and a Class 2 error message output.

Unless the "Critical" option of the PERFORM Statement is specified, the execution priority of the performed subroutine will be the same as that of the caller. A GOAL program's execution priority (critical / non-critical) will return to its state prior to the subroutine perform upon return from a subroutine.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. A maximum of 512 words may be passed to a subroutine via parameters.

B. A subroutine may only be performed by the parent program within which it is defined or included, or by another subroutine that is defined or included within its parent GOAL program.

C. All parameters received by a subroutine must be declared or defined within the subroutine.

D. The level of nesting for subroutine performs is a maximum of three.

E. Recursion is not supported; however, no attempt will be made to detect and/or prevent recursive (nesting of same subroutine) performs.

Legal Function Designators

Any Function Designator describable on a DEFINE Statement may be passed as a parameter.

THIS PAGE INTENTIONALLY LEFT BLANK.

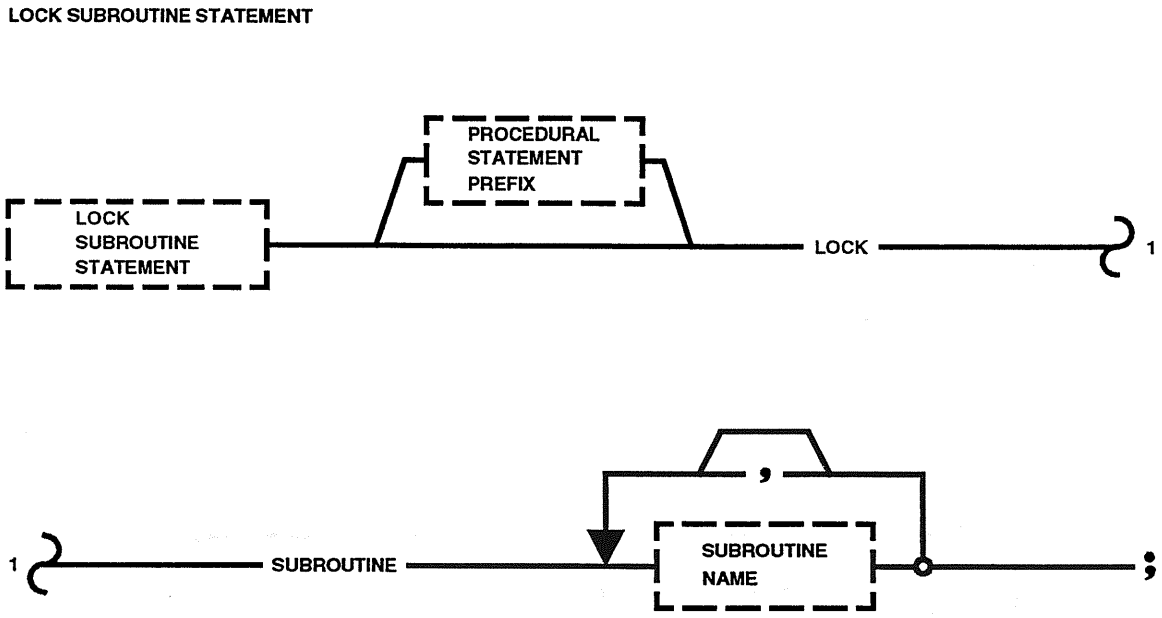


Figure 14-6 LOCK SUBROUTINE Statement

14.6 LOCK SUBROUTINE STATEMENT

Function

The LOCK SUBROUTINE Statement is used to enhance the performance of frequently called subroutines by bringing them into memory and keeping them there until they are no longer needed.

Description

The LOCK SUBROUTINE Statement will cause the subroutines specified to be read into memory from the disk. If the subroutines are not consecutive, all intervening subroutines will also be read into memory. The subroutines will remain memory resident while the parent GOAL program is the current active program level unless an UNLOCK SUBROUTINES Statement is executed. Level performs will destroy locked subroutines; however, they will be restored when a performed level terminates.

Examples

```
    .
    .
    .
  LOCK SUBROUTINE (A),(GPROC1),(GPROC2);
    .
    .
  UNLOCK SUBROUTINES;
  LOCK SUBROUTINE (A),(GPROC2);
    .
    .
    .
    .
  INCLUDE SUBROUTINE (A);
  BEGIN SUBROUTINE (GPROC1);
    .
    .
  END SUBROUTINE;
  BEGIN SUBROUTINE (GPROC2);
    .
    .
  END SUBROUTINE (GPROC2);
```

The first LOCK SUBROUTINE Statement will cause subroutines A, GPROC1 and GPROC2 to be read into memory. The second LOCK SUBROUTINE

Statement will have the same results as the first. Although GPROC1 is not specified in this statement, it will be brought into memory because it is between A and GPROC2.

Statement Execution

The Executor will verify that the GOAL program has no other currently locked subroutines and that the size of the specified subroutines spans no more than three 128 word interpretive code blocks. The lock request will be ignored and a Class IV error will be output if this is not the case. After the validity of the request has been established, a maximum of three interpretive code blocks (128, 256 or 384 words) in memory owned by the GOAL task will be allocated to the subroutines. Subsequent performs of these subroutines will not cause disk accesses to obtain interpretive code. The memory allocated to the subroutines is taken from the parent GOAL program's interpretive code area which may cause its performance to be degraded. The degree of degradation depends on the logic flow of the GOAL program.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The subroutines specified (and any intervening subroutines must not exceed 384 words nor span more than 3 sectors. The compiler will generate an error if this limit is exceeded. The *BLOCK directive may be used, if necessary, to cause a subroutine to begin on a sector boundary.

B. If a LOCK SUBROUTINE Statement is performed while the subroutines specified are already locked into memory, then execution will continue without error. However, if the subroutines that are already locked into memory are different from those specified on the LOCK Statement, a Class IV error will be processed.

THIS PAGE INTENTIONALLY LEFT BLANK.

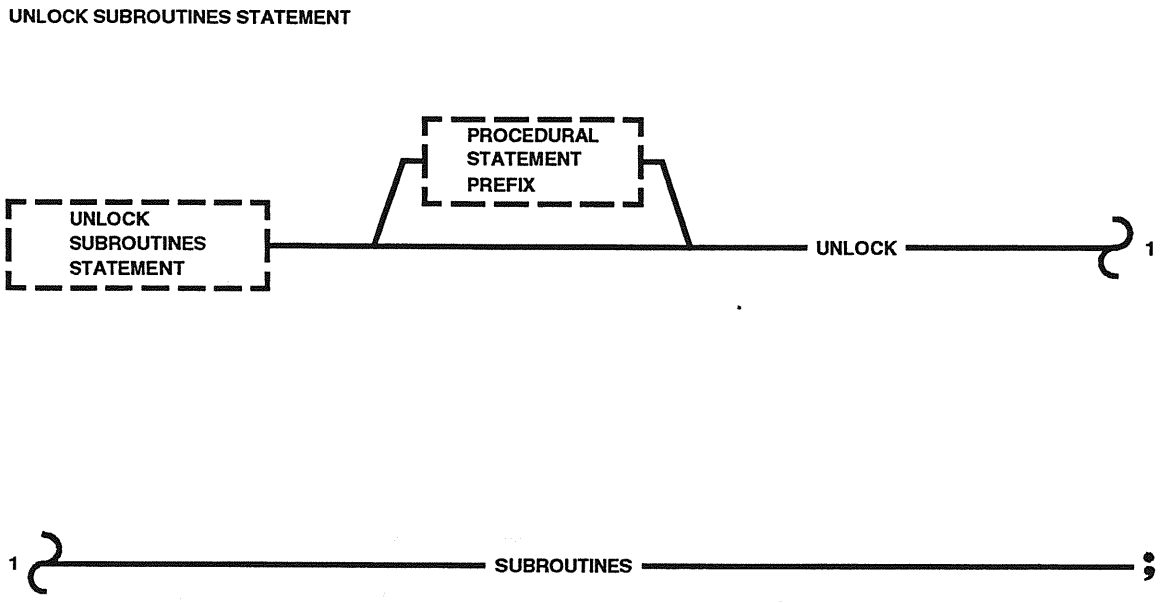


Figure 14-7 UNLOCK SUBROUTINES Statement

14.7 UNLOCK SUBROUTINES STATEMENT

Function

The UNLOCK SUBROUTINES Statement is used to specify that all currently locked subroutines no longer need to stay in memory.

Description

The UNLOCK SUBROUTINES Statement allows the memory that was used to lock subroutines into memory to be reallocated to the main program of the active program level.

Examples

```
    .
    .
    .
  LOCK SUBROUTINE (A);
    .
    .
    .
  UNLOCK SUBROUTINES;
    .
    .
    .
```

The above UNLOCK SUBROUTINES Statement will unlock subroutine A.

Statement Execution

When the UNLOCK SUBROUTINES Statement is executed, the Executor will reclaim the memory allocated to the subroutines by the LOCK SUBROUTINE Statement. This memory is then used to expand the amount of memory used by the parent program for interpretive code paging back to its original value. This action will not cause a disk access for interpretive code at this time. New interpretive code will be read from the disk when all currently memory resident code has been executed.

Restrictions/Limitations

The UNLOCK SUBROTINES Statement will always unlock all subroutines currently locked. It cannot be used to unlock only a portion of the subroutines currently locked into memory.

THIS PAGE INTENTIONALLY LEFT BLANK.

15. SYSTEM STATEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The System Statements primarily direct the Language Processor. The statements to be presented in the indicated sections are as follows:

BEGIN PROGRAM STATEMENT (Section 15.1)

END PROGRAM STATEMENT (Section 15.2)

COMMENT STATEMENT (Section 15.3)

MACROS (Section 15.4)

- BEGIN MACRO STATEMENT (Section 15.4.2)
- END MACRO STATEMENT (Section 15.4.3)
- EXPAND MACRO STATEMENT (Section 15.4.4)

BEGIN SEQUENCE STATEMENT (Section 15.5.1)

END SEQUENCE STATEMENT (Section 15.5.2)

THIS PAGE INTENTIONALLY LEFT BLANK.

15.1 BEGIN PROGRAM STATEMENT

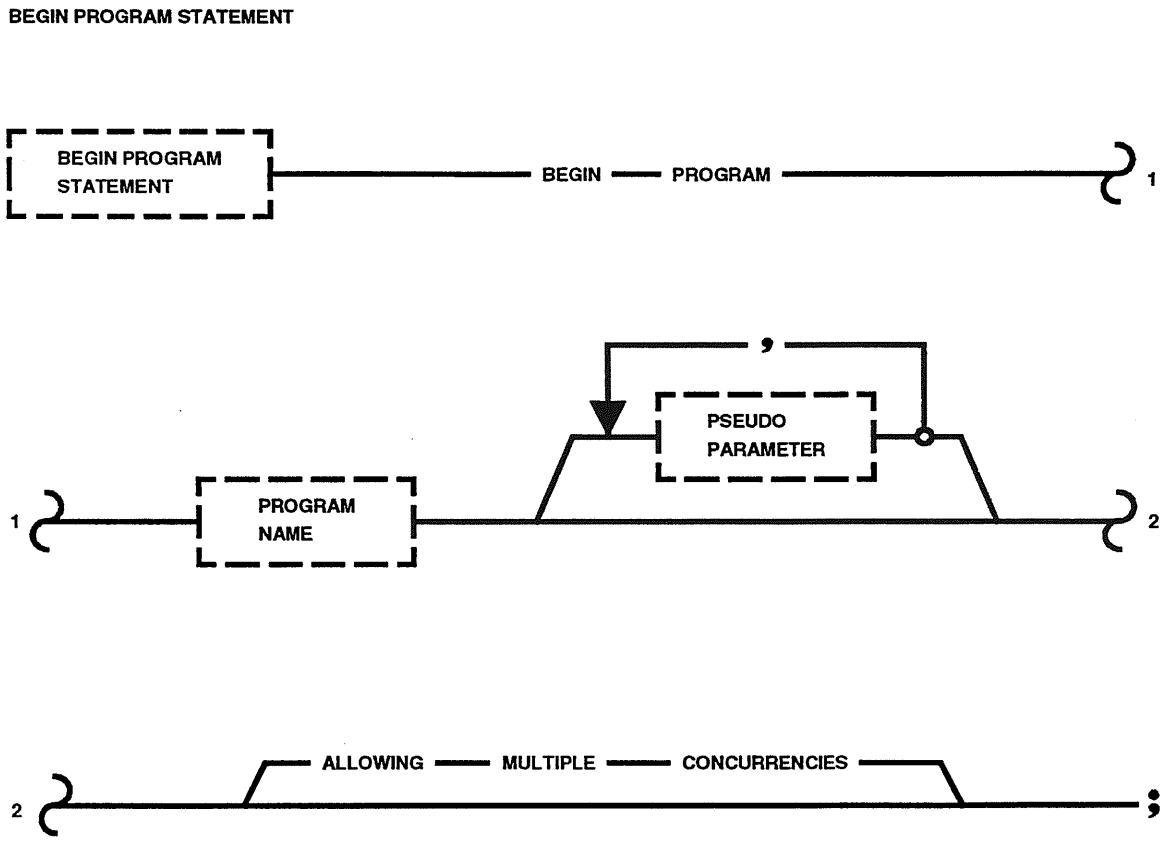


Figure 15-1 BEGIN PROGRAM Statement

Function

The BEGIN PROGRAM Statement is used to indicate the start of a GOAL procedure, specify its name, identify any required input parameters, and specify whether this procedure may run concurrently with itself.

Description

The BEGIN PROGRAM Statement must be the first statement in a GOAL procedure and may appear only once. The Program Name follows the standard format of Name and may be up to 8 characters in length. All references to the procedure must use this name. Pseudo Parameters may optionally be provided in this statement to identify data which is provided by a calling procedure at execution time. Once declared or defined, the Pseudo Parameters may be referenced by GOAL Statements in the program. There are 512 memory words available for receiving parameters at execution time. Any legal combination of data types may be received provided this limit is not exceeded. Memory words required for each data type areas follows:

Numeric - one word

Quantity - two words

State - one word

Text - two characters per memory word

Refer to the CONCURRENT and PERFORM PROGRAM Statements for additional information on parameter passing.

The Allowing Multiple Concurrencies option of the statement will allow this GOAL procedure to be executed in multiple level 1 concurrencies simultaneously. This option of the statement is designed to be used in conjunction with the keyboard PERFORM PROGRAM command for this function.

Examples

```
BEGIN PROGRAM (EXAMPLE1);
```

This example will be the first statement in the GOAL procedure EXAMPLE1 and indicates the start of the procedure.

```
BEGIN PROGRAM (TEST25) (PARM1), < A >, (PARM2);
```

This example indicates the beginning of the GOAL procedure TEST25 and identifies the input pseudo parameters PARM1, A, and PARM2.

```
BEGIN PROGRAM (TEST1) ALLOWING MULTIPLE CONCURRENCIES;
```

This example indicates the beginning of the GOAL procedure TEST1 and allows that procedure to be run concurrently with itself.

Statement Execution

The BEGIN Statement is transparent to the GOAL Executor if no parameters are expected by the procedure. If parameters are expected, the Executor will obtain the data from the task work area and store it in the variable data area via store operations. If the Allowing Multiple Concurrencies option of the statement is taken, the GOAL Executor will retrieve the bit set in the program control block and store it in the task work area for later use by the Initiator/Terminator software.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. If pseudo parameters are used they must be declared (if Names) or defined (if Function Designators) in the GOAL procedure.

B. There is a limit of 512 16-bit memory words available for receiving parameters.

C. Only compiler directives are allowed before the BEGIN PROGRAM statement.

Legal Function Designators

All Function Designators are legal for use by this statement.

15.2 END PROGRAM STATEMENT

THIS PAGE INTENTIONALLY LEFT BLANK.

END PROGRAM STATEMENT

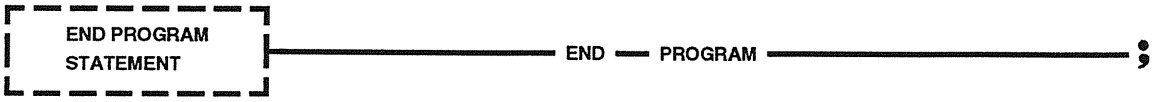


Figure 15-2 END PROGRAM Statement

Function

The END PROGRAM Statement is used to indicate to the Language Processor the end of a GOAL program.

Description

The END PROGRAM Statement is a required statement for all GOAL programs and must appear as the last statement.

Any data following the END PROGRAM Statement is flagged as an error with the exception of blank lines.

If a GOAL program has no subroutines, an END PROGRAM Statement must be preceded by an unconditional GO TO Statement, TERMINATE Statement or restrictive STOP Statement. None of the above may have a Verify Prefix. If there are any subroutines, the END PROGRAM Statement must immediately follow the END SUBROUTINE Statement or INCLUDE SUBROUTINE Statement of the last subroutine. Also, the first subroutine must be immediately preceded by an unconditional GO TO Statement, TERMINATE Statement or restrictive STOP Statement.

THIS PAGE INTENTIONALLY LEFT BLANK.

15.3 COMMENT STATEMENT

THIS PAGE INTENTIONALLY LEFT BLANK.

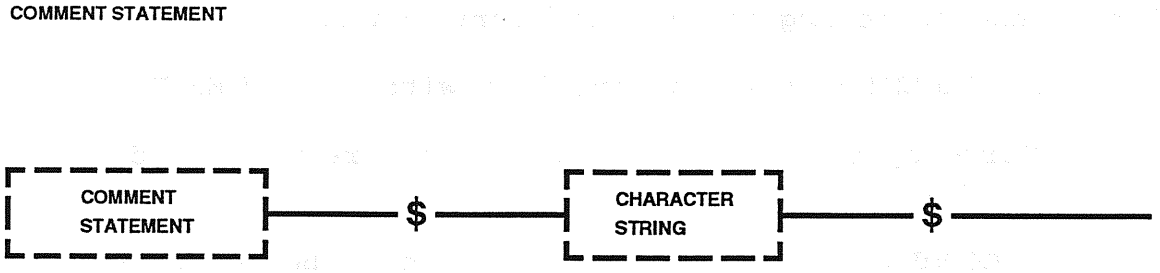


Figure 15-3 COMMENT Statement

Function

The COMMENT Statement provides the capability for annotating a GOAL procedure with descriptive information.

Description

COMMENTS must begin with a currency symbol ($) and terminate with a currency symbol. They may be inserted at any point in a GOAL procedure except where prohibited by syntax notes. COMMENTS are printed on the GOAL procedure listing and have no effect on the execution of the procedure. Comments are not placed into the interpretive code that is executed by the GOAL Executor.

Example

```
$ COMMENT EXAMPLE $
LET (Q1) $ TEXT $ = (Q2) $ 2.0 VOLTS $ + (Q3) $ 50 % OF Q2 $;
```

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. A COMMENT may not be embedded within a COMMENT.

B. Currency symbol ($) and semicolon are not valid in a COMMENT.

C. A COMMENT must contain at least one character between currency symbols.

15.4 MACROS

THIS PAGE INTENTIONALLY LEFT BLANK.

15.4.1 INTRODUCTION

The following statements will be discussed in this section:

BEGIN MACRO STATEMENT (Section 15.4.2)
END MACRO STATEMENT (Section 15.4.3)
EXPAND MACRO STATEMENT (Section 15.4.4)

A Macro is a named Character String. A Macro must start with a BEGIN MACRO Statement and end with an END MACRO Statement. The Character String, between the BEGIN MACRO Statement and END MACRO Statement, is defined as a Macro skeleton and is made up of any combination of GOAL characters. The only exception is the Character String END MACRO which is invalid.

Once the Macro is defined, each reference to the Macro Name by an EXPAND MACRO Statement will cause the Macro skeleton for that Macro to be inserted into the source program. A Macro may be inserted anywhere within a program as long as it is defined following the BEGIN Statement of the procedure, prior to its reference by an EXPAND MACRO Statement, or is resident in the standalone macro library. By convention, Macros should be grouped together and should follow the BEGIN Statement of a GOAL program, prior to any procedural statement.

The BEGIN MACRO Statement allows the test writer to assign an arbitrary Name to Pseudo Parameters. The Pseudo Parameters may be any legal Name and must be unique only within the Macro. In other words, a Pseudo Parameter defined within a Macro may also be defined as a Name within the main program. The same Pseudo Parameter may also appear in several Macros. A Pseudo Parameter may be referenced as many times as desired within the Macro skeleton.

Three options are available for specifying the printed output of Macros. These options are illustrated in the example. They are as follows:

A. EXPAND Option

This option inhibits the printing of the EXPAND MACRO Statement and allows for the printing of the macro source after the parameters have been substituted.

B. EXECUTE Option

This option inhibits the printing of the Macro source and allows for the printing of the Macro Name along with the substitution parameters.

C. EXPAND AND EXECUTE Option

This option is a combination of the Execute option followed by the Expand option.

Example

GOAL SOURCE	EXPANDED SOURCE LISTING
BEGIN PROGRAM (EXAMPLE); BEGIN MACRO (TEST) < A >, (B), (C); VERIFY < A > IS (B); SET < DS1 > TO (C); END MACRO;	BEGIN PROGRAM (EXAMPLE); BEGIN MACRO (TEST) < A >, (B), (C); VERIFY &1 IS &2; SET < DS1 > TO &3; END MACRO;
EXPAND (TEST),< DM1 >, ON, (STATE1),;	VERIFY <DM1> IS ON; SET < DS1 > TO (STATE1);
EXECUTE (TEST), < DM1 >, ON, (STATE1),;	TEST, <DM1>, ON, (STATE1),;
EXPAND AND EXECUTE (TEST), < DM1 >, ON, (STATE1),;	TEST, < DM1 >, ON, (STATE1),; VERIFY < DM1 > IS ON; SET < DS1 > TO (STATE1);
END PROGRAM;	END PROGRAM;

THIS PAGE INTENTIONALLY LEFT BLANK.

BEGIN MACRO STATEMENT

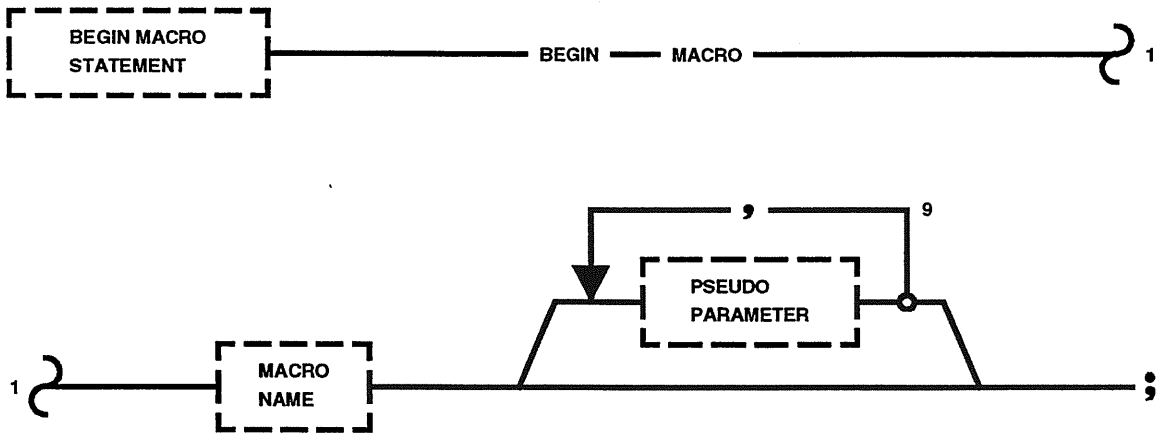


Figure 15-4 BEGIN MACRO Statement

15.4.2 BEGIN MACRO STATEMENT

Function

The BEGIN MACRO Statement is used to indicate the start of a Macro, specify its name, and identify any required substitution parameters.

Description

The BEGIN MACRO Statement must be the first statement of a Macro. It may appear only once in a Macro. It uniquely identifies the Macro, and all references to the Macro must use this name.

Macro Pseudo Parameters are specified in the form of a Name or a Function Designator. Pseudo Parameters must be unique only within the Macro; e.g., a Pseudo Parameter may also be defined as a Name within the main program. Pseudo Parameters identify substitutable arguments within a Macro which are replaced by corresponding character strings defined in the EXPAND MACRO Statement. This replacement is done during compilation by the Language Processor.

Options

Examples

```
BEGIN MACRO (TEST 2);
BEGIN MACRO (TEST 3) (PARM 1), < DS2 >, (PARM 2);
```

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. A Macro must be embedded in a GOAL program and may not be compiled as a standalone GOAL component.

B. By convention, Macros should be grouped together immediately following the BEGIN PROGRAM Statement in a GOAL program.

C. A Macro Pseudo Parameter may not be repeated on a BEGIN MACRO Statement.

D. The BEGIN MACRO Statement cannot be used inside a subroutine.

Acceptable Conditions

A Macro Pseudo Parameter may not span multiple lines.

END MACRO STATEMENT

Figure 15-5 END MACRO Statement

15.4.3 END MACRO STATEMENT

Function

The END MACRO Statement is used to indicate to the Language Processor the end of a Macro definition.

Description

The END MACRO Statement is a required statement for all Macros and must appear at the end of a Macro definition.

The END MACRO Statement may appear only once in a Macro definition.

Acceptable Conditions

The END MACRO Statement may not span multiple lines.

EXPAND MACRO STATEMENT

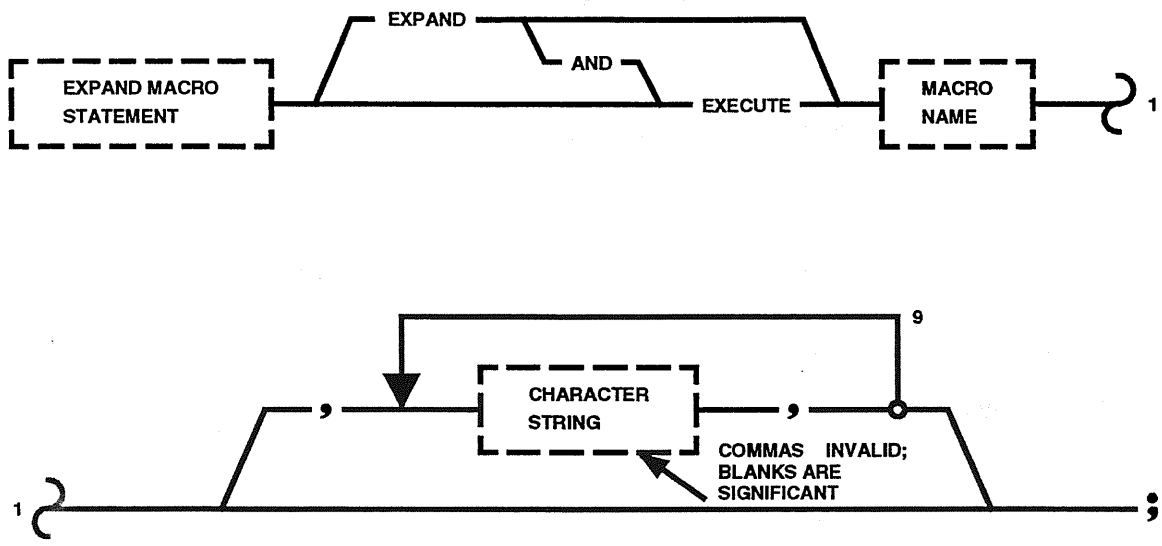


Figure 15-6 EXPAND MACRO Statement

15.4.4 EXPAND MACRO STATEMENT

Function

The EXPAND MACRO Statement allows a predefined sequence of Macro source code to be inserted into a GOAL source program and allows any required substitution parameters to be specified.

Description

The EXPAND MACRO Statement causes Macro source code to be inserted at the point in the GOAL procedure occupied by the EXPAND MACRO Statement. The MACRO NAME uniquely identifies the Macro which is to be inserted into the GOAL procedure. Substitution parameters are specified as CHARACTER STRINGS (see syntax diagram). These CHARACTER STRINGS replace corresponding pseudo parameters in the Macro source code. Substitution is performed on a one-to-one basis; i.e., the first CHARACTER STRING replaces the first pseudo parameter, the second replaces the second, until all pseudo parameters are replaced by CHARACTER STRINGS. Note that a comma must be entered before the first CHARACTER STRING and after the last.

If the Macro has not been defined within the program, the compiler will search the standalone Macro file for the Macro.

The following options are available for specifying the printouts of Macros on the Expanded Source Listing:

A. EXPAND Option

This option inhibits the printing of the EXPAND MACRO Statement and allows for the printing of the Macro source after the parameters have been substituted.

B. EXECUTE Option

This option inhibits the printing of the Macro source and allows for the printing of the Macro Name along with the substitution parameters.

C. EXPAND AND EXECUTE Option

This option is a combination of the EXECUTE option followed by the EXPAND option.

Refer to Section 15.4.2 for examples of the three options.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The number of substitution parameters must agree with the number of pseudo parameters defined in the BEGIN MACRO Statement.

B. Blanks and dollar signs are significant in a substitution parameter.

C. Each substitution parameter must be delimited by commas.

D. Each substitution parameter will be treated as a character string delimited by commas; therefore, any GOAL character (except commas and ampersands) will be passed as part of the pseudo parameter.

E. Since dollar signs are significant, all comments embedded between commas will be passed as part of a substitution parameter.

F. No more than 79 characters may be specified for a substitution parameter.

G. Commas and ampersands are not allowed in a substitution parameter.

H. Substitution parameters must agree with the type and subtype required by procedural statements within the Macro Skeleton.

I. Any substitution parameters which are passed Function Designators must be defined via the DEFINE Statement.

J. Any substitution parameters which are Internal Names must be defined via the DECLARATION Statement.

K. The EXPAND MACRO Statement cannot be used inside a subroutine.

15.4.5 STANDALONE MACROS (APLM)

The standalone macro capability allows the user to maintain a library of frequently used macros on a system file which is directly accessible by a GOAL program. When an EXPAND MACRO Statement is compiled and the macro specified is not defined inline, the standalone macro library will be searched for the macro.

MACRO NAMING/USAGE CONVENTION

For retrieval of MACRO's from the APLM LIBRARY the following convention must be used:

```
EXPAND (namexxxx);
```

Where name is the MACRONAME and xxxx is the 4 digit revision number.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Macro Name must be one to eight characters.

B. Revision Number, if specified, must be four digits.

C. If there are four trailing digits they will be interpreted as the Revision Number of the MACRO.

D. No Revision Number being specified implies a default to the highest one in the APLM.

Examples

```
  EXPAND (MAC10002);
```

Retrieve revision number two of MACRO "MAC1".

```
  EXPAND (MAC1);
```

Retrieve the highest revision of MACRO "MAC1".

```
  EXPAND (M0001);
```

Retrieve revision number one of MACRO "M".

```
  EXPAND (M001);
```

Retrieve the highest revision of MACRO "M001".

15.5 SEQUENCES

THIS PAGE INTENTIONALLY LEFT BLANK.

The following statements will be discussed in this section:

A. BEGIN SEQUENCE STATEMENT (Section 15.5.1)

B. END SEQUENCE STATEMENT (Section 15.5.2)

The BEGIN and END SEQUENCE Statements provide the boundaries for grouping several statements together and thus provide a convenient means to execute desired statements without having to perform tests and GO TO's to separate sections of code and branches back to the main line again.

The following example points out some of the capabilities:

```
VERIFY <CDT> IS LESS THAN 2 HR CDT THEN
    BEGIN SEQUENCE;
        ACTIVATE CRITICAL MODE;
        PERFORM STATEMENT GROUPS ON (N1);
            STATEMENT GROUP FOR (N1) EQUAL TO (1)
                BEGIN SEQUENCE;
                    APPLY 6.0 V to <AO-2001>;
                    APPLY 10.0 V to <AO-02>;
                END SEQUENCE;
            STATEMENT GROUP FOR (N1) EQUAL TO OTHER VALUES
                BEGIN SEQUENCE;
                    SET <DO-01> TO OFF;
                    SET <DO-02> TO ON;
                END SEQUENCE;
            END STATEMENT GROUPS FOR (N1);
            INHIBIT CRITICAL MODE;
            RECORD TEXT (2 HR CDT VALUES SET);
        END SEQUENCE;
  ELSE
            RECORD TEXT (CDT > 2 HR - OLD VALUES STILL SET);
```

A. A test (VERIFY) can precede an entire sequence to determine if the statements should be executed. A branch around (GO TO) does not have to be created.

B. Sequences can be embedded/nested within one another to provide a convenient way to make additional checks and still remain within the main line code.

C. Additionally, (not shown in the example), the REPEAT Statement enhances the sequence capability, along with the PERFORM STATEMENT GROUPS.

THIS PAGE INTENTIONALLY LEFT BLANK.

15.5.1 BEGIN SEQUENCE STATEMENT

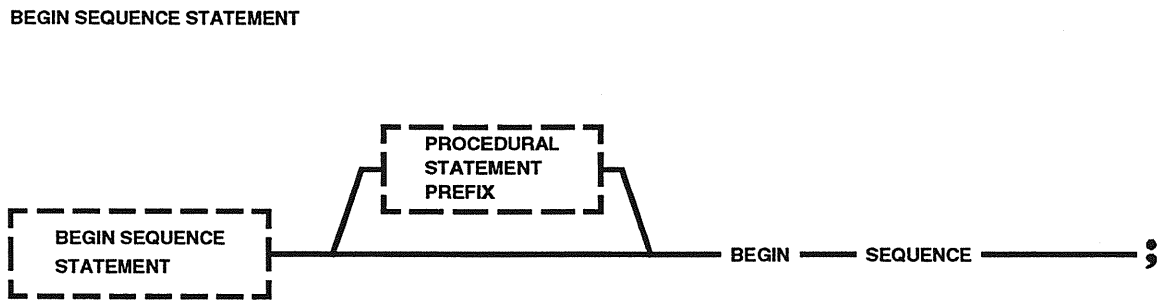


Figure 15-7 BEGIN SEQUENCE Statement

Function

The BEGIN SEQUENCE Statement indicates the start of a sequence of statements to be performed in the functional place of a single procedure statement.

Description

The BEGIN SEQUENCE Statement identifies the beginning of a sequence of procedural statements. A sequence is used:

A. To identify a sequence of statements which may be considered as a logical unit.

B. In conjunction with the REPEAT Statement to indicate those statements to be repetitively executed.

C. In conjunction with the Verify Prefix to create IF/ VERIFY-THEN-ELSE structures.

D. In conjunction with the PERFORM STATEMENT GROUPS Statement to indicate a sequence of statements to be executed for a particular condition.

Any number of procedural statements may appear in a sequence. The end of a sequence is defined by an END SEQUENCE Statement. A sequence can be embedded within another sequence. However, an embedded sequence must lie entirely within the boundaries of the sequence in which it is embedded. An example showing the use of sequences appears in Section 15.5.

Statement execution

The BEGIN SEQUENCE Statement is not an executable statement. It merely serves to indicate the beginning of the sequence.

Error Processing

Not Applicable

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. A BEGIN SEQUENCE Statement can appear only once in a sequence unless it defines the start of an embedded sequence.

B. An END SEQUENCE Statement must be specified for each BEGIN SEQUENCE Statement to define the end of the sequence.

C. Branching into a sequence is illegal; however, branching out of the sequence is permitted as if branching within a sequence.

Legal Function Designators

None

15.5.2 END SEQUENCE STATEMENT

END SEQUENCE STATEMENT

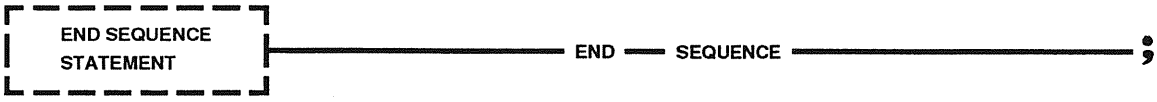


Figure 15-8 END SEQUENCE Statement

Function

The END SEQUENCE Statement indicates the end of a sequence of statements to be performed in the functional place of a single procedural statement.

Description

The END SEQUENCE Statement identifies the end of a sequence of procedural statements. Usage of a sequence is discussed in Section 15.5 and Section 15.5.1.

Statement Execution

The END SEQUENCE Statement is not an executable statement. It merely serves to indicate the end of a sequence of procedural statements.

Error Processing

Not Applicable.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. One and only one END SEQUENCE Statement must be specified for every BEGIN SEQUENCE Statement and must follow the last statement of a sequence.

B. The END SEQUENCE Statement can appear only once in a sequence unless it defines the end of an embedded sequence.

Legal Function Designators

None

THIS PAGE INTENTIONALLY LEFT BLANK.

16. GOAL ELEMENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

The GOAL elements are the basic syntax elements which make up the GOAL Statements. Some of the GOAL elements have already been discussed along with the statements in which they appear. The following classifications of GOAL elements arranged in alphabetical order by class will be discussed:

CHARACTERS (Section 16.1)

COMPARISON TEST (Section 16.2)

ENGINEERING UNITS (DIMENSIONS) (Section 16.3)

FUNCTION DESIGNATORS (Section 16.4)

INTERNAL NAME (Section 16.5)

NAMES (Section 16.6)

NUMBER REPRESENTATION (Section 16.7)

PROCEDURAL STATEMENT PREFIXES (Section 16.8)

TEXT CONSTANT (Section 16.9)

THIS PAGE INTENTIONALLY LEFT BLANK.

16.1 CHARACTERS

THIS PAGE INTENTIONALLY LEFT BLANK.

The following elements (in alphabetical order) will be discussed in this section:

CHARACTER

CHARACTER STRING

LETTER

NUMERAL

SYMBOL

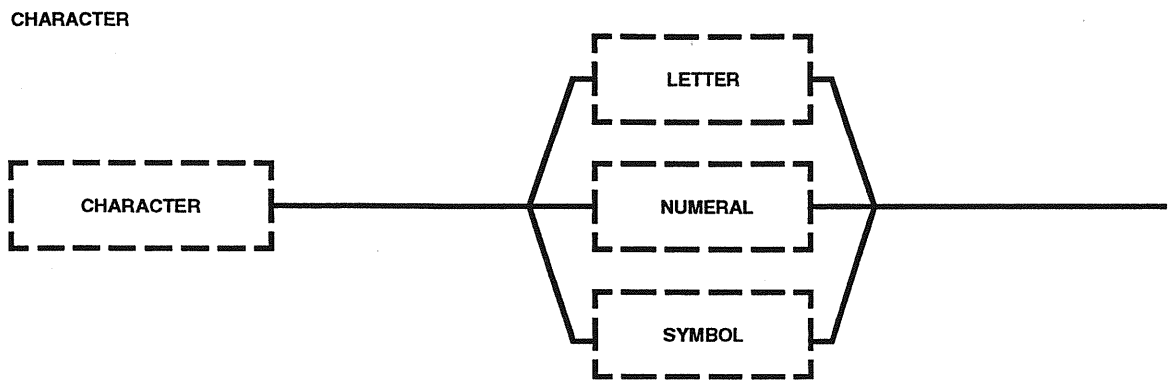


Figure 16-1 CHARACTER

Function

A CHARACTER specifies any element of the GOAL character set.

Description

Characters can be any Letter, Numeral, or Symbol.

The Processor will recognize each CHARACTER by its unique code.

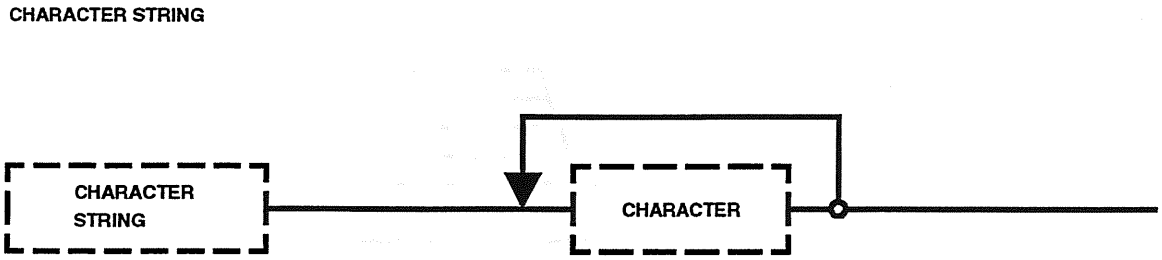


Figure 16-2 Character String

Function

A CHARACTER STRING specifies a group of characters.

Description

A CHARACTER STRING is any combination of Letters, Symbols or Numerals placed side by side to form a string.

The Processor will recognize each Character in the CHARACTER STRING by its unique character code.

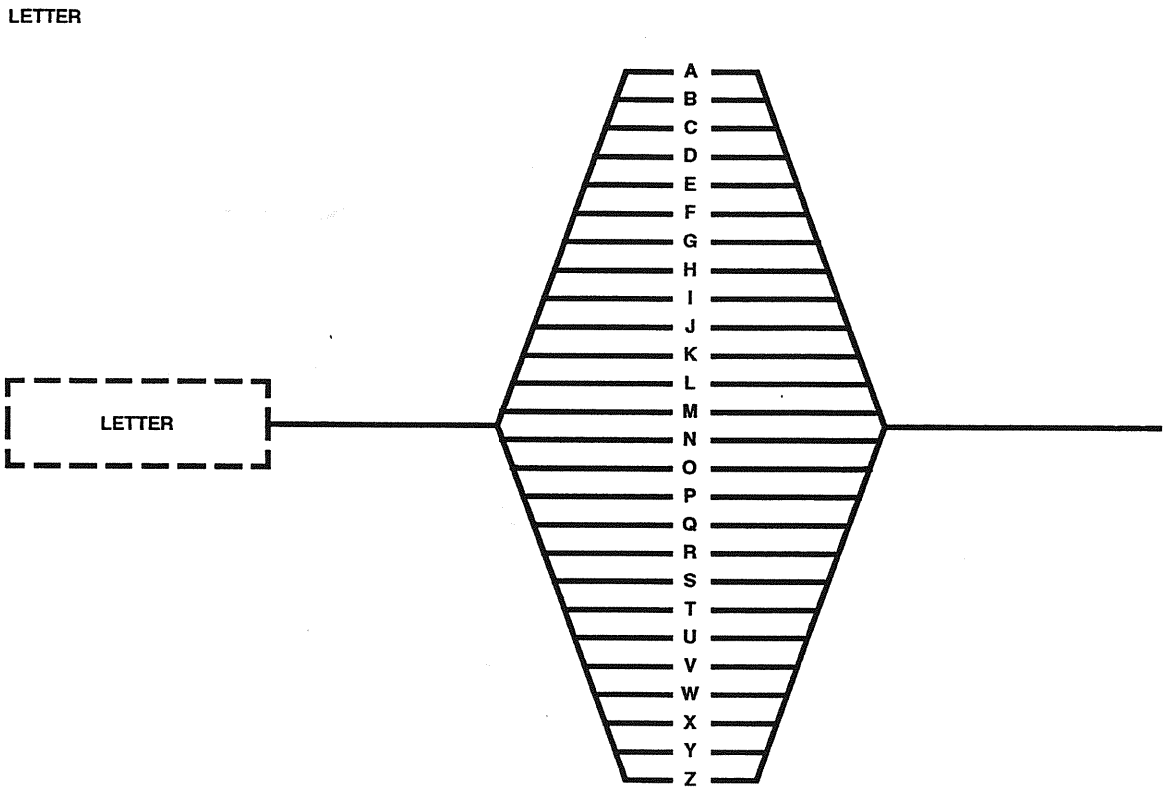


Figure 16-3 LETTER

Function

A LETTER specifies a letter of the alphabet.

Description

Letters are those characters in the GOAL Language character set that correspond to the upper case symbols A through Z.

The Processor will recognize LETTERS by their unique character code.

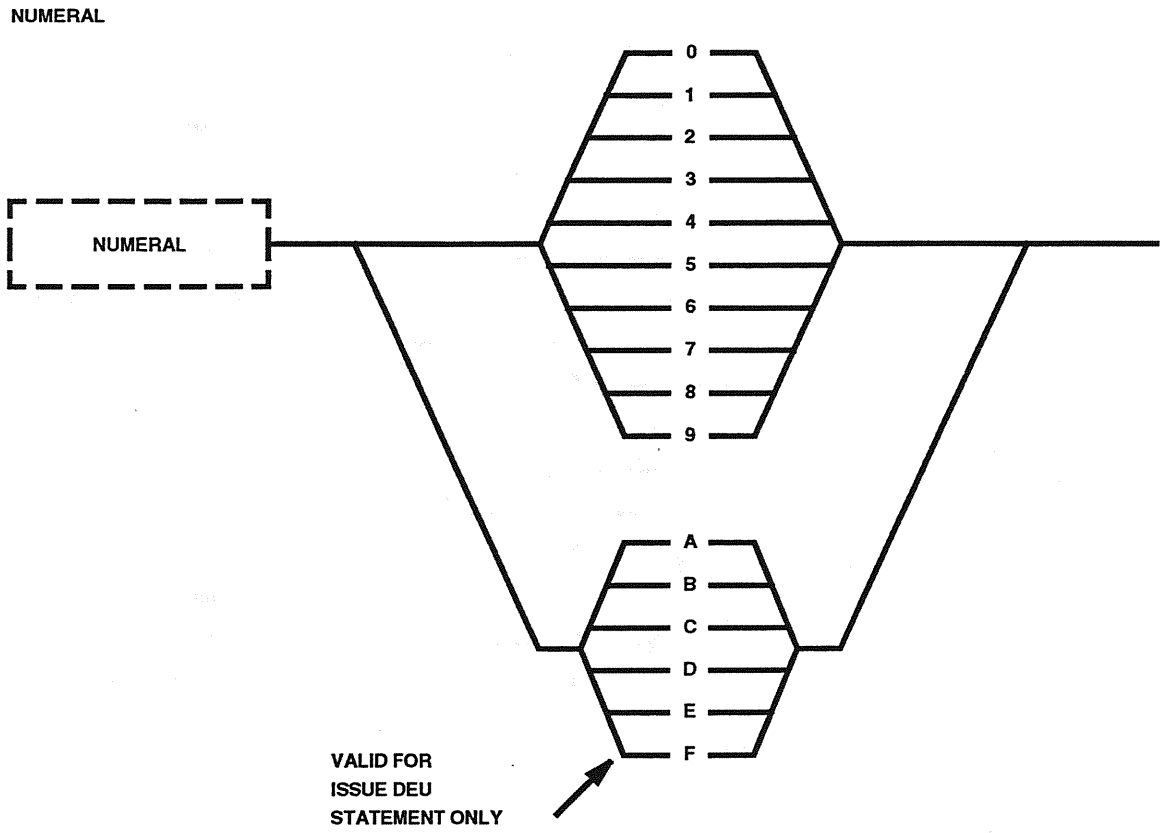


Figure 16-4 NUMERAL

Function

A NUMERAL specifies a decimal digit.

Description

Numerals are those characters in the GOAL Language character set that correspond to the digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

The Processor will recognize NUMERALS by their unique character code.

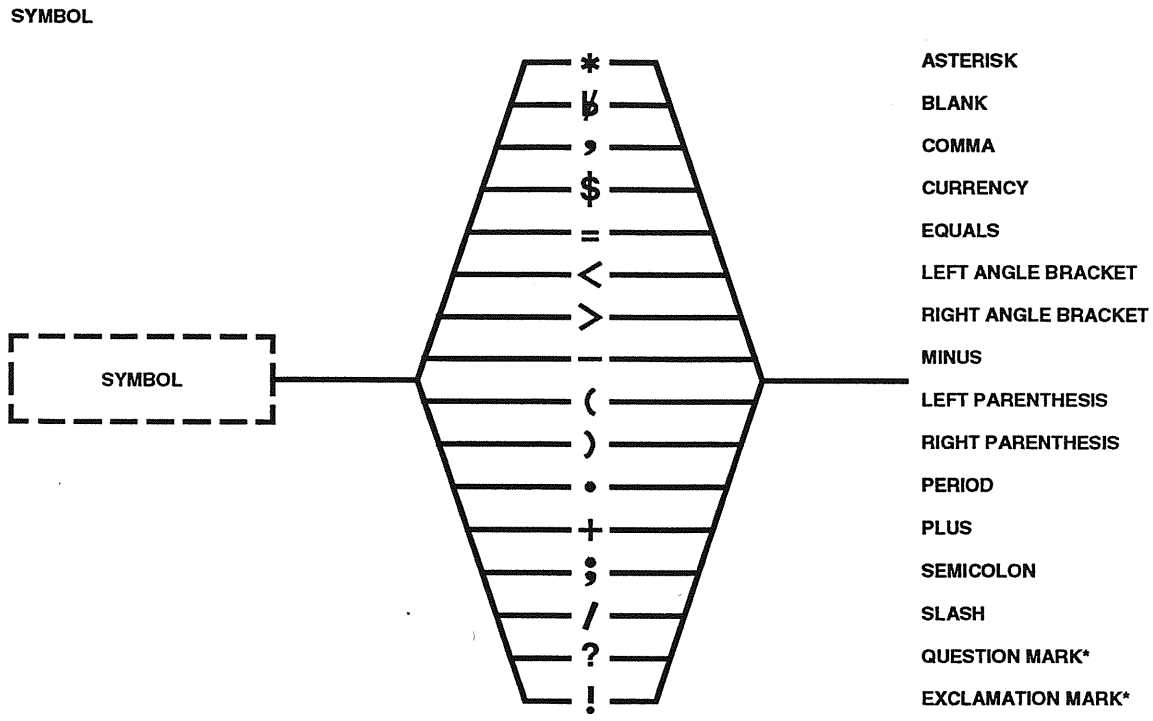


Figure 16-5 SYMBOL

Function

A SYMBOL specifies a non-alphanumeric character.

Description

Symbols are those characters in the GOAL Language character set that correspond to the following: Asterisk, Blank, Comma, Currency, Equal, Left Angle Bracket, Right Angle Bracket, Minus, Left Parenthesis, Right Parenthesis, Period, Plus, Semicolon and Slash.

All Symbols must be ASCII and EBCDIC compatible.

The Processor will recognize SYMBOLS by their unique character code.

16.2 COMPARISON TEST

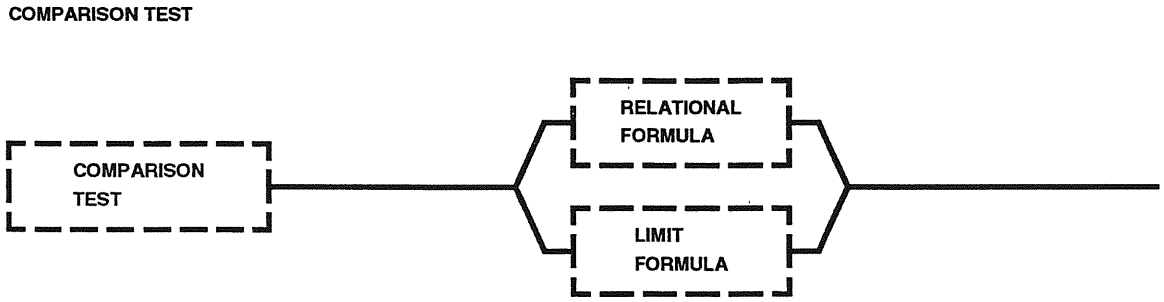


Figure 16-6 Comparison Test

Function

The COMPARISON TEST is used to allow a choice between the Relational Formula and the Limit Formula.

Description

The COMPARISON TEST may be used with Internal Names or External Designators. The data types of the items being compared must be compatible. If the COMPARISON TEST is used with the Function Designators, the type of Comparison Test must agree with the Function Designator type according to the following table:

Function Designator Type	Comparison Test Type
Analog (AM, AS, AMS, ASS)	QUANTITY
Analog Double Precision (AMDP)	QUANTITY
Discrete (DM, DS, DMS, DSS)	STATE
Digital Pattern (DPM, DPS, DPMS, DPSS)	NUMERIC
PFP Light (PFPL)	STATE (ON/OFF)
Greenwich Mean Time (GMT)	QUANTITY (GMT)
Countdown Time (CDT)	QUANTITY (CDT)
Interval Time (TIMR)	QUANTITY (INTERVAL TIME)
Julian Time of Year (CDT)	QUANTITY (JTOY)
Floating Point (FP)	QUANTITY
Multi-Word Digital Pattern (MWDP)	NUMERIC (LIST)
Mission Elapsed Time (CDT)	QUANTITY (MET)

A. For Quantity type Comparison Tests, the Dimensions (Engineering Units) of the item(s) being tested must be compatible with the Dimensions of the test criteria.

B. For State type Comparison Tests, the Subtype, (e.g., ON/OFF, WET/DRY), of the item(s) being tested must agree with the subtype of the test criteria.

C. When a units mismatch occurs while using the Table Functions option, a compiler warning message will be output.

D. If the Comparison Test is used with Function Designators specified as Table Function, the test is not performed on any Row Designators that are inhibited. These comparisons are regarded as being successful.

LIMIT FORMULA

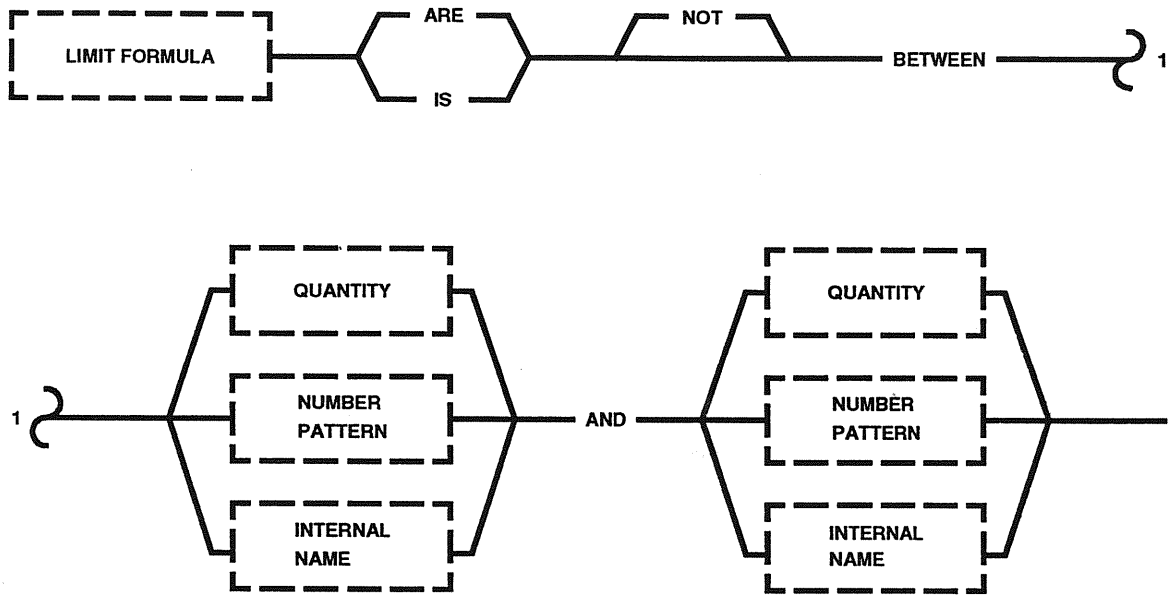


Figure 16-7 Limit Formula

Function

The LIMIT FORMULA is used to test the range of limits of a value.

Description

The LIMIT FORMULA is an option of the COMPARISON TEST which may be used to test either Internal Names or Function Designators. The values of the limits may be Text type data, Quantity or Numeric data. If quantity data is mixed with numeric data, the numeric item will be converted to quantity.

Data items may be tested to determine if they are within limits by using the IS/ARE BETWEEN option, or they may be tested to determine if they are out of limits by using the NOT BETWEEN option. Refer to RELATIONAL FORMULA for text comparisons.

BETWEEN Quantity and Quantity Option

Examples

```
VERIFY < AM1 > IS BETWEEN -5V AND 5V;
IF (Q7) IS BETWEEN 1.5 AMP AND 3.5 AMP, GO TO STEP 5;
VERIFY < AM1 >
       < AM2 >
       < AM3 > ARE NOT BETWEEN -5V and 5V;
```

BETWEEN Number Pattern and Number Pattern Option

Examples

```
VERIFY < DPM > IS BETWEEN B101 AND B1101;
IF (N8) IS BETWEEN 5 and 10, GO TO STEP 5;
```

BETWEEN Internal Name and Internal Name Option

Examples

```
VERIFY < AM1 > IS BETWEEN (Q1) and (Q5);
IF (QLIST1) IS BETWEEN (QLIST4) AND (QLIST4), GO TO STEP 5;
IF (QLIST1) IS BETWEEN (QLIST3) AND (Q2), GO TO STEP 5;
IF (NTABLE1) COLUMN 2 IS BETWEEN COLUMN 1 AND COLUMN 3,
    GO TO STEP 5;
VERIFY (QTABLE1) FUNCTIONS ARE NOT BETWEEN COLUMN 1
    AND COLUMN 2;
```

BETWEEN Text and Text Option

Examples

```
VERIFY (TEXT1) IS BETWEEN TEXT (AABB) AND TEXT (BBCC);
IF (TEXTWORD) IS NOT BETWEEN (TEXTFIRST) AND (TEXTLAST) GO
    TO 200;
```

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The limits may be specified by single values, lists, table columns, or a single value and a list or table column.

B. If more than one value is specified, the limits test must be satisfied for all values in order to have a successful comparison.

C. Refer to the COMPARISON TEST for details on compatibility of data types and comparison types.

D. The end points of a limit test are inclusive; that is, a value equal to an end point is considered between the end points.

THIS PAGE INTENTIONALLY LEFT BLANK.

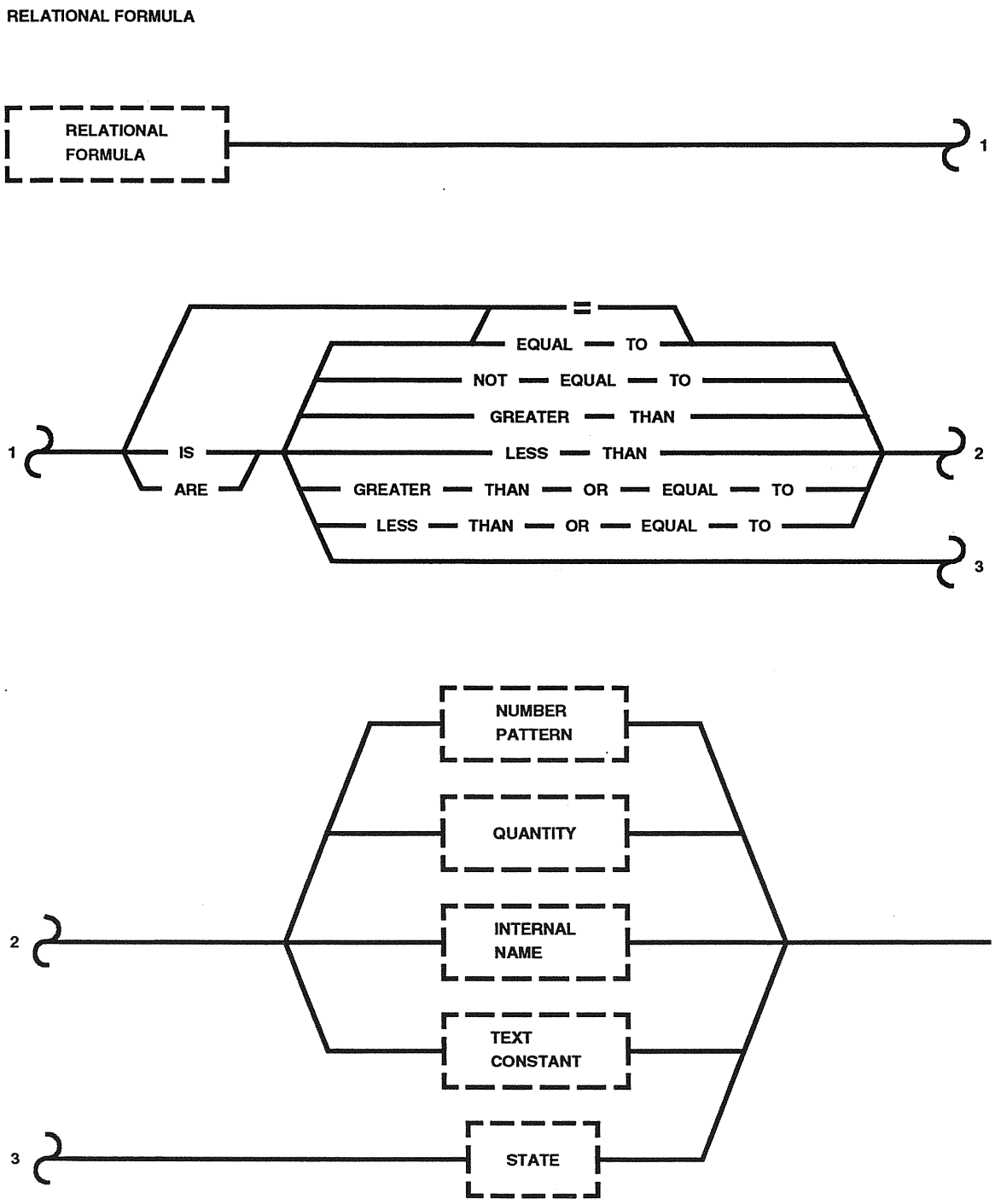


Figure 16-8 Relational Formula

Function

The RELATIONAL FORMULA is used to specify the comparison criteria for a relational test.

Description

The RELATIONAL FORMULA is an option of the COMPARISON TEST which may be used to test either Internal Names or Function Designators. The items to be tested may be compared against Numeric, Quantity, State, or Text type data. The relational tests which may be performed are: EQUAL TO, NOT EQUAL TO, GREATER THAN, LESS THAN, GREATER THAN OR EQUAL TO, and LESS THAN OR EQUAL TO. State constants may be tested using IS/ARE only.

The following table defines the data types which are legal for use with each relational operator:

Relational Operator	Numeric	Quantity	State	Text
EQUAL TO	OK	-	-	OK
NOT EQUAL TO	OK	-	-	OK
GREATER THAN	OK	OK	-	OK
LESS THAN	OK	OK	-	OK
GREATER THAN OR EQUAL TO	OK	OK	-	OK
LESS THAN OR EQUAL TO	OK	OK	-	OK
IS/ARE	-	-	OK	-

It should be noted that EQUAL and NOT EQUAL may not be used with Quantity data. Time Values are an exception to the rules. EQUAL and NOT EQUAL may be specified when referencing a Time Value quantity constant or Internal Name. The only relational operator that applies to State data is IS/ARE. Also, text data may not be compared using IS/ARE.

Text comparison

Any mixture of text Internal Names and Text Constants may be compared. The text entries do not have to be of equal length. Text entries will be compared from left to right, one character at a time, until the characters are not equivalent or until the end of one or both of the entries is encountered. The shorter text entry will be considered to be less than (i.e., come before) the longer entry, unless the text difference is the number of trailing blank characters. In this case the text entries will be considered to be equal.

The following collating sequence will be used for text comparisons:

blank	;	a
!	<	b
"	=	c
#	>	.
%	?	.
&	@	.
'	A	x
(	B	y
)	C	z
*	.	{
+	.	|
,	.	|
-	X	}
.	Y	~
/	Z	
0	[	
1	\	
2	]	
.	^	
.	_	
.	\	
9		
:		

Figure 16-9 Special DG Character Codes Collating Sequence

RELATIONAL FORMULA Number Pattern Option

Examples

```
IF (N9) = 9, GO TO STEP 5;
IF (N9) IS LESS THAN 10, GO TO STEP 5;
VERIFY < DPM1 > = XFF;
```

RELATIONAL FORMULA Quantity Option

Examples

```
IF (Q1) IS GREATER THAN 5V, GO TO STEP 5;
VERIFY < CDT > = -20 MIN CDT THEN GO TO STEP 5;
```

RELATIONAL FORMULA Internal Name Option

Examples

```
IF (TEXT1) = (TEXT2), GO TO STEP 5;
IF (QLIST1) IS GREATER THAN (QLIST3), GO TO STEP 5;
IF (QTABLE1) COLUMN 1 IS LESS THAN COLUMN 2, GO TO STEP 5;
VERIFY < AM1 > IS GREATER THAN 3 V THEN GO TO STEP 5;
```

RELATIONAL FORMULA Text Constraint Option

Examples

```
IF (TEXT1) IS LESS THAN (TEXT2), GO TO STEP 15;
IF (TEXT1) = TEXT (OPTION 1), GO TO STEP 5;
IF (TLIST1) IS NOT EQUAL TO TEXT (POWER OFF), GO TO STEP 5;
IF (TEXT1) IS GREATER THAN (TEXT2), GO TO STEP 10;
```

RELATIONAL FORMULA State Option

Examples

```
IF (STATE1) IS ON, GO TO STEP 5;
VERIFY (STABLE1) FUNCTIONS ARE OFF ELSE RECORD EXCEPTION
    AND GO TO STEP 5;
```

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The relational criteria may be a single value or a list or column of values. If more than one value is specified the relational comparisons must be satisfied for all values in order to have a successful comparison.

B. Refer to the COMPARISON TEST for additional details regarding repetitive operations and compatibility of data types.

THIS PAGE INTENTIONALLY LEFT BLANK.

16.3 ENGINEERING UNITS (DIMENSIONS)

THIS PAGE INTENTIONALLY LEFT BLANK.

The following elements used to specify Engineering Units are included in this section:

DIMENSION

QUANTITY

STATE

TIME VALUE

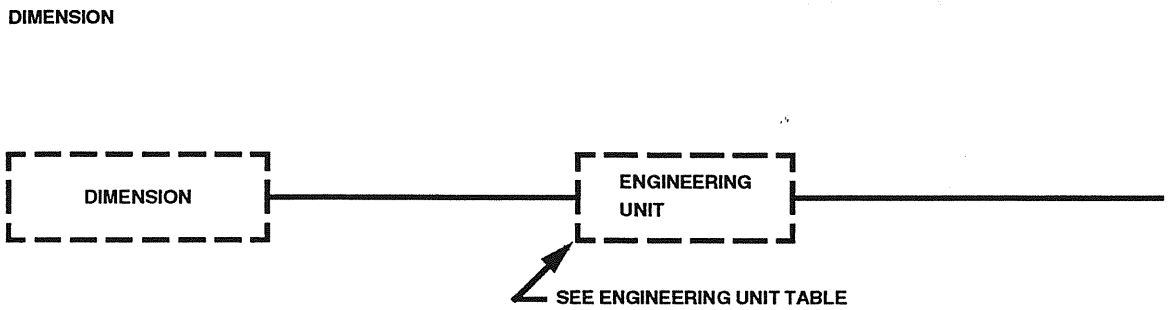


Figure 16-10 DIMENSION

Function

The DIMENSION element provides a readable description of the engineering units associated with Quantity data and supports compatibility checks and usage validation of Quantity data.

Description

Dimensions must be referenced by the abbreviations given in Appendix A.

The DIMENSIONS of quantity data must be compatible with associated Function Designators in the following statements:

READ
AVERAGE
APPLY
VERIFY PREFIX
REPEAT UNTIL

The compiler checks these statements to insure that the DIMENSIONS of quantity data and the DIMENSIONS of the associated Function Designators are the same.

The DIMENSIONS of internal data items being compared must be compatible. This is the case with the Verify Prefix, IF option.

Also, the DIMENSIONS of quantity parameters must be compatible between the calling and called programs.

Dimensions used in the LET EQUAL Statement need not be compatible.

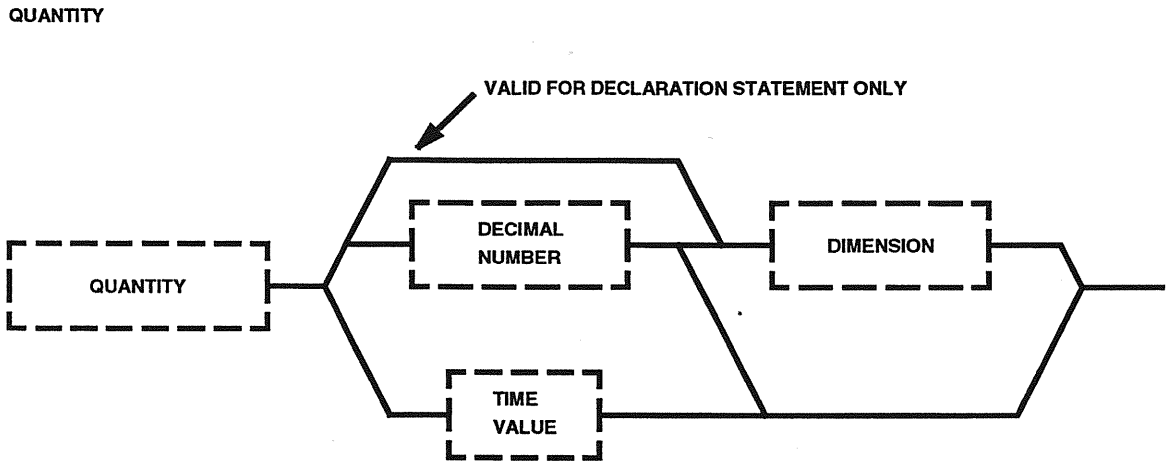


Figure 16-11 QUANTITY

Function

QUANTITY specifies the value of a QUANTITY type data item and optionally assigns dimensional characteristics to it.

Description

A QUANTITY may be specified as Time Value.

A QUANTITY may be a Decimal Number followed by DIMENSION. If DIMENSION is omitted the quantity is treated as a dimensionless constant; e.g., a ratio or coefficient.

Only the DIMENSION need be provided when an initial value is not specified in a QUANTITY declaration statement. This form must be used if the data being declared is a pseudo parameter.

If only the DIMENSION is specified and the value being declared is not a pseudo parameter, it must be initialized in the program by a Procedural Statement, such as LET EQUAL.

THIS PAGE INTENTIONALLY LEFT BLANK.

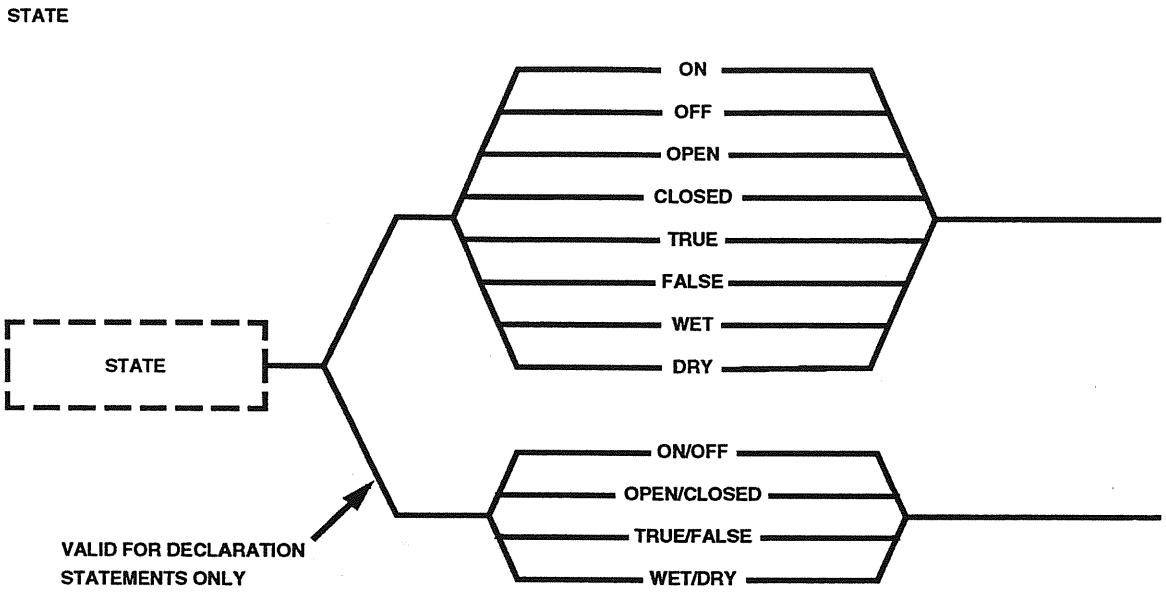


Figure 16-12 STATE

Function

The STATE element specifies the value of a state type data item in any of the permissible formats.

Description

The allowable STATE values are: ON, OFF, OPEN, CLOSED, TRUE, FALSE, WET and DRY.

The formats ON/OFF, OPEN/CLOSED, TRUE/FALSE, and WET/DRY are used to indicate the operational sense when an actual initial value is not specified in a STATE DECLARATION Statement. This form must be used if the data being declared is a Pseudo Parameter.

If the operational sense format is being used and the value being declared is not a Pseudo Parameter, it must be initialized in the program by a Procedural Statement, such as ASSIGN.

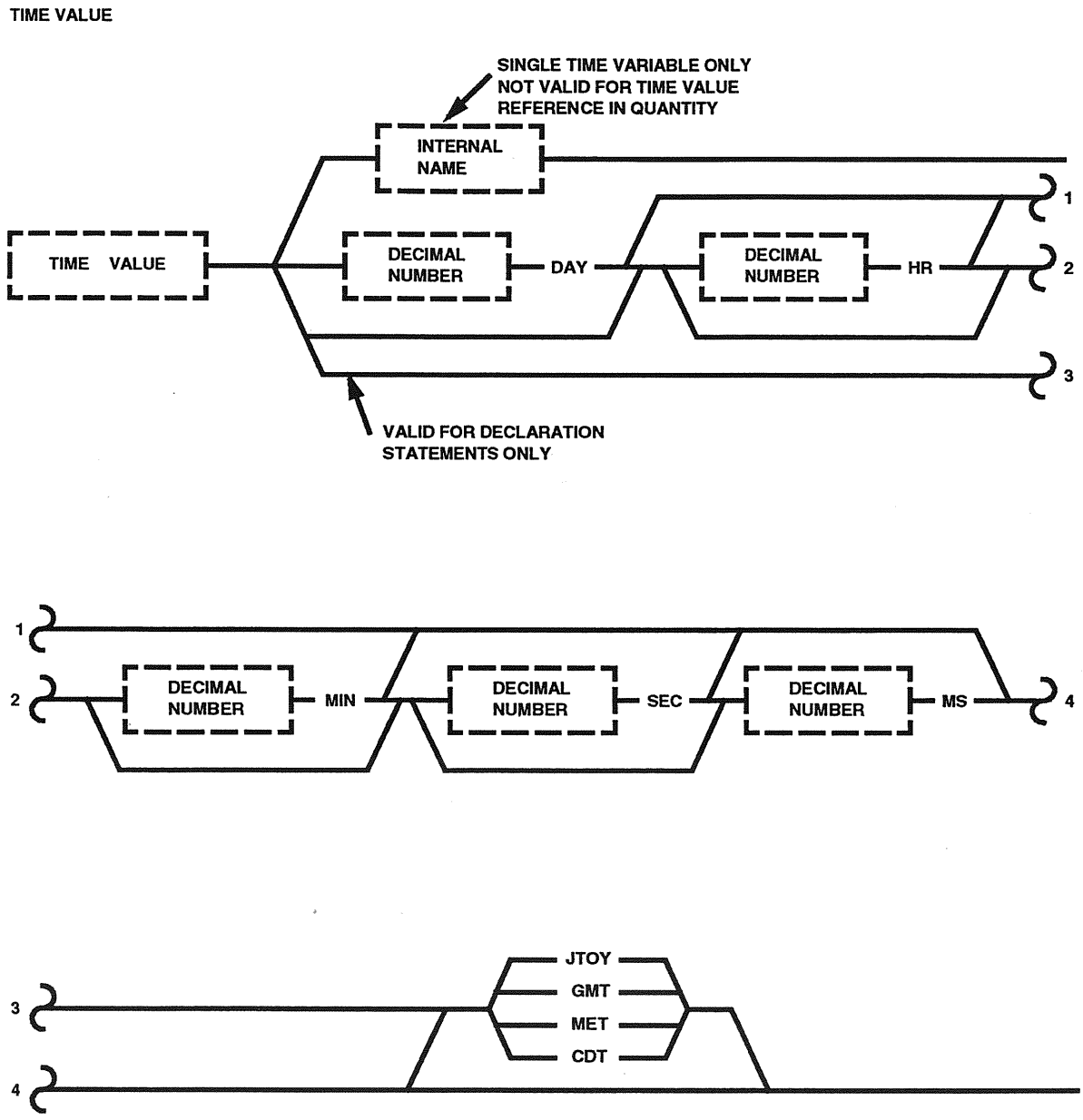


Figure 16-13 Time Value

Function

The TIME VALUE element is used to specify a time.

Description

TIME VALUE may be specified in the form of an Internal Name or as a constant. If an Internal Name is used, it must be declared as Quantity data with the dimension of time. A constant Time Value may be specified as a combination of days, hours, minutes, seconds, and milliseconds. At least one of these time units must be specified if this option is used. If a sign is used, it applies to the total of the individual time units which are specified.

Greenwich Mean Time (GMT), Count Down Time (CDT), Julian Time of Year (JTOY) and Mission Elapsed Time (MET) may be used as optional suffixes to a constant Time Value. If suffixes are not used, the value is considered to be a time interval. GMT, CDT, JTOY, and MET may also be used along to specify dimensional type of an uninitialized entry on a DECLARE QUANTITY Statement.

NOTE: *MET is a Cargo Integration Test Equipment (CITE) unique suffix. If MET is used in a CCMS system, it will default to CDT.*

Time value constants are converted to binary integers by the GOAL Language Processor. All other quantity constants are processed as floating point values.

Examples

```
DELAY (TIME1);
DELAY 1 HR 5 MIN 5 SEC;
VERIFY < CDT > = -7 HR CDT THEN GO TO STEP 5;
DECLARE QUANTITY (TIME4) = 3 DAY JTOY;
DECLARE QUANTITY (TIME3) = GMT;
```

JTOY, CDT and MET are converted to seconds. GMT and time intervals are converted to milliseconds.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. The maximum/minimum time values that may be specified are as follows:

	MAXIMUM	MINIMUM
Greenwich Mean Time:	24 hrs 0 min 0 sec 0 ms	0 hrs 0 min 0 sec 0 ms
Count Down Time:	+99 days 23 hrs 59 min 59 sec	-99 days 23 hrs 59 min 59 sec
Interval Time:	24 hrs 0 min 0 sec 0 ms	-23 hrs 59 min 59 sec 999 ms
Julian Time of Year:	396 days 23 hrs 59 min 59 sec	1 day 0 hrs 0 min 0 sec
Mission Elapsed Time:	99 days 23 hrs 59 min 59 sec	-99 days 23 hrs 59 min 59 sec

B. Recorded output of time values does not follow this format. Refer to RECORD Statement.

C. The Internal Name form of a Time Value may not be used as an initial value in declaration statements. Also, if Time Value is specified as an Internal Name, the name must be declared as Quantity data with the Dimension of time.

16.4 FUNCTION DESIGNATORS

THIS PAGE INTENTIONALLY LEFT BLANK.

The following elements will be discussed in this section:

EXTERNAL DESIGNATOR
FUNCTION DESIGNATOR
ROW DESIGNATOR

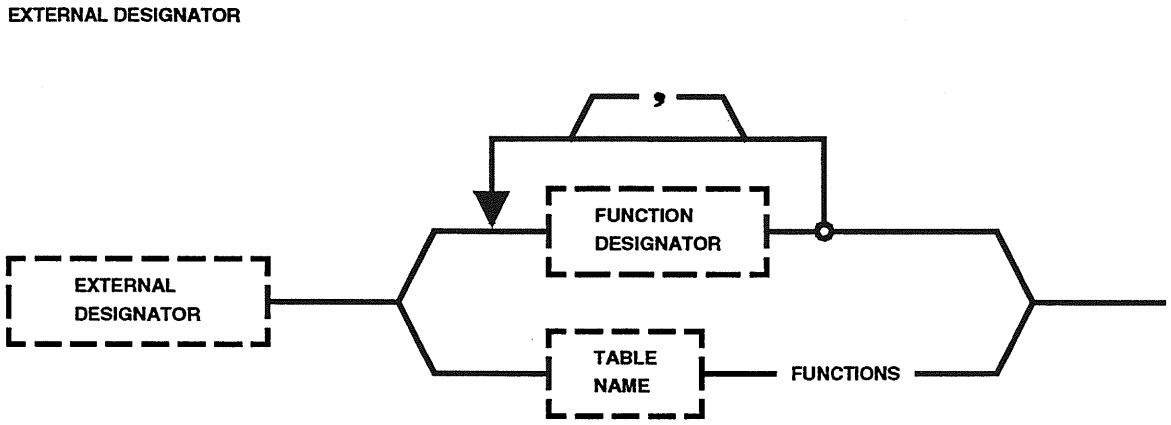


Figure 16-14 External Designator

Function

The External Designator Element enables reference to one or more Function Designators.

Description

The External Designator may be a single Function Designator or a list of Function Designators which are optionally separated by commas.

An External Designator may be in a form of a Table Name followed by the keyword FUNCTIONS to indicate the list of Row Designators specified.

When the TABLE FUNCTIONS option is used, the Row Designators are processed as a group. Each Row Designator of a table is associated with an inhibit indicator which controls its processing during execution. If a Row Designator is inhibited, it is skipped over in any operation involving a reference to Table Functions. The inhibit indicator is controlled by the ACTIVATE and INHIBIT Statements.

Examples

```
APPLY 5 V to < AS1 >
             < AS2 >
             < AS3 > ;
APPLY 5 V TO (QTABLE1) FUNCTIONS;
```

Restrictions/Limitations

Refer to the following for restrictions and limitations:

When a units mismatch occurs while using the TABLE FUNCTIONS option, a compiler warning message will be output.

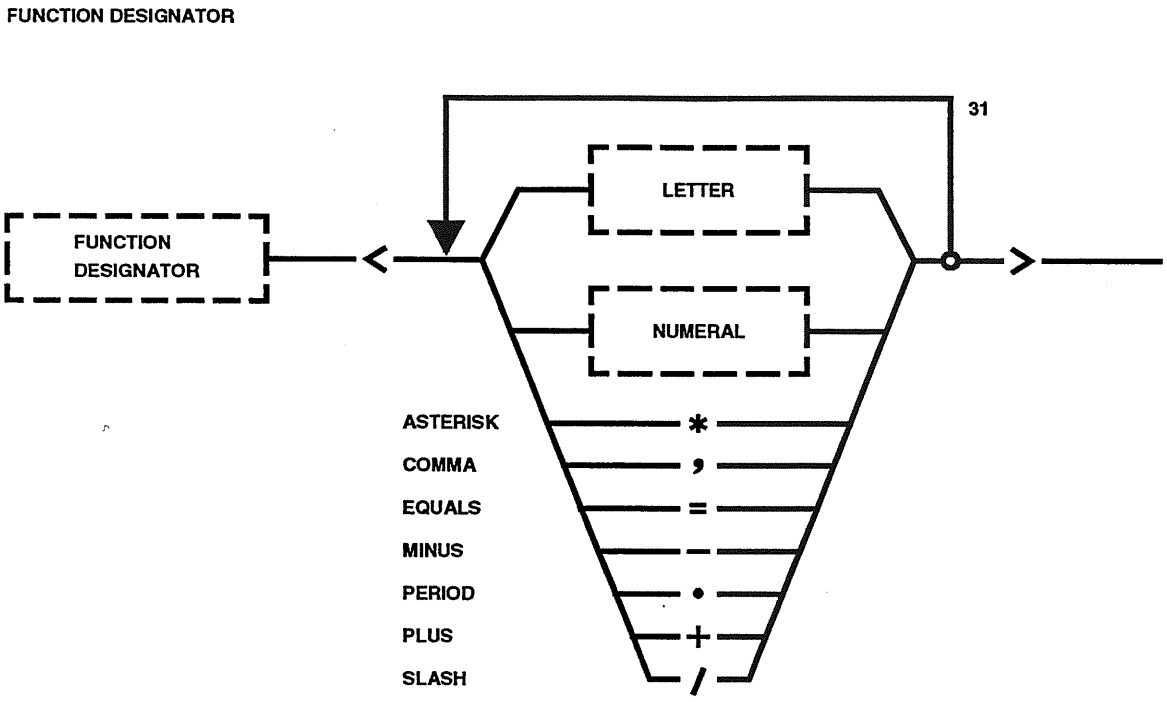


Figure 16-15 Function Designator

Function

The Function Designator element identifies by name items which are external to a GOAL procedure.

Description

Function Designators are items which interface via the Data Bank with the system under test or checkout system software.

A Function Designator must be delimited by angle brackets and may consist of any combination of Letters, Numerals or a Symbol subset. Blanks and commas are allowed within the Function Designator and are not significant.

A Function Designator is normally defined in a data bank; however, it may be defined in a GOAL procedure when it is used as a Pseudo Parameter of a program. Refer to the DEFINE Statement for additional information.

A Function Designator identifies by name a specific item which is external to a GOAL procedure. Function Designators are used to access measurements and commands associated with the system under test. They are also used to control peripheral devices and timers which are part of the checkout system.

Function Designators are normally defined in the Data Bank. They are identified in the Data Bank by their unique names. For each Function Designator name, the Data Bank contains characteristic information which is used to validate its usage in GOAL statement. The Data Bank also associates the Function Designator with a test configuration in which specific address and routing data link it to the external item which it identifies.

In the Function Designator cross-reference listing, Function Designators whose values will be altered during execution are flagged. An asterisk (*) is printed after the Function Designator name and after each Internal Sequence Number at which the value of the Function Designator is altered.

Function Designators are classified according to type. Many different types of Function Designators have been defined. The statement descriptions identify the types which are legal for particular GOAL statements.

Following is a list of Function Designator types available for use in the GOAL Language. The descriptions are intended to provide a

general understanding of the role of each Function Designator type. Specific details concerning operation and use are not provided in this document.

A. MEASUREMENTS AND STIMULI

1. LOAD ANALOG

This type of Function Designator identifies an analog stimulus to the system under test. It may take on a range of values in accordance with the characteristics of the end item which receives the stimulus. Quantity data from the GOAL procedure is converted to raw data stimulus format according to scaling factors and engineering units defined in the Data Bank. The APPLY Statement may be used to issue quantity type data to a Load Analog Function Designator. This data may be accessed by the READ Statement and the Verify Prefix.

2. LOAD DIGITAL PATTERN

This type of Function Designator identifies a digital pattern stimulus to the system under test. It may take on values which may be considered to be a string of binary bits (ones and zeroes). Numeric type data from the GOAL procedure is shifted and masked according to criteria defined in the Data Bank. The resulting binary bits are issued in parallel to the end item as a digital pattern stimulus. The ISSUE Statement may be used to issue numeric type data to a Load Digital Pattern Function Designator. This data may be accessed by the READ Statement and the Verify Prefix.

3. LOAD DISCRETE

This type of Function Designator identifies a discrete stimulus to the system under test. It may take on either of two values whose meaning depends upon the characteristics and use of the discrete which receives the stimulus. State type data, (ON/OFF, OPEN/CLOSED, etc.), from the GOAL procedure is converted to raw data command format (bit) in accordance with conversion criteria defined in the Data Bank. The SET Statement may be used to issue state type data to a Load Discrete Function

Designator. This data may be accessed by the READ Statement and the Verify Prefix.

4. PSEUDO

This type of Function Designator identifies a common communication word which is accessible by any GOAL procedure. Data may be passed from one GOAL procedure to another via Pseudo Function Designators. Pseudo Function Designators may be defined in any one of the following data types: analog, discrete or digital pattern. They may be referenced by the ISSUE, SET, APPLY, READ, RECORD Statements or Verify Prefix. They may not be referenced by the INTERRUPT Statements.

5. SENSOR ANALOG

This type of Function Designator identifies an analog measurement in the system under test. It may take on a range of values in accordance with the characteristics of the sensor. Raw data from the sensor is converted to engineering values according to the scaling factors and engineering units defined in the Data Bank. This engineering value is available to the GOAL procedure for use in conjunction with quantity type data in GOAL Statements such as READ, AVERAGE, VERIFY, and RECORD.

6. SENSOR DIGITAL PATTERN

This type of Function Designator identifies a digital pattern measurement in the system under test. It may take on values which may be considered to be a string of binary bits (ones and zeroes). Raw data from the sensor (or data stream) is shifted and masked according to criteria defined in the Data Bank. The resulting digital pattern is available to the GOAL procedure for use in conjunction with numeric type data in GOAL Statements such as READ, RECORD and VERIFY.

7. SENSOR DISCRETE

This type of Function Designator identifies a discrete measurement in the system under test. It may take on either of two values whose meaning

depends upon the characteristics and use of the sensor. The raw data (bit) from the sensor is converted to an engineering value, (ON/OFF, WET/DRY, etc.), in accordance with conversion criteria defined in the Data Bank. This engineering value is available for use in conjunction with state type data in GOAL statements such as READ, RECORD and VERIFY.

B. SYSTEM FUNCTION DESIGNATORS

1. ALARM

This type of Function Designator identifies the audible alarms on the CCMS consoles. It may be used with the SET DISCRETE Statement. The alarms may be set to ON and OFF.

2. CDT

This type of Function Designator identifies the Count Down Time which is maintained by CCMS. It may be used with Quantity Data which has the engineering units of CDT time. A CDT Function Designator is provided to allow control of Count Down Time and to allow an interrupt to be taken at a specified CDT. CDT is controlled using the TIMER CONTROL Statement. The WHEN INTERRUPT Statement may then be used to generate a response when the Count Down Time becomes equal to a specified value.

3. CONSOLE

This type of Function Designator identifies a CCMS console which contains GOAL procedures that are being executed. It is used to reference specific consoles with the SEND, PERFORM and CONCURRENT.

4. DATE

This type of Function Designator identifies a text string which contains the current Month-Day-Year information. It may be used with the READ and RECORD DATA Statements, and is compatible with text type data.

5. DISPLAY

This type of Function Designator identifies a Display Generator (DG) page or on-board display. DG pages may be used for manual interaction with a GOAL procedure during its execution in real time. The RECORD Statement may be used to send data from a GOAL procedure to the display page (or on-board CRT) for viewing by the operator. This statement also provides an option to request that data be entered by the operator through the keyboard associated with the specified ground display Function Designator.

6. DISPLAY FUNCTION KEY

This type of Function Designator is used to identify certain keys which are located on the keyboard of the online CRT terminal. These keys may be used with the INTERRUPT Statements to respond to operator inputs in real time.

7. FILE

This type of Function Designator identifies a Data File which may be used to store output information from a GOAL Program for subsequent processing. The data is stored as raw data (i.e., binary). It is normally used in the WRITE option of the RECORD Statement. Data Files in CCMS are located on the console moving head disk, PDR Tape Recorders, and CDS Tape Recorders.

8. GMT

This Function Designator identifies the current value of Greenwich Mean Time. It may be used with statements such as READ and VERIFY. It is compatible with quantity type data which has the engineering units of GMT time.

9. PFP FUNCTION KEY

This type of Function Designator identifies a push button which is located adjacent to a LED halfline. It may be used with the INTERRUPT Statements to respond to operator inputs in real time. In this case, the LED would normally be used to display

information related to current use of the Function Key.

10. PFP LIGHT

This Function Designator type identifies an indicator light which is located in the PFP Function Key. Each Function Key has two indicator lights, which are normally used to indicate ARM and EXECUTE actions. These Function Designators may be used with READ, SET, and VERIFY. They may take on values of ON and OFF and may be used in conjunction with state type internal data.

11. FPG1

This type of Function Designator identifies a text string which contains a "fixed Page 1 location in the Common Data Buffer." It may be used with the READ and RECORD DATA statements, and is compatible with text type data.

12. PFP LED MATRIX

This type of Function Designator identifies a string of 17 characters which are located on a Programmable Function Panel Presentation. Each presentation contains six strings which are normally referred to as half lines.

The LED type Function Designator may be used with the Record Statement to display up to 17 characters of text for viewing by the operator. It may also be used with the CLEAR DISPLAY and MODIFY DISPLAY Statements.

13. PRINTER

This type of Function Designator identifies an online device which is able to print text information from a GOAL Program. It is normally used in the Print Option of the RECORD Statement. Printers in CCMS include the Console Printer/Plotter and SPA Line Printer.

14. RECORDER

This type Function Designator is used to record ASCII data to PDR and CDS tape recorders. It may be used with the PRINT option of the RECORD DATA Statement.

15. REMOTE COMMUNICATION

This type of Function Designator is used by GOAL procedures to communicate with each other. It is used in the SEND INTERRUPT Statement to notify other GOAL procedures of a special condition and used in the INTERRUPT Statements to detect the interrupt condition sent by another procedure.

16. SYSTEM INTERRUPT

This type of Function Designator is used to identify special conditions which may be detected by the system. It may be used with the INTERRUPT Statements to respond to these conditions. For instance, a CPU which goes offline in real time could cause a SYSTEM INTERRUPT in a GOAL procedure. System Interrupt Function Designators include the following:

(A) Measurement Validity Changes

(B) CPU Status Changes

(C) Class III Errors

(D) Interrupt Event Occurrence in a Higher Level

(E) CDT Counting

(F) CDT Holding

(G) MET Counting

(H) MET Holding

17. TIMER

This type of Function Designator identifies a resetable time which is supported in the checkout system. The Timer may be set to a positive interval Time Value by using the TIMER CONTROL Statement.

The Timer automatically decrements until its value reaches zero. The WHEN INTERRUPT Statement may be used to react to the timeout condition. The Timer may be used in conjunction with interval time quantity data in GOAL statements such as READ and VERIFY. The accuracy of the Timer is determined by its count rate, which for CCMS is one count per second.

18. MET

This type of Function Designator identifies the Mission Elapsed Time which is maintained by the CITE System. It may be used with Quantity Data which has engineering units of MET time. A MET Function Designator is provided to allow control of mission elapsed time and to allow an interrupt to be taken at a specified MET. MET is controlled using the TIMER CONTROL Statement. The WHEN INTERRUPT Statement may then be used to generate a response when the Mission Elapsed Time becomes equal to a specified value.

19. FEP

This type of Function Designator identifies the various CCMS Front End Processors (FEP's). It is used to reference specific FEP's on the ACTIVATE/ INHIBIT SYSTEM and CHANGE Statements.

20. CURVE

This type of Function Designator identifies the linearization curve name for non-linear data. It is used in the CHANGE Statement for reassigning or assigning linearization data to specific Function Designators that are non-linear.

ROW DESIGNATOR

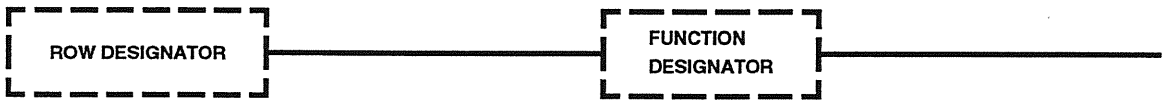


Figure 16-16 Row Designator

Function

The Row Designator element uniquely identifies a row of a table and allows reference to the row by a symbolic name.

Description

A Row Designator follows the standard format of Function Designator.

A Row Designator is defined in a DECLARE TABLE Statement.

A Row Designator may be used in forming an Internal Name.

A Row Designator must be unique within a table.

Example

```
< AM1 >

LET (QTABLE1) < AM1 > COLUMN 1 = (Q1);
```

THIS PAGE INTENTIONALLY LEFT BLANK.

16.5 INTERNAL NAME

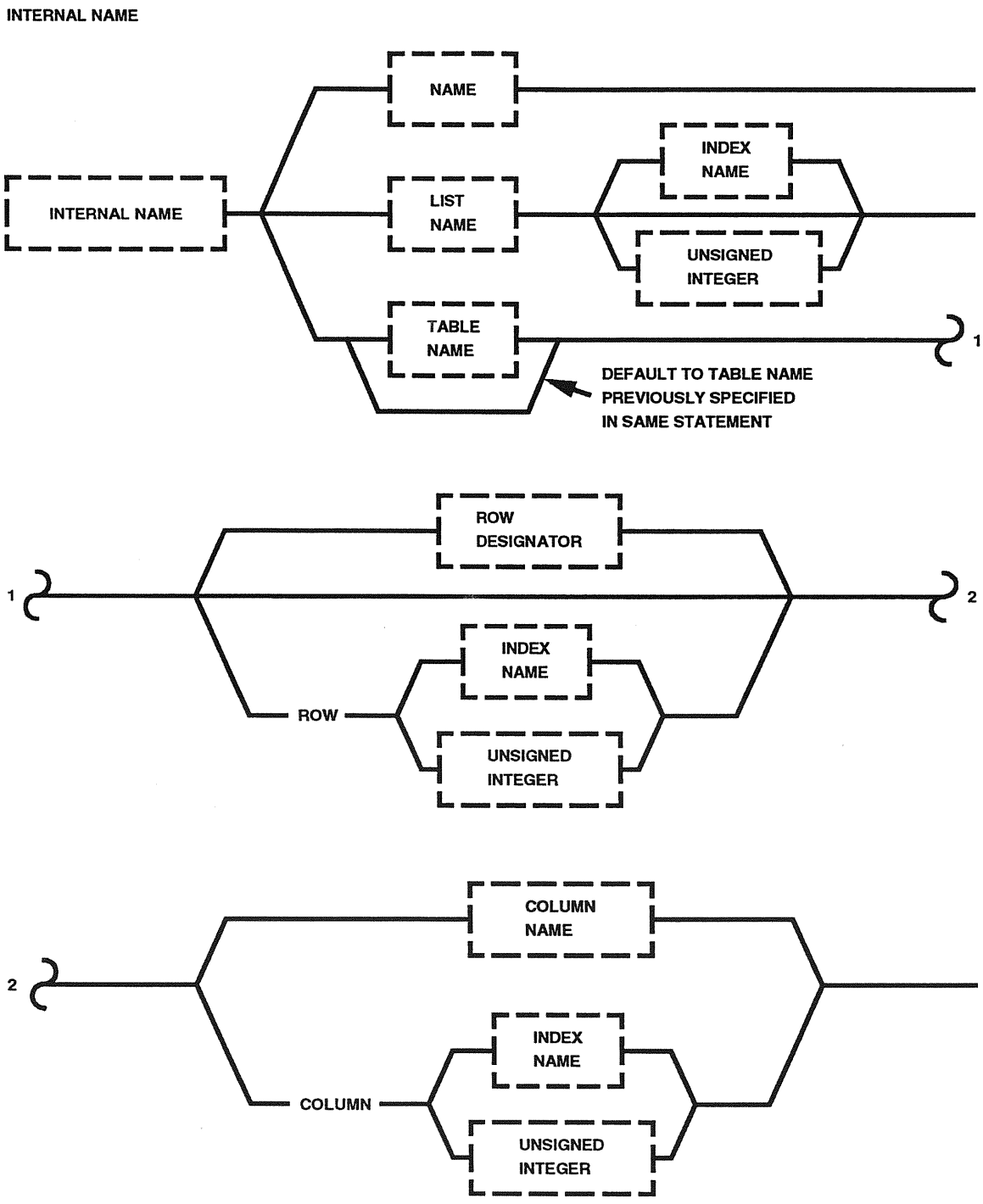


Figure 16-17 Internal Name

Function

The Internal Name element enables reference to internal data by its symbolic name.

Description

An Internal Name is used to reference data which has been previously declared in the GOAL procedure. An Internal Name may represent the following:

A. A NAME - a single data item

B. A LIST or LIST element

C. A TABLE element or TABLE column.

A list element may be specified by means of a List Name followed by an Unsigned Integer or by means of a List Name followed by an Index Name. The latter option (LIST NAME and INDEX NAME) provides the capability of a variable index into a list, which is defined at execution time.

An element of a table may be specified in the same manner as an element of a List; i.e., by specifying ROW and COLUMN with Index Name or Unsigned Integer. In addition, a row number of a table may optionally be specified by giving the Row Designator assigned to the row in the Table Declaration Statement. A column number of a table may optionally be specified by giving the Column Name assigned to the column in the Table Declaration Statement.

An entire column of a table may be selected for use as a list of data items by specifying the Table Name and the desired column. Once a Table Name is specified in a GOAL Statement, it need not be given again unless a different table is to be referenced.

In the Internal Name cross-reference listing, Internal Names whose values will be altered during execution are flagged. An asterisk (*) is printed after the Internal Name and after each Internal Sequence Number at which the value of the Internal Name is altered.

NAME Option

Examples

```
(N3), (Q1), (STATE5), (B101), (T377), (X33FF)
LET (Q5) = (Q1) + (Q2);
```

LIST NAME Option

Examples

```
(NLIST1), (QLIST1), (SLIST1), (TLIST1)
(NLIST1)2 (QLIST1)3 (SLIST1)2 (TLIST1)3

LET (NLIST1) = (NLIST2)5;
```

The first example illustrates List Names. The second example illustrates list elements. The third example illustrates the use of a list and a list element in the LET EQUAL Statement.

TABLE NAME - ROW COLUMN Option

Examples

```
(NTABLE1) ROW 1 COLUMN 2
(QTABLE2) ROW (N1) COLUMN (N2)

 LET (QTABLE2) ROW (N1) COLUMN (N2) = (QTABLE2) ROW 1 COLUMN 2;
```

The first example specifies row 1, column 2 of (NTABLE1). The second example specifies the element of (QTABLE2) located at the row defined by variable (N1) and the column defined by variable (N2). The third example illustrates the use of a table element in the LET EQUAL Statement.

TABLE NAME - ROW DESIGNATOR Option

Examples

```
(STABLE2) < DM3 > COLUMN 1
(QTABLE1) < AS2 > COLUMN (N1)

LET (Q5) = (QTABLE1) < AS2 > COLUMN 1;
```

Table Name - Column Name and Table Column Options

Examples

```
(QTABLE1) (COLUMN A)
(QTABLE1) COLUMN 1

LET (QTABLE2) COLUMN 1 = (Q5);
```

TABLE NAME - IMPLICIT TABLE REFERENCE Option

Examples

```
RECORD (QTABLE1) ROW 1 COLUMN 2, ROW 2 COLUMN 2 TO < PAGE-B >;
VERIFY (QTABLE1) FUNCTIONS ARE BETWEEN COLUMN 1 AND COLUMN 2;
```

Once a Table Name is specified in a GOAL Statement, it need not be repeated until a different table is referenced. The previous examples illustrate this option.

Error Processing

If the value of the Index Name used to specify a list or table element causes addressing beyond the limits of the list or table, a Class III error will result.

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. An Internal Name is made up of Letters and Numerals only. The first character must be a Letter.

B. Blanks and Commas are insignificant.

THIS PAGE INTENTIONALLY LEFT BLANK.

16.6 NAMES

THIS PAGE INTENTIONALLY LEFT BLANK.

The GOAL elements which refer to the element Name are discussed in this section. First Name is discussed and then the references to it are discussed in alphabetical order. The following elements will be discussed in this section:

NAME

COLUMN NAME

CURSOR NAME

DISK FILE NAME

INDEX NAME

LIST NAME

MACRO NAME

PROGRAM NAME

SKELETON NAME

SUBROUTINE NAME

SYSTEM AREA NAME

TABLE NAME

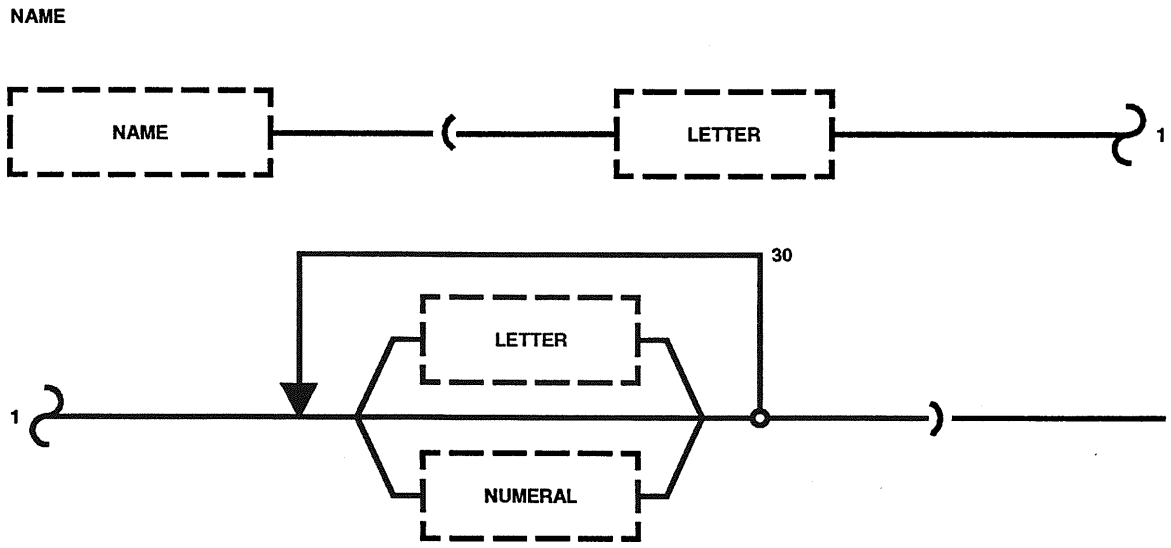


Figure 16-18 NAME

Function

The element Name enable symbolic reference to Internal Data, Programs, Table Columns, Index Variables, Display Skeletons, and Symbolic Cursor Positions and Macros.

Description

A Name must be delimited by parentheses.

A Name must begin with a Letter and may consist of any combination of up to 32 Letters and Numerals.

Blanks and Comments are allowed in a Name and are ignored by the Processor.

Except for Programs, Display Skeletons, and Symbolic Cursor Positions, a name is defined by a Declaration.

All names defined by a Declaration may be up to 32 characters.

COLUMN NAME

Figure 16-19 Column Name

Function

The Column Name element uniquely identifies a column of a table and allows reference to the column by a symbolic name.

Description

A Column Name follows the standard format of Name.

A Column Name is defined in a DECLARE TABLE Statement.

A Column Name may be used in forming an Internal Name.

The Column Name must be unique within a table.

A Column Name must not be the same as the name of a declared data item.

CURSOR NAME

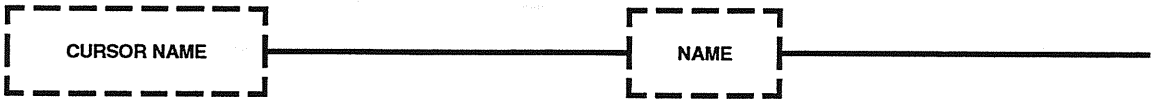


Figure 16-20 Cursor Name

Function

The Cursor Name element identifies a symbolic cursor coordinate position on a display skeleton and allows reference to the cursor position by specifying its symbolic name along with its related skeleton name. Cursor names are defined at Skeleton Build time according to specifications in the LPS CCMS System Generation User Guide KSC-LPS-OP-033-05.

Description

The Cursor Name follows the standard format of Name.

The Cursor Name is used in the SPECIFY INTERRUPT Statement and RECORD DATA Statement.

A Cursor Name may be up to ten characters in length.

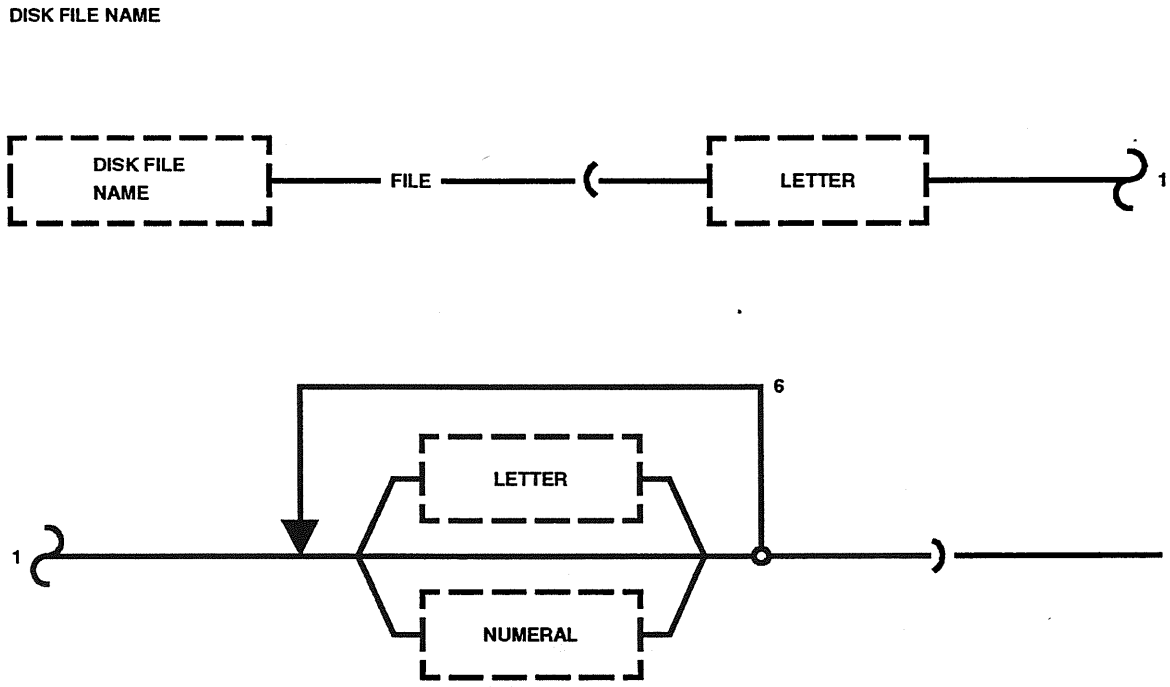


Figure 16-21 Disk File Name

Function

The Disk File Name element identifies a GOAL disk file uniquely, and allows reference to the disk file by its symbolic name.

Description

The Disk File Name follows the standard format of name.

INDEX NAME

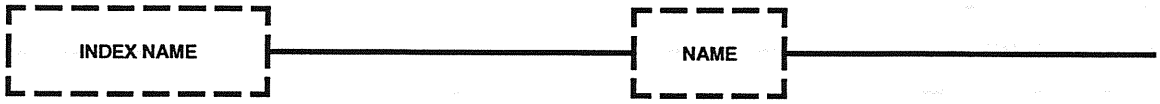


Figure 16-22 Index Name

Function

The Index Name element enables the use of a Name as a variable index.

Description

The Index Name follows the standard format of Name.

An Index Name must be defined as a single numeric data item in a DECLARE DATA Statement.

An Index Name may be used in forming an Internal Name.

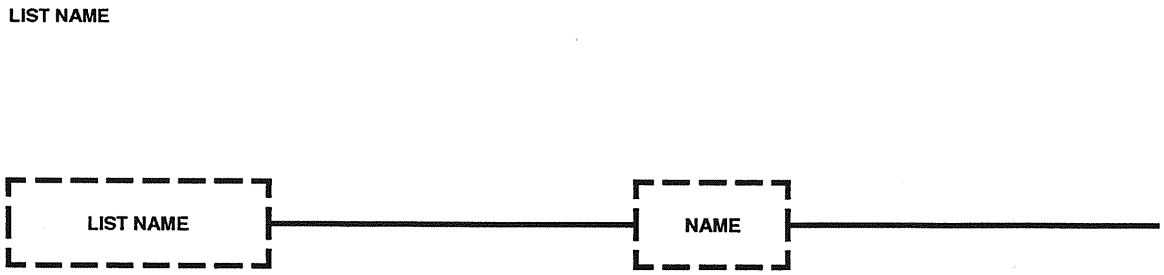


Figure 16-23 List Name

Function

The List Name element uniquely identifies a List and allows reference to the List by its symbolic name.

Description

The List Name follows the standard format of Name.

A List Name is defined in a DECLARE LIST Statement and may be used in GOAL statements where reference to internal data is permitted.

MACRO NAME

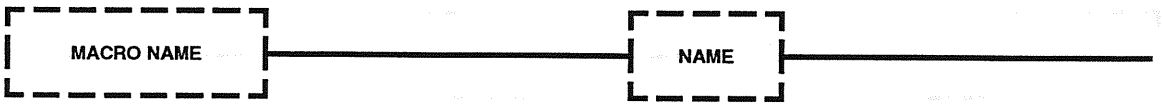


Figure 16-24 Macro Name

Function

The Macro Name element uniquely identifies a macro and allows reference to the macro by its symbolic name.

Description

The Macro Name follows the standard format of Name.

A Macro Name is used in the BEGIN MACRO Statement.

A Macro Name may be up to 32 characters in length.

A Macro Name that refers to a standalone macro in APLM can be a maximum of 8 characters in length.

PROGRAM NAME

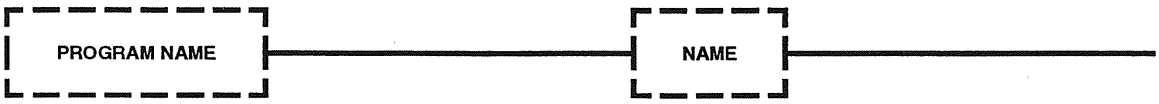


Figure 16-25 Program Name

Function

The Program Name element uniquely identifies a program and allows reference to the Program by its symbolic name.

Description

The Program Name follows the standard format of Name.

A Program Name is used in the BEGIN PROGRAM Statement, PERFORM PROGRAM Statement, and CONCURRENT Statement.

A Program Name may be up to eight characters in length.

SKELETON NAME

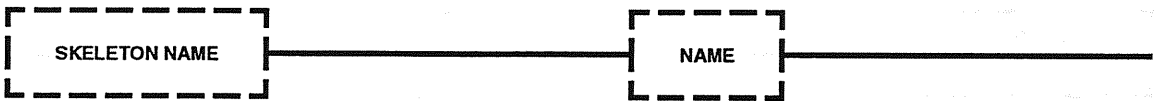


Figure 16-26 Skeleton Name

Function

The Skeleton Name element uniquely identifies a display skeleton and allows reference to the skeleton by its symbolic name.

Description

The Skeleton Name follows the standard format of Name.

A Skeleton Name is used in the DISPLAY SKELETON Statement, RECORD DATA Statement, and SPECIFY INTERRUPT Statement.

A Skeleton Name may be up to ten characters in length.

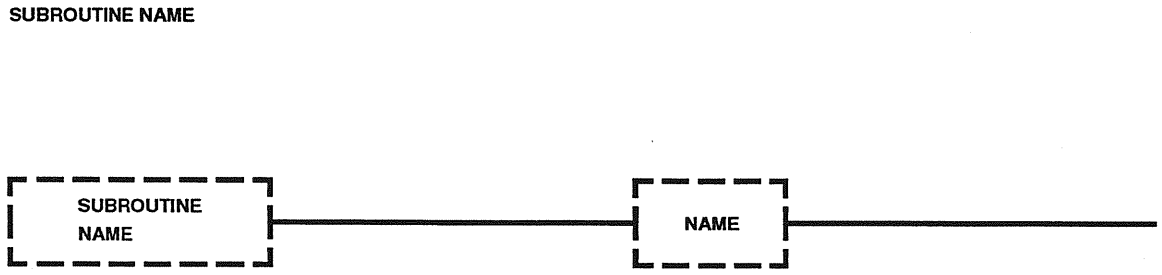


Figure 16-27 Subroutine Name

Function

The Subroutine Name element uniquely identifies a subroutine and allows reference to the Subroutine by its symbolic name.

Description

The Subroutine Name follows the standard format of Name.

A Subroutine Name is used in the BEGIN SUBROUTINE Statement, PERFORM SUBROUTINE Statement, INCLUDE SUBROUTINE Statement and LOCK SUBROUTINE Statement.

A Subroutine Name may be up to 8 characters in length.

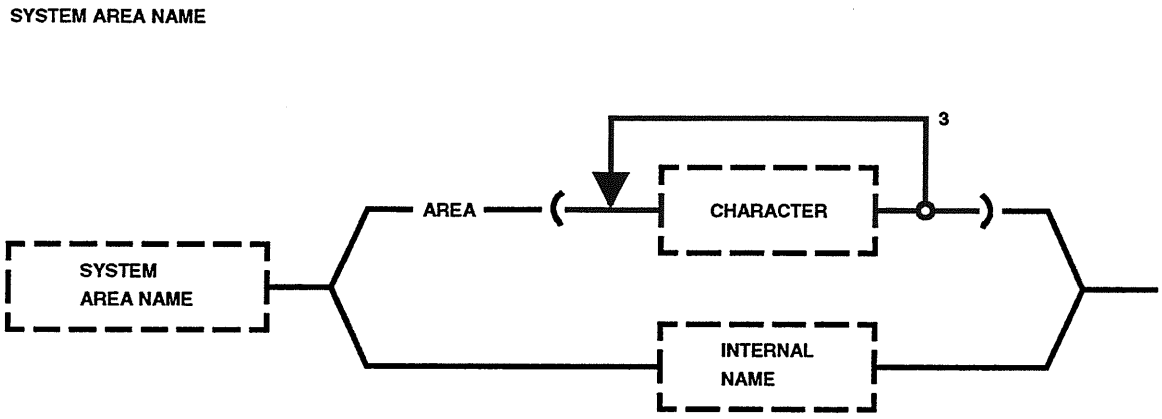


Figure 16-28 System Area Name

Function

The System Area Name element identifies an area in console bulk memory which is to be used for disk file data manipulation. It is not required to be defined as a responsible system name in the CCMS Data Bank.

Description

Legal Internal Name type is text.

TABLE NAME

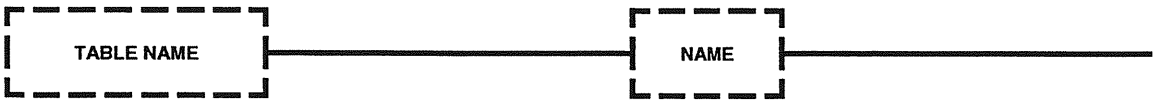


Figure 16-29 Table Name

Function

The Table Name element uniquely identifies a Table and allows reference to the Table by its symbolic name.

Description

The Table Name follows the standard format of Name.

A Table Name is defined in a DECLARE TABLE Statement and may be used in GOAL Statements where reference to internal data is permitted.

16.7 NUMBER REPRESENTATION

THIS PAGE INTENTIONALLY LEFT BLANK.

The elements for Number Representation will be discussed and are presented in alphabetical order:

BINARY NUMBER

DECIMAL NUMBER

HEXADECIMAL NUMBER

INTEGER NUMBER

NUMBER PATTERN

OCTAL NUMBER

UNSIGNED INTEGER

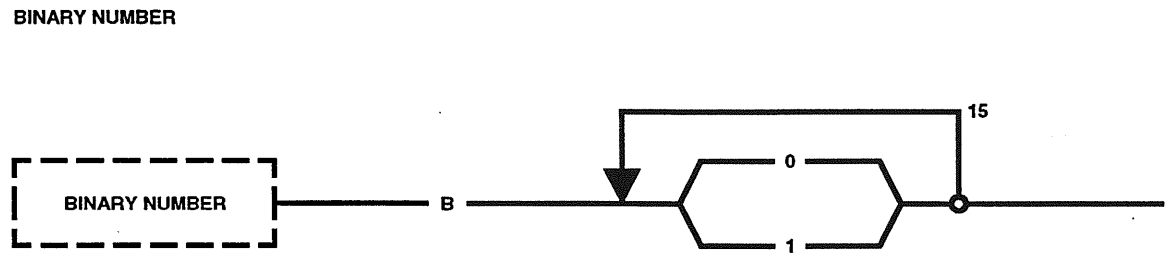


Figure 16-30 Binary Number

Function

The Binary Number element specifies the value of a Digital Pattern data item in binary format.

Description

A Binary Number must begin with the letter "B" and followed by any combination of Numerals 1 and 0.

No more than 16 numerals may be specified.

If less than 16 are specified, leading zeroes are assumed.

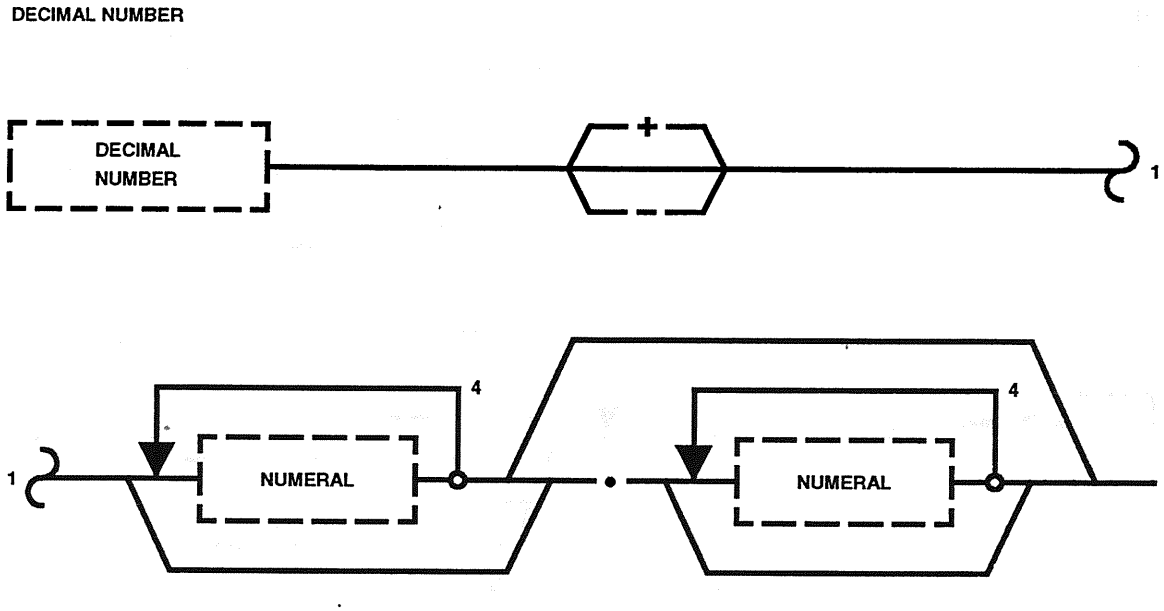


Figure 16-31 Decimal Number

Function

The Decimal Number element specifies the value of a Quantity data item in conventional Decimal format.

Description

A Decimal Number may be positive or negative.

A Decimal Number may contain any combination of the numerals 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

A Decimal Number may contain a fractional portion which is preceded by a decimal point.

The use of leading and trailing zeroes is optional.

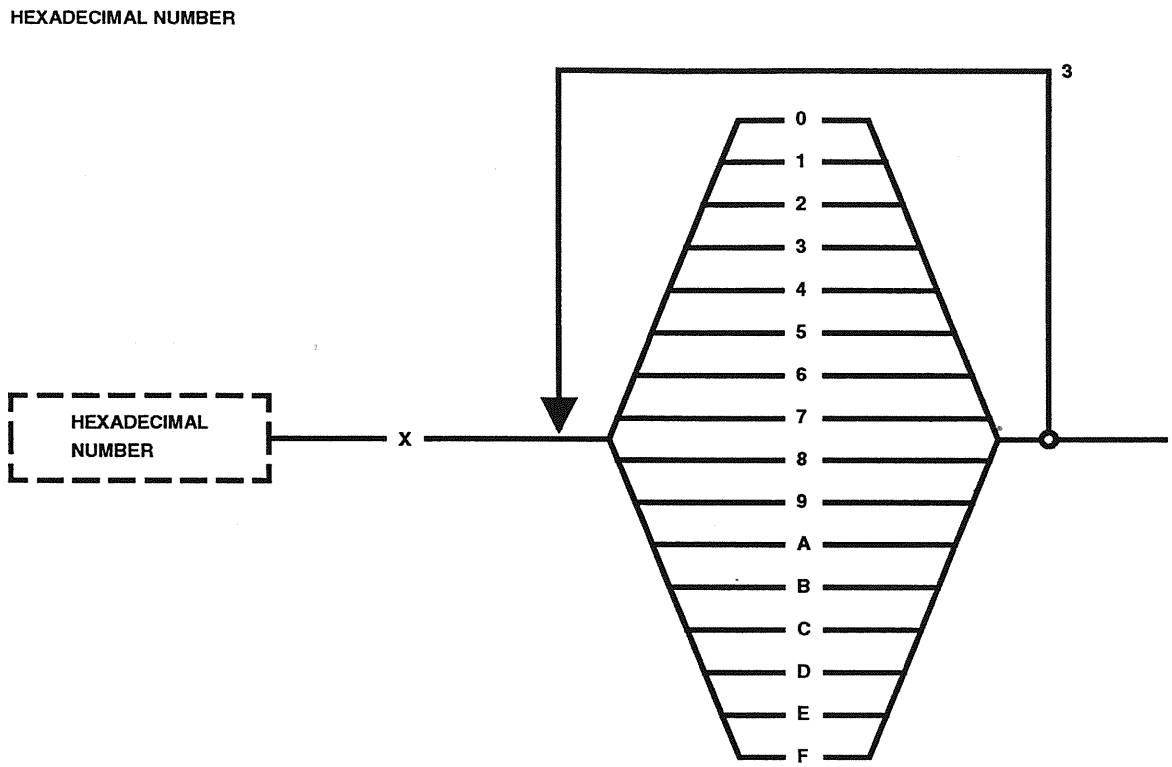


Figure 16-32 Hexadecimal Number

Function

The Hexadecimal Number element specifies the value of a Number Pattern data item in Hexadecimal format.

Description

A Hexadecimal Number must begin with the letter X followed by any combination of the Numerals 0 through 9 and the Letters A, B, C, D, E or F.

Hexadecimal digits are right justified in a data item.

No more than four Hexadecimal digits may be specified.

INTEGER NUMBER

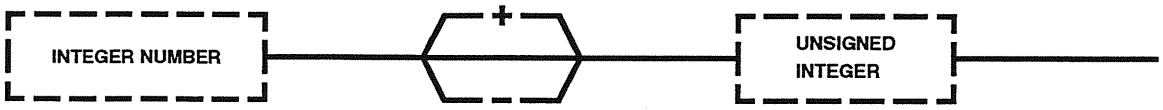


Figure 16-33 Integer Number

Function

The Integer Number element specifies the value of a Numeric data item.

Description

An Integer Number may be positive or negative.

If no sign is provided, the value is assumed to be positive.

An Integer Number may contain any combination of the numerals 0 through 9.

The Integer Number can be a maximum value of +32767 and a minimum value of -32768, except in the LET EQUAL Statement where the minimum value is -32767.

No more than five digits may be specified for an Integer Number.

The use of leading zeroes is optional.

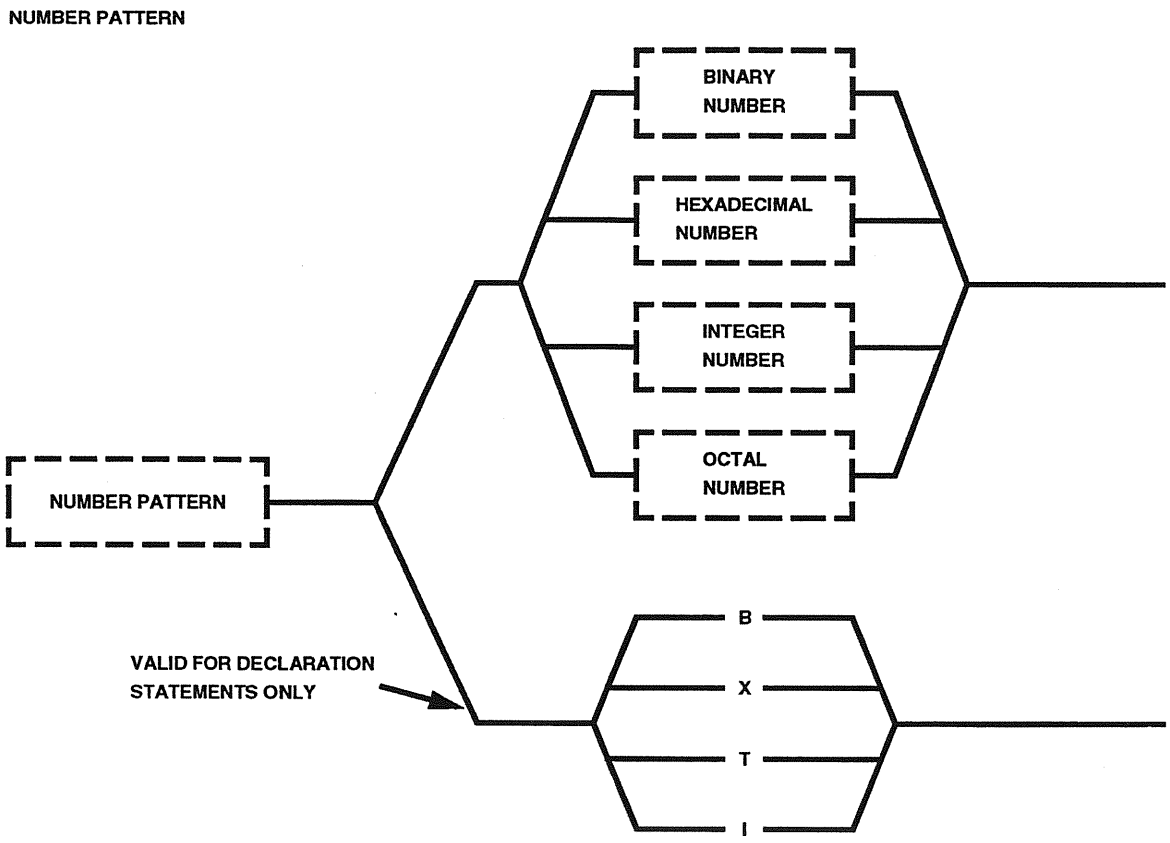


Figure 16-34 Number Pattern

Function

The Number Pattern element specifies the value of a numeric type data item in any of the permissible formats.

Description

The Number Pattern element refers to the following types of numbers:

A. Binary

B. Octal

C. Hexadecimal

D. Integer

In a numeric declaration statement, the letters B, X, T and I may be used to specify binary, hexadecimal, octal, and integer numbers. When B, X, T and I are used, the initial value at execution time is zero. If the number being declared is a Pseudo Parameter, B, X, T and I must be used.

Examples

```
B1101     X 3ABF       T 377        32760

DECLARE NUMBER (B101) = B 101,
               (XNUM) = X FFFF,
               (TNUM) = T 377,
               (N9)   = 9;
DECLARE NUMBER (B101) = B,
               (XNUM) = X,
               (TNUM) = T,
               (N9)   = I;
```

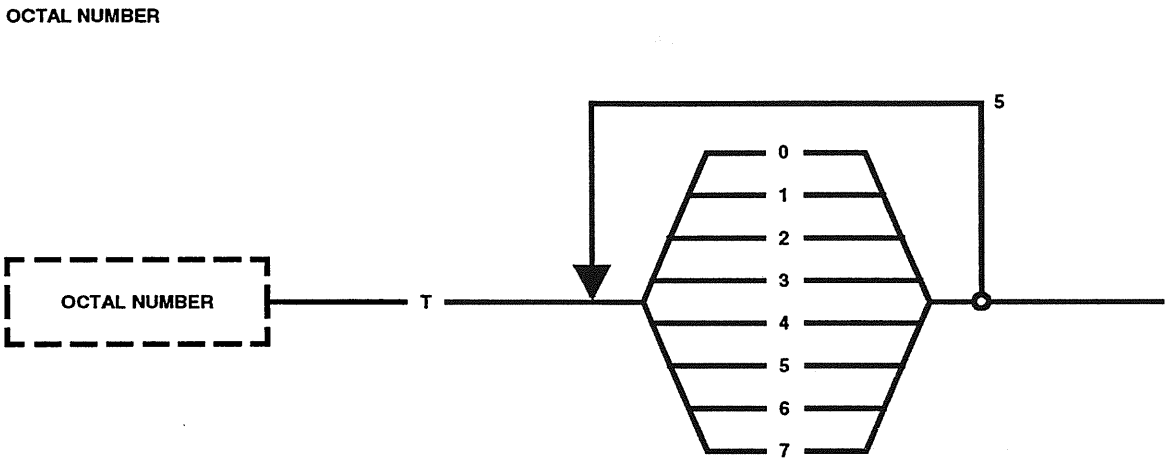


Figure 16-35 Octal Number

Function

The Octal Number element specifies the value of a Number Pattern data item in Octal format.

Description

An Octal Number must begin with the letter T followed by any combination of the Numerals 0 through 7.

An Octal Number may not exceed the value 177777.

No more than six Octal digits may be specified.

Octal digits are right justified in a data item.

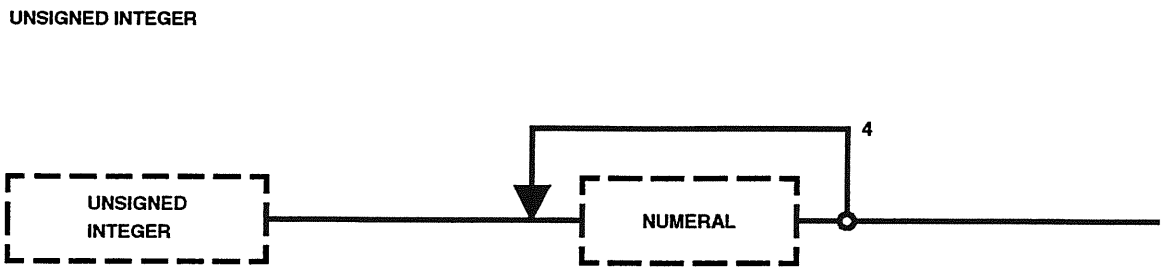


Figure 16-36 Unsigned Integer

Function

The Unsigned Integer element specifies the value of an unsigned numeric constant in an Integer format.

Description

An Unsigned Integer may contain any combination of the Numerals 0 through 9.

The Unsigned Integer can be a maximum value of 32767 and a minimum value of 0.

No more than five digits may be specified for an Unsigned Integer.

The use of leading zeroes is optional.

THIS PAGE INTENTIONALLY LEFT BLANK.

16.8 PROCEDURAL STATEMENT PREFIXES

THIS PAGE INTENTIONALLY LEFT BLANK.

The Procedural Statement Prefix, including the following elements, will be discussed in this section:

PROCEDURAL STATEMENT PREFIX

STEP NUMBER

TIME PREFIX

The Verify Prefix and the Statement Group Label are also Procedural Statement Prefixes; but they will not be discussed in this section, since they were presented with the Branching Statements.

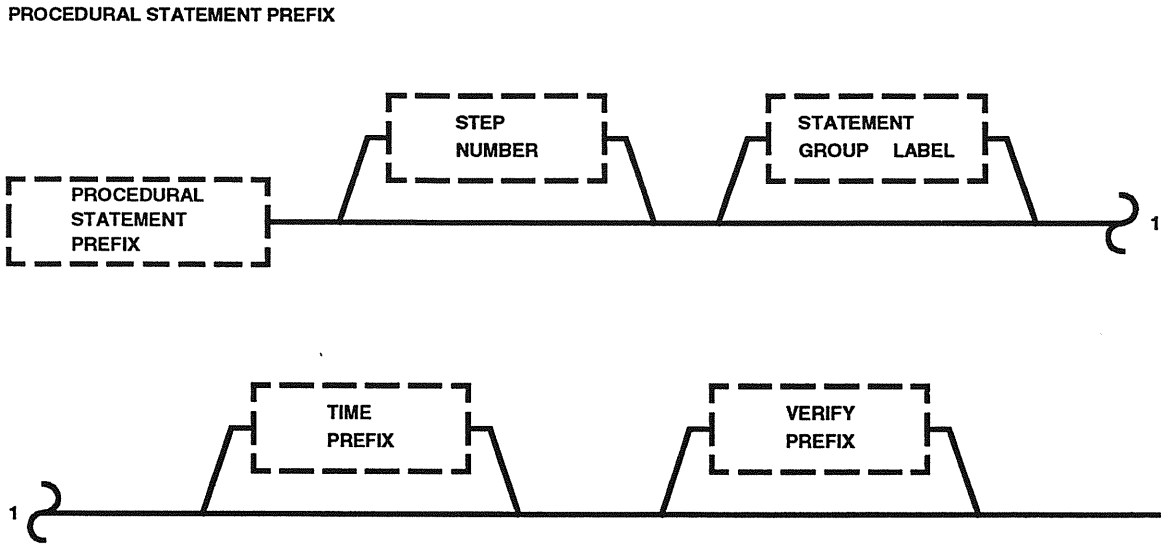


Figure 16-37 Procedural Statement Prefix

Function

The Procedural Statement Prefix element allows a Procedural Statement to be referenced by a symbolic label, to be executed dependent upon a time condition, and/or to be executed dependent upon an event occurrence.

Description

A Procedural Statement may be preceded by a combination of the following: Step Number, Time Prefix, Verify Prefix, Statement Group Label.

If more than one prefix is specified, the prefixes must be ordered as follows: Step Number, Statement Group Label, Time and Verify Prefixes.

A Step Number and a Statement Group Label cannot appear more than once per GOAL Procedural Statement.

Example

```
STEP 5 AFTER < CDT > IS -20 MIN CDT,
    VERIFY < DM1 > IS ON THEN
    GO TO STEP 7;
```

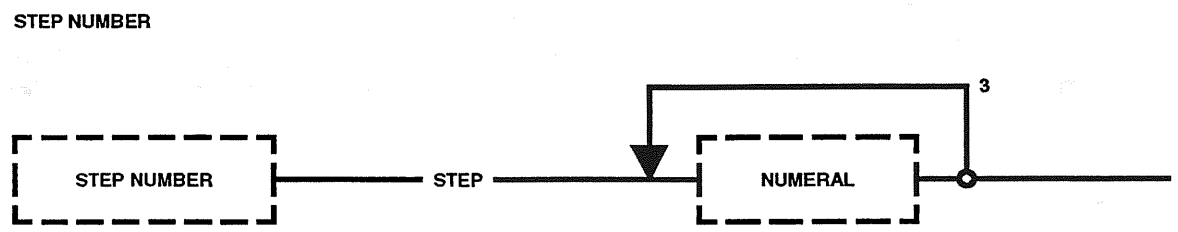


Figure 16-38 Step Number

Function

The Step Number element provides means of labeling or referencing a GOAL Procedural Statement.

Description

Step Number is specified by the word STEP followed by up to four numerals. At least one of the numerals must be non-zero.

Leading zeroes are not considered as part of the Step Number.

A Step Number may be the first item of a Procedural Statement Prefix.

Step Numbers are not required to be in any particular order.

Step Numbers defined in a GOAL procedure must be unique.

All Step Numbers must be defined in the procedure in which they are referenced.

Example

```
STEP 9999 GO TO STEP 125;
```

THIS PAGE INTENTIONALLY LEFT BLANK.

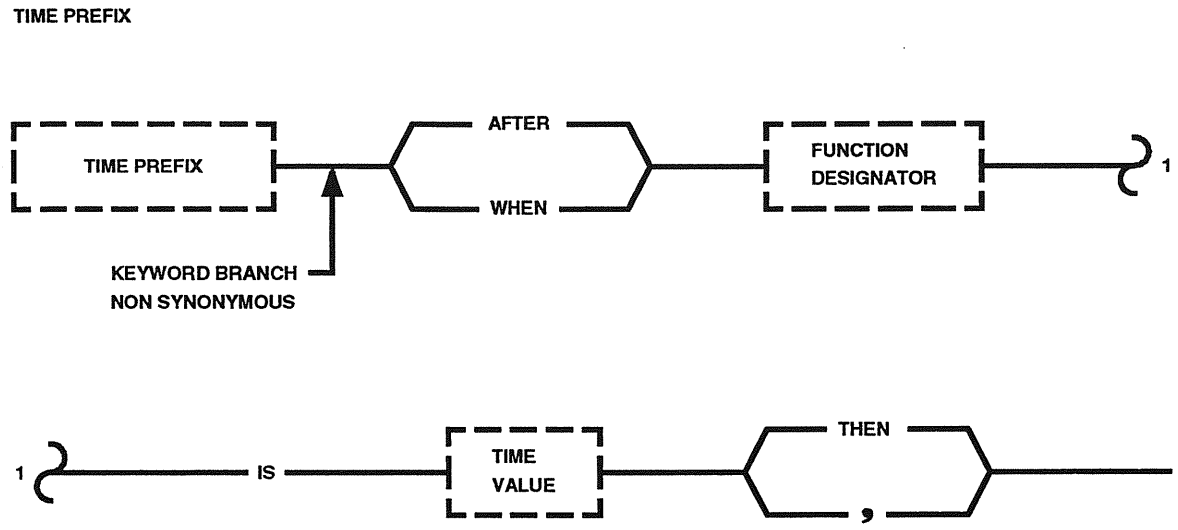


Figure 16-39 Time Prefix

Function

The Time Prefix element delays the execution of a GOAL Program until a specified time.

Description

The Time Prefix is optional for procedural statements.

The Time Prefix may only be used once per statement. Multiple occurrences of the Time Prefix are allowed by nesting inside the BEGIN/END sequence.

The Function Designator may indicate either GMT, CDT, or MET.

The keyword AFTER indicates that the delay is to be in effect up to and including the specified time value.

The keyword WHEN indicates that the delay is to be in effect only up to the specified Time Value. When this time occurs, execution will resume.

The keyword THEN and the comma are synonymous.

Examples

```
AFTER < CDT > IS -20 MIN CDT, SET < DS1 > TO ON;
AFTER < GMT > IS (GMT2) THEN GO TO STEP 105;
WHEN < CDT > IS -2 HR CDT, GO TO STEP 99;
WHEN < METIME > IS -1 HR MET, GO TO STEP 100;
```

The keywords AFTER and WHEN are processed synonymously when referencing GMT according to the WHEN option.

The keywords AFTER and WHEN are processed differently when referencing CDT. The WHEN option results in processing coincident with the time specified, whereas, for AFTER option processing is not performed until 1 second after the specified time. Thus, if CDT is halted at the specified time, the WHEN option initiates processing and the AFTER option does not. Upon resumption of countdown, the AFTER option would allow processing to occur at the first 1 second count.

Legal Function Designators

MET, CDT, GMT
JTOY is not allowed.

THIS PAGE INTENTIONALLY LEFT BLANK.

16.9 TEXT CONSTANT

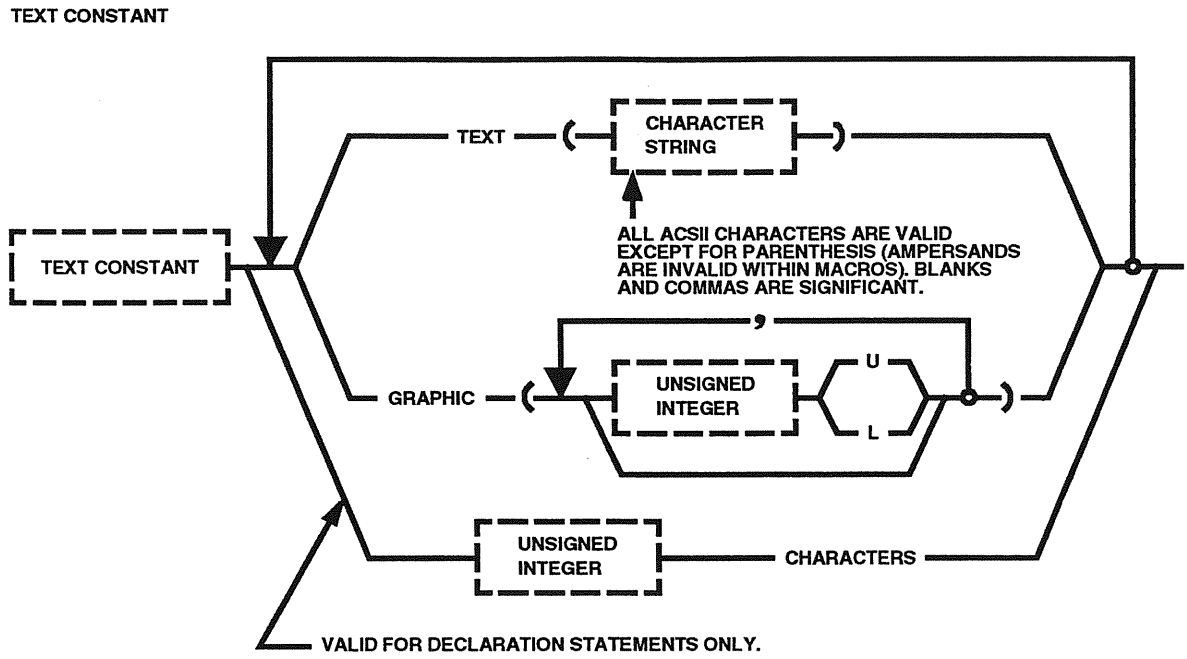


Figure 16-40 Text Constant

Function

The Text Constant element specifies the value of a text type data item.

Description

Text data may be in TEXT format or GRAPHIC format. In the TEXT format, the characters are processed using their standard ASCII codes. In the GRAPHIC format, the character names are converted to special codes which represent the Graphic characters of the on-line display terminal. For the TEXT option, all ASCII characters are valid except parenthesis and ampersand (&).

The GRAPHIC characters are named by their key cap numbers as shown on the chart of Graphic Function Keys in the Appendix. Successive GRAPHIC character names must be separated by commas. Two consecutive commas are used to indicate a blank. The bypass option should only be used for this purpose.

When an initial text value is not provided on a Text Declaration Statement, a character count may be used to specify the length of the Text Constant. This form must be used if the value being declared is a Pseudo Parameter.

Examples

```
TEXT(ABC  1 2 3 * , / . )
GRAPHIC(44L, , , , ,44U)

DECLARE TEXT (TEXT1) = TEXT(ABCD) GRAPHIC(25U);
DECLARE TEXT (TEXT2) = 25 CHARACTERS;
RECORD TEXT (*** ABC 1 2 ***), GRAPHIC (25U) TO < PAGE-B >;
```

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A. Parentheses may not be used in a Text Constant. (Ampersands are invalid within macros.)

B. Blanks and commas are significant, that is, they are included in the character count.

C. A maximum of 80 characters may be specified in a Text Constant.

THIS PAGE INTENTIONALLY LEFT BLANK.

17. GOAL COMPILER DIRECTIVES

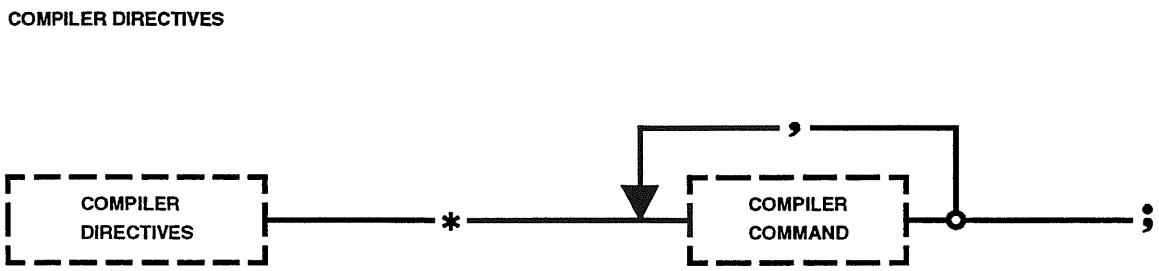


Figure 17-1 COMPILER Directives

17.1 INTRODUCTION

The COMPILER DIRECTIVES are instructions to the GOAL Language Processor which are used to change the format of compiler output listings, list specific parts of the output (e.g., sources, diagnostics, Function Designators), and block interpretive code.

Function

COMPILER DIRECTIVES are used to give directions to the GOAL Language Processor regarding the compiling and translating of a GOAL procedure.

Description

COMPILER DIRECTIVES must begin with an asterisk and end with a semicolon. The feedback loop on the syntax diagram permits a number of Compiler Commands to be entered in one statement. The COMPILER DIRECTIVES are optional. Compiler Commands of Engineering Units and Block appear in the Expanded Source Listing. The remaining commands are found only in the Source Listing of the GOAL procedure, and only the result of the action requested appears in the Expanded Source Listing.

Examples

```
*PAGE;
*TITLE (TEST SEQUENCE 15), DATE (6-23-77), PAGE;
```

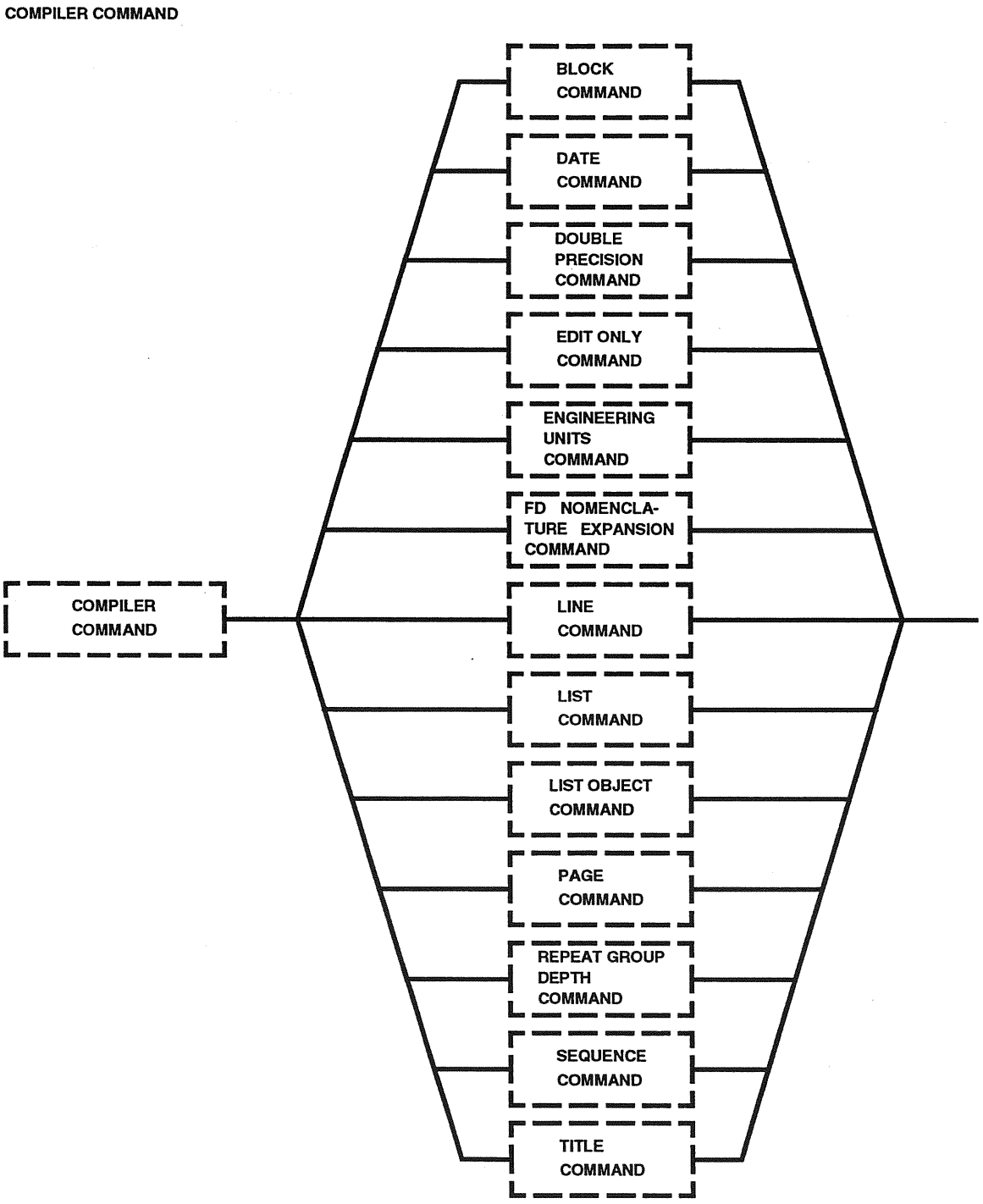


Figure 17-2 COMPILER Command

Function

The Compiler Command is an element of the COMPILER DIRECTIVES Statement.

Description

The Compiler Commands which may be used to define COMPILER DIRECTIVES are shown in the syntax diagram. The individual Compiler Commands will be discussed in the text which follows.

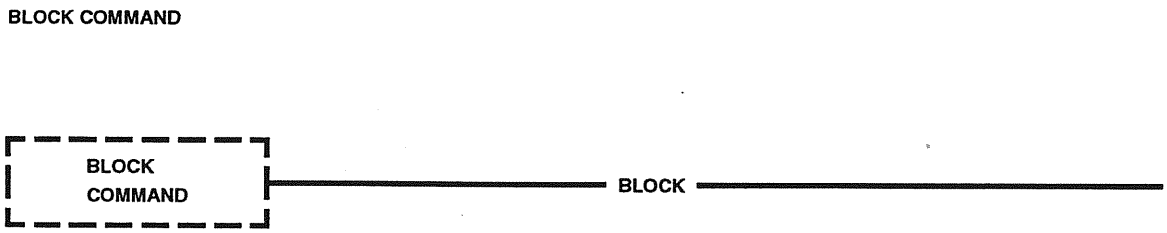


Figure 17-3 BLOCK Command

Function

The BLOCK command is used to force the end of the current block of interpretive code and start another block.

Description

Interpretive code is blocked every 128 words. The BLOCK command makes it possible to group a limited number of GOAL Statements into one block of interpretive code. This ensures that the interpretive code for the statements will be resident in the console memory at the same time and will not have to be retrieved from disk during execution. This increases the speed of execution of the GOAL Statements which have been blocked. Refer to Section 15.5 for further information on procedure efficiency considerations related to blocking interpretive code.

Example

```
*BLOCK;
```

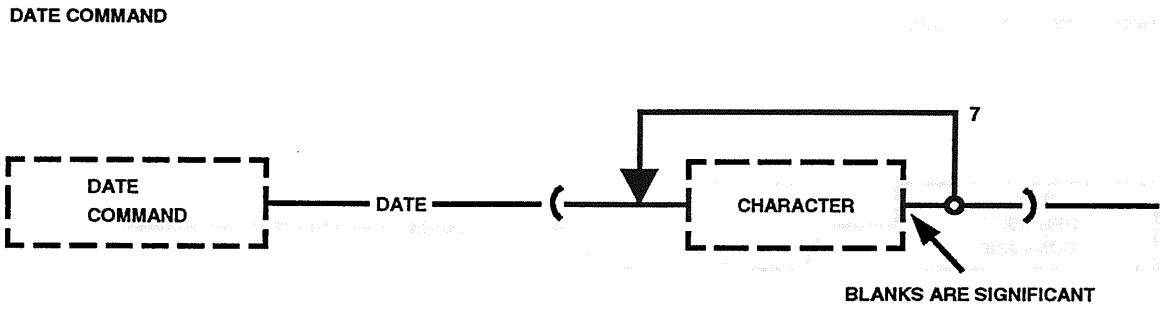


Figure 17-4 DATE Command

Function

The DATE command is used to print the date at the top of the pages of the expanded listing.

Description

This command causes the date specified within parentheses to be printed at the top of the pages of the GOAL expanded source listing. If the DATE Command appears prior to any source line other than another DATE Command or a TITLE Command, blank lines are significant, the date specified will appear on page 1 of the expanded source listing. The date is printed at the top right and left of the page. A limit of eight characters may be specified within parentheses. Blanks are significant.

Example

```
*DATE(10-23-77);
```

If the DATE command is not specified, the current day's date will be printed on each page of the expanded source listing.

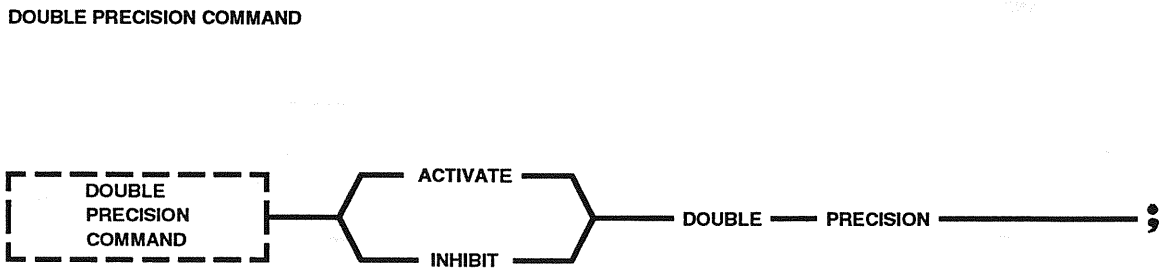


Figure 17-5 DOUBLE PRECISION Command

Function

The DOUBLE PRECISION command is used to specify the mode of integer arithmetic used in LET EQUAL Statements.

Description

Activation of double precision mode will cause all integer arithmetic in LET EQUAL Statements to be performed using double precision (64-bit intermediate; 32-bit final result) operations. Inhibition of double precision mode (the default case if precision mode is not specified) will cause all integer arithmetic in LET EQUAL Statements to be performed using single precision (32-bit intermediate; 16-bit final result) operations unless a double precision variable (time variable) is introduced into the expression. Note that the effect of this command will be evident only in the calculation of intermediate results. The final precision is governed by the length of the variable to which a value is assigned. Use of double precision mode in expressions involving only numeric Internal Names or variables will drastically increase CCMS execution time while providing greater accuracy for intermediate results.

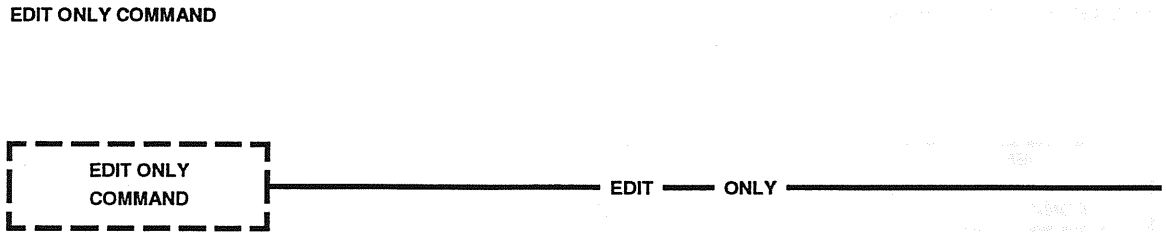


Figure 17-6 EDIT ONLY Command

Function

The EDIT ONLY command is used to suppress generation of compile object output.

Description

The EDIT ONLY command expedites compilation for the purpose of error checking and obtaining listings. Compiler output reports (e.g., Expanded Source Listing and Diagnostic Summary) are not affected by this command.

Example

```
* EDIT ONLY;
```

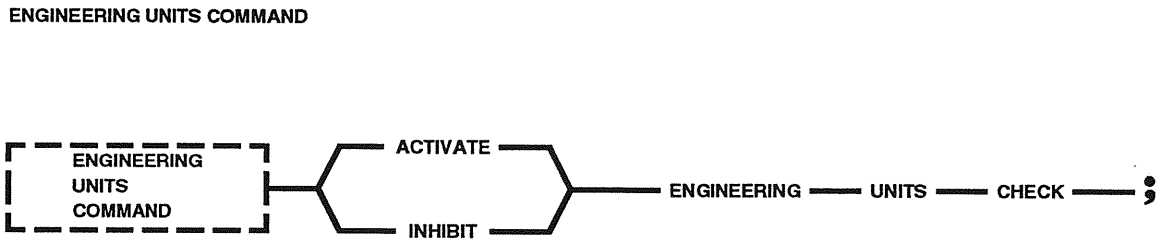


Figure 17-7 ENGINEERING UNITS Command

Function

The ENGINEERING UNITS command is used to control engineering unit validation.

Description

The ENGINEERING UNITS command is used to activate or inhibit engineering unit compatibility checks for all quantity type data. Once this command is used in the GOAL procedure, the option specified (ACTIVATE or INHIBIT) is in effect for all subsequent statements until it is changed by the use of another ENGINEERING UNITS command.

Care should be taken that the engineering unit check is not allowed to remain inhibited if the checks are necessary to ensure proper procedure execution. Note also that the INHIBIT option should be used only for those statements in which known incompatibility exists.

This command may be used many times in a GOAL procedure. Engineering unit checks are activated at the beginning of procedure compilation.

Example

```
INHIBIT ENGINEERING UNITS CHECK;
```

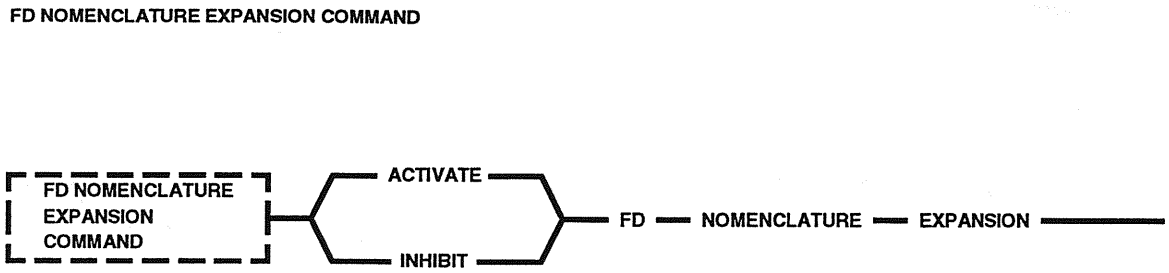


Figure 17-8 FD NOMENCLATURE EXPANSION Command

Function

The FD NOMENCLATURE EXPANSION command is used to control expansion of Function Designator nomenclature within the expanded source listing.

Description

The FD NOMENCLATURE EXPANSION command is used to activate or inhibit Function Designator nomenclature expansion in the expanded source listing. This command affects only those statements which follow it in a GOAL procedure.

The default is to insert the nomenclature (ACTIVATE) in the expanded source.

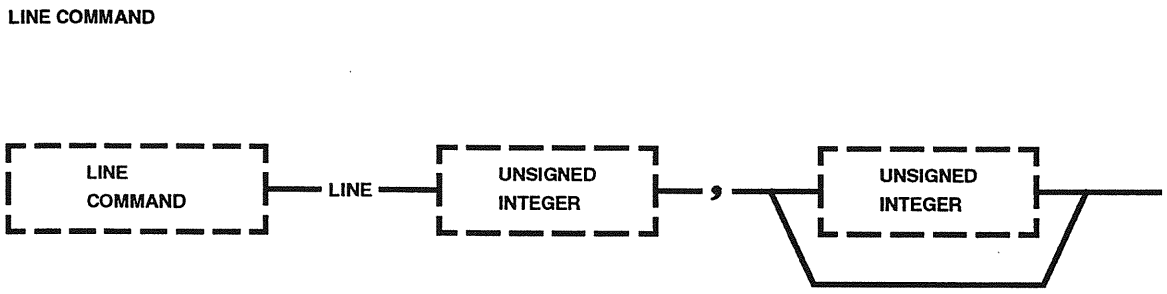


Figure 17-9 LINE Command

Function

The LINE command is used to specify the number of lines and columns to be printed per page on the Expanded Source Listing.

Description

This command is of the form

```
*LINE a,b;
```

where a and b represent the number of lines per page and the number of columns per page respectively for the expanded source listing. After printing a lines a new page will be started, and after printing b characters a new line will be started. Both a and b must be integer numbers. The range of a is 1 to 32767 and the range of b is 80 to 131. Note that the b value applies only when the expanded statement cannot be printed on a single line. If this compiler directive is not used the compiler defaults to 50 for a and 111 for b. The default value of 111 for b accounts for eleven leader columns (for * DIAGNOSTIC *, etc.) and 100 columns for expanded source.

Examples

```
*LINE 60, 80;
*LINE 120;
*LINE 80;
```

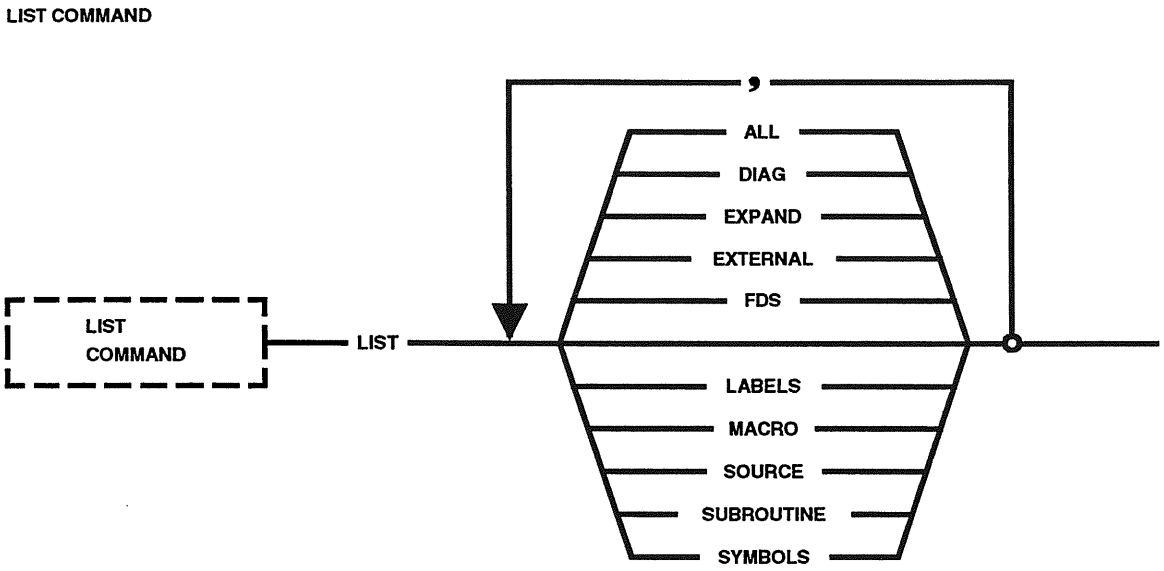


Figure 17-10 LIST Command

Function

The LIST command is used to select the reports to be generated with a GOAL compile.

Description

The LIST command allows the following output reports to be selected for a GOAL compile:

SOURCE:	Output a Source Listing
EXPAND:	Output the Expanded Source Listing
LABELS:	Output the Statement Label (i.e., step number) Cross Reference Table
SYMBOLS:	Output the symbols (i.e., Internal Names Cross Reference Table)
FDS:	Output the Function Designator Cross Reference Table
DIAG:	Output the Diagnostic Summary
ALL:	Output all of the above
EXTERNAL:	Output External Reference Cross-References
MACRO or SUBROUTINE:	Output Macro/Subroutine Cross-References (MACRO and SUBROUTINE are interchangeable)

If LIST is used with no options, no reports will be generated. The diagnostic summary will always be generated when errors or warnings are detected during compilation. If the LIST command is not used, all of the output reports will be generated except the Source Listing.

The SOURCE and EXPAND Options activate printing lines of code at the point the command is encountered. Once the command is invoked, the remainder of the program is listed. Including a *LIST (only) command at a point later in the program will terminate printing. To print the entire source program, a *LIST SOURCE command should be the first line in the program.

Examples

```
*LIST SOURCE, EXPAND;
*LIST SYMBOLS, EXPAND;
*LIST LABELS, FDS, SYMBOLS;
```

THIS PAGE INTENTIONALLY LEFT BLANK.

LIST OBJECT COMMAND

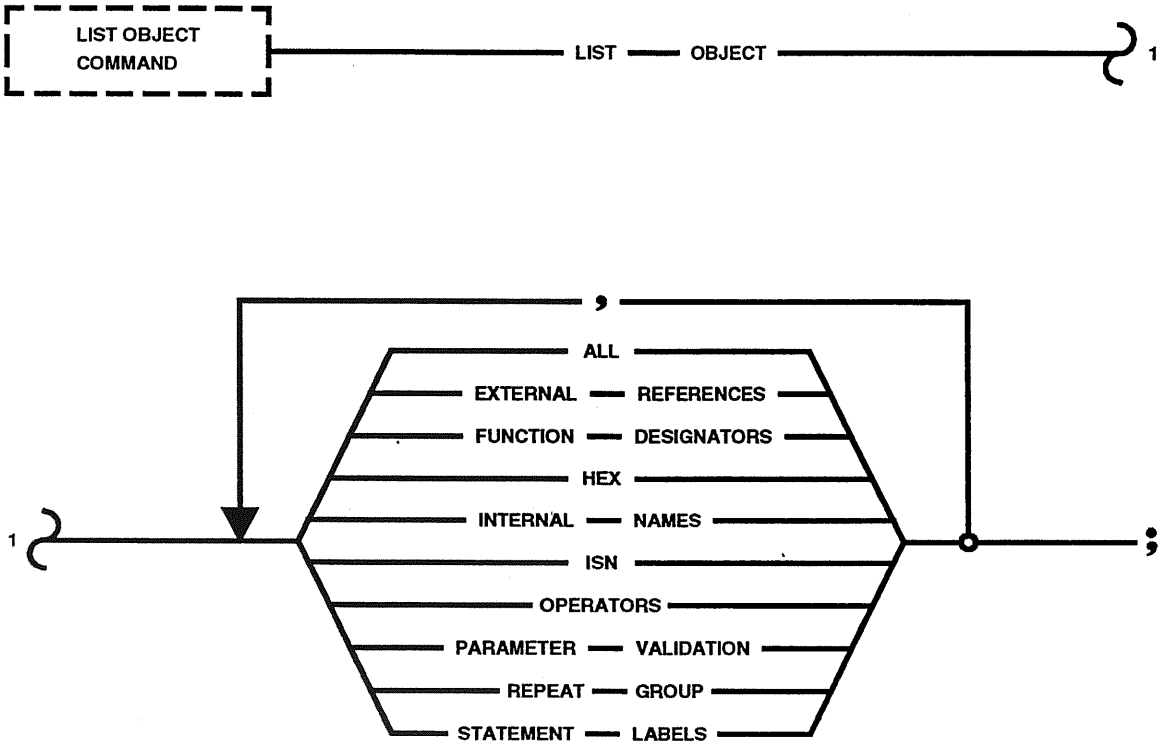


Figure 17-11 LIST OBJECT Command

Function

The LIST OBJECT command is used to obtain listings related to compiler object code.

Description

The following optional object reports may be obtained by means of the OBJECT command:

ALL:	List all options. All reports will be generated in hexadecimal.
HEX:	List output in hexadecimal.
EXTERNAL REFERENCES:	List the External Reference Table, which includes those references which are not accessible to the GLP; e.g., program names on PERFORM Statements, display skeletons.
FUNCTION DESIGNATORS:	List the Function Designator Names Table.
INTERNAL NAMES:	List the Internal Names Table.
ISN:	List the Internal Statement Numbers Table.
OPERATORS:	List the Operators.
STATEMENT LABELS:	List the Step Numbers; i.e., the Statement Label Table.
PARAMETER VALIDATION:	List the Parameter Validation Table.
REPEAT GROUP:	List the Repeat Group Table.

Examples

```
*LIST OBJECT ALL;
*LIST OBJECT INTERNAL NAMES;
*LIST OBJECT OPERATORS;
*LIST OBJECT ISN, OPERATORS, STATEMENT LABELS;
```

For examples of output generated by the LIST OBJECT command, see Figure 17-12.

The Function Designator source codes used in the Function Designator Names Table are listed below:

G - GSE
L - LDB
P - PCM
R - RTU
U - UPLK
X - GSE/PCM *
Y - GSE/RTU *
Z - GSE/PCM/RTU *

Restrictions/Limitations

Refer to the following for restrictions and limitations:

A LIST OBJECT command in a GOAL disk file definition will cause a warning message to be generated and the command will be ignored.

*These combined source codes indicate that a Function Designator used as a passed parameter may have any one of the sources in the combination.

```
======F O R M A T T E D   L I S T I N G   O F   F I L E======                    PAGE#  1

****** FILE CONTROL BLOCK (DECIMAL)
FILE CONTROL BLOCK SIZE                                          16
NUMBER OF 128 WORD BLOCKS                                         7
NUMBER OF EXTERNAL REFERENCE TABLE BLOCKS                         1
NUMBER OF EXTERNAL REFERENCE TABLE ENTRIES                        1
NUMBER OF FUNCTION DESIGNATOR NAME TABLE BLOCKS                   1
NUMBER OF FUNCTION DESIGNATOR NAME TABLE ENTRIES                  2
NUMBER OF PARAMETER VALIDATION TABLE BLOCKS                       1
NUMBER OF PARAMETER VALIDATION TABLE ENTRIES                      1
NUMBER OF RESIDENT DATA AREA BLOCKS                               1
NUMBER OF INTERPRETIVE CODE BLOCKS                                1
TOTAL BLOCKS FOR INT, ISNT, SLT, RGT, SDT                         2
DATA BANK REVISION NUMBER                                     99739
NUMBER OF INTERRUPT VECTOR TABLE BLOCKS                           0
NUMBER OF ENTRIES IN RULE TABLE                                   3
NUMBER OF BLOCKS IN RULE TABLE                                    1
NUMBER OF ENTRIES IN STATEMENT ID MATRIX                          6
NUMBER OF BLOCKS IN STATEMENT ID MATRIX                           1
SPARE                                                             0

****** EXTERNAL REFERENCE TABLE
PROCEDURE NAME                    EXTERNAL REFERENCE TYPE CODE
                                  (HEXADECIMAL)      (DECIMAL)
WK                                   00001               1

****** FUNCTION DESIGNATOR NAME TABLE
FD NAME     FD NOMENCLATURE                    TYPE SUBTYPE  REV   STATE BTUC  SOURCE PASSED SYSTEM RSYS-I RSYS-II
N001ECS     COM INTERRUPT                       27     0     30010   0    0           N      Y      N      N
PAGE-A      DISPLAY APPLICATION PAGE A          12     0     20021   0    0           N      Y      N      N

****** PARAMETER VALIDATION TABLE (OCTAL)
WORD #1          WORD #2          WORD #3          WORD #4          WORD #5          WORD #6
664220202020     202020202020     000000000001     000000000000     000000000000     000000000000

****** CONFIGURATOR RULE TABLE
RULE NUMBER (DECIMAL)         I/C ADDRESS - OFFSET (HEXADECIMAL)
          11                          000000081
          10                          0000000AB
          58                          0000000C1

****** RDA HEADER (HEXADECIMAL)
SIZE OF RDA                                  00000000F
PTR TO IVT                                   000000000
PTR TO MEASUREMENT WIB                       000000000
PTR TO DG WIB                                000000000
PTR TO PFP WIB                               000000000
PTR TO RC WIB                                000000000
PTR TO SI WIB                                000000000
PTR TO CDT WIB                               000000000
PTR TO IT WIB                                000000000
PTR TO VDA                                   00000000E
CHECKSUM OF VDA                              000000000
CHECKSUM OF FIRST 128 WORDS OF PROC          000000000
```

Figure 17-12 Output Generated by the LIST OBJECT Command (1 of 5)

```
======F O R M A T T E D   L I S T I N G   O F   F I L E======          PAGE#  2

****** RDA HEADER (HEXADECIMAL)                        CONTINUED
WIB CHECKSUM                                 000000000
NUMBER OF WORDS OF I/C                       000000044

****** PROGRAM CONTROL BLOCK (HEX/ASCII)
LENGTH/OP CODE WORD                                    000001582
PROGRAM NAME                                           WK
PROCEDURE ID                                           000000000
PROCEDURE STATUS                                       0000000000000000
PROCEDURE REVISION NUMBER                              000000000
SYSTEM CONFIGURATOR SERIAL NUMBER
CONFIGURE DATE
DATA BANK REVISION NUMBER                              00001859B
CONSOLE DISTRIBUTION MASK                              000000000
COMPILE DATE                                           070196
PROCEDURE CONTROL WORD                                 000000001
RELATIVE SECTOR ADDRESS OF INT/RNT                     000000002
RELATIVE SECTOR ADDRESS OF ISN TABLE                   000000003
RELATIVE SECTOR ADDRESS OF SLT                         000000000
RELATIVE SECTOR ADDRESS OF RGT                         000000000
I/C ADDRESS OF FIRST STATEMENT                         00000002B
NUMBER OF SECTORS IN PROC. INCLUDING RDA, I/C & TABLES 000000004
HIGHEST I/C ADDRESS IN PROCEDURE                       000000044
CONFIGURATOR SYSTEM (ASCII)
COMPILER SYSTEM (ASCII)                                SG24
VDA ADDRESS OF CURRENT ENTRY POINTER OF LCT            000000001
MAXIMUM DEPTH PROVIDED IN LCT                          000000000
TOTAL # OF RECORDS IN DISK FILE                        000000000
# RECORDS/SUBFILE (DF ONLY)                           000000000
# WORDS/RECORD (DF ONLY)                              000000000
RELATIVE SECTOR ADDRESS OF SDT                         000000000
VDA OFFSET TO START OF SCT                             000000000
SPARE                                                  000000000
SPARE                                                  000000000
SPARE                                                  000000000
SPARE                                                  000000000
SPARE                                                  000000000
```

Figure 17-12 Output Generated by the LIST OBJECT Command (2 of 5)

```
=======H E X A D E C I M A L   L I S T I N G   O F   F I L E=======          PAGE#  1

****** FILE CONTROL BLOCK
  1   1
        0  0000    0010  0007  0001  0001  0001  0002  0001  0001
        8  0008    0001  0001  0002  859B  0000  0001  0001  0000

****** EXTERNAL REFERENCE TABLE
  1  17
        0  0000    0001  0000  0410  0410

****** FUNCTION DESIGNATOR NAME TABLE
  5   1
        0  0000    001B  753A  0000  1553  0410  0665  9D27  0410
        8  0008    0410  0410  0400  000C  4E35  0000  5A91  0410
       16  0010    78D1  79E3  3666  15D5  0410  0400

****** PARAMETER VALIDATION TABLE
  3   1
        0  0000    0410  0410  0001  0000  0000  0000

****** STATEMENT ID MATRIX
  7   1
        0  0000    0000  0000  0000  0000  0000  0000

****** CONFIGURATOR RULE TABLE
  9   1
        0  0000    0081  00AB  00C1

****** RDA HEADER
 11   1
        0  0000    000F  0000  0000  0000  0000  0000  0000  0000
        8  0008    0000  000E  0000  0000  0000  0044

****** WHEN INTERRUPT BLOCKS
 11  15


****** VARIABLE DATA AREA
 11  15
        0  0000    0000

****** PROGRAM CONTROL BLOCK
 12   1
        0  0000    1582  574B  2020  2020  2020  0000  0000  0000
        8  0008    0000  0000  0000  0000  0000  859B  0000  3037
       16  0010    3031  3936  0001  0002  0003  0000  0000  002B
       24  0018    0004  0044  0000  0000  5347  3234  0001  0000
       32  0020    0000  0000  0000  0000  0000  0000  0000  0000
       40  0028    0000  0000

****** OPERATORS
 12   1
        0  0000    1582  574B  2020  2020  2020  0000  0000  0000
        8  0008    0000  0000  0000  0000  0000  859B  0000  3037
```

Figure 17-12 Output Generated by the LIST OBJECT Command (3 of 5)

```
=======H E X A D E C I M A L   L I S T I N G   O F   F I L E=======          PAGE#  2
****** OPERATORS                          CONTINUED
 12  16
       16  0010    3031  3936  0001  0002  0003  0000  0000  002B
       24  0018    0004  0044  0000  0000  5347  3234  0001  0000
       32  0020    0000  0000  0000  0000  0000  0000  0000  0000
       40  0028    0000  0000  0000  8680  0003  0277  0011  4845
       48  0030    4C4C  4F20  4352  5545  4C20  574F  524C  4400
       56  0038    02C3  0001  0000  4001  00FE  82C1  0004  0701
       64  0040    0000  4000  8184  0005

****** INTERNAL NAMES TABLE
 13   1
        0  0000    0662  0001  0006  0400  4E55  4D31  0000  0101
        8  0008    0000  0000  0000  0000  0000  0000  0000  0000
       16  0010    0000  0000  0000  0000  0000  0000  0000  0000
       24  0018    0000  0000  0000  0000  0000  0000  0000  0000
       32  0020    0000  0000  0000  0000  0000  0000  0000  0000
       40  0028    0000  0000  0000  0000  0000  0000  0000  0000
       48  0030    0000  0000  0000  0000  0000  0000  0000  0000
       56  0038    0000  0000  0000  0000  0000  0000  0000  0000
       64  0040    0000  0000  0000  0000  0000  0000  0000  0000
       72  0048    0000  0000  0000  0000  0000  0000  0000  0000
       80  0050    0000  0000  0000  0000  0000  0000  0000  0000
       88  0058    0000  0000  0000  0000  0000  0000  0000  0000
       96  0060    0000  0000  0000  0000  0000  0000  0000  0000
      104  0068    0000  0000  0000  0000  0000  0000  0000  0000
      112  0070    0000  0000  0000  0000  0000  0000  0000  0000
      120  0078    0000  0000  0000  0000  0000  0000  0000  0000

****** INTERNAL STATEMENT NUMBER TABLE
 14   1


****** STATEMENT LABEL TABLE
 11   1


****** REPEAT GROUP TABLE
 11   1


****** INTERRUPT VECTOR TABLE
 12   1
```

Figure 17-12 Output Generated by the LIST OBJECT Command (4 of 5)

```
STEP  ISN  WORD OFST  I/C ADDR  OPCODE  #ENTS  OPERATOR          I/C
----  ---  --------  --------  ------  -----  --------          ---

           0081      0000        2             PCB               1582* 574B  2020  2020  2020  0000  0000  0000    ..WK      ......
                                                                 0000  0000  0000  0000  0000  859B  0000  3037    ..............07
                                                                 3031  3936  0001  0002  0003  0000  0000  002B    0196...........+
                                                                 0004  0044  0000  0000  5347  3234  0001  0000    ...D....SG24....
                                                                 0000  0000  0000  0000  0000  0000  0000  0000    ................

                                                                0000  0000  0000*                                  ......

         3 00AC      002B        0             COMPUTE           8680  0003                                        ....
           00AE      002D        0.55      1   FORMAT TEXT       0277  0011  4845  4C4C  4F20  4352  5545  4C20    ....HELLO CRUEL
                                                                 574F  524C  4400                                  WORLD.
                                                                 FR1 = IC                          TO = TS

           00B9      0038        67. 0     1   WRITE TO PAGE     02C3  0001  0000  4001  00FE                    ..........

         4 00BE      003D        65. 7     1   SEND INTERRUPT    82C1  0004  0701  0000* 4000                    ..........

         5 00C3      0042        4. 8          LEVEL TERMINATE   8184  0005  0007                              ......
```

Figure 17-12 Output Generated by the LIST OBJECT Command (5 of 5)

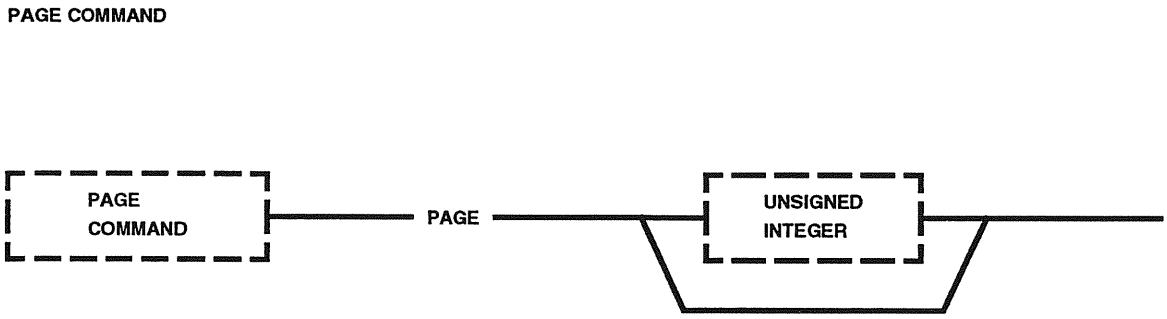


Figure 17-13 PAGE Command

Function

The PAGE command is used to specify the beginning of a new page.

Description

When the PAGE command is encountered, it causes printing to be terminated on the current page of the expanded source listing and to continue on the next page. If the optional Unsigned Integer parameter is specified, printing will begin on the next page and the page number will be the integer value specified. If the integer number is not specified, printing will just begin on the next page. The range of allowable values for Unsigned Integer is 1 to 999.

Examples

```
*PAGE;
*PAGE 55;
```

REPEAT GROUP DEPTH COMMAND

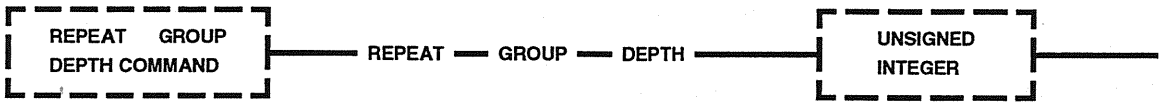


Figure 17-14 REPEAT GROUP DEPTH Command

Function

The purpose of this command is to reserve space in the Variable Data Area (VDA) for repeat loop control information.

Description

If this command is not used, the compiler automatically reserves enough table space in the VDA's Loop Control Table (LCT) to handle a total loop depth equal to the maximum depth of repeat loops in the program (twice this amount if any SPECIFY or WHEN INTERRUPT Statements appear in the program). If an AVERAGE, SEND INTERRUPT WITH DATA, RELAY INTERRUPT or RETURN INTERRUPT Statement exists in the program, the size of the reserved table space is incremented by one. This action could result in the allocation of more VDA space than is actually needed by the program.

To reserve the exact amount of VDA space needed, the REPEAT GROUP DEPTH Command should be specified. The depth specified on the command should be the maximum depth of repeat loops in the interruptable portions of the program plus the maximum depth of repeat loops in the interrupt routines of the program. If an AVERAGE, SEND INTERRUPT WITH DATA, RELAY INTERRUPT or RETURN INTERRUPT Statement exists in the deepest repeat loop of the interrupt routines, the depth specified should be incremented by one. Also, if the interrupt routines contain no repeat loops and one or more of these statements are present, the depth specified should be incremented by one.

During program execution, looping information is saved in the VDA. Assuming that only enough room is reserved in the VDA for the maximum repeat depth by the REPEAT GROUP DEPTH Command, the following situation could arise. If the procedure is executing statements in that area of maximum depth, then all the VDA locations for loop control will be in use. If at that time an interrupt occurs, program execution will be diverted to the interrupt routine. If the interrupt routine then attempts a repeat loop, it can be seen that there is no space available in the VDA for saving the looping information. At this point, in order to continue execution without error stops (and assuming the procedure will not RETURN but do a GOTO to a main line), the executor will overlay the oldest repeat loop information in the VDA with this newest repeat loop. If it is necessary to prevent this loss of information (the program may want to RETURN), more space must be reserved in the VDA for loop control information. Reserving space may be done in either of two ways:

A. Use the REPEAT GROUP DEPTH directive to specify the correct amount of VDA space needed (use the calculation described in the previous paragraph to determine the correct amount), or

B. Do not use the directive and allow the compiler to reserve the default amount of space.

Failure to do so (and then returning to the interrupted repeat loops) will eventually cause an error stop when the overwritten loop information is accessed at the CONTINUE Statement of the oldest (least depth) repeat group. The error message output will say "ISN MISMATCH" which means the Internal Step Number (ISN) of the repeat loop being accessed does not match the ISN of the CONTINUE Statement currently being executed.

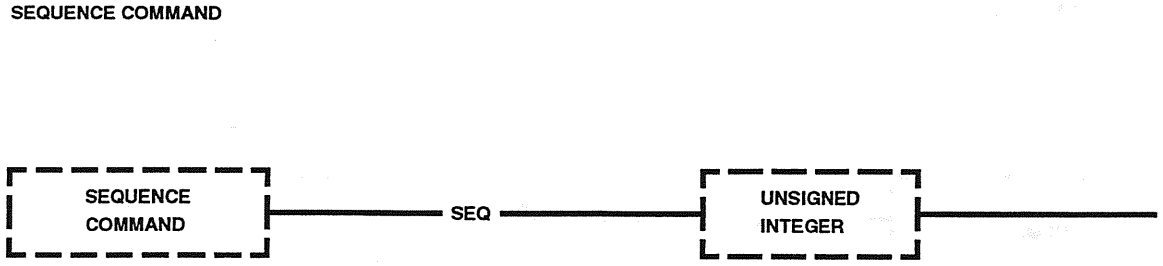


Figure 17-15 SEQUENCE Command

Function

The SEQUENCE command is used to direct the compiler to ignore sequence numbers in the source file during a compile.

Description

The SEQUENCE command causes the last (n) characters of each source record to be ignored during a compile. It is useful when a card deck with sequence numbers in the last (n) columns needs to be compiled. The maximum sequence number which may be specified is 79.

NOTE: *This command cannot be used to strip the sequence numbers (located at the beginning of each record) from a time-sharing file.*

Example

```
*SEQ 8;
```

In the example the last 8 characters of each source record will be ignored by the compiler.

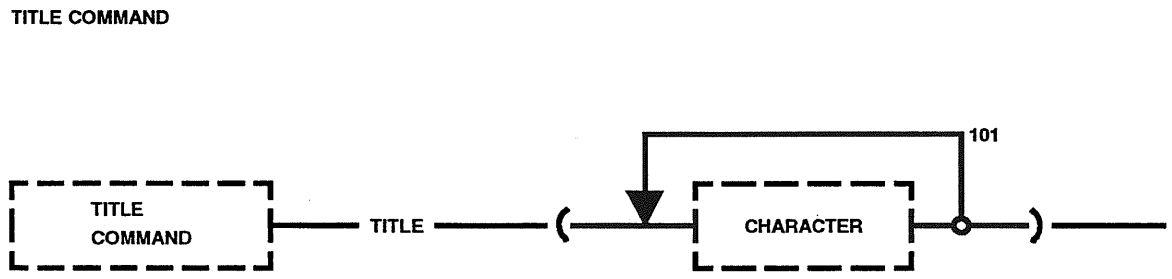


Figure 17-16 TITLE Command

Function

The TITLE command is used to print a title at the top of each page.

Description

The TITLE command causes a specified title to be printed at the top of each page of the Expanded Source Listing. If the TITLE Command appears prior to any source line other than another TITLE Command or a DATE Command, blank lines are significant, the title specified will appear on page 1 of the expanded source listing. A maximum of 102 characters may be specified within parentheses for the title. Blanks are significant.

Example

```
*TITLE(*******SAMPLE TITLE************);
```

18. PROCEDURE EFFICIENCY CONSIDERATIONS

THIS PAGE INTENTIONALLY LEFT BLANK.

18.1 RDA UTILIZATION AND SIZING

The Resident Data Area (RDA) is a 1280 word block for a GOAL Task's use in processing interrupts, declaring variable data and defining Function Designators. The interrupt portion of the RDA is limited to 1024 words.

Each RDA consists of a header of fourteen words for the use of the Executor, interrupt blocks for events referenced on SPECIFY INTERRUPT and WHEN INTERRUPT Statements, data areas reserved by DEFINE Statements, and data declared by the procedure.

The RDA is distinctly divided into two parts: Interrupt Blocks and Variable Data. The Interrupt Block portion of the RDA is created for each procedure from the SPECIFY and WHEN INTERRUPT Statements. These statements may reference different types of Function Designators, as follows:

A. Measurement Function Designators

B. Display Generator Function Keys

C. Programmable Function Panel Function Keys

D. Remote Communications

E. System Function Designators

F. Interval Timer Function Designator

G. Count Down Time or Mission Elapsed Time

Each type of Function Designator referenced will cause an interrupt block header to be reserved in the RDA. These headers are four words long. Each Function Designator referenced will cause a two word entry to be created in the RDA.

The memory required in the RDA for interrupt blocks for a procedure is dependent upon the Interrupt Blocks in the higher level. That is, Interrupt Blocks are accumulated for all active levels of a GOAL task. For example, the RDA size for a level 3 procedure must be based on the interrupts specified in the levels above - level 1 and level 2. Each task's Interrupt Blocks require enough memory for one four-word header for each Function Designator type referenced in the task and one two-word entry for each unique Function Designator referenced for each of the active levels of the task.

The Variable Data Area (VDA) is unique for each level of a GOAL concurrency. When a level 1 procedure performs another procedure, level 1's VDA is saved on an auxiliary storage device and level 2's VDA may occupy the real memory allocated to the RDA for the concurrency.

Data which may reside in the RDA and the storage requirements of each type areas follows:

A. Text data - 1 word for each 2 characters, including blanks

B. Quantity data - 2 words per data entry

C. Numeric data - 1 word per data entry

D. State data - 1/2 word per data entry

E. Lists - 2 word header per list plus 1 or more words for each data entry per above data types

F. Tables - 2 word header per table plus 1 or more words for each data entry per above data types plus 1 word for each 16 rows in the table inhibit/active status bits

G. Text Lists and Tables - Space reserved for every entry is the same as the length of the largest entry

H. Defined Analog Measurements (AM) and Analog Measurements SIO (AMS) for Function Designator type pseudo parameters - 14 words each

I. Defined Analog Stimuli (AS) and Analog Stimulus SIO (ASS) for Function Designator type pseudo parameters - 14 words each

J. Defined Discrete Measurements (DM) and Discrete Measurements SIO (DMS) for Function Designator type pseudo parameters - 5 words each

K. Defined Discrete Stimulus (DS) and Discrete Stimulus SIO (DSS) for Function Designator type pseudo parameters - 5 words each

L. Defined Digital Pattern Measurements (DPM) and Digital Pattern Measurements SIO (DPMS) for Function Designator type pseudo parameters - 6 words each

M. Defined Digital Pattern Stimuli (DPS) and Digital Pattern Stimuli SIO (DPSS) for Function Designator type pseudo parameters - 6 words each

N. Defined Pseudo Analogs (PA) for Function Designator type pseudo parameters - 3 words each

O. Defined Pseudo Discrete (PD) for Function Designator type pseudo parameters - 6 words each

P. Defined Pseudo Digital Pattern (PDP) for Function Designator type pseudo parameters - 3 words each

Q. Defined Analog Double Precision (AMDP) for Function Designator type pseudo parameters - 15 words each

R. Defined Digital Pattern Stimulus with Data (DPSD) for Function Designator type pseudo parameters - 8 words each

S. Defined Multi-word Digital Pattern (MWDP) for Function Designator type pseudo parameters - 7 words each

T. Defined Floating Point (FP) for Function Designator type pseudo parameters - 8 words each

U. Defined Data Set (THDS) for Function Designator type pseudo parameters - 3 words each

V. Defined Bus Terminal Unit (BTU) for Function Designator type pseudo parameters - 4 words each

W. All other defined Function Designators for Function Designator type pseudo parameters - 1 word each

X. LOOP CONTROL INFORMATION

The VDA also contains storage space reserved for runtime loop control information. The amount of RDA space reserved is based upon the deepest depth of nested repeat loops in the program or the depth as specified on the REPEAT GROUP DEPTH directive. Refer to Section 17, REPEAT GROUP DEPTH COMMAND, for a description of this directive and how the compiler calculates the total loop depth if the directive is not specified. If the total loop depth is zero, then no VDA space is reserved. Otherwise, the number of RDA words allocated to loop control information storage is twice the total loop depth plus one.

Y. If one or more AVERAGE, SEND INTERRUPT WITH DATA, RELAY INTERRUPT, or RETURN INTERRUPT Statements exists within a GOAL program, then one extra word might be needed for loop control information even if no repeat loops are coded in the program. To determine whether or not this is true for a specific GOAL program, see the preceding bullet on Loop Control Information and Section 17, REPEAT GROUP DEPTH COMMAND.

18.2 BLOCKING INTERPRETIVE CODE

The interpretive code resident on the Modcomp disk is blocked every 128 words. The BLOCK compiler directive may be used to force the end of the current 128 word block and start another. This capability may prove useful when a procedure contains a relatively short iterative process such as a REPEAT group or a DELAY UNTIL.

The real memory available in the Modcomp computer for interpretive code is dependent upon the RDA size. Each GOAL procedure is allocated 2048 words of memory for RDA and interpretive code. The number of interpretive code blocks which may be resident is equal to 1 block for each 128 words not used by RDA. When the interpretive code resident in memory is exhausted a disk access is required to obtain the next interpretive code blocks before the procedure can continue. Therefore, if an iterative process can be contained in the available memory, it is advantageous to the execution time of that sequence to ensure this capability by blocking the sequence in a new interpretive code block.

18.3 GROUPING FUNCTION DESIGNATORS AND DATA

Every time the GOAL Executor requests an I/O operation, the procedure requiring the I/O is suspended, and other tasks are given an opportunity to execute. Therefore, any time data to be written to a CRT, read from the CDBFR, etc., is grouped such that one operation can be performed, and both I/O and CPU time can be minimized.

18.3.1 OBTAINING MEASUREMENT AND COMMAND STATUS

When more than one measurement or stimulus value must be obtained from the CDBFR, the GOAL procedure will execute more efficiently if all Function Designators are grouped in one statement. Each time the GOAL Executor reads a value from the CDBFR the execution of the procedure is suspended until the I/O required to obtain the value is completed. If all Function Designators are grouped in one statement, the GOAL Executor can obtain all required values with the minimum number of CDBFR I/O requests. If only one Function Designator is used per statement, each statement will require a separate CDBFR I/O request. The number of CDBFR I/O requests required to read multiple Function Designators on a single statement can be minimized by grouping like types together (i.e., analogs, discretes, digital patterns). This technique applies to Row Designators for tables also. Grouping like types as contiguous Row Designators will minimize CDBFR I/O wait time also. Passed Function Designators (pseudo parameters) and real Function Designators are considered different types by the compiler even if they are the same data bank type.

If the status of a group of measurements must be tested, it can be accomplished most efficiently by either:

A. Using a single Verify Prefix, or

B. Using a single READ Statement followed by the IF option of the Verify Prefix.

Examples

Most Efficient	Least Efficient
VERIFY < DM1 > < DM2 > < DM3 > ARE ON;	VERIFY < DM1 > IS ON; VERIFY < DM2 > IS ON; VERIFY < DM3 > IS ON;

READ < AM1 >
< AM2 >
< AM3 > AND SAVE AS (QLIST1);
IF (QLIST)1 IS LESS THAN 5V,
GO TO STEP 10;
IF (QLIST1)2 IS LESS THAN 4V,
GO TO STEP 15;
IF (QLIST1)3 IS LESS THAN -5V,
GO TO STEP 20;
	VERIFY < AM1 > IS LESS THAN
	5V THEN GO TO STEP 10;
	VERIFY < AM2 > IS LESS THAN
	4V THEN GO TO STEP 15;
	VERIFY < AM3 > IS LESS THAN
	-5V THEN GO TO STEP 20;

In the first example, grouping the three discrete measurement Function Designators in a single Verify Prefix is more efficient than executing three Verify Prefixes. In the second example, a single READ followed by the IF option of the Verify Prefix is more efficient than using three Verify Prefixes.

18.3.2 RECORDING DATA

When using the RECORD DATA Statement, it is more efficient to group all of the data to be recorded before the keyword TO. When printing on the console printer/plotter, this technique gives the added advantage of insuring that data from other procedures will not be merged with the data being printed.

Examples

Most Efficient

```
RECORD TEXT(MESSAGE 1) NEXT
    TEXT(MESSAGE 2) NEXT
    TEXT(MESSAGE 3)
    TO < PAGE-B >;
```

Least Efficient

```
RECORD TEXT(MESSAGE 1)
    TO < PAGE-B >;
RECORD TEXT(MESSAGE 2)
    TO < PAGE-B >;
RECORD TEXT(MESSAGE 3)
    TO < PAGE-B >;
```

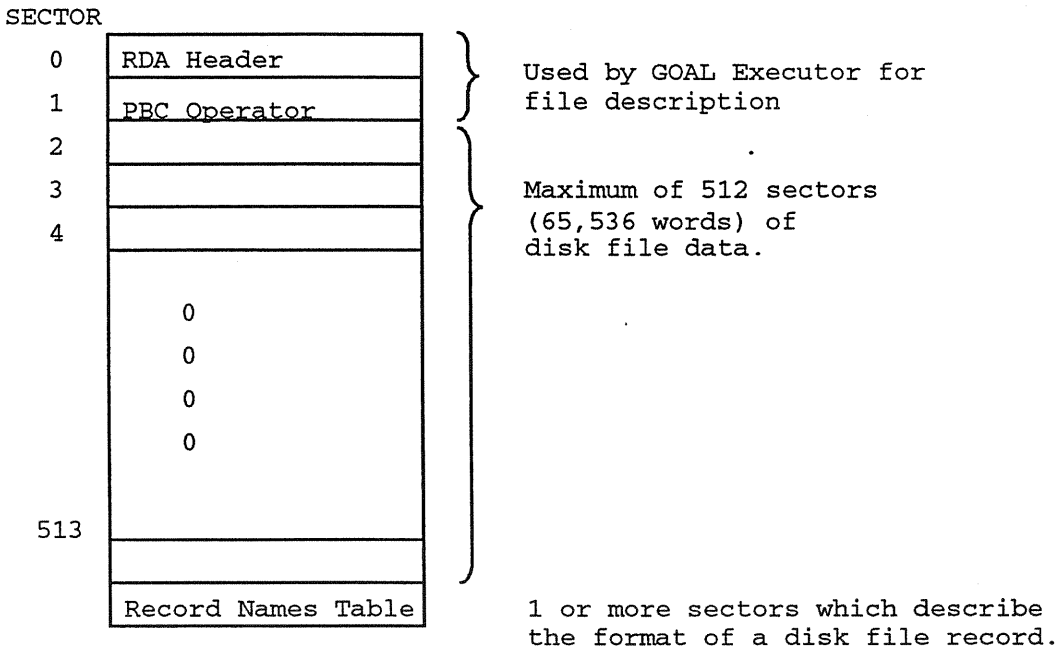
```
RECORD TEXT(MESSAGE 1)
    TO < PAGE-B >,
    TEXT(MESSAGE 2)
    TO < PAGE-B >,
    TEXT(MESSAGE 3)
    TO < PAGE-B >;
```

All of these examples accomplish the same purpose. However, the one on the left is the most efficient, since all of the text to be displayed is output in a single I/O request. The examples on the right illustrate two different techniques for displaying the same text. Both require three I/O requests each.

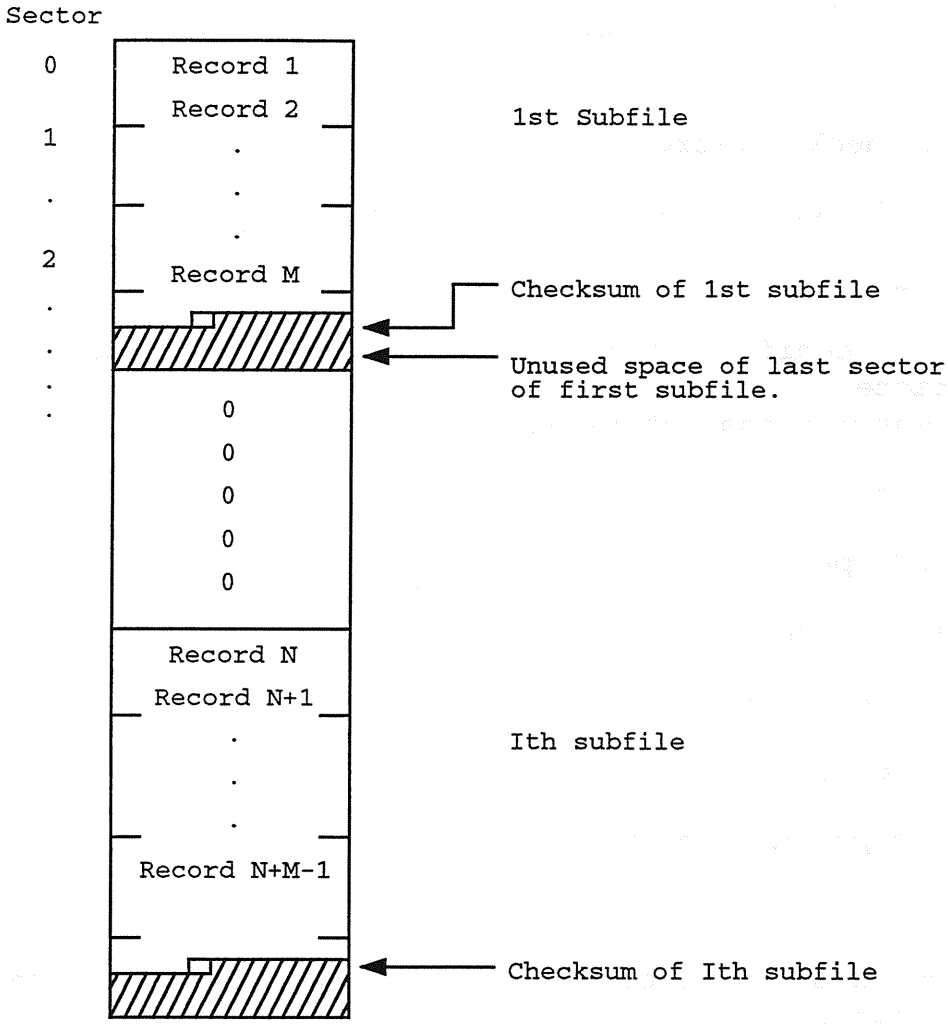
18.4 DISK FILE SIZING AND UTILIZATION

Sizing

The type of disk files created on CDS via the compiler statements defined in Sections 13, 13.1 and 13.2, have the following format on a CCMS disk.



The disk file data is organized into logical records and subfiles (blocks of records) as specified in the definition of the disk file. The mapping of logical records and subfiles to physical sectors (128 words) of a disk is as follows:



Each subfile starts at the beginning of a sector and the last word of the subfile is the checksum of the subfile. Space left between the checksum and the start of the next subfile is unused (wasted) disk space. Therefore, for efficient disk space utilization, a subfile should be close to a 128 word multiple (without going over) in size. The unused space is included in the 65,536 word limit.

SS = Subfile Size = (M)*(RS) + 1

M = # Records/Subfile = # Records specified on "BLOCKED AT" option of the BEGIN DISK FILE DEFINITION Statement.

RS = Record Size = Sum of the sizes of all variables within the record, not exceed 384 words.

1 = size of checksum word

The size of each type of variable is determined by the following:

A. Text data

1 word specifying # characters + 1 word for each 2 characters including blanks.
(integer((# characters +1)/2)+ 1)

B. Quantity data

2 words per data entry.

C. Numeric data

1 word per data entry.

D. State data

1 word per data entry.

E. Lists

(# entries)*(size of one entry as specified by above data types).

The amount of bulk memory occupied by a disk file after being opened into a system area equals the subfile size minus one (no checksum).

Example 1

```
BEGIN DISK FILE DEFINITION FOR FILE (XYZ) WITH 100 RECORDS
    BLOCKED AT 10 RECORDS;
DECLARE QUANTITY (Q1) = V;
DECLARE STATE (S1) = ON/OFF;
DECLARE NUMBER (N1) = I;
```

```
DECLARE TEXT (T1) = 9 CHARACTERS;
DECLARE QUANTITY LIST (QL1) WITH 3 ENTRIES V;
DECLARE STATE LIST (SL1) WITH 5 ENTRIES ON/OFF;
DECLARE NUMERIC LIST (NL1) WITH 8 ENTRIES X;
DECLARE TEXT LIST (TL1) WITH 3 ENTRIES 4 CHARACTERS;
END DISK FILE DEFINITION;

RS =    Q1   =    2 Words
        S1   =    1 Word
        N1   =    1 Word
        T1   =    6 Words
        QL1  =    6 Words
        SL1  =    5 Words
        NL1  =    8 Words
        TL1  =    9 Words
                 38 Words/Record

M  =    10 Records/Subfile

SS =    (10 Records/Subfile)(38 Words/Record) +1 = 381
        Words/Subfile

Wasted Disk Space = ((3 * 128) - 381) = 3 words/subfile

Bulk Memory Used  = SS - 1 = 380 Words
```

Example 2

```
BEGIN DISK FILE DEFINITION FOR FILE (ABC) WITH 512 RECORDS
    BLOCKED AT 1 RECORD;
DECLARE NUMBER (N1) = I;
END DISK FILE DEFINITION;

RS = Size of (N1) = 1 Word
M  = 1 Record/Subfile
SS = (1 Record/Subfile) (1 Word/Record) +1 = 2 words/subfile
Wasted Disk Space = ((1 * 128) -2) = 126 words/subfile
                                   = 126 words/record

Bulk Memory Used = SS - 1 = 1 word
```

Performance Considerations

Disk files should be organized to minimize runtime console disk activity. This minimization is accomplished by maximizing the use of bulk memory as a cache storage area for frequently accessed records.

There are 10,112 words of bulk memory reserved for use by a maximum of 18 open files. As indicated above, the amount of bulk memory used by a file is based upon its subfile size. After the first access, one subfile of a file's data is kept in bulk memory. Any subsequent accesses to records within this subfile will cause no disk activity. One or more disk accesses will result from the reference of a non-bulk memory resident record. Having the entire file bulk memory resident all of the time is the most ideal situation. In this case, the file has only one subfile containing all the records. Except for small files, this organization is probably impractical. For larger files, the records must be blocked into subfiles via the "Blocked At" option of the BEGIN DISK FILE DEFINITION Statement. The method of deciding the number of records to block together should be based on the record size and anticipated access method. Access methods fall into three basic categories; random, sequential, other.

If records are to be accessed at random, disk accesses are minimized by the following:

A. If the record size is less than or equal to 128 words, then let the number of records/subfile = integer (383/ record size)

B. If the record size is greater than 128 words, then let the number of records/subfile = 1

Sequentially accessed files should have subfile sizes as follows:

records/subfile = integer (amount bulk memory available/record size)

If a relationship between records within a file exists such that the probability of accessing a record within a subset of the records is higher than the probability of accessing a record outside the subset, then the records of the subset should be contained within the same subfile. The possibility of more than one GOAL program, within the "system," accessing the same file should also be considered when organizing subfiles.

NOTE: The above discussion does not take disk space utilization into consideration. In specific situations, a compromise between performance and disk space utilization may have to be made.

18.5 SPECIAL CONSIDERATIONS FOR PERFORM STATEMENT GROUPS

The PERFORM STATEMENT GROUP processing routines utilize dynamic storage capabilities during the processing and generation of interpretive code for the comparison tests specified by all statement group labels between the 'PERFORM STATEMENT GROUPS' and 'END STATEMENT GROUPS' Statements. Since the amount of dynamic storage available is limited by the CDS operating system, memory may be exhausted if a large number of statement groups and/or comparison tests are part of the same group structure. This number is typically in the range of 400 to 500 statement groups. The actual number may vary depending on the number and type of comparisons specified. If this condition arises, the statement group structure may be divided into two or more smaller structures and recompiled successfully.

THIS PAGE INTENTIONALLY LEFT BLANK.

19. GOAL SHORT FORMS

THIS PAGE INTENTIONALLY LEFT BLANK.

KEYWORDS	SHORT FORM
ACTIVATE	ACT
AND SAVE AS	ASA
AND INDICATE RESTART LABELS	AIRL
BEGIN MACRO	BMAC
BEGIN PROGRAM	BPRO
BITE STATUS REGISTER	BSR
CHARACTERS	CHAR
COLUMNS	COL
CONCURRENTLY	CONC
DECLARE NUMBER	DN
DECLARE NUMERIC LIST	DNL
DECLARE NUMERIC TABLE	DNT
DECLARE QUANTITY	DQ
DECLARE QUANTITY LIST	DQL
DECLARE QUANTITY TABLE	DQT
DECLARE STATE	DS
DECLARE STATE LIST	DSL
DECLARE STATE TABLE	DST
DECLARE TEXT	DT
DECLARE TEXT LIST	DTL
DECLARE TEXT TABLE	DTT
END MACRO	EMAC
END PROGRAM	EPRO
ENTRIES	ENT
EQUAL TO	EQ, =
GREATER THAN	GT
GREATER THAN OR EQUAL TO	GE
INHIBIT	INH
LESS THAN	LT
LESS THAN OR EQUAL TO	LE

KEYWORDS	SHORT FORM
LIST OBJECT	LISTX
NOT EQUAL TO	NE
PERFORM	PER
PRESENT VALUE OF	PVO
REGISTER	REG
ROWS AND	RA
READINGS OF	RO
SHIFT LEFT	SL
SHIFT RIGHT	SR
STEP	S
SYSTEM	SYS
TERMINATE	TER
USING MESSAGE FROM	UMF
KEYWORD	SHORT FORM
VERIFY	VER
WHEN INTERRUPT	WI
WITH ENTRIES	WENT

APPENDIX A. GOAL DIMENSIONS

THIS PAGE INTENTIONALLY LEFT BLANK.

The GOAL compiler, GOAL Executor and CCMS Data Bank will support the following list of Dimensions (engineering units). Each Dimension may be described as follows:

DESCRIPTION - Name of the dimension.

GOAL INPUT - Actual character string (abbreviated name), which must be specified when referencing a Dimension in a GOAL Procedure.

DESCRIPTION	GOAL INPUT
AMPERE AC	AMPAC
AMPERE DC	AMP
ARCMINUTE	ARC MIN
ARCSECOND	ARCS
ASTRONOMICAL UNITS	AU
BRITISH THERMAL UNITS PER SQUARE FOOT PER SECOND	BFS
COUNT(S)	CNT
COUNT DOWN TIME	CDT
CYCLE(S)	C
DECIBEL(S)	DB
DEGREE(S)	DEG
DEGREE(S) CENTIGRADE	DEGC
DEGREE(S) FAHRENHEIT	DEGF
DEGREE(S) KELVIN	DEGK
DEGREE(S) PER HOUR	DEG/HR
DEGREE(S) PER HOUR PER G	DEG/HR/G
DEGREE(S) PER SECOND	DEG/S
DEGREE(S) RANKIN	DEGR
DIMENSIONLESS QUANTITY	NULL
FEET	FT
FEET PER SEC	FT/S
FEET PER SEC2	FT/S2
G	G
G PEAK TO PEAK	GP-P
GALLON(S)	GAL
GALLON(S) PER MIN	GAL/MIN
GIGA HERTZ	GHZ
GRAINS PER POUND OF DRY AIR	GR/LBDA
GRAM(S)	GM
GREENWICH MEAN TIME	GMT
HERTZ	HZ
INCH(ES)	IN
INCH(ES) OF MERCURY	INHG
INCH(ES) OF WATER	INH20

DESCRIPTION	GOAL INPUT
JULIAN TIME OF YEAR	JTOY
KILOFEET	KFT
KILOFEET PER SECOND	KFT/S
KILOGALLON(S)	KGAL
KILOGALLONS PER MINUTE	KGAL/MIN
KILOHERTZ	KHZ
KILOMICRO G	KUG
KILOMICRO G PER G	KUG/G
KILORPM	KRPM
KILOWATT(S)	KW
KILOWATT(S) PER METER2	KW/M2
KNOT(S)	KT
KNOT(S)/SECOND	KT/S
MACH	MACH
MEGAFEET	MEGAFT
MEGAFEET PER SECOND	MEGAFT/S
MEGAHERTZ	MEGAHZ
METER(S)	M
METER(S) PER SECOND	M/S
MICROGRAM	UG
MICROGRAM PER G	UG/G
MICROGRAM PER METER CUBED	UGM/M3
MICROINCH	UIN
MICROINCH PER INCH	UIN/IN
MICROMETER(S)	UM
MICROMETER PER METER	UM/M
MILE(S)	MI
MILE(S) PER SECOND	MI/S
MILLI AMP(S)	MAMP
MILLI JOULE(S)	MJ
MILLIMETER	MM
MILLIMETER OF MERCURY	MMHG
MILLIRADIANS	MRAD
MILLIRADIANS PER SECOND	MRAD/S
MILLIVOLTS	MV
NAUTICAL MILE	NM
NON-DIMENSION	NULL
OHM(S)	OHM

DESCRIPTION	GOAL INPUT
PART(S) PER MILLION	PPM
PERCENT	PCT
POUND FORCE	LBF
POUND MASS	LBM
POUND(S) PER CUBIC FOOT	LBM/FT3
POUND(S) PER HOUR	LBM/HR
POUND(S) PER SQUARE FOOT	LBF/FT2
POUND(S) PER SQUARE FOOT PER SECOND	LBF/F2/S
POUND(S) PER SQUARE INCH	PSI
PSI ATMOSPHERE	PSIA
PSI DIFFERENTIAL	PSID
PSI GAUGE	PSIG
PSI PER MINUTE	PSI/MIN
PULSE(S)	P
PULSE(S) PER SECOND	PPS
RADIAN(S)	RAD
RADIAN(S) PER SECOND	RAD/S
RADIAN(S) PER SECOND2	RAD/S2
RADIAN(S) PER SECOND PER G	RAD/S/G
REVOLUTION(S) PER MINUTE	RPM
SLUGS	SLUGS
TIME/INTERVAL TIME (DAY, HOUR, MINUTE, SECONDS, MILLISECONDS)	DAY, HR, MIN, SEC, MS
TORR	TORR
UNIT(S)	UNITS
VOLT(S)	V
VOLT(S) AC	VAC
VOLT(S) PEAK TO PEAK	VP-P
VOLT(S) RMS	VRMS
WATT(S)	W

APPENDIX B. SYSTEM FUNCTION DESIGNATORS

THIS PAGE INTENTIONALLY LEFT BLANK.

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
TYPE = ALARM		
ALARM-1	ALRM	STATION #1 ALARM
ALARM-2	ALRM	STATION #2 ALARM
ALARM-3	ALRM	STATION #3 ALARM
TYPE = CDT		
CDT	CDT	COUNTDOWN TIME
JTOY	CDT	JULIAN TIME OF YEAR
METIME	CDT	MISSION ELAPSED TIME
TYPE = CONS		
APU	CONS	RESP. SYSTEM CONSOLE
ARMS	CONS	RESP. SYSTEM CONSOLE
BACKUP	CONS	RESP. SYSTEM CONSOLE
BELEC	CONS	RESP. SYSTEM CONSOLE
BHYD	CONS	RESP. SYSTEM CONSOLE
BINST	CONS	RESP. SYSTEM CONSOLE
BIOMED	CONS	RESP. SYSTEM CONSOLE
BMDM	CONS	RESP. SYSTEM CONSOLE
BPYRO	CONS	RESP. SYSTEM CONSOLE
BREC	CONS	RESP. SYSTEM CONSOLE
BRS	CONS	RESP. SYSTEM CONSOLE
C1	CONS	RESP. SYSTEM CONSOLE
C2	CONS	RESP. SYSTEM CONSOLE
C3	CONS	RESP. SYSTEM CONSOLE
C4	CONS	RESP. SYSTEM CONSOLE
C5	CONS	RESP. SYSTEM CONSOLE
C6	CONS	RESP. SYSTEM CONSOLE
C7	CONS	RESP. SYSTEM CONSOLE
C8	CONS	RESP. SYSTEM CONSOLE
C9	CONS	RESP. SYSTEM CONSOLE
C10	CONS	RESP. SYSTEM CONSOLE
C11	CONS	RESP. SYSTEM CONSOLE
C12	CONS	RESP. SYSTEM CONSOLE
CARGO	CONS	RESP. SYSTEM CONSOLE
CEOA	CONS	RESP. SYSTEM CONSOLE
CITEM	CONS	RESP. SYSTEM CONSOLE
CITEOP	CONS	RESP. SYSTEM CONSOLE
COMM	CONS	RESP. SYSTEM CONSOLE
COPL	CONS	RESP. SYSTEM CONSOLE

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
DISCON	CONS	RESP. SYSTEM CONSOLE
DPS	CONS	RESP. SYSTEM CONSOLE
DPSGSW	CONS	RESP. SYSTEM CONSOLE
ECLSS	CONS	RESP. SYSTEM CONSOLE
ECS	CONS	RESP. SYSTEM CONSOLE
EPDC	CONS	RESP. SYSTEM CONSOLE
FALM	CONS	RESP. SYSTEM CONSOLE
FCP	CONS	RESP. SYSTEM CONSOLE
FCPRSD	CONS	RESP. SYSTEM CONSOLE
FIREX	CONS	RESP. SYSTEM CONSOLE
FRCS	CONS	RESP. SYSTEM CONSOLE
GFRCS	CONS	RESP. SYSTEM CONSOLE
GLSAPU	CONS	RESP. SYSTEM CONSOLE
GLSARM	CONS	RESP. SYSTEM CONSOLE
GLSBEL	CONS	RESP. SYSTEM CONSOLE
GLSBHD	CONS	RESP. SYSTEM CONSOLE
GLSBIN	CONS	RESP. SYSTEM CONSOLE
GLSBPY	CONS	RESP. SYSTEM CONSOLE
GLSBRS	CONS	RESP. SYSTEM CONSOLE
GLSCOM	CONS	RESP. SYSTEM CONSOLE
GLSDPS	CONS	RESP. SYSTEM CONSOLE
GLSECL	CONS	RESP. SYSTEM CONSOLE
GLSECS	CONS	RESP. SYSTEM CONSOLE
GLSEPD	CONS	RESP. SYSTEM CONSOLE
GLSFCP	CONS	RESP. SYSTEM CONSOLE
GLSFUL	CONS	RESP. SYSTEM CONSOLE
GLSGNC	CONS	RESP. SYSTEM CONSOLE
GLSHYD	CONS	RESP. SYSTEM CONSOLE
GLSH20	CONS	RESP. SYSTEM CONSOLE
GLSINS	CONS	RESP. SYSTEM CONSOLE
GLSLH2	CONS	RESP. SYSTEM CONSOLE
GLSL02	CONS	RESP. SYSTEM CONSOLE
GLSMEC	CONS	RESP. SYSTEM CONSOLE
GLSMPS	CONS	RESP. SYSTEM CONSOLE
GLSNAV	CONS	RESP. SYSTEM CONSOLE
GLSOX	CONS	RESP. SYSTEM CONSOLE
GLSPLB	CONS	RESP. SYSTEM CONSOLE
GLSPRD	CONS	RESP. SYSTEM CONSOLE
GLSPVD	CONS	RESP. SYSTEM CONSOLE
GLSSSM	CONS	RESP. SYSTEM CONSOLE
GLSTIN	CONS	RESP. SYSTEM CONSOLE
GLSTRS	CONS	RESP. SYSTEM CONSOLE
GNC	CONS	RESP. SYSTEM CONSOLE

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
GOXARM	CONS	RESP. SYSTEM CONSOLE
HAZGAS	CONS	RESP. SYSTEM CONSOLE
HVAC	CONS	RESP. SYSTEM CONSOLE
HYD	CONS	RESP. SYSTEM CONSOLE
HYDORB	CONS	RESP. SYSTEM CONSOLE
HYDSRB	CONS	RESP. SYSTEM CONSOLE
HYFUEL	CONS	RESP. SYSTEM CONSOLE
HYHEGN	CONS	RESP. SYSTEM CONSOLE
HYOXID	CONS	RESP. SYSTEM CONSOLE
INSTR	CONS	RESP. SYSTEM CONSOLE
INTEG	CONS	RESP. SYSTEM CONSOLE
LAFT	CONS	RESP. SYSTEM CONSOLE
LANDGR	CONS	RESP. SYSTEM CONSOLE
LAPS	CONS	RESP. SYSTEM CONSOLE
LFWD	CONS	RESP. SYSTEM CONSOLE
LGEN	CONS	RESP. SYSTEM CONSOLE
LHLDFD	CONS	RESP. SYSTEM CONSOLE
LH2	CONS	RESP. SYSTEM CONSOLE
L02	CONS	RESP. SYSTEM CONSOLE
LPSTST	CONS	RESP. SYSTEM CONSOLE
MECH	CONS	RESP. SYSTEM CONSOLE
MET	CONS	RESP. SYSTEM CONSOLE
MPS	CONS	RESP. SYSTEM CONSOLE
MSRB	CONS	RESP. SYSTEM CONSOLE
MSTR1	CONS	RESP. SYSTEM CONSOLE
MSTR2	CONS	RESP. SYSTEM CONSOLE
MSTR3	CONS	RESP. SYSTEM CONSOLE
MSTR4	CONS	RESP. SYSTEM CONSOLE
NAVAID	CONS	RESP. SYSTEM CONSOLE
NONE	CONS	RESP. SYSTEM CONSOLE
OIGSW	CONS	RESP. SYSTEM CONSOLE
OSTA-1	CONS	RESP. SYSTEM CONSOLE
PBIC	CONS	RESP. SYSTEM CONSOLE
PBK	CONS	RESP. SYSTEM CONSOLE
PL	CONS	RESP. SYSTEM CONSOLE
PLBD	CONS	RESP. SYSTEM CONSOLE
PNEU	CONS	RESP. SYSTEM CONSOLE
PVD	CONS	RESP. SYSTEM CONSOLE
RAFT	CONS	RESP. SYSTEM CONSOLE
RAPS	CONS	RESP. SYSTEM CONSOLE
RFWD	CONS	RESP. SYSTEM CONSOLE
RGEN	CONS	RESP. SYSTEM CONSOLE
SPPOW	CONS	RESP. SYSTEM CONSOLE
SSME	CONS	RESP. SYSTEM CONSOLE

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SW	CONS	RESP. SYSTEM CONSOLE
TINST	CONS	RESP. SYSTEM CONSOLE
TPOWR	CONS	RESP. SYSTEM CONSOLE
TPROP	CONS	RESP. SYSTEM CONSOLE
TRS	CONS	RESP. SYSTEM CONSOLE
UCS	CONS	RESP. SYSTEM CONSOLE
VAL1	CONS	RESP. SYSTEM CONSOLE
VAL2	CONS	RESP. SYSTEM CONSOLE
VAL3	CONS	RESP. SYSTEM CONSOLE
VAL4	CONS	RESP. SYSTEM CONSOLE
VEALL	CONS	RESP. SYSTEM CONSOLE
VECPRS	CONS	RESP. SYSTEM CONSOLE
VEGSW	CONS	RESP. SYSTEM CONSOLE
VEIDME	CONS	RESP. SYSTEM CONSOLE
VEIDPS	CONS	RESP. SYSTEM CONSOLE
WATER	CONS	RESP. SYSTEM CONSOLE
60HZ	CONS	RESP. SYSTEM CONSOLE

TYPE = CPP

CNSL-PP	CPP	CONSOLE PRINTER PLOTTER

TYPE = DATE

DATE	DATE	MONTH DAY YEAR

TYPE = DGKY

DISARM-PA	DGKY	DISARM COMMAND FUNCT KEY PAGE A
DISARM-PB	DGKY	DISARM COMMAND FUNCT KEY PAGE B
DISARM-P1A	DGKY	DISARM COMMAND FUNCT KEY PAGE 1A
DISARM-P1B	DGKY	DISARM COMMAND FUNCT KEY PAGE 1B
DISARM-P2A	DGKY	DISARM COMMAND FUNCT KEY PAGE 2A
DISARM-P2B	DGKY	DISARM COMMAND FUNCT KEY PAGE 2B
DISARM-P3A	DGKY	DISARM COMMAND FUNCT KEY PAGE 3A

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
DISARM-P3B	DGKY	DISARM COMMAND FUNCT KEY PAGE 3B
DISARM-P4A	DGKY	DISARM COMMAND FUNCT KEY PAGE 4A
DISARM-P4B	DGKY	DISARM COMMAND FUNCT KEY PAGE 4B
DISARM-P5A	DGKY	DISARM COMMAND FUNCT KEY PAGE 5A
DISARM-P5B	DGKY	DISARM COMMAND FUNCT KEY PAGE 5B
DISARM-P6A	DGKY	DISARM COMMAND FUNCT KEY PAGE 6A
DISARM-P6B	DGKY	DISARM COMMAND FUNCT KEY PAGE 6B
EXEC-PA	DGKY	EXECUTE COMMAND FUNCT KEY PAGE A
EXEC-PB	DGKY	EXECUTE COMMAND FUNCT KEY PAGE B
EXEC-P1A	DGKY	EXECUTE COMMAND FUNCT KEY PAGE 1A
EXEC-P1B	DGKY	EXECUTE COMMAND FUNCT KEY PAGE 1B
EXEC-P2A	DGKY	EXECUTE COMMAND FUNCT KEY PAGE 2A
EXEC-P2B	DGKY	EXECUTE COMMAND FUNCT KEY PAGE 2B
EXEC-P3A	DGKY	EXECUTE COMMAND FUNCT KEY PAGE 3A
EXEC-P3B	DGKY	EXECUTE COMMAND FUNCT KEY PAGE 3B
EXEC-P4A	DGKY	EXECUTE COMMAND FUNCT KEY PAGE 4A
EXEC-P4B	DGKY	EXECUTE COMMAND FUNCT KEY PAGE 4B
EXEC-P5A	DGKY	EXECUTE COMMAND FUNCT KEY PAGE 5A
EXEC-P5B	DGKY	EXECUTE COMMAND FUNCT KEY PAGE 5B
EXEC-P6A	DGKY	EXECUTE COMMAND FUNCT KEY PAGE 6A
EXEC-P6B	DGKY	EXECUTE COMMAND FUNCT KEY PAGE 6B
XMIT-PA	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE A
XMIT-PB	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE B
XMIT-P1A	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE 1A
XMIT-P1B	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE 1B
XMIT-P2A	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE 2A
XMIT-P2B	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE 2B
XMIT-P3A	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE 3A
XMIT-P3B	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE 3B
XMIT-P4A	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE 4A
XMIT-P4B	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE 4B
XMIT-P5A	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE 5A
XMIT-P5B	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE 5B
XMIT-P6A	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE 6A
XMIT-P6B	DGKY	TRANSMIT CURSOR FUNCT KEY PAGE 6B

TYPE = FEP

CDL1	FEP	CITE DOWNLINK FEP #1
CGS1	FEP	CITE GSE FEP #1
CKUB	FEP	CITE KUB AND FEP
CPI	FEP	CITE POCC INTERFACE FEP
GPCA	FEP	128KB GPC PCM FEP - ACTIVE
GPCS	FEP	128KB GPC PCM FEP - STANDBY

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
GS1A	FEP	GSE FEP #1 - ACTIVE
GS1S	FEP	GSE FEP #1 - STANDBY
GS2A	FEP	GSE FEP #2 - ACTIVE
GS2S	FEP	GSE FEP #2 - STANDBY
GS3	FEP	GSE FEP #3
GS4	FEP	GSE FEP #4
GS5	FEP	GSE FEP #5
HOSC	FEP	HUNTSVILLE OPERATIONS SUPPORT CENTER
LDBA	FEP	LAUNCH DATA BUS FEP - ACTIVE
LDBD	FEP	LAUNCH DATA BUS FEP - DUAL
LDBS	FEP	LAUNCH DATA BUS - STANDBY
ME1	FEP	MAIN ENGINE #1 60KB OFI PCM FEP
ME2	FEP	MAIN ENGINE #2 60KB OFI PCM FEP
ME3	FEP	MAIN ENGINE #3 60KB OFI PCM FEP
OIA	FEP	128KB OI PCM FEP - ACTIVE
OIS	FEP	128KB OI PCM FEP - STANDBY
PLD	FEP	PAYLOAD FEP
SDB1	FEP	SRB DATA BUS FEP 1
SDB2	FEP	SRB DATA BUS FEP 2
SWDF	FEP	SWINGER PCM FEP 128KB OFI PCM DATA
SWOF	FEP	SWINGER PCM FEP 64KB OFI PCM DATA
UCS	FEP	UTILITY CONTROL SET FEP
UPLK	FEP	UPLINK COMMAND PCM FEP
TYPE = FILE		
CDS-RCDRR	FILE	CDS RECORDER RAW DATA
DISK-F1	FILE	DISK FILE 1
DISK-F2	FILE	DISK FILE 2
DISK-F3	FILE	DISK FILE 3
DISK-F4	FILE	DISK FILE 4
DISK-F5	FILE	DISK FILE 5
DISK-F6	FILE	DISK FILE 6
PDR-RCDRR	FILE	PDR RECORDER RAW DATA
TYPE = FPG1		
TCIDNAME	FPG1	TCID NAME
BFRREFDES	FPG1	CDBFR REFDES INDICATOR
TYPE = GMT		
GMT	GMT	GREENWICH MEAN TIME

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
TYPE = GMTD		
GMTDELGPCA	GMTD	GMT DELTA FROM GPC ACTIVE
GMTDELGPCS	GMTD	GMT DELTA FROM GPC SPARE
GMTDELOIA	GMTD	GMT DELTA FROM OI ACTIVE
GMTDELOIS	GMTD	GMT DELTA FROM OI STANDBY
TYPE = LED		
LED1	LED	LED 1 DEFAULT
LED1-P1	LED	LED 1 PRESENTATION 1
LED1-P2	LED	LED 1 PRESENTATION 2
LED1-P3	LED	LED 1 PRESENTATION 3
LED1-P4	LED	LED 1 PRESENTATION 4
LED1-P5	LED	LED 1 PRESENTATION 5
LED1-P6	LED	LED 1 PRESENTATION 6
LED2	LED	LED 2 DEFAULT
LED2-P1	LED	LED 2 PRESENTATION 1
LED2-P2	LED	LED 2 PRESENTATION 2
LED2-P3	LED	LED 2 PRESENTATION 3
LED2-P4	LED	LED 2 PRESENTATION 4
LED2-P5	LED	LED 2 PRESENTATION 5
LED2-P6	LED	LED 2 PRESENTATION 6
LED3	LED	LED 3 DEFAULT
LED3-P1	LED	LED 3 PRESENTATION 1
LED3-P2	LED	LED 3 PRESENTATION 2
LED3-P3	LED	LED 3 PRESENTATION 3
LED3-P4	LED	LED 3 PRESENTATION 4
LED3-P5	LED	LED 3 PRESENTATION 5
LED3-P6	LED	LED 3 PRESENTATION 6
LED4	LED	LED 4 DEFAULT
LED4-P1	LED	LED 4 PRESENTATION 1
LED4-P2	LED	LED 4 PRESENTATION 2
LED4-P3	LED	LED 4 PRESENTATION 3
LED4-P4	LED	LED 4 PRESENTATION 4
LED4-P5	LED	LED 4 PRESENTATION 5
LED4-P6	LED	LED 4 PRESENTATION 6
LED5	LED	LED 5 DEFAULT
LED5-P1	LED	LED 5 PRESENTATION 1
LED5-P2	LED	LED 5 PRESENTATION 2
LED5-P3	LED	LED 5 PRESENTATION 3
LED5-P4	LED	LED 5 PRESENTATION 4
LED5-P5	LED	LED 5 PRESENTATION 5

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
LED5-P6	LED	LED 5 PRESENTATION 6
LED6	LED	LED 6 DEFAULT
LED6-P1	LED	LED 6 PRESENTATION 1
LED6-P2	LED	LED 6 PRESENTATION 2
LED6-P3	LED	LED 6 PRESENTATION 3
LED6-P4	LED	LED 6 PRESENTATION 4
LED6-P5	LED	LED 6 PRESENTATION 5
LED6-P6	LED	LED 6 PRESENTATION 6
TYPE = OCRT		
GPCCRT	OCRT	ONBOARD GPC CRT DISPLAY
TYPE = PA		
DITVARTIME	PA	DIT VARIABLE START TIME
TYPE = PAGE		
PAGE-A	PAGE	DISPLAY APPLICATION PAGE A
PAGE-B	PAGE	DISPLAY APPLICATION PAGE B
PAGE-SA	PAGE	DISPLAY SHARED APPLICATION PAGE
PAGE-1A	PAGE	DISPLAY APPLICATION PAGE 1A
PAGE-1B	PAGE	DISPLAY APPLICATION PAGE 1B
PAGE-2A	PAGE	DISPLAY APPLICATION PAGE 2A
PAGE-2B	PAGE	DISPLAY APPLICATION PAGE 2B
PAGE-3A	PAGE	DISPLAY APPLICATION PAGE 3A
PAGE-3B	PAGE	DISPLAY APPLICATION PAGE 3B
PAGE-4A	PAGE	DISPLAY APPLICATION PAGE 4A
PAGE-4B	PAGE	DISPLAY APPLICATION PAGE 4B
PAGE-5A	PAGE	DISPLAY APPLICATION PAGE 5A
PAGE-5B	PAGE	DISPLAY APPLICATION PAGE 5B
PAGE-6A	PAGE	DISPLAY APPLICATION PAGE 6A
PAGE-6B	PAGE	DISPLAY APPLICATION PAGE 6B
TYPE = PD		
DITABORT	PD	DIT ABORT
DITCRIT1	PD	CRITICAL FUNCTION 1
DITCRIT2	PD	CRITICAL FUNCTION 2
DITCRIT3	PD	CRITICAL FUNCTION 3
DITCRIT4	PD	CRITICAL FUNCTION 4
DITCRIT5	PD	CRITICAL FUNCTION 5
DITCRIT6	PD	CRITICAL FUNCTION 6

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
DITCRIT7	PD	CRITICAL FUNCTION 7
DITCRIT8	PD	CRITICAL FUNCTION 8
DITCRIT9	PD	CRITICAL FUNCTION 9
DITCRIT10	PD	CRITICAL FUNCTION 10
DITCRIT11	PD	CRITICAL FUNCTION 11
DITCRIT12	PD	CRITICAL FUNCTION 12
DITCRIT13	PD	CRITICAL FUNCTION 13
DITCRIT14	PD	CRITICAL FUNCTION 14
DITCRIT15	PD	CRITICAL FUNCTION 15
DITCRIT16	PD	CRITICAL FUNCTION 16
DITGPCINIT	PD	DIT GPC INIT
DITPRINT	PD	DIT PRINT
DITVARFLAG	PD	DIT VARIABLE START TIME INDICATOR
NCPRSCDL1	PD	CITE DOWNLINK FEP CHECKPOINT STATUS
NCPRSCGS1	PD	CITE GSE FEP CHECKPOINT STATUS
NCPRSCUL1	PD	CITE UPLINK FEP CHECKPOINT STATUS
NCPRSGPCA	PD	GPC FEP CHECKPOINT STATUS
NCPRSGS1A	PD	GSE FEP #1 CHECKPOINT STATUS
NCPRSGS2A	PD	GSE FEP #2 CHECKPOINT STATUS
NCPRSGS3	PD	GSE FEP #3 CHECKPOINT STATUS
NCPRSGS4	PD	GSE FEP #4 CHECKPOINT STATUS
NCPRSGS5	PD	GSE FEP #5 CHECKPOINT STATUS
NCPRSLDBA	PD	LDB FEP CHECKPOINT STATUS
NCPRSLDBD	PD	LDBD FEP CHECKPOINT STATUS
NCPRSME1	PD	ME1 FEP CHECKPOINT STATUS
NCPRSME2	PD	ME2 FEP CHECKPOINT STATUS
NCPRSME3	PD	ME3 FEP CHECKPOINT STATUS
NCPRSOIA	PD	OI FEP CHECKPOINT STATUS
NCPRSPLD	PD	PLD FEP CHECKPOINT STATUS
NCPRSSDB1	PD	SDB1 FEP CHECKPOINT STATUS
NCPRSSDB2	PD	SDB2 FEP CHECKPOINT STATUS
NCPRSTATUS	PD	CHECKPOINT RESTART STATUS
NCPRSUCS	PD	UCS FEP CHECKPOINT STATUS
NCPRSUPLK	PD	PLK FEP CHECKPOINT STATUS
TYPE - PDP		
DITCRITSEL	PDP	DIT CRIT 1 THRU 16 SELECT MASK
DITFILESEL	PDP	DIT FILE SELECT
DITSTATUS	PDP	DIT PHASE STATUS
DITTYPE	PDP	DIT SCENARIO TYPE
NCPRCONTRL	PDP	CHECKPOINT RESTART CONTROL
NCPRRCDL1	PDP	CITE DOWNLINK FEP REQUEST OPTION

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
NCPRRCGS1	PDP	CITE GSE FEP REQUEST OPTION
NCPRRCUL1	PDP	CITE UPLINK FEP REQUEST OPTION
NCPRRGPCA	PDP	GPC FEP REQUEST OPTION
NCPRRGS1A	PDP	GSE FEP #1 REQUEST OPTION
NCPRRGS2A	PDP	GSE FEP #2 REQUEST OPTION
NCPRRGS3	PDP	GSE FEP #3 REQUEST OPTION
NCPRRGS4	PDP	GSE FEP #4 REQUEST OPTION
NCPRRGS5	PDP	GSE FEP #5 REQUEST OPTION
NCPRRLDBA	PDP	LDB FEP REQUEST OPTION
NCPRRLDBD	PDP	LDBD FEP REQUEST OPTIONS
NCPRRME1	PDP	ME1 FEP REQUEST OPTION
NCPRRME2	PDP	ME2 FEP REQUEST OPTION
NCPRRME3	PDP	ME3 FEP REQUEST OPTION
NCPRROIA	PDP	OI FEP REQUEST OPTION
NCPRRPLD	PDP	PLD FEP REQUEST OPTIONS
NCPRRSWDF	PDP	SWDF FEP REQUEST OPTION
NCPRRSWOF	PDP	SWOF FEP REQUEST OPTION
NCPRRUCS	PDP	UCS FEP REQUEST OPTION
NCPRRUPLK	PDP	UPLK FEP REQUEST OPTION
NFEPLDB	PDP	LDB FEP ROUTING INDICATOR
NFSWMCNFG	PDP	FLIGHT SOFTWARE CONFIGURATION ID
NGPCBUSSEL	PDP	GPC BUS SELECT FOR DUAL MODE
NGPCLMCNFG	PDP	CURRENT LDB GPC MEMORY CONFIGURATION
NGPC1MCNFG	PDP	GPC1 MEM CONFIG AND STATUS
NGPC2MCNFG	PDP	GPC2 MEM CONFIG AND STATUS
NGPC3MCNFG	PDP	GPC3 MEM CONFIG AND STATUS
NGPC4MCNFG	PDP	GPC4 MEM CONFIG AND STATUS
NGPC5MCNFG	PDP	GPC5 MEM CONFIG AND STATUS
NMEPTCHBUF	PDP	ME PATCH BUFFER
NME1VALCK	PDP	MAIN ENGINE #1 PCM VALIDITY
NME2VALCK	PDP	MAIN ENGINE #2 PCM VALIDITY
NME3VALCK	PDP	MAIN ENGINE #3 PCM VALIDITY
NPRVUSTAT	PDP	CITE END-TO-END CNSL PREVU STAT
NSSCONTROL	PDP	PCM SOURCE SELECT CONTROL WORD
NSSFEPSTS	PDP	PCM SOURCE SELECT FEP STATUS
NSSPARMSTS	PDP	PCM SOURCE SELECT PARAMETER STATUS
NTCPCONTRL	PDP	TIME CRITICAL PATCHER CONTROL
NTCPPARM01	PDP	TIME CRITICAL PATCHER PARAMETER 1
NTCPPARM02	PDP	TIME CRITICAL PATCHER PARAMETER 2
NTCPPARM03	PDP	TIME CRITICAL PATCHER PARAMETER 3
NTCPPARM04	PDP	TIME CRITICAL PATCHER PARAMETER 4
NTCPPARM05	PDP	TIME CRITICAL PATCHER PARAMETER 5
NTCPPARM06	PDP	TIME CRITICAL PATCHER PARAMETER 6

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
NTCPPARM07	PDP	TIME CRITICAL PATCHER PARAMETER 7
NTCPPARM08	PDP	TIME CRITICAL PATCHER PARAMETER 8
NTCPPARM09	PDP	TIME CRITICAL PATCHER PARAMETER 9
NTCPPARM10	PDP	TIME CRITICAL PATCHER PARAMETER 10
NTOCALLOW	PDP	TEST OPERATIONS CHANGE ALLOW
NTOCSTATUS	PDP	TOC ACTIVE, NORMAL/ABNORMAL TERM
N72IV057D	PDP	OCF STATUS
N72IV058D	PDP	GPC/MMU ADDRESS OR PATCH ID
N72IV059D	PDP	C-TO-C EVENT CODE BLOCK
N72IV060D	PDP	GPC STATUS WORD
N72IV061D	PDP	NUMBER OF MISCOMPARES
TYPE = PDRR		
CDS-RCDR	PDRR	CDS RECORDER ASCII DATA
PDR-RCDR	PDRR	PDR RECORDER ASCII DATA
TYPE = PFK		
PFK1-PA	PFK	PROG. FUNCTION KEY 1 PAGE-A DEFAULT
PFK1-PB	PFK	PROG. FUNCTION KEY 1 PAGE-B DEFAULT
PFK1-P1A	PFK	PROG. FUNCTION KEY 1 PAGE-1A
PFK1-P1B	PFK	PROG. FUNCTION KEY 1 PAGE-1B
PFK1-P2A	PFK	PROG. FUNCTION KEY 1 PAGE-2A
PFK1-P2B	PFK	PROG. FUNCTION KEY 1 PAGE-2B
PFK1-P3A	PFK	PROG. FUNCTION KEY 1 PAGE-3A
PFK1-P3B	PFK	PROG. FUNCTION KEY 1 PAGE-3B
PFK1-P4A	PFK	PROG. FUNCTION KEY 1 PAGE-4A
PFK1-P4B	PFK	PROG. FUNCTION KEY 1 PAGE-4B
PFK1-P5A	PFK	PROG. FUNCTION KEY 1 PAGE-5A
PFK1-P5B	PFK	PROG. FUNCTION KEY 1 PAGE-5B
PFK1-P6A	PFK	PROG. FUNCTION KEY 1 PAGE-6A
PFK1-P6B	PFK	PROG. FUNCTION KEY 1 PAGE-6B
PFK2-PA	PFK	PROG. FUNCTION KEY 2 PAGE-A DEFAULT
PFK2-PB	PFK	PROG. FUNCTION KEY 2 PAGE-B DEFAULT
PFK2-P1A	PFK	PROG. FUNCTION KEY 2 PAGE-1A
PFK2-P1B	PFK	PROG. FUNCTION KEY 2 PAGE-1B
PFK2-P2A	PFK	PROG. FUNCTION KEY 2 PAGE-2A
PFK2-P2B	PFK	PROG. FUNCTION KEY 2 PAGE-2B
PFK2-P3A	PFK	PROG. FUNCTION KEY 2 PAGE-3A
PFK2-P3B	PFK	PROG. FUNCTION KEY 2 PAGE-3B
PFK2-P4A	PFK	PROG. FUNCTION KEY 2 PAGE-4A
PFK2-P4B	PFK	PROG. FUNCTION KEY 2 PAGE-4B
PFK2-P5A	PFK	PROG. FUNCTION KEY 2 PAGE-5A

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
PFK2-P5B	PFK	PROG. FUNCTION KEY 2 PAGE-5B
PFK2-P6A	PFK	PROG. FUNCTION KEY 2 PAGE-6A
PFK2-P6B	PFK	PROG. FUNCTION KEY 2 PAGE-6B
PFK3-PA	PFK	PROG. FUNCTION KEY 3 PAGE-A DEFAULT
PFK3-PB	PFK	PROG. FUNCTION KEY 3 PAGE-B DEFAULT
PFK3-P1A	PFK	PROG. FUNCTION KEY 3 PAGE-1A
PFK3-P1B	PFK	PROG. FUNCTION KEY 3 PAGE-1B
PFK3-P2A	PFK	PROG. FUNCTION KEY 3 PAGE-2A
PFK3-P2B	PFK	PROG. FUNCTION KEY 3 PAGE-2B
PFK3-P3A	PFK	PROG. FUNCTION KEY 3 PAGE-3A
PFK3-P3B	PFK	PROG. FUNCTION KEY 3 PAGE-3B
PFK3-P4A	PFK	PROG. FUNCTION KEY 3 PAGE-4A
PFK3-P4B	PFK	PROG. FUNCTION KEY 3 PAGE-4B
PFK3-P5A	PFK	PROG. FUNCTION KEY 3 PAGE-5A
PFK3-P5B	PFK	PROG. FUNCTION KEY 3 PAGE-5B
PFK3-P6A	PFK	PROG. FUNCTION KEY 3 PAGE-6A
PFK3-P6B	PFK	PROG. FUNCTION KEY 3 PAGE-6B
PFK4-PA	PFK	PROG. FUNCTION KEY 4 PAGE-A DEFAULT
PFK4-PB	PFK	PROG. FUNCTION KEY 4 PAGE-B DEFAULT
PFK4-P1A	PFK	PROG. FUNCTION KEY 4 PAGE-1A
PFK4-P1B	PFK	PROG. FUNCTION KEY 4 PAGE-1B
PFK4-P2A	PFK	PROG. FUNCTION KEY 4 PAGE-2A
PFK4-P2B	PFK	PROG. FUNCTION KEY 4 PAGE-2B
PFK4-P3A	PFK	PROG. FUNCTION KEY 4 PAGE-3A
PFK4-P3B	PFK	PROG. FUNCTION KEY 4 PAGE-3B
PFK4-P4A	PFK	PROG. FUNCTION KEY 4 PAGE-4A
PFK4-P4B	PFK	PROG. FUNCTION KEY 4 PAGE-4B
PFK4-P5A	PFK	PROG. FUNCTION KEY 4 PAGE-5A
PFK4-P5B	PFK	PROG. FUNCTION KEY 4 PAGE-5B
PFK4-P6A	PFK	PROG. FUNCTION KEY 4 PAGE-6A
PFK4-P6B	PFK	PROG. FUNCTION KEY 4 PAGE-6B
PFK5-PA	PFK	PROG. FUNCTION KEY 5 PAGE-A DEFAULT
PFK5-PB	PFK	PROG. FUNCTION KEY 5 PAGE-B DEFAULT
PFK5-P1A	PFK	PROG. FUNCTION KEY 5 PAGE-1A
PFK5-P1B	PFK	PROG. FUNCTION KEY 5 PAGE-1B
PFK5-P2A	PFK	PROG. FUNCTION KEY 5 PAGE-2A
PFK5-P2B	PFK	PROG. FUNCTION KEY 5 PAGE-2B
PFK5-P3A	PFK	PROG. FUNCTION KEY 5 PAGE-3A
PFK5-P3B	PFK	PROG. FUNCTION KEY 5 PAGE-3B
PFK5-P4A	PFK	PROG. FUNCTION KEY 5 PAGE-4A
PFK5-P4B	PFK	PROG. FUNCTION KEY 5 PAGE-4B
PFK5-P5A	PFK	PROG. FUNCTION KEY 5 PAGE-5A
PFK5-P5B	PFK	PROG. FUNCTION KEY 5 PAGE-5B
PFK5-P6A	PFK	PROG. FUNCTION KEY 5 PAGE-6A

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
PFK5-P6B	PFK	PROG. FUNCTION KEY 5 PAGE-6B
PFK6-PA	PFK	PROG. FUNCTION KEY 6 PAGE-A DEFAULT
PFK6-PB	PFK	PROG. FUNCTION KEY 6 PAGE-B DEFAULT
PFK6-P1A	PFK	PROG. FUNCTION KEY 6 PAGE-1A
PFK6-P1B	PFK	PROG. FUNCTION KEY 6 PAGE-1B
PFK6-P2A	PFK	PROG. FUNCTION KEY 6 PAGE-2A
PFK6-P2B	PFK	PROG. FUNCTION KEY 6 PAGE-2B
PFK6-P3A	PFK	PROG. FUNCTION KEY 6 PAGE-3A
PFK6-P3B	PFK	PROG. FUNCTION KEY 6 PAGE-3B
PFK6-P4A	PFK	PROG. FUNCTION KEY 6 PAGE-4A
PFK6-P4B	PFK	PROG. FUNCTION KEY 6 PAGE-4B
PFK6-P5A	PFK	PROG. FUNCTION KEY 6 PAGE-5A
PFK6-P5B	PFK	PROG. FUNCTION KEY 6 PAGE-5B
PFK6-P6A	PFK	PROG. FUNCTION KEY 6 PAGE-6A
PFK6-P6B	PFK	PROG. FUNCTION KEY 6 PAGE-6B
PFK7-PA	PFK	PROG. FUNCTION KEY 7 PAGE-A DEFAULT
PFK7-PB	PFK	PROG. FUNCTION KEY 7 PAGE-B DEFAULT
PFK7-P1A	PFK	PROG. FUNCTION KEY 7 PAGE-1A
PFK7-P1B	PFK	PROG. FUNCTION KEY 7 PAGE-1B
PFK7-P2A	PFK	PROG. FUNCTION KEY 7 PAGE-2A
PFK7-P2B	PFK	PROG. FUNCTION KEY 7 PAGE-2B
PFK7-P3A	PFK	PROG. FUNCTION KEY 7 PAGE-3A
PFK7-P3B	PFK	PROG. FUNCTION KEY 7 PAGE-3B
PFK7-P4A	PFK	PROG. FUNCTION KEY 7 PAGE-4A
PFK7-P4B	PFK	PROG. FUNCTION KEY 7 PAGE-4B
PFK7-P5A	PFK	PROG. FUNCTION KEY 7 PAGE-5A
PFK7-P5B	PFK	PROG. FUNCTION KEY 7 PAGE-5B
PFK7-P6A	PFK	PROG. FUNCTION KEY 7 PAGE-6A
PFK7-P6B	PFK	PROG. FUNCTION KEY 7 PAGE-6B
PFK8-PA	PFK	PROG. FUNCTION KEY 8 PAGE-A DEFAULT
PFK8-PB	PFK	PROG. FUNCTION KEY 8 PAGE-B DEFAULT
PFK8-P1A	PFK	PROG. FUNCTION KEY 8 PAGE-1A
PFK8-P1B	PFK	PROG. FUNCTION KEY 8 PAGE-1B
PFK8-P2A	PFK	PROG. FUNCTION KEY 8 PAGE-2A
PFK8-P2B	PFK	PROG. FUNCTION KEY 8 PAGE-2B
PFK8-P3A	PFK	PROG. FUNCTION KEY 8 PAGE-3A
PFK8-P3B	PFK	PROG. FUNCTION KEY 8 PAGE-3B
PFK8-P4A	PFK	PROG. FUNCTION KEY 8 PAGE-4A
PFK8-P4B	PFK	PROG. FUNCTION KEY 8 PAGE-4B
PFK8-P5A	PFK	PROG. FUNCTION KEY 8 PAGE-5A
PFK8-P5B	PFK	PROG. FUNCTION KEY 8 PAGE-5B
PFK8-P6A	PFK	PROG. FUNCTION KEY 8 PAGE-6A
PFK8-P6B	PFK	PROG. FUNCTION KEY 8 PAGE-6B
PFK9-PA	PFK	PROG. FUNCTION KEY 9 PAGE-A DEFAULT

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
PFK9-PB	PFK	PROG. FUNCTION KEY 9 PAGE-B DEFAULT
PFK9-P1A	PFK	PROG. FUNCTION KEY 9 PAGE-1A
PFK9-P1B	PFK	PROG. FUNCTION KEY 9 PAGE-1B
PFK9-P2A	PFK	PROG. FUNCTION KEY 9 PAGE-2A
PFK9-P2B	PFK	PROG. FUNCTION KEY 9 PAGE-2B
PFK9-P3A	PFK	PROG. FUNCTION KEY 9 PAGE-3A
PFK9-P3B	PFK	PROG. FUNCTION KEY 9 PAGE-3B
PFK9-P4A	PFK	PROG. FUNCTION KEY 9 PAGE-4A
PFK9-P4B	PFK	PROG. FUNCTION KEY 9 PAGE-4B
PFK9-P5A	PFK	PROG. FUNCTION KEY 9 PAGE-5A
PFK9-P5B	PFK	PROG. FUNCTION KEY 9 PAGE-5B
PFK9-P6A	PFK	PROG. FUNCTION KEY 9 PAGE-6A
PFK9-P6B	PFK	PROG. FUNCTION KEY 9 PAGE-6B
PFK10-PA	PFK	PROG. FUNCTION KEY 10 PAGE-A DEFAULT
PFK10-PB	PFK	PROG. FUNCTION KEY 10 PAGE-B DEFAULT
PFK10-P1A	PFK	PROG. FUNCTION KEY 10 PAGE-1A
PFK10-P1B	PFK	PROG. FUNCTION KEY 10 PAGE-1B
PFK10-P2A	PFK	PROG. FUNCTION KEY 10 PAGE-2A
PFK10-P2B	PFK	PROG. FUNCTION KEY 10 PAGE-2B
PFK10-P3A	PFK	PROG. FUNCTION KEY 10 PAGE-3A
PFK10-P3B	PFK	PROG. FUNCTION KEY 10 PAGE-3B
PFK10-P4A	PFK	PROG. FUNCTION KEY 10 PAGE-4A
PFK10-P4B	PFK	PROG. FUNCTION KEY 10 PAGE-4B
PFK10-P5A	PFK	PROG. FUNCTION KEY 10 PAGE-5A
PFK10-P5B	PFK	PROG. FUNCTION KEY 10 PAGE-5B
PFK10-P6A	PFK	PROG. FUNCTION KEY 10 PAGE-6A
PFK10-P6B	PFK	PROG. FUNCTION KEY 10 PAGE-6B
PFK11-PA	PFK	PROG. FUNCTION KEY 11 PAGE-A DEFAULT
PFK11-PB	PFK	PROG. FUNCTION KEY 11 PAGE-B DEFAULT
PFK11-P1A	PFK	PROG. FUNCTION KEY 11 PAGE-1A
PFK11-P1B	PFK	PROG. FUNCTION KEY 11 PAGE-1B
PFK11-P2A	PFK	PROG. FUNCTION KEY 11 PAGE-2A
PFK11-P2B	PFK	PROG. FUNCTION KEY 11 PAGE-2B
PFK11-P3A	PFK	PROG. FUNCTION KEY 11 PAGE-3A
PFK11-P3B	PFK	PROG. FUNCTION KEY 11 PAGE-3B
PFK11-P4A	PFK	PROG. FUNCTION KEY 11 PAGE-4A
PFK11-P4B	PFK	PROG. FUNCTION KEY 11 PAGE-4B
PFK11-P5A	PFK	PROG. FUNCTION KEY 11 PAGE-5A
PFK11-P5B	PFK	PROG. FUNCTION KEY 11 PAGE-5B
PFK11-P6A	PFK	PROG. FUNCTION KEY 11 PAGE-6A
PFK11-P6B	PFK	PROG. FUNCTION KEY 11 PAGE-6B
PFK12-PA	PFK	PROG. FUNCTION KEY 12 PAGE-A DEFAULT
PFK12-PB	PFK	PROG. FUNCTION KEY 12 PAGE-B DEFAULT
PFK12-P1A	PFK	PROG. FUNCTION KEY 12 PAGE-1A

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
PFK12-P1B	PFK	PROG. FUNCTION KEY 12 PAGE-1B
PFK12-P2A	PFK	PROG. FUNCTION KEY 12 PAGE-2A
PFK12-P2B	PFK	PROG. FUNCTION KEY 12 PAGE-2B
PFK12-P3A	PFK	PROG. FUNCTION KEY 12 PAGE-3A
PFK12-P3B	PFK	PROG. FUNCTION KEY 12 PAGE-3B
PFK12-P4A	PFK	PROG. FUNCTION KEY 12 PAGE-4A
PFK12-P4B	PFK	PROG. FUNCTION KEY 12 PAGE-4B
PFK12-P5A	PFK	PROG. FUNCTION KEY 12 PAGE-5A
PFK12-P5B	PFK	PROG. FUNCTION KEY 12 PAGE-5B
PFK12-P6A	PFK	PROG. FUNCTION KEY 12 PAGE-6A
PFK12-P6B	PFK	PROG. FUNCTION KEY 12 PAGE-6B
PFK13-PA	PFK	PROG. FUNCTION KEY 13 PAGE-A DEFAULT
PFK13-PB	PFK	PROG. FUNCTION KEY 13 PAGE-B DEFAULT
PFK13-P1A	PFK	PROG. FUNCTION KEY 13 PAGE-1A
PFK13-P1B	PFK	PROG. FUNCTION KEY 13 PAGE-1B
PFK13-P2A	PFK	PROG. FUNCTION KEY 13 PAGE-2A
PFK13-P2B	PFK	PROG. FUNCTION KEY 13 PAGE-2B
PFK13-P3A	PFK	PROG. FUNCTION KEY 13 PAGE-3A
PFK13-P3B	PFK	PROG. FUNCTION KEY 13 PAGE-3B
PFK13-P4A	PFK	PROG. FUNCTION KEY 13 PAGE-4A
PFK13-P4B	PFK	PROG. FUNCTION KEY 13 PAGE-4B
PFK13-P5A	PFK	PROG. FUNCTION KEY 13 PAGE-5A
PFK13-P5B	PFK	PROG. FUNCTION KEY 13 PAGE-5B
PFK13-P6A	PFK	PROG. FUNCTION KEY 13 PAGE-6A
PFK13-P6B	PFK	PROG. FUNCTION KEY 13 PAGE-6B
PFK14-PA	PFK	PROG. FUNCTION KEY 14 PAGE-A DEFAULT
PFK14-PB	PFK	PROG. FUNCTION KEY 14 PAGE-B DEFAULT
PFK14-P1A	PFK	PROG. FUNCTION KEY 14 PAGE-1A
PFK14-P1B	PFK	PROG. FUNCTION KEY 14 PAGE-1B
PFK14-P2A	PFK	PROG. FUNCTION KEY 14 PAGE-2A
PFK14-P2B	PFK	PROG. FUNCTION KEY 14 PAGE-2B
PFK14-P3A	PFK	PROG. FUNCTION KEY 14 PAGE-3A
PFK14-P3B	PFK	PROG. FUNCTION KEY 14 PAGE-3B
PFK14-P4A	PFK	PROG. FUNCTION KEY 14 PAGE-4A
PFK14-P4B	PFK	PROG. FUNCTION KEY 14 PAGE-4B
PFK14-P5A	PFK	PROG. FUNCTION KEY 14 PAGE-5A
PFK14-P5B	PFK	PROG. FUNCTION KEY 14 PAGE-5B
PFK14-P6A	PFK	PROG. FUNCTION KEY 14 PAGE-6A
PFK14-P6B	PFK	PROG. FUNCTION KEY 14 PAGE-6B
PFK15-PA	PFK	PROG. FUNCTION KEY 15 PAGE-A DEFAULT
PFK15-PB	PFK	PROG. FUNCTION KEY 15 PAGE-B DEFAULT
PFK15-P1A	PFK	PROG. FUNCTION KEY 15 PAGE-1A
PFK15-P1B	PFK	PROG. FUNCTION KEY 15 PAGE-1B
PFK15-P2A	PFK	PROG. FUNCTION KEY 15 PAGE-2A

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
PFK15-P2B	PFK	PROG. FUNCTION KEY 15 PAGE-2B
PFK15-P3A	PFK	PROG. FUNCTION KEY 15 PAGE-3A
PFK15-P3B	PFK	PROG. FUNCTION KEY 15 PAGE-3B
PFK15-P4A	PFK	PROG. FUNCTION KEY 15 PAGE-4A
PFK15-P4B	PFK	PROG. FUNCTION KEY 15 PAGE-4B
PFK15-P5A	PFK	PROG. FUNCTION KEY 15 PAGE-5A
PFK15-P5B	PFK	PROG. FUNCTION KEY 15 PAGE-5B
PFK15-P6A	PFK	PROG. FUNCTION KEY 15 PAGE-6A
PFK15-P6B	PFK	PROG. FUNCTION KEY 15 PAGE-6B

TYPE = PFPK

PFPK1	PFPK	PFP KEY 1 DEFAULT
PFPK1-P1	PFPK	PFP KEY 1 PRESENTATION 1
PFPK1-P2	PFPK	PFP KEY 1 PRESENTATION 2
PFPK1-P3	PFPK	PFP KEY 1 PRESENTATION 3
PFPK1-P4	PFPK	PFP KEY 1 PRESENTATION 4
PFPK1-P5	PFPK	PFP KEY 1 PRESENTATION 5
PFPK1-P6	PFPK	PFP KEY 1 PRESENTATION 6
PFPK2	PFPK	PFP KEY 2 DEFAULT
PFPK2-P1	PFPK	PFP KEY 2 PRESENTATION 1
PFPK2-P2	PFPK	PFP KEY 2 PRESENTATION 2
PFPK2-P3	PFPK	PFP KEY 2 PRESENTATION 3
PFPK2-P4	PFPK	PFP KEY 2 PRESENTATION 4
PFPK2-P5	PFPK	PFP KEY 2 PRESENTATION 5
PFPK2-P6	PFPK	PFP KEY 2 PRESENTATION 6
PFPK3	PFPK	PFP KEY 3 DEFAULT
PFPK3-P1	PFPK	PFP KEY 3 PRESENTATION 1
PFPK3-P2	PFPK	PFP KEY 3 PRESENTATION 2
PFPK3-P3	PFPK	PFP KEY 3 PRESENTATION 3
PFPK3-P4	PFPK	PFP KEY 3 PRESENTATION 4
PFPK3-P5	PFPK	PFP KEY 3 PRESENTATION 5
PFPK3-P6	PFPK	PFP KEY 3 PRESENTATION 6
PFPK4	PFPK	PFP KEY 4 DEFAULT
PFPK4-P1	PFPK	PFP KEY 4 PRESENTATION 1
PFPK4-P2	PFPK	PFP KEY 4 PRESENTATION 2
PFPK4-P3	PFPK	PFP KEY 4 PRESENTATION 3
PFPK4-P4	PFPK	PFP KEY 4 PRESENTATION 4
PFPK4-P5	PFPK	PFP KEY 4 PRESENTATION 5
PFPK4-P6	PFPK	PFP KEY 4 PRESENTATION 6
PFPK5	PFPK	PFP KEY 5 DEFAULT
PFPK5-P1	PFPK	PFP KEY 5 PRESENTATION 1
PFPK5-P2	PFPK	PFP KEY 5 PRESENTATION 2
PFPK5-P3	PFPK	PFP KEY 5 PRESENTATION 3

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
PFPK5-P4	PFPK	PFP KEY 5 PRESENTATION 4
PFPK5-P5	PFPK	PFP KEY 5 PRESENTATION 5
PFPK5-P6	PFPK	PFP KEY 5 PRESENTATION 6
PFPK6	PFPK	PFP KEY 6 DEFAULT
PFPK6-P1	PFPK	PFP KEY 6 PRESENTATION 1
PFPK6-P2	PFPK	PFP KEY 6 PRESENTATION 2
PFPK6-P3	PFPK	PFP KEY 6 PRESENTATION 3
PFPK6-P4	PFPK	PFP KEY 6 PRESENTATION 4
PFPK6-P5	PFPK	PFP KEY 6 PRESENTATION 5
PFPK6-P6	PFPK	PFP KEY 6 PRESENTATION 6
PFPK7	PFPK	PFP KEY 7 DEFAULT
PFPK7-P1	PFPK	PFP KEY 7 PRESENTATION 1
PFPK7-P2	PFPK	PFP KEY 7 PRESENTATION 2
PFPK7-P3	PFPK	PFP KEY 7 PRESENTATION 3
PFPK7-P4	PFPK	PFP KEY 7 PRESENTATION 4
PFPK7-P5	PFPK	PFP KEY 7 PRESENTATION 5
PFPK7-P6	PFPK	PFP KEY 7 PRESENTATION 6
TDCDTCOUNT	PFPK	GOAL/CDT RESUME REQUEST
TDCDTHOLD	PFPK	GOAL/CDT HOLD REQUEST

TYPE = PFPL

PFPK1-L1	PFPL	PFP KEY 1 LIGHT 1 DEFAULT
PFPK1-L1P1	PFPL	PFP KEY 1 LIGHT 1 PRESENTATION 1
PFPK1-L1P2	PFPL	PFP KEY 1 LIGHT 1 PRESENTATION 2
PFPK1-L1P3	PFPL	PFP KEY 1 LIGHT 1 PRESENTATION 3
PFPK1-L1P4	PFPL	PFP KEY 1 LIGHT 1 PRESENTATION 4
PFPK1-L1P5	PFPL	PFP KEY 1 LIGHT 1 PRESENTATION 5
PFPK1-L1P6	PFPL	PFP KEY 1 LIGHT 1 PRESENTATION 6
PFPK1-L2	PFPL	PFP KEY 1 LIGHT 2 DEFAULT
PFPK1-L2P1	PFPL	PFP KEY 1 LIGHT 2 PRESENTATION 1
PFPK1-L2P2	PFPL	PFP KEY 1 LIGHT 2 PRESENTATION 2
PFPK1-L2P3	PFPL	PFP KEY 1 LIGHT 2 PRESENTATION 3
PFPK1-L2P4	PFPL	PFP KEY 1 LIGHT 2 PRESENTATION 4
PFPK1-L2P5	PFPL	PFP KEY 1 LIGHT 2 PRESENTATION 5
PFPK1-L2P6	PFPL	PFP KEY 1 LIGHT 2 PRESENTATION 6
PFPK2-L1	PFPL	PFP KEY 2 LIGHT 1 DEFAULT
PFPK2-L1P1	PFPL	PFP KEY 2 LIGHT 1 PRESENTATION 1
PFPK2-L1P2	PFPL	PFP KEY 2 LIGHT 1 PRESENTATION 2
PFPK2-L1P3	PFPL	PFP KEY 2 LIGHT 1 PRESENTATION 3
PFPK2-L1P4	PFPL	PFP KEY 2 LIGHT 1 PRESENTATION 4
PFPK2-L1P5	PFPL	PFP KEY 2 LIGHT 1 PRESENTATION 5
PFPK2-L1P6	PFPL	PFP KEY 2 LIGHT 1 PRESENTATION 6
PFPK2-L2	PFPL	PFP KEY 2 LIGHT 2 DEFAULT

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
PFPK2-L2P1	PFPL	PFP KEY 2 LIGHT 2 PRESENTATION 1
PFPK2-L2P2	PFPL	PFP KEY 2 LIGHT 2 PRESENTATION 2
PFPK2-L2P3	PFPL	PFP KEY 2 LIGHT 2 PRESENTATION 3
PFPK2-L2P4	PFPL	PFP KEY 2 LIGHT 2 PRESENTATION 4
PFPK2-L2P5	PFPL	PFP KEY 2 LIGHT 2 PRESENTATION 5
PFPK2-L2P6	PFPL	PFP KEY 2 LIGHT 2 PRESENTATION 6
PFPK3-L1	PFPL	PFP KEY 3 LIGHT 1 DEFAULT
PFPK3-L1P1	PFPL	PFP KEY 3 LIGHT 1 PRESENTATION 1
PFPK3-L1P2	PFPL	PFP KEY 3 LIGHT 1 PRESENTATION 2
PFPK3-L1P3	PFPL	PFP KEY 3 LIGHT 1 PRESENTATION 3
PFPK3-L1P4	PFPL	PFP KEY 3 LIGHT 1 PRESENTATION 4
PFPK3-L1P5	PFPL	PFP KEY 3 LIGHT 1 PRESENTATION 5
PFPK3-L1P6	PFPL	PFP KEY 3 LIGHT 1 PRESENTATION 6
PFPK3-L2	PFPL	PFP KEY 3 LIGHT 2 DEFAULT
PFPK3-L2P1	PFPL	PFP KEY 3 LIGHT 2 PRESENTATION 1
PFPK3-L2P2	PFPL	PFP KEY 3 LIGHT 2 PRESENTATION 2
PFPK3-L2P3	PFPL	PFP KEY 3 LIGHT 2 PRESENTATION 3
PFPK3-L2P4	PFPL	PFP KEY 3 LIGHT 2 PRESENTATION 4
PFPK3-L2P5	PFPL	PFP KEY 3 LIGHT 2 PRESENTATION 5
PFPK3-L2P6	PFPL	PFP KEY 3 LIGHT 2 PRESENTATION 6
PFPK4-L1	PFPL	PFP KEY 4 LIGHT 1 DEFAULT
PFPK4-L1P1	PFPL	PFP KEY 4 LIGHT 1 PRESENTATION 1
PFPK4-L1P2	PFPL	PFP KEY 4 LIGHT 1 PRESENTATION 2
PFPK4-L1P3	PFPL	PFP KEY 4 LIGHT 1 PRESENTATION 3
PFPK4-L1P4	PFPL	PFP KEY 4 LIGHT 1 PRESENTATION 4
PFPK4-L1P5	PFPL	PFP KEY 4 LIGHT 1 PRESENTATION 5
PFPK4-L1P6	PFPL	PFP KEY 4 LIGHT 1 PRESENTATION 6
PFPK4-L2	PFPL	PFP KEY 4 LIGHT 2 DEFAULT
PFPK4-L2P1	PFPL	PFP KEY 4 LIGHT 2 PRESENTATION 1
PFPK4-L2P2	PFPL	PFP KEY 4 LIGHT 2 PRESENTATION 2
PFPK4-L2P3	PFPL	PFP KEY 4 LIGHT 2 PRESENTATION 3
PFPK4-L2P4	PFPL	PFP KEY 4 LIGHT 2 PRESENTATION 4
PFPK4-L2P5	PFPL	PFP KEY 4 LIGHT 2 PRESENTATION 5
PFPK4-L2P6	PFPL	PFP KEY 4 LIGHT 2 PRESENTATION 6
PFPK5-L1	PFPL	PFP KEY 5 LIGHT 1 DEFAULT
PFPK5-L1P1	PFPL	PFP KEY 5 LIGHT 1 PRESENTATION 1
PFPK5-L1P2	PFPL	PFP KEY 5 LIGHT 1 PRESENTATION 2
PFPK5-L1P3	PFPL	PFP KEY 5 LIGHT 1 PRESENTATION 3
PFPK5-L1P4	PFPL	PFP KEY 5 LIGHT 1 PRESENTATION 4
PFPK5-L1P5	PFPL	PFP KEY 5 LIGHT 1 PRESENTATION 5
PFPK5-L1P6	PFPL	PFP KEY 5 LIGHT 1 PRESENTATION 6
PFPK5-L2	PFPL	PFP KEY 5 LIGHT 2 DEFAULT
PFPK5-L2P1	PFPL	PFP KEY 5 LIGHT 2 PRESENTATION 1
PFPK5-L2P2	PFPL	PFP KEY 5 LIGHT 2 PRESENTATION 2

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
PFPK5-L2P3	PFPL	PFP KEY 5 LIGHT 2 PRESENTATION 3
PFPK5-L2P4	PFPL	PFP KEY 5 LIGHT 2 PRESENTATION 4
PFPK5-L2P5	PFPL	PFP KEY 5 LIGHT 2 PRESENTATION 5
PFPK5-L2P6	PFPL	PFP KEY 5 LIGHT 2 PRESENTATION 6
PFPK6-L1	PFPL	PFP KEY 6 LIGHT 1 DEFAULT
PFPK6-L1P1	PFPL	PFP KEY 6 LIGHT 1 PRESENTATION 1
PFPK6-L1P2	PFPL	PFP KEY 6 LIGHT 1 PRESENTATION 2
PFPK6-L1P3	PFPL	PFP KEY 6 LIGHT 1 PRESENTATION 3
PFPK6-L1P4	PFPL	PFP KEY 6 LIGHT 1 PRESENTATION 4
PFPK6-L1P5	PFPL	PFP KEY 6 LIGHT 1 PRESENTATION 5
PFPK6-L1P6	PFPL	PFP KEY 6 LIGHT 1 PRESENTATION 6
PFPK6-L2	PFPL	PFP KEY 6 LIGHT 2 DEFAULT
PFPK6-L2P1	PFPL	PFP KEY 6 LIGHT 2 PRESENTATION 1
PFPK6-L2P2	PFPL	PFP KEY 6 LIGHT 2 PRESENTATION 2
PFPK6-L2P3	PFPL	PFP KEY 6 LIGHT 2 PRESENTATION 3
PFPK6-L2P4	PFPL	PFP KEY 6 LIGHT 2 PRESENTATION 4
PFPK6-L2P5	PFPL	PFP KEY 6 LIGHT 2 PRESENTATION 5
PFPK6-L2P6	PFPL	PFP KEY 6 LIGHT 2 PRESENTATION 6
TYPE = PRTR		
SPA-PRNTR	PRTR	SPA PRINTER
TYPE = SCON		
SPA	SCON	SPA CONSOLE
TYPE = SI		
SCT1	SI	SCT CHANGE SYS. INTERRUPT LPORT1
SCT2	SI	SCT CHANGE SYS. INTERRUPT LPORT2
SCT3	SI	SCT CHANGE SYS. INTERRUPT LPORT3
SCT4	SI	SCT CHANGE SYS. INTERRUPT LPORT4
SCT5	SI	SCT CHANGE SYS. INTERRUPT LPORT5
SCT6	SI	SCT CHANGE SYS. INTERRUPT LPORT6
SCT7	SI	SCT CHANGE SYS. INTERRUPT LPORT7
SCT8	SI	SCT CHANGE SYS. INTERRUPT LPORT8
SCT9	SI	SCT CHANGE SYS. INTERRUPT LPORT9
SCT10	SI	SCT CHANGE SYS. INTERRUPT LPORT10
SCT11	SI	SCT CHANGE SYS. INTERRUPT LPORT11
SCT12	SI	SCT CHANGE SYS. INTERRUPT LPORT12
SCT13	SI	SCT CHANGE SYS. INTERRUPT LPORT13
SCT14	SI	SCT CHANGE SYS. INTERRUPT LPORT14
SCT15	SI	SCT CHANGE SYS. INTERRUPT LPORT15

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SCT16	SI	SCT CHANGE SYS. INTERRUPT LPORT16
SCT17	SI	SCT CHANGE SYS. INTERRUPT LPORT17
SCT18	SI	SCT CHANGE SYS. INTERRUPT LPORT18
SCT19	SI	SCT CHANGE SYS. INTERRUPT LPORT19
SCT20	SI	SCT CHANGE SYS. INTERRUPT LPORT20
SCT21	SI	SCT CHANGE SYS. INTERRUPT LPORT21
SCT22	SI	SCT CHANGE SYS. INTERRUPT LPORT22
SCT23	SI	SCT CHANGE SYS. INTERRUPT LPORT23
SCT24	SI	SCT CHANGE SYS. INTERRUPT LPORT24
SCT25	SI	SCT CHANGE SYS. INTERRUPT LPORT25
SCT26	SI	SCT CHANGE SYS. INTERRUPT LPORT26
SCT27	SI	SCT CHANGE SYS. INTERRUPT LPORT27
SCT28	SI	SCT CHANGE SYS. INTERRUPT LPORT28
SCT29	SI	SCT CHANGE SYS. INTERRUPT LPORT29
SCT30	SI	SCT CHANGE SYS. INTERRUPT LPORT30
SCT31	SI	SCT CHANGE SYS. INTERRUPT LPORT31
SCT32	SI	SCT CHANGE SYS. INTERRUPT LPORT32
SCT33	SI	SCT CHANGE SYS. INTERRUPT LPORT33
SCT34	SI	SCT CHANGE SYS. INTERRUPT LPORT34
SCT35	SI	SCT CHANGE SYS. INTERRUPT LPORT35
SCT36	SI	SCT CHANGE SYS. INTERRUPT LPORT36
SCT37	SI	SCT CHANGE SYS. INTERRUPT LPORT37
SCT38	SI	SCT CHANGE SYS. INTERRUPT LPORT38
SCT39	SI	SCT CHANGE SYS. INTERRUPT LPORT39
SCT40	SI	SCT CHANGE SYS. INTERRUPT LPORT40
SCT41	SI	SCT CHANGE SYS. INTERRUPT LPORT41
SCT42	SI	SCT CHANGE SYS. INTERRUPT LPORT42
SCT43	SI	SCT CHANGE SYS. INTERRUPT LPORT43
SCT44	SI	SCT CHANGE SYS. INTERRUPT LPORT44
SCT45	SI	SCT CHANGE SYS. INTERRUPT LPORT45
SCT46	SI	SCT CHANGE SYS. INTERRUPT LPORT46
SCT47	SI	SCT CHANGE SYS. INTERRUPT LPORT47
SCT48	SI	SCT CHANGE SYS. INTERRUPT LPORT48
SCT49	SI	SCT CHANGE SYS. INTERRUPT LPORT49
SCT50	SI	SCT CHANGE SYS. INTERRUPT LPORT50
SCT51	SI	SCT CHANGE SYS. INTERRUPT LPORT51
SCT52	SI	SCT CHANGE SYS. INTERRUPT LPORT52
SCT53	SI	SCT CHANGE SYS. INTERRUPT LPORT53
SCT54	SI	SCT CHANGE SYS. INTERRUPT LPORT54
SCT55	SI	SCT CHANGE SYS. INTERRUPT LPORT55
SCT56	SI	SCT CHANGE SYS. INTERRUPT LPORT56
SCT57	SI	SCT CHANGE SYS. INTERRUPT LPORT57
SCT58	SI	SCT CHANGE SYS. INTERRUPT LPORT58
SCT59	SI	SCT CHANGE SYS. INTERRUPT LPORT59

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SCT60	SI	SCT CHANGE SYS. INTERRUPT LPORT60
SCT61	SI	SCT CHANGE SYS. INTERRUPT LPORT61
SCT62	SI	SCT CHANGE SYS. INTERRUPT LPORT62
SCT63	SI	SCT CHANGE SYS. INTERRUPT LPORT63
SCT64	SI	SCT CHANGE SYS. INTERRUPT LPORT64
CDTCOUNT	SI	CDT STATUS CHANGE TO COUNT
CDTHOLD	SI	CDT STATUS CHANGE TO HOLD
DG-INT	SI	DG KEY FD TYPE SYSTEM INTERRUPT
ERROR3	SI	GOAL PROCEDURE CLASS 3 ERROR
MEAS-INT	SI	MEASUREMENT FD TYPE SYSTEM INTERRUPT
METCOUNT	SI	MET STATUS CHANGE TO COUNT
METHOLD	SI	MET STATUS CHANGE TO HOLD
PFP-INT	SI	PFP FD TYPE SYSTEM INTERRUPT
REMCOM-INT	SI	REMOTE COMMUNICATION FD TYPE SYSTEM INTERRUPT
SYS-INT	SI	SYSTEM FD TYPE SYSTEM INTERRUPT

TYPE = SSA1

SBKUPGO	SSA1	BACKUP TYPE II CONSOLE GO MODE
SBKUPSTAT	SSA1	BACKUP TYPE II CONSOLE STATUS
SBSDDATAV	SSA1	BSD PBIC DATA VALID
SBSDSTAT	SSA1	BSD PBIC STATUS
SCDL1AREA	SSA1	COMMON DOWNLINK AREA STATUS
SCDL1DATAV	SSA1	COMMON DOWNLINK DATA VALID
SCDL1STAT	SSA1	COMMON DOWNLINK STATUS
SCGS1DATAV	SSA1	CITE GSE DATA VALID
SCGS1STAT	SSA1	CITE GSE STATUS
SCPIDATAV	SSA1	CPI PROCESSOR DATA VALID
SCPISTAT	SSA1	CPI PROCESSOR ACTIVE STATUS
SC1GO	SSA1	OPERATIONS CONSOLE #1 GO MODE
SC1STAT	SSA1	OPERATIONS CONSOLE #1 STATUS
SC2GO	SSA1	OPERATIONS CONSOLE #2 GO MODE
SC2STAT	SSA1	OPERATIONS CONSOLE #2 STATUS
SC3GO	SSA1	OPERATIONS CONSOLE #3 GO MODE
SC3STAT	SSA1	OPERATIONS CONSOLE #3 STATUS
SC4GO	SSA1	OPERATIONS CONSOLE #4 GO MODE
SC4STAT	SSA1	OPERATIONS CONSOLE #4 STATUS
SC5GO	SSA1	OPERATIONS CONSOLE #5 GO MODE
SC5STAT	SSA1	OPERATIONS CONSOLE #5 STATUS
SC6GO	SSA1	OPERATIONS CONSOLE #6 GO MODE
SC6STAT	SSA1	OPERATIONS CONSOLE #6 STATUS
SC7GO	SSA1	OPERATIONS CONSOLE #7 GO MODE
SC7STAT	SSA1	OPERATIONS CONSOLE #7 STATUS

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SC8GO	SSA1	OPERATIONS CONSOLE #8 GO MODE
SC8STAT	SSA1	OPERATIONS CONSOLE #8 STATUS
SC9GO	SSA1	OPERATIONS CONSOLE #9 GO MODE
SC9STAT	SSA1	OPERATIONS CONSOLE #9 STATUS
SC10GO	SSA1	OPERATIONS CONSOLE #10 GO MODE
SC10STAT	SSA1	OPERATIONS CONSOLE #10 STATUS
SC11GO	SSA1	OPERATIONS CONSOLE #11 GO MODE
SC11STAT	SSA1	OPERATIONS CONSOLE #11 STATUS
SC12GO	SSA1	OPERATIONS CONSOLE #12 GO MODE
SC12STAT	SSA1	OPERATIONS CONSOLE #12 STATUS
SGPCADATAV	SSA1	GPC FEP ACTIVE DATA VALID
SGPCAREAOI	SSA1	GPC FEP OI AREA STATUS
SGPCAREA1	SSA1	GPC FEP AREA 1 STATUS
SGPCAREA2	SSA1	GPC FEP AREA 2 STATUS
SGPCAREA3	SSA1	GPC FEP AREA 3 STATUS
SGPCAREA4	SSA1	GPC FEP AREA 4 STATUS
SGPCASTAT	SSA1	GPC FEP ACTIVE STATUS
SGPCDISWSQ	SSA1	LDBD GPC INVALID SWITCH SEQUENCE
SGPCDPOLL	SSA1	LDBD GPC POLLING
SGPCISWSEQ	SSA1	GPC INVALID SWITCH SEQUENCE
SGPCPOLL	SSA1	LDB GPC POLLING
SGPCSDATAV	SSA1	GPC FEP STANDBY DATA VALID
SGPCSSTAT	SSA1	GPC FEP STANDBY STATUS
SGS1ADATAV	SSA1	GSE FEP #1 ACTIVE DATA VALID
SGS1ASTAT	SSA1	GSE FEP #1 ACTIVE STATUS
SGS1SDATAV	SSA1	GSE FEP #1 STANDBY DATA VALID
SGS1SSTAT	SSA1	GSE FEP #1 STANDBY STATUS
SGS2ADATAV	SSA1	GSE FEP #2 ACTIVE DATA VALID
SGS2ASTAT	SSA1	GSE FEP #2 ACTIVE STATUS
SGS2SDATAV	SSA1	GSE FEP #2 STANDBY DATA VALID
SGS2SSTAT	SSA1	GSE FEP #2 STANDBY STATUS
SGS3ADATAV	SSA1	GSE FEP #3 ACTIVE DATA VALID
SGS3ASTAT	SSA1	GSE FEP #3 ACTIVE STATUS
SGS3SDATAV	SSA1	STANDBY DATA VALID
SGS3SSTAT	SSA1	STANDBY STATUS
SGS4DATAV	SSA1	GSE FEP #4 DATA VALID
SGS4STAT	SSA1	GSE FEP #4 STATUS
SGS5DATAV	SSA1	GSE FEP #5 DATA VALID
SGS5STAT	SSA1	GSE FEP #5 STATUS
SHIM001ST	SSA1	HIM 1 STATUS

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SHIM002ST	SSA1	HIM 2 STATUS
SHIM003ST	SSA1	HIM 3 STATUS
SHIM004ST	SSA1	HIM 4 STATUS
SHIM005ST	SSA1	HIM 5 STATUS
SHIM006ST	SSA1	HIM 6 STATUS
SHIM007ST	SSA1	HIM 7 STATUS
SHIM010ST	SSA1	HIM 10 STATUS
SHIM011ST	SSA1	HIM 11 STATUS
SHIM012ST	SSA1	HIM 12 STATUS
SHIM013ST	SSA1	HIM 13 STATUS
SHIM014ST	SSA1	HIM 14 STATUS
SHIM015ST	SSA1	HIM 15 STATUS
SHIM016ST	SSA1	HIM 16 STATUS
SHIM017ST	SSA1	HIM 17 STATUS
SHIM020ST	SSA1	HIM 20 STATUS
SHIM021ST	SSA1	HIM 21 STATUS
SHIM022ST	SSA1	HIM 22 STATUS
SHIM023ST	SSA1	HIM 23 STATUS
SHIM024ST	SSA1	HIM 24 STATUS
SHIM025ST	SSA1	HIM 25 STATUS
SHIM026ST	SSA1	HIM 26 STATUS
SHIM027ST	SSA1	HIM 27 STATUS
SHIM030ST	SSA1	HIM 30 STATUS
SHIM031ST	SSA1	HIM 31 STATUS
SHIM032ST	SSA1	HIM 32 STATUS
SHIM033ST	SSA1	HIM 33 STATUS
SHIM034ST	SSA1	HIM 34 STATUS
SHIM035ST	SSA1	HIM 35 STATUS
SHIM036ST	SSA1	HIM 36 STATUS
SHIM037ST	SSA1	HIM 37 STATUS
SHIM040ST	SSA1	HIM 40 STATUS
SHIM041ST	SSA1	HIM 41 STATUS
SHIM042ST	SSA1	HIM 42 STATUS
SHIM043ST	SSA1	HIM 43 STATUS
SHIM044ST	SSA1	HIM 44 STATUS
SHIM045ST	SSA1	HIM 45 STATUS
SHIM046ST	SSA1	HIM 46 STATUS
SHIM047ST	SSA1	HIM 47 STATUS
SHIM050ST	SSA1	HIM 50 STATUS
SHIM051ST	SSA1	HIM 51 STATUS
SHIM052ST	SSA1	HIM 52 STATUS
SHIM053ST	SSA1	HIM 53 STATUS
SHIM054ST	SSA1	HIM 54 STATUS
SHIM055ST	SSA1	HIM 55 STATUS

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SHIM056ST	SSA1	HIM 56 STATUS
SHIM057ST	SSA1	HIM 57 STATUS
SHIM060ST	SSA1	HIM 60 STATUS
SHIM061ST	SSA1	HIM 61 STATUS
SHIM062ST	SSA1	HIM 62 STATUS
SHIM063ST	SSA1	HIM 63 STATUS
SHIM064ST	SSA1	HIM 64 STATUS
SHIM065ST	SSA1	HIM 65 STATUS
SHIM066ST	SSA1	HIM 66 STATUS
SHIM067ST	SSA1	HIM 67 STATUS
SHIM070ST	SSA1	HIM 70 STATUS
SHIM071ST	SSA1	HIM 71 STATUS
SHIM072ST	SSA1	HIM 72 STATUS
SHIM073ST	SSA1	HIM 73 STATUS
SHIM074ST	SSA1	HIM 74 STATUS
SHIM075ST	SSA1	HIM 75 STATUS
SHIM076ST	SSA1	HIM 76 STATUS
SHIM077ST	SSA1	HIM 77 STATUS
SHIM100ST	SSA1	HIM 100 STATUS
SHIM101ST	SSA1	HIM 101 STATUS
SHIM102ST	SSA1	HIM 102 STATUS
SHIM103ST	SSA1	HIM 103 STATUS
SHIM104ST	SSA1	HIM 104 STATUS
SHIM105ST	SSA1	HIM 105 STATUS
SHIM106ST	SSA1	HIM 106 STATUS
SHIM107ST	SSA1	HIM 107 STATUS
SHIM110ST	SSA1	HIM 110 STATUS
SHIM111ST	SSA1	HIM 111 STATUS
SHIM112ST	SSA1	HIM 112 STATUS
SHIM113ST	SSA1	HIM 113 STATUS
SHIM114ST	SSA1	HIM 114 STATUS
SHIM115ST	SSA1	HIM 115 STATUS
SHIM116ST	SSA1	HIM 116 STATUS
SHIM117ST	SSA1	HIM 117 STATUS
SHIM120ST	SSA1	HIM 120 STATUS
SHIM121ST	SSA1	HIM 121 STATUS
SHIM122ST	SSA1	HIM 122 STATUS
SHIM123ST	SSA1	HIM 123 STATUS
SHIM124ST	SSA1	HIM 124 STATUS
SHIM125ST	SSA1	HIM 125 STATUS
SHIM126ST	SSA1	HIM 126 STATUS
SHIM127ST	SSA1	HIM 127 STATUS
SHIM130ST	SSA1	HIM 130 STATUS
SHIM131ST	SSA1	HIM 131 STATUS

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SHIM132ST	SSA1	HIM 132 STATUS
SHIM133ST	SSA1	HIM 133 STATUS
SHIM134ST	SSA1	HIM 134 STATUS
SHIM135ST	SSA1	HIM 135 STATUS
SHIM136ST	SSA1	HIM 136 STATUS
SHIM137ST	SSA1	HIM 137 STATUS
SHIM140ST	SSA1	HIM 140 STATUS
SHIM141ST	SSA1	HIM 141 STATUS
SHIM142ST	SSA1	HIM 142 STATUS
SHIM143ST	SSA1	HIM 143 STATUS
SHIM144ST	SSA1	HIM 144 STATUS
SHIM145ST	SSA1	HIM 145 STATUS
SHIM146ST	SSA1	HIM 146 STATUS
SHIM147ST	SSA1	HIM 147 STATUS
SHIM150ST	SSA1	HIM 150 STATUS
SHIM151ST	SSA1	HIM 151 STATUS
SHIM152ST	SSA1	HIM 152 STATUS
SHIM153ST	SSA1	HIM 153 STATUS
SHIM154ST	SSA1	HIM 154 STATUS
SHIM155ST	SSA1	HIM 155 STATUS
SHIM156ST	SSA1	HIM 156 STATUS
SHIM157ST	SSA1	HIM 157 STATUS
SHIM160ST	SSA1	HIM 160 STATUS
SHIM161ST	SSA1	HIM 161 STATUS
SHIM162ST	SSA1	HIM 162 STATUS
SHIM163ST	SSA1	HIM 163 STATUS
SHIM164ST	SSA1	HIM 164 STATUS
SHIM165ST	SSA1	HIM 165 STATUS
SHIM166ST	SSA1	HIM 166 STATUS
SHIM167ST	SSA1	HIM 167 STATUS
SHIM170ST	SSA1	HIM 170 STATUS
SHIM171ST	SSA1	HIM 171 STATUS
SHIM172ST	SSA1	HIM 172 STATUS
SHIM173ST	SSA1	HIM 173 STATUS
SHIM174ST	SSA1	HIM 174 STATUS
SHIM175ST	SSA1	HIM 175 STATUS
SHIM176ST	SSA1	HIM 176 STATUS
SHIM177ST	SSA1	HIM 177 STATUS
SHIM200ST	SSA1	HIM 200 STATUS
SHIM201ST	SSA1	HIM 201 STATUS
SHIM202ST	SSA1	HIM 202 STATUS
SHIM203ST	SSA1	HIM 203 STATUS
SHIM204ST	SSA1	HIM 204 STATUS
SHIM205ST	SSA1	HIM 205 STATUS

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SHIM206ST	SSA1	HIM 206 STATUS
SHIM207ST	SSA1	HIM 207 STATUS
SHIM210ST	SSA1	HIM 210 STATUS
SHIM211ST	SSA1	HIM 211 STATUS
SHIM212ST	SSA1	HIM 212 STATUS
SHIM213ST	SSA1	HIM 213 STATUS
SHIM214ST	SSA1	HIM 214 STATUS
SHIM215ST	SSA1	HIM 215 STATUS
SHIM216ST	SSA1	HIM 216 STATUS
SHIM217ST	SSA1	HIM 217 STATUS
SHIM220ST	SSA1	HIM 220 STATUS
SHIM221ST	SSA1	HIM 221 STATUS
SHIM222ST	SSA1	HIM 222 STATUS
SHIM223ST	SSA1	HIM 223 STATUS
SHIM224ST	SSA1	HIM 224 STATUS
SHIM225ST	SSA1	HIM 225 STATUS
SHIM226ST	SSA1	HIM 226 STATUS
SHIM227ST	SSA1	HIM 227 STATUS
SHIM230ST	SSA1	HIM 230 STATUS
SHIM231ST	SSA1	HIM 231 STATUS
SHIM232ST	SSA1	HIM 232 STATUS
SHIM233ST	SSA1	HIM 233 STATUS
SHIM234ST	SSA1	HIM 234 STATUS
SHIM235ST	SSA1	HIM 235 STATUS
SHIM236ST	SSA1	HIM 236 STATUS
SHIM237ST	SSA1	HIM 237 STATUS
SHIM240ST	SSA1	HIM 240 STATUS
SHIM241ST	SSA1	HIM 241 STATUS
SHIM242ST	SSA1	HIM 242 STATUS
SHIM243ST	SSA1	HIM 243 STATUS
SHIM244ST	SSA1	HIM 244 STATUS
SHIM245ST	SSA1	HIM 245 STATUS
SHIM246ST	SSA1	HIM 246 STATUS
SHIM247ST	SSA1	HIM 247 STATUS
SHIM250ST	SSA1	HIM 250 STATUS
SHIM251ST	SSA1	HIM 251 STATUS
SHIM252ST	SSA1	HIM 252 STATUS
SHIM253ST	SSA1	HIM 253 STATUS
SHIM254ST	SSA1	HIM 254 STATUS
SHIM255ST	SSA1	HIM 255 STATUS
SHIM256ST	SSA1	HIM 256 STATUS
SHIM257ST	SSA1	HIM 257 STATUS
SHIM260ST	SSA1	HIM 260 STATUS
SHIM261ST	SSA1	HIM 261 STATUS

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SHIM262ST	SSA1	HIM 262 STATUS
SHIM263ST	SSA1	HIM 263 STATUS
SHIM264ST	SSA1	HIM 264 STATUS
SHIM265ST	SSA1	HIM 265 STATUS
SHIM266ST	SSA1	HIM 266 STATUS
SHIM267ST	SSA1	HIM 267 STATUS
SHIM270ST	SSA1	HIM 270 STATUS
SHIM271ST	SSA1	HIM 271 STATUS
SHIM272ST	SSA1	HIM 272 STATUS
SHIM273ST	SSA1	HIM 273 STATUS
SHIM274ST	SSA1	HIM 274 STATUS
SHIM275ST	SSA1	HIM 275 STATUS
SHIM276ST	SSA1	HIM 276 STATUS
SHIM277ST	SSA1	HIM 277 STATUS
SHIM300ST	SSA1	HIM 300 STATUS
SHIM301ST	SSA1	HIM 301 STATUS
SHIM302ST	SSA1	HIM 302 STATUS
SHIM303ST	SSA1	HIM 303 STATUS
SHIM304ST	SSA1	HIM 304 STATUS
SHIM305ST	SSA1	HIM 305 STATUS
SHIM306ST	SSA1	HIM 306 STATUS
SHIM307ST	SSA1	HIM 307 STATUS
SHIM310ST	SSA1	HIM 310 STATUS
SHIM311ST	SSA1	HIM 311 STATUS
SHIM312ST	SSA1	HIM 312 STATUS
SHIM313ST	SSA1	HIM 313 STATUS
SHIM314ST	SSA1	HIM 314 STATUS
SHIM315ST	SSA1	HIM 315 STATUS
SHIM316ST	SSA1	HIM 316 STATUS
SHIM317ST	SSA1	HIM 317 STATUS
SHIM320ST	SSA1	HIM 320 STATUS
SHIM321ST	SSA1	HIM 321 STATUS
SHIM322ST	SSA1	HIM 322 STATUS
SHIM323ST	SSA1	HIM 323 STATUS
SHIM324ST	SSA1	HIM 324 STATUS
SHIM325ST	SSA1	HIM 325 STATUS
SHIM326ST	SSA1	HIM 326 STATUS
SHIM327ST	SSA1	HIM 327 STATUS
SHIM330ST	SSA1	HIM 330 STATUS
SHIM331ST	SSA1	HIM 331 STATUS
SHIM332ST	SSA1	HIM 332 STATUS
SHIM333ST	SSA1	HIM 333 STATUS
SHIM334ST	SSA1	HIM 334 STATUS
SHIM335ST	SSA1	HIM 335 STATUS

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SHIM336ST	SSA1	HIM 336 STATUS
SHIM337ST	SSA1	HIM 337 STATUS
SHIM340ST	SSA1	HIM 340 STATUS
SHIM341ST	SSA1	HIM 341 STATUS
SHIM342ST	SSA1	HIM 342 STATUS
SHIM343ST	SSA1	HIM 343 STATUS
SHIM344ST	SSA1	HIM 344 STATUS
SHIM345ST	SSA1	HIM 345 STATUS
SHIM346ST	SSA1	HIM 346 STATUS
SHIM347ST	SSA1	HIM 347 STATUS
SHIM350ST	SSA1	HIM 350 STATUS
SHIM351ST	SSA1	HIM 351 STATUS
SHIM352ST	SSA1	HIM 352 STATUS
SHIM353ST	SSA1	HIM 353 STATUS
SHIM354ST	SSA1	HIM 354 STATUS
SHIM355ST	SSA1	HIM 355 STATUS
SHIM356ST	SSA1	HIM 356 STATUS
SHIM357ST	SSA1	HIM 357 STATUS
SHIM360ST	SSA1	HIM 360 STATUS
SHIM361ST	SSA1	HIM 361 STATUS
SHIM362ST	SSA1	HIM 362 STATUS
SHIM363ST	SSA1	HIM 363 STATUS
SHIM364ST	SSA1	HIM 364 STATUS
SHIM365ST	SSA1	HIM 365 STATUS
SHIM366ST	SSA1	HIM 366 STATUS
SHIM367ST	SSA1	HIM 367 STATUS
SHIM370ST	SSA1	HIM 370 STATUS
SHIM371ST	SSA1	HIM 371 STATUS
SHIM372ST	SSA1	HIM 372 STATUS
SHIM373ST	SSA1	HIM 373 STATUS
SHIM374ST	SSA1	HIM 374 STATUS
SHIM375ST	SSA1	HIM 375 STATUS
SHIM376ST	SSA1	HIM 376 STATUS
SH010SH161	SSA1	HIM 10 OR 161 STATUS
SH013SH202	SSA1	HIM 13 OR 202 STATUS
SH016SH056	SSA1	HIM 15, 16, OR 56 STATUS
SH020SH206	SSA1	HIM 20 OR 206 STATUS
SH022SH151	SSA1	HIM 22 OR 151 STATUS
SH023SH303	SSA1	HIM 23 OR 303 STATUS
SH025SH325	SSA1	HIM 25 OR 325 STATUS
SH026SH246	SSA1	HIM 26 OR 246 STATUS
SH027SH260	SSA1	HIM 11, 27 OR 260 STATUS
SH031SH227	SSA1	HIM 31 OR 227 STATUS
SH032SH311	SSA1	HIM 32 OR 311 STATUS

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SH034SH256	SSA1	HIM 34 OR 256 STATUS
SH035SH061	SSA1	HIM 12, 35 OR 061 STATUS
SH037SH236	SSA1	HIM 37, 162, OR 236 STATUS
SH040SH101	SSA1	HIM 40, 101, OR 321 STATUS
SH043SH077	SSA1	HIM 43, 77, OR 333 STATUS
SH045SH045	SSA1	HIM 45 OR 45 STATUS
SH046SH065	SSA1	HIM 46, 65, OR 306 STATUS
SH047SH062	SSA1	HIM 14, 47, OR 62 STATUS
SH051SH137	SSA1	HIM 51, 137, OR 323 STATUS
SH052SH071	SSA1	HIM 52, 71, OR 317 STATUS
SH055SH263	SSA1	HIM 17, 55, OR 263 STATUS
SH064SH072	SSA1	HIM 60, 64, OR 72 STATUS
SH070SH102	SSA1	HIM 70 OR 102 STATUS
SH073SH073	SSA1	HIM 73 OR 146 STATUS
SH074SH265	SSA1	HIM 74, 76, OR 265 STATUS
SH075SH175	SSA1	HIM 75, 131, OR 175 STATUS
SH100SH100	SSA1	HIM 100 OR 100 STATUS
SH103SH103	SSA1	HIM 103 OR 103 STATUS
SH104SH241	SSA1	HIM 104, 241, OR 324 STATUS
SH105SH105	SSA1	HIM 105 OR 105 STATUS
SH106SH106	SSA1	HIM 106 OR 106 STATUS
SH107SH242	SSA1	HIM 107, 242, OR 340 STATUS
SH110SH266	SSA1	HIM 110, 124, OR 266 STATUS
SH111SH111	SSA1	HIM 111 OR 111 STATUS
SH112SH112	SSA1	HIM 112 OR 112 STATUS
SH113SH253	SSA1	HIM 113, 253, OR 330 STATUS
SH114SH114	SSA1	HIM 114 OR 114 STATUS
SH115SH250	SSA1	HIM 115, 250, OR 335 STATUS
SH135SH244	SSA1	HIM 135, 157, OR 244 STATUS
SH136SH264	SSA1	HIM 136, 143, OR 264 STATUS
SH140SH274	SSA1	HIM 127, 140, OR 274 STATUS
SH141SH141	SSA1	HIM 141 OR 141 STATUS
SH155SH235	SSA1	HIM 155 OR 235 STATUS
SH164SH201	SSA1	HIM 164, 201, OR 315 STATUS
SH165SH365	SSA1	HIM 165 OR 365 STATUS
SH166SH211	SSA1	HIM 166 OR 211 STATUS
SH173SH212	SSA1	HIM 173 OR 212 STATUS
SH204SH305	SSA1	HIM 204 OR 305 STATUS
SH205SH255	SSA1	HIM 205 OR 255 STATUS
SH213SH221	SSA1	HIM 213 OR 221 STATUS
SH217SH230	SSA1	HIM 217 OR 230 STATUS
SH222SH275	SSA1	HIM 222 OR 275 STATUS
SH224SH320	SSA1	HIM 224 OR 320 STATUS
SH233SH316	SSA1	HIM 233 OR 316 STATUS

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SH257SH357	SSA1	HIM 257 OR 357 STATUS
SH261SH261	SSA1	HIM 150, 261, OR 261 STATUS
SH271SH271	SSA1	HIM 271 OR 271 STATUS
SH272SH272	SSA1	HIM 272 OR 272 STATUS
SH301SH301	SSA1	HIM 301 OR 101 STATUS
SH302SH302	SSA1	HIM 302 OR 302 STATUS
SH304SH304	SSA1	HIM 304 OR 304 STATUS
SH307SH307	SSA1	HIM 307 OR 307 STATUS
SH310SH310	SSA1	HIM 310 OR 310 STATUS
SH313SH313	SSA1	HIM 313 OR 313 STATUS
SHOSCDATAV	SSA1	HOSC FEP DATA VALID
SHOSCSTAT	SSA1	HOSC FEP STATUS
SHOSRDATAV	SSA1	HOSR FEP DATA VALID
SHOSRSTAT	SSA1	HOSR FEP STATUS
SHSP1STAT	SSA1	HOT SPARE CONSOLE #1 STATUS
SHSP1GO	SSA1	HOT SPARE CONSOLE #1 GO MODE
SHSP2STAT	SSA1	HOT SPARE CONSOLE #2 STATUS
SHSP2GO	SSA1	HOT SPARE CONSOLE #2 GO MODE
SHSP3STAT	SSA1	HOT SPARE CONSOLE #3 STATUS
SHSP3GO	SSA1	HOT SPARE CONSOLE #3 GO MODE
SINTGGO	SSA1	INTEGRATION CONSOLE GO MODE
SINTGSTAT	SSA1	INTEGRATION CONSOLE STATUS
SLDBADATAV	SSA1	LDB FEP ACTIVE DATA VALID
SLDBASTAT	SSA1	LDB FEP ACTIVE STATUS
SLDBDDATAV	SSA1	LDBD FEP ACTIVE DATA VALID
SLDBDDIOMD	SSA1	LDBD FEP DIRECT I/O MODE
SLBDGPCMD	SSA1	LDBD FEP GPC MODE
SLDBDIOMD	SSA1	LDB FEP DIRECT I/O MODE
SLDBDSTAT	SSA1	LDBD FEP ACTIVE STATUS
SLDBD1STAT	SSA1	LDBD BUS 1 ACTIVE
SLDBD2STAT	SSA1	LDBD BUS 2 ACTIVE
SLDBGPCMD	SSA1	LDB FEP GPC MODE
SLDBSDATAV	SSA1	LDB FEP STANDBY DATA VALID
SLDBSSTAT	SSA1	LDB FEP STANDBY STATUS
SLDB1STAT	SSA1	LDB BUS 1 ACTIVE
SLDB2STAT	SSA1	LDB BUS 2 ACTIVE
SME1DATAV	SSA1	MAIN ENGINE FEP #1 DATA VALID
SME1STAT	SSA1	MAIN ENGINE FEP #1 STATUS
SME2DATAV	SSA1	MAIN ENGINE FEP #2 DATA VALID
SME2STAT	SSA1	MAIN ENGINE FEP #2 STATUS
SME3DATAV	SSA1	MAIN ENGINE FEP #3 DATA VALID
SME3STAT	SSA1	MAIN ENGINE FEP #3 STATUS
SMSTRGO	SSA1	MASTER CONSOLE GO MODE
SMSTRSTAT	SSA1	MASTER CONSOLE STATUS

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
SMTUDATAV	SSA1	MASTER TIMING UNIT DATA VALID
SMTUSTAT	SSA1	MASTER TIMING UNIT STATUS
SOIADATAV	SSA1	128 OI FEP ACTIVE DATA VALID
SOIAREAOI	SSA1	OI FEP OI AREA STATUS
SOIASTAT	SSA1	128 OI FEP ACTIVE STATUS
SOISDATAV	SSA1	128 OI FEP STANDBY DATA VALID
SOISSTAT	SSA1	128 OI FEP STANDBY STATUS
SPDRGO	SSA1	PROCESSED DATA RECORDER GO
SPDRSTAT	SSA1	PROCESSED DATA RECORDER
SPLDAREAOI	SSA1	PLD FEP OI AREA STATUS
SPLDAREA1	SSA1	PLD FEP AREA 1 STATUS
SPLDAREA2	SSA1	PLD FEP AREA 2 STATUS
SPLDAREA3	SSA1	PLD FEP AREA 3 STATUS
SPLDAREA4	SSA1	PLD FEP AREA 4 STATUS
SPLDADATAV	SSA1	PLD FEP ACTIVE DATA VALID
SPLDASTAT	SSA1	PLD FEP ACTIVE STATUS
SPLDSDATAV	SSA1	PLD FEP STANDBY DATA VALID
SPLDSSTAT	SSA1	PLD FEP STANDBY STATUS
SPP1GO	SSA1	PRINTER-PLOTTER #1 GO MODE
SPP1STAT	SSA1	PRINTER-PLOTTER #1 STATUS
SPP2GO	SSA1	PRINTER-PLOTTER #2 GO MODE
SPP2STAT	SSA1	PRINTER-PLOTTER #2 STATUS
SRTU001ST	SSA1	RTU 1 STATUS
SRTU002ST	SSA1	RTU 2 STATUS
SRTU003ST	SSA1	RTU 3 STATUS
SRTU004ST	SSA1	RTU 4 STATUS
SRTU005ST	SSA1	RTU 5 STATUS
SRTU006ST	SSA1	RTU 6 STATUS
SRTU014ST	SSA1	RTU 14 STATUS
SRTU017ST	SSA1	RTU 17 STATUS
SRTU018ST	SSA1	RTU 18 STATUS
SSDB1DATAV	SSA1	SDB1 FEP DATA VALID
SSDB1STAT	SSA1	SDB1 FEP STATUS
SSDB2DATAV	SSA1	SDB2 FEP DATA VALID
SSDB2STAT	SSA1	SDB2 FEP STATUS
SSDB1BUS1	SSA1	SDB1 BUS #1 ACTIVE
SSDB1BUS2	SSA1	SDB1 BUS #2 ACTIVE
SSDB1M1STA	SSA1	SDB1 MDM #1 STATUS
SSDB1M2STA	SSA1	SDB2 MDM #1 STATUS
SSDB2BUS1	SSA1	SDB2 BUS #1 ACTIVE
SSDB2BUS2	SSA1	SDB2 BUS #2 ACTIVE
SSDB2M1STA	SSA1	SDB2 BUS #1 STATUS
SSDB2M2STA	SSA1	SDB2 BUS #2 STATUS
SSPAGO	SSA1	SHARED PERIPHERAL AREA GO MODE
SSPASTAT	SSA1	SHARED PERIPHERAL AREA STATUS
STCG1DATAV	SSA1	TIME CODE GEN PBIC #1 DATA VALID

NAME	TYPE	DESCRIPTIVE NOMENCLATURE
STCG1STAT	SSA1	TIME CODE GENERATOR PBIC #1 STATUS
STCG2DATAV	SSA1	TIME CODE GEN PBIC #2 DATA VALID
STCG2STAT	SSA1	TIME CODE GENERATOR PBIC #2 STATUS
STMDSDATAV	SSA1	TIMING PBIC DATA VALID
STMDSSTAT	SSA1	TIMING DISTRIBUTION PBIC STATUS
SUCSDATAV	SSA1	UTILITY CONTROL SET DATA VALID
SUCSSTAT	SSA1	UTILITY CONTROL SET STATUS
SUPLKDATAV	SSA1	UPLINK COMMAND FEP DATA VALID
SUPLKSTAT	SSA1	UPLINK COMMAND FEP STATUS
SVDSGO	SSA1	V&DA PBIC GO MODE
SVDSSTAT	SSA1	V&DA PBIC STATUS
TYPE = SSA2		
SCDL1FID	SSA2	COMMON DOWNLINK AREA FORMAT ID
SGPCFIDAOI	SSA2	GPC FEP OI AREA FORMAT ID
SGPCFIDA1	SSA2	GPC FEP AREA 1 FORMAT ID
SGPCFIDA2	SSA2	GPC FEP AREA 2 FORMAT ID
SGPCFIDA3	SSA2	GPC FEP AREA 3 FORMAT ID
SGPCFIDA4	SSA2	GPC FEP AREA 4 FORMAT ID
SOIFID	SSA2	OI FEP FORMAT ID
SPLDFIDAOI	SSA2	PLD FEP OI AREA FORMAT ID
SPLDFIDA1	SSA2	PLD FEP AREA 1 FORMAT ID
SPLDFIDA2	SSA2	PLD FEP AREA 2 FORMAT ID
SPLDFIDA3	SSA2	PLD FEP AREA 3 FORMAT ID
SPLDFIDA4	SSA2	PLD FEP AREA 4 FORMAT ID
SSDB1M1ADR	SSA2	SDB1 MDM #1 BTU ADDRESS
SSDB1M2ADR	SSA2	SDB1 MDM #2 BTU ADDRESS
SSDB2M1ADR	SSA2	SDB2 MDM #1 BTU ADDRESS
SSDB2M2ADR	SSA2	SDB2 MDM #2 BTU ADDRESS
SSWOFIDAOI	SSA2	SWOF FEP OI AREA FORMAT ID
SSWOFIDA1	SSA2	SWOF FEP AREA 1 FORMAT ID
SSWOFIDA2	SSA2	SWOF FEP AREA 2 FORMAT ID
SSWOFIDA3	SSA2	SWOF FEP AREA 3 FORMAT ID
SSWOFIDA4	SSA2	SWOF FEP AREA 4 FORMAT ID
TYPE = TIMR		
TIMER	TIMR	INTERVAL TIMER
TYPE = UPAL		
UPLKACTI	UPAI	UPLINK ACTIVITY INDICATOR

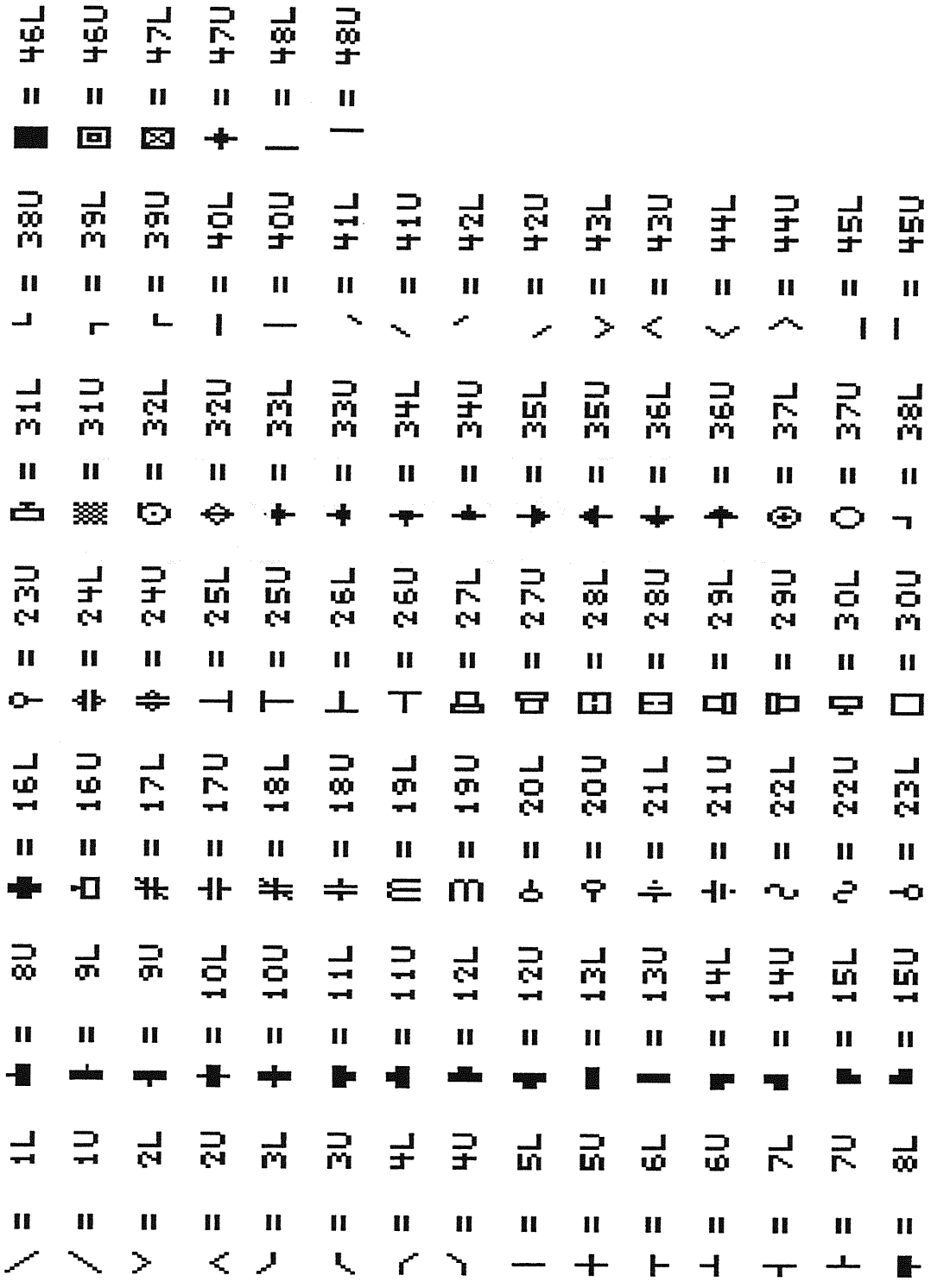


Figure B-1 Graphic Function Keys

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX C. GLOSSARY

THIS PAGE INTENTIONALLY LEFT BLANK.

Analog

A signal which has a magnitude that varies with respect to time and which is a measurable Quantity (e.g., a voltage or pressure measurement).

Application Page

One of the console display presentations to which GOAL or machine language application programs may display data.

Application Program

A GOAL or machine language program which runs under the control of the Operating System or GOAL Executor and performs a user-oriented function.

Audible Alarm

An acoustic device located on the console PFP which produces a sound that warns the operator of danger. The device may be controlled from GOAL or the DG keyboard.

Bulk Memory

A 128K solid state memory addition to the console which emulates core or fixed head disk. It is used to store the GOAL high speed directory, Control Logic sequences, and display pages.

Cathode Ray Tube (CRT)

An on-line electronic display device which is often used as a computer peripheral to display information from a computer subsystem.

Central Data Subsystem (CDS)

A large scale central computer facility which houses Launch Processing System (LPS) support software including the GOAL Language Processor, TCS Compiler, Control Logic Compiler, and System Build Software.

Central Data Subsystem (CDS) Recorder

One of the magnetic tape recorders located in the CDS.

Central Processing Unit (CPU)

The part of a computer which handles the execution of instructions. This term is often loosely used to refer to a computer mainframe including CPU and memory but excluding peripherals.

Checkout, Control, and Monitor Subsystem (CCMS)

A distributed computer network at KSC, which is used to accomplish test, checkout, and control of the Space Shuttle Vehicle and Ground Support Equipment.

Commanded Status

The value of a stimulus stored in the CDBFR for reference by GOAL procedures. When a stimulus is issued to GSE, the commanded status is the echo of the stimulus returned from the HIM and stored in the CDBFR by the GSE FEP.

Common Data Buffer (CDBFR)

A high speed memory centrally located in CCMS, which is used to store measurement and commanded status and for computer-to-computer communication.

Component

A collection of software modules which performs a single, exclusive function.

Compiler

See GOAL Compiler.

Compiler Directive

A statement which begins with an asterisk and ends with a semicolon and is used to give instructions to the GOAL Compiler regarding the generation of output reports, intermediate text, or interpretive code.

Concurrency

A procedure or task which runs independently of and in parallel with another; a concurrent program.

Concurrent Program

A procedure or task which runs in parallel with another--synonymous with concurrency.

Console

The computer subsystem in CCMS in which GOAL procedures are executed. The console subsystem includes a CPU (Type I), three color CRT's and keyboards, two moving head disks, and an electrostatic printer/plotter.

Console Subsystem

The computer hardware, software, and peripherals associated with a console.

Control Logic

A software component in the Console Subsystem which executes prerequisite and reactive sequences.

Countdown Time (CDT)

A time (measured in days, hours, minutes and seconds) maintained in CCMS and referenced to the time of liftoff. Negative time is prior to liftoff; Count Down Time equal to zero is the time of liftoff; and positive time is after liftoff.

Critical Program

A GOAL program which runs at a higher priority than normal GOAL programs.

Cursor

A marker displayed on a CRT, which indicates the position of the next character to be displayed.

Data Bank

The collection of all compiler and hardware related records for Function Designators.

Declaration Statement

A non-test action GOAL statement that is required in CCMS to reserve storage, supply initial data conditions, or to declare the "type" of data that is valid for specified actions.

Digital Pattern

A measurement or stimulus which is a string of up to 16 binary bits (ones and zeroes).

Dimension

The engineering units associated with a Quantity value (e.g., volts, amperes, PSI).

Discrete

A measurement or stimulus which may take on a binary value (e.g., logical 0 or 1).

Disk File

A storage file located on console moving head disk which may be used by GOAL procedures to store and retrieve data.

Display Electronics Unit (DEU)

The on-board man-machine interface, consisting of a CRT and keyboard.

Display Electronics Unit (DEU) Equivalent

An emulation of an on-board DEU keyboard input.

Display Generator (DG)

A hardware interface between the console computer and three color CRT's and associated keyboards. The DG writes data to the color CRT's and routes user keyboard inputs back to the console computer.

Display Generator (DG) Function Key

One of the keys on the DG keyboard used to transmit interrupts to the console computer. DG Function Key interrupts are handled by the Control and Display component of the console Operating System and are used to do such things as communicate with GOAL procedures and issue keyboard commands.

Display Page

One of the display presentations (live and/or background) which contains data recorded by system or application tasks and which may be selected for viewing by an operator.

Engineering Value

A dimension.

Exception Condition

An out-of-limits value in the case of an analog measurement; a state which is not normal in the case of a discrete measurement; or a digital pattern which is equal to an exception value, not equal to a predefined value, or any change in a predefined value.

Exception Monitoring

The process of detecting exception conditions in an FEP, notifying a console, and displaying the exception condition to the operator.

Explicity Coded Program (ECP)

A machine language application program which runs in the on-board GPC.

External Designator

A Function Designator or string of Function Designators.

File

A storage area in which raw (binary) data is written. Files are located on the console moving head disks, the PDR tape recorders, and the CDS tape recorders.

Function Designator

The name given to a reference which is external to a GOAL procedure and which interfaces via the Data Bank with the system under test or checkout system software.

General Memory (G-MEM)

The mainframe memory of the on-board General Purpose Computer (GPC).

GOAL

The acronym for Ground Operations Aerospace Language. GOAL is the higher order computer language tailored to aerospace test, checkout, and control operations.

GOAL Compiler

A component of the GOAL Language Processor which syntactically validates GOAL source statements and outputs intermediate text to the translator.

GOAL Element

A syntactical part of a GOAL statement which is discussed as a separate unit for convenience.

GOAL Executor

A component in the Console Subsystem which runs under the control of the Operating System and controls the execution of GOAL procedures.

GOAL Language Processor (GLP)

A software subsystem, composed of a compiler and translator, which syntactically validates the GOAL source statements and converts them into interpretive code for input to the GOAL Executor.

Graphic Character

A special symbol which may be displayed on the DG CRT and may be used to construct engineering illustrations.

Ground Support Equipment (GSE)

Non-flight hardware, facilities, and subsystems required to support test, checkout, and launch of the Space Shuttle Vehicle.

Ground Support Equipment Front End Processor (GSE FEP)

One of the CCMS FEP's which interfaces with GSE via a HIM.

Hardware Interface Module (HIM)

A hardware interface between the GSE FEP and GSE. A HIM contains discrete output drivers, digital to analog converters for analog stimuli, analog to digital converters for analog measurements, discrete input registers, relay closure cards, and special cards for handling digital patterns.

Input/Output (I/O)

A general term which refers to the communication between a hardware device or software element and the outside world. For example, communications between a computer and a peripheral device are referred to as I/O.

Internal Name

A symbolic reference to internal data in a GOAL procedure. An internal name may be a name (a single data item), a list or list element, or a table or table element.

Interpretive Code

An ordered set of operators which specify the functions with associated parameters resulting from compile and translation, via the GLP, of GOAL Source statements. An operator is an ordered set of data which specify a function and contain the parameters required to perform the function as specified by the GOAL source statements. These parameters provide the necessary information for the GOAL Executor to accomplish the statements specified in the GOAL source.

Interrupt

An external event occurrence which may be used to alter the execution path of a program. Interrupts may be passed to GOAL procedures by the GOAL Executor.

Interrupt Block

The space reserved in the RDA for defining an interrupt.

Interval Timer

One of the six timers per console allocated to GOAL procedures on a per task basis. When a timer is set to a positive value, it decrements to zero at a one count per second rate. When it reaches zero, it may generate an interrupt to a GOAL procedure.

Julian Time of Year (JTOY)

A time maintained in console memory by timer services. It is generated from Julian Day and time of day which are maintained in the CDBFR, and is updated once per second.

Launch Data Bus Front End Processor (LDB FEP)

The CCMS computer which interfaces with the GPC via the Launch Data Bus. It formats TCS-1/SACS commands upon request from GOAL procedures or keyboard commands and routes TCS responses back to the originating console.

Level

One of the four GOAL procedures which may be executing within a task at any point in time. The mainline is considered level one and the fourth nested level is considered level four.

Light Emitting Diode (LED)

The displays located on the Programmable Function Panels. Six half lines of 17 characters each may be displayed on the LED's of a single PFP.

List

A group of data items referred to by a single Name.

Load

The description of a stimulus type Function Designator; e.g., Load Analog; the opposite of sensor.

Macro

A named Character String. Once a Macro is defined, it may be inserted at any point within a GOAL procedure by referencing the macro name.

Mainline

The procedure executing at level 1 within a Task.

Master Console

The computer in the CCMS which provides load, initialization, termination, status monitoring, and recovery/reconfiguration functions.

Mission Elapsed Time (MET)

A time (measured in days, hours, minutes, and seconds) maintained by the CITE System and referenced to the time of liftoff. Mission Elapsed Time equal to zero is the time prior to liftoff or the time of liftoff; positive time is after liftoff.

Moving Head Disk

A disk with a capacity of approximately five million 16-bit words, which is located on a CCMS console. GOAL and TCS programs reside on this disk.

Multiword Measurement

A measurement which is more than 16 bits in length; e.g., 32 or 64 bit floating point words.

Name

An internal name which refers to a single data item as opposed to a list name or table name.

Non-GOAL Program

A machine language applications program which may be called from GOAL and executed on an operations console.

Numeric Data

Fixed point data. Numeric data may be integer, binary, hexadecimal, or octal.

On-Board Interface

The GOAL capability of sending single commands from the ground to the on-board GPC for execution by the Test Control Supervisor. Also known as TCS-1 and TCS-1-1.

On-Line

A term which refers to a hardware device or a program which is powered up and/or active but not executing and dynamically responding in real-time.

Operating System

A collection of software components required to manage computer resources, control the execution of programs, interface with peripherals, and handle the routing of transactions among the various components of a software subsystem.

Parameter

A constant or variable which may be passed from one program to another when performing or concurrently performing a program.

Parameter Passing

Transferring constants, internal variables, and Function Designators from one program to another.

Performer

A program which calls another via the PERFORM PROGRAM Statement.

Polling

Data acquisition; i.e., periodically sampling measurements from HIM's in the case of the GSE FEP's or acquiring measurement data from the PCM interface in the case of the PCM FEP's.

Prerequisite Sequence

A group of Control Logic statements which describe conditions which must be satisfied before a command may be issued. A Prerequisite Sequence is equivalent to software interlocks.

Present Value Of (PVO)

The value of a measurement at the current time.

Printer/Plotter

An electrostatic hardcopy device (or printer) located on the console and the Printer/Plotter Subsystem. The console printer/plotter is used for hardcopying the data on a DG CRT and for printing messages from system and application programs in the console. The printer/plotters on the Printer/Plotter Subsystem are used as a strip chart recorder.

Printer/Plotter Subsystem

The CCMS computer hardware and software required to generate strip chart printouts of measurement data in CCMS.

Procedural Statement

A statement which is executed sequentially in a step-by-step manner.

Procedure

An automated set of executable GOAL statements written by a GOAL user, acceptable to the language processor, and executable by the GOAL Executor. The term procedure is synonymous with GOAL program, GOAL applications program, applications procedure, and test procedure.

Procedure Error Override

A capability to pre-plan error handling by a GOAL procedure. If Procedure Error Override is activated, the GOAL Executor will output an error message and continue execution of the procedure upon detection of a Class III error. If Procedure Error Override is inhibited, the GOAL Executor will display an error message and stop the GOAL procedure in the event of a Class III error.

Processed Data Recorder (PDR)

The computer subsystem in CCMS which handles the logging of transactions within CCMS. Transactions which are logged to PDR are recorded to disk for near real-time recall and to magnetic tape for historical retrieval.

Processed Data Recorder (PDR) Recorder

The magnetic tape recording device on the Processed Data Recorder Subsystem.

Processing

Change checking, exception monitoring, and linearization performed by a FEP on acquired measurement data.

Programmable Function Key (PFK)

One of the push buttons on the DG keyboard used to generate an interrupt to a GOAL procedure or other applications program.

Programmable Function Panel (PFP)

The LED's and adjacent push buttons which may be used to display messages to GOAL procedures, turn on indicator lights, or generate interrupts to GOAL procedures.

Programmable Function Panel (PFP) Function Key

One of the push buttons on a PFP used to generate interrupts to application programs in the console.

Programmable Function Panel (PFP) Light

One of the lights located on the PFP Function Keys. A PFP Light may be turned ON or OFF, read, or verified from an applications program.

Programmable Function Panel (PFP) Presentation

One of the six PFP displays assigned to GOAL tasks. A PFP Presentation is stored on bulk memory and may be called up for viewing on a PFP by operator selection from a menu.

Prompt

A displayed message used to key the operator to enter a reply; i.e., to prompt the operator to reply. The RECORD AND SAVE REPLY AS option generates a prompt.

Pseudo Function Designator

A Function Designator which references a CDBFR address and may be read, verified, or altered by a GOAL procedure. Pseudo Function Designators have the properties of both stimuli and measurements.

Pulse Code Modulation Front End Processor (PCM FEP)

A computer in CCMS which processes a downlink PCM stream. Processing includes checking measurements for significant change, exception monitoring, linearizing measurements, and storing measurement changes in the CDBFR.

Quantity

A Floating point constant which may have an associated dimension; e.g., 5.2 volts, 55.55, and 100.50 PSI.

Reactive Sequence

A group of Control Logic statements which are executed when triggered by an exception condition. Reactive sequences are resident in bulk memory and provide much faster response time than GOAL procedures.

Read-BITE-After-Issue

The execution of a BITE (Built in Test Equipment) test after the issue of a command to an MDM.

Real Time

A. A term which refers to the present time

B. A reference to events which are currently executing in a computer

C. A type of programming which responds dynamically to external events.

Remote Communication (REM COM)

A term which refers to passing information to another console in CCMS.

Resident Data Area (RDA)

The part of the memory partition allocated to a task which contains the internal variables and interrupt blocks used by a program in the task.

Run Time

Execution time; the time at which a program is executed.

Scatter/Gather Read

The capability of the CDBFR Interface Processor to read a string of Function Designators from different CDBFR addresses based upon a single request from GOAL and return the collected values to GOAL.

Scratch Write

A special display to a CRT, which is not backed up on Bulk Memory. Scratch writes require less computer resources than displays to pages but have the disadvantage of being lost when another display page is selected for viewing.

Sensor

A measurement type Function Designator; e.g., Sensor Analog, Sensor Discrete.

Shared Peripheral Area (SPA)

One of the computer subsystems in CCMS which manages line printers and bulk tapes which it shares with the PDR. Post processing reduction programs may be run in the SPA to retrieve historical data from tape. Data from GOAL procedures may be printed as report data by specifying the SPA printer.

Shared Peripheral Area (SPA) Printer

A line printer, located on the SPA Subsystem, which may be used by GOAL procedures.

Significant Change

The amount a measurement must change before FEP will update its value in the CDBFR.

Skeleton

A permanent, disk-resident copy of a display, which may contain text and/or graphic characters, which may be displayed from a GOAL procedure, and updated from a GOAL procedure.

State

A constant which may take on the values: ON or OFF, OPEN or CLOSED, TRUE or FALSE, WET or DRY.

Subtype

The reference to the dimension type of a declared value; e.g., volts, ON/OFF, binary.

Symbolic Cursor

A reference to cursor position by means of a name rather than numerical values.

System Exception Monitor

The exception monitor which results in displays of exceptions to the console Exception Monitor Pages rather than routing of the exceptions to GOAL.

System Interrupt

Notification of an external event due to conditions within CCMS rather than notification of a measurement exception by a FEP. System Interrupts include measurement validity changes, CPU status changes, Class III errors, interrupt event occurrence in a higher level, and CDT count/hold status changes.

Systems Software Avionics Command Support (SACS)

An on-board software subsystem which provides the equivalent of TCS support in the ascent load.

Table

A grouping of data in a row and column format referred to by a name.

Task

A mainline program and all the lower level programs performed by it; a concurrency.

Test Control Supervisor (TCS)

An on-board executive which performs on-board test and checkout programs or ground-initiated TCS-1 commands.

Test Control Supervisor (TCS) Program/Sequence (TCS-S)

A program which is executed on board by the Test Control Supervisor. A TCS sequence may be called from Mass Memory (i.e., magnetic tape) on board or sent up the LDB from the ground for execution.

Test Control Supervisor (TCS) Response

Data generated for the ground by the Test Control Supervisor as a result of executing a TCS-1/SACS command or TCS sequence statement.

Text

A constant in GOAL which may be composed of characters or graphic symbols.

Time Homogeneous Read

A read of multiple words from the CDBFR which ensures by checking the lock word that all of the data is taken from a single time sample. The lock word indicates whether data is being updated in the CDBFR.

Timer

See Interval Timer.

Translator

A software component of the GOAL Language Processor, which translates the output of the compiler into interpretive code capable of being executed by the GOAL Executor.

Uplink Command Front End Processor

One of the computers in the CCMS which formats and issues commands to the on board system via the 72 kilobit uplink at the request of applications programs in the consoles.

Variable Data Area (VDA)

The part of the Resident Data Area in which internal variables used in a GOAL procedure are stored.

APPENDIX D. LIST OF ACRONYMS

THIS PAGE INTENTIONALLY LEFT BLANK.

AMP	Ampere
ASCII	American Standard Code for Information Interchange
BCD	Binary Coded Decimal
CCMS	Checkout, Control, and Monitor Subsystem
CDBFR	Common Data Buffer
CDS	Central Data Subsystem
CDT	Countdown Time
CPU	Central Processing Unit
CRT	Cathode Ray Tube
CTC	Computer-To-Computer
DEU	Display Electronics Unit
DG	Display Generator
EBCDIC	Extended Binary Coded Decimal Interchange Code
EIU	Engine Interface Unit
ECP	Explicitly Coded Program
ET	External Tank
FEP	Front End Processor
FIFO	First In-First Out
GLP	GOAL Language Processor
G-MEM	General Memory
GMT	Greenwich Mean Time
GOAL	Ground Operations Aerospace Language
GPC	General Purpose Computer
GSE	Ground Support Equipment
HIM	Hardware Interface Module
ID	Identification
I/O	Input/Output
IOS	Input Output Supervision
JTOY	Julian Time of Year
KBS	Kilobits per Second
KSC	Kennedy Space Center
LDB	Launch Data Bus
LED	Light Emitting Diode
LPS	Launch Processing System
ME	Main Engine
MEC	Main Engine Controller Master Events Controller
MET	Mission Elapsed Time
OFI	Operational Flight Instrumentation
PBIC	Programmable Buffer Interface Card
PCM	Pulse Coded Modulation
PDR	Processed Data Recorder
PFK	Programmable Function Key
PFP	Programmable Function Panel
PWA	Private Write Area
RDA	Resident Data Area

SACS	Systems Software Avionics Command Support
SPA	Shared Peripheral Area
TBD	To Be Determined
TCS	Test Control Supervisor
V	Volt
VDA	Variable Data Area
V&DA	Video and Data Assembly

APPENDIX E. FEP EXCEPTION RE-REPORTING

THIS PAGE INTENTIONALLY LEFT BLANK.

E.1 INTRODUCTION

This appendix is to be used as a general reference to the problems which can occur when conditions exist in a FEP which result in re-reporting of Function Designator exception conditions. GOAL and Control Logic procedure writers should be aware of the possibility of receiving multiple exception notifications on a Function Designator for a single out-of-tolerance condition, and code their procedures in such a way that this condition will not result in undesirable actions being taken.

This appendix consists of several sections: a high-level overview which explains why these problems occur; detailed examples of the problems as they exist for the various FEP types; and finally, a list of suggestions which will help a procedure writer avoid these problems.

E.2 HIGH LEVEL OVERVIEW

In normal FEP operations, a GOAL procedure can activate exception notification for Control Logic, Exception Monitoring, or GOAL. If an exception condition is detected by the FEP, an exception message is sent to the responsible console, indicating which type of exception(s) occurred. If a GOAL exception is detected, the FEP will automatically inhibit GOAL notification on the Function Designator, preventing any further notifications from being reported by the FEP. If a Control Logic or Exception Monitor (EMON) exception is detected, the FEP will not inhibit Control Logic or Exception Monitor notification on the Function Designator. This condition will result in an exception being sent by the FEP for each transition of the Function Designator from out-of-tolerance to in-tolerance.

The processing of a particular Function Designator by the FEP is controlled by a Measurement Descriptor Table (MDT). The structure of the FEP's MDT's are such that, in certain particular situations, exceptions can be reported more than once by the FEP for a single out-of-tolerance condition. This "re-reporting" of a Control Logic exception, for example, can result in the Function Designators associated Control Logic Reactive Sequence being executed again. A "re-reported" GOAL exception would result in the GOAL procedure receiving a duplicate interrupt on the Function Designator. Depending on exactly how the Control Logic Reactive Sequence/GOAL procedure is written, this duplicate interrupt could have undesirable results. "Re-reported" EMON exceptions will result induplicate messages on the responsible console's EMON page(s). This appendix will attempt to describe

those situations where the FEP "re-reports" exception conditions. It is divided into a section describing those conditions which affect GSE FEP's only, followed by a section which describes conditions which affect all FEP's.

E.3 GSE EXCEPTION RE-REPORTING CONDITIONS

Effectivity: All flows, F3X and beyond.

Typical Scenario:

A. Notification is activated on a Function Designator for a particular routing (Control Logic or EMON).

B. An out-of-tolerance exception is generated on the Function Designator for the notification activated in Step 1.

C. The [Function] is inhibited (Processing or Polling).

D. Notification is activated on the Function Designator for the other routing (EMON or Control Logic).

E. The [Function] inhibited in Step 3 is re-activated.

F. An out-of-tolerance exception is generated for the notification activated in Step 4.

G. An out-of-tolerance exception is generated for the notification activated in Step 1. This is the "re-reported" exception.

Assumptions: The Function Designator is out-of-tolerance at the start of the scenario, and remains out-of-tolerance throughout the scenario.

Cause: Combined Control Logic and EMON status fields in the Measurement Descriptor Table (MDT).

A. GSE FEP's can re-report exceptions due to the structure of the FEP's MDT's.

Analog MDTs contain a combined Control Logic/EMON Exception Status word, which does not allow for separate activate/inhibit of Control Logic/EMON notifications to be detected by the GSE FEP microcode. This prohibition can result in both Control Logic and EMON exceptions being "re-reported" when notification is activated and

the Function Designator is out-of-tolerance. GOAL exceptions are not "re-reported" on activate notification requests. (GOAL notification would be auto-inhibited following the original exception, and the exception generated on a subsequent activate GOAL request is not considered a "re-reported" exception).

Discrete MDT's contain a combined Control Logic/EMON/GOAL exception status word, which does not allow for separate activate/inhibit of any of the three types of notification to be detected by the GSE FEP microcode. This prohibition can result in "re-reporting" of Control Logic and EMON exceptions when notification is activated and the Function Designator is out-of-tolerance. GOAL exceptions are not re-reported in activate requests. (GOAL notification would be auto-inhibited following the original exception, and the exception generated on a subsequent activate GOAL request is not considered a "re-reported" exception).

B. The following table lists those functions which, if inhibited when notification is activated, will result in exceptions being "re-reported" when the function is re-activated. These functions will be referred to as [Function] in the example sequences given on the following pages.

[FUNCTION]	GOAL STATEMENT	COMMAND PROCESSOR
FEP Data Aquisition Inhibit	N/A	I DA <FEP NAME>
FEP Processing Inhibit	N/A	I PR <FEP NAME>
FD Processing Inhibit	INHIBIT PROCESSING FOR...	I PR <FD>
FD Polling Inhibit	INHIBIT POLLING FOR...	I DA <FD>
HIM Polling Inhibit	N/A	I HIM <HIM #X FEP NAME>

E.3.1 GSE ANALOG RE-REPORTING PROBLEMS

A. Only affected when activating Control Logic or EMON

B. No problem when activating GOAL

C. Function Designator is currently out-of-tolerance

D. Scenario:

1. Activate Function Designator notification for Control Logic or EMON.

   Exception sent (Function Designator out-of-tolerance).

2. Inhibit [Function]

3. Activate Function Designator notification for the other type EMON or Control Logic.

   No action here since [Function] inhibited on Function Designator.

4. Activate [Function]

   Exception sent for notification activated in Step 3.

   Exception "re-reported" for notification activated in Step 1.

E.3.2 GSE DISCRETE RE-REPORTING PROBLEMS

E.3.2.1 Single Discrete In A Group (Only 1 discrete function designator is defined for that CDBFR location)

A. Affected when activating any notification routing

B. Function Designator is currently out-of-tolerance

C. Control Logic/EMON and GOAL exception states (normal states) are the same.

D. Scenario:

1. Activate Function Designator notification for CL or EMON or GOAL.

Exception sent (Function Designator out of tolerance).

2. Inhibit [Function]

3. Activate Function Designator notification for another type EMON or CL or GOAL.

   No action here since [Function] inhibited on Function Designator.

4. Activate [Function]

   Exception sent for notification activated in Step 3.

   Exception "re-reported" for notification activated in Step 1[1].

E.3.2.2 Packed Discretes (Multiple discretes function designators are defined for that CDBFR location)

EXAMPLE 1

A. Affected when activating any notification routing

B. Function Designator is currently out-of-tolerance

C. Control Logic/EMON and GOAL exception states (normal states) are the same.

D. Scenario:

   1. Activate Function Designator notification on <BIT ZERO> for Control Logic or EMON.

      Exception sent (Function Designator out-of-tolerance).

   2. Inhibit Function Designator processing
      or
      Inhibit Function Designator polling

   3. Activate Function Designator notification for <BIT ZERO> for EMON or Control Logic or GOAL (other notification).

1. Will not be "re-reported" if GOAL is activated in Step 1.

No action since Function Designator polling/ processing inhibited.

4. Activate Function Designator processing/polling.

   Exception sent for <BIT ZERO> notification activated in Step 3.

   Exception re-reported for <BIT ZERO> notification activated in Step 1 [1].

EXAMPLE 2

A. Affected when activating any notification routing.

B. Function Designator is currently out-of-tolerance.

C. Control Logic/EMON and GOAL exception states (normal states) are the same.

D. Scenario:

   1. Activate Function Designator notification for <BIT ZERO> for Control Logic or EMON or GOAL.

      Exception sent (Function Designator out-of-tolerance).

   2. Inhibit FEP processing
      or
      Inhibit FEP polling

   3. Activate Function Designator notification for <BIT ONE> for Control Logic or EMON or GOAL.

      No action since [Function] inhibited.

   4. Activate FEP polling/processing and <BIT ONE> is out-of-tolerance.

      Exception sent for <BIT ONE> that was activated in Step 3.

      Exception re-reported for <BIT ZERO> that was activated in Step 1[1].

1. Will not be "re-reported" if GOAL is activated in Step 1.

E.3.2.3 Single or Packed Discretes

EXAMPLE 1

A. Affected when activating any notification routing.

B. Function Designator is currently out-of-tolerance.

C. Control Logic/EMON and GOAL exception states (normal states) are different.

D. Scenario:

1. Notification is currently active on some Function Designators (bits) for EMON/Control Logic/GOAL and exceptions sent.

2. Inhibit [Function]

3. Activate GOAL on a Function Designator (bit) that reported an exception for Control Logic or EMON (Control Logic/EMON and GOAL exception states (normal states) are different).

4. Activate [Function]

   Get exception on notification activated in Step 3 and Step 1 for the particular Function Designator (bit) activated in Step 3.

   "Re-report" exceptions, for all other Function Designators (bits) sent in Step 1[1].

EXAMPLE 2

A. Affected when activating any notification routing.

B. Function Designator is currently out-of-tolerance.

C. Control Logic/EMON and GOAL exception states (normal states) are different.

D. Scenario:

1. Notification is currently active on some Function Designators (bits) for EMON/Control Logic/GOAL and exceptions sent.

1. Will not be "re-reported" if GOAL is activated in Step 1.

2. Inhibit [Function]

3. Activate Control Logic or EMON on some Function Designator (bit) that reported an exception for EMON or Control Logic - but GOAL is not active for that Function Designator (bit) even though exception states differ.

4. Activate [Function]

   Get (Control Logic or EMON) exceptions for Step 3 and Step 1 for the bit activated in Step 3. No other Function Designators (bits) in this group re-reported in this case.

E.3.2.3.1 Summary for GSE Discretes when Control Logic/EM and GOAL Exception States (Normal States) Differ

A. Activate notification on GOAL when Control Logic/EMON is out-of-tolerance and exception state (normal state) differs and [Function] inhibited - you will get "re-reporting" of all the EMON/Control Logic in that group when the [Function] is re-activated.

B. If exception state (normal state) differs for a Function Designator (bit) but only Control Logic/EMON is active (and Function Designator is out-of-tolerance) - any EMON/Control Logic activation on any Function Designator (bit) in the group when a [Function] is inhibited will "re-report" Control Logic/EMON for that Function Designator (bit) only when the [Function] is re-activated.

C. GOAL exceptions are never "re-reported".

E.3.3 GSE SWITCHOVER PROBLEMS

EXAMPLE 1

A. Affects analog and discrete Function Designators.

B. Affects all notification routings.

C. Function Designators are currently out-of-tolerance.

D. Scenario:

1. Active/Standby GSE FEP switchovers which occur when [Function(s)] are inhibited will result in

"re-reporting" of exceptions when [Function(s)] are re-activated as noted in cases previously listed.

2. Active/Standby GSE FEP switchovers which occur at any other time will not result in extra exception reporting except in the following condition:

   Active/Standby GSE FEP switchover occurs during the data synchronization process, before standby FEP completes updates to its tables.

EXAMPLE 2

A. Affects discrete function designators.

B. Control Logic/EMON and GOAL limits are different.

C. Control Logic/EMON and GOAL notification are active.

D. Function Designator is currently out of Control Logic/ EMON limits.

E. Scenario:

   1. GSE Active/Standby Switchover occurs.

      EMON is re-reported.

EXAMPLE 3

A. Affects discrete function designators.

B. Control logic/EMON and GOAL limits are different.

C. Control Logic/EMON and GOAL notification are currently inhibited but were active when limits were different.

D. Function Designator is currently out of Control Logic/ EMON limits.

E. Scenario:

   1. GSE Active/Standby Switchover occurs.

   2. Activate GOAL/EMON Exception Reporting.

      Measurement is out of EMON limits so an EMON is re-reported.

3. An additional re-report will occur if any of the following actions are performed.

   Data Acquisition or Processing is inhibited and then activated.

   An Activate HIM command (no previous Inhibit HIM required) is executed.

   $SWSCAN is run. $SWSCAN executes an Activate HIM command for all HIMS in a GSE FEP. $SWSCAN is run periodically by LPS operations when LPS is supporting Hardware testing.

E.4 COMMON FEP EXCEPTION RE-REPORTING CONDITIONS

Effectivity: All GSE, PCM, SDB, and UCS FEP's; F3X and beyond.

Typical Scenario:

A. FEP tables contain some MDT's (Function Designators) which are currently out-of-tolerance and have exception notification active. (Exceptions have been generated as appropriate).

B. FEP tables are "checkpointed" via the Checkpoint Restart Program ($FEPCPR).

C. FEP tables are reloaded via the table load programs ($CLAI, $TOC).

D. Data acquisition is restarted on the FEP (A DA).

E. All out-of-tolerance conditions which still exist, for Function Designators which have console notification active in the checkpointed tables, will be "re-reported" to the appropriate responsible consoles.

Cause: Exception status fields in the MDT are not checkpointed. GOAL notification indicators may have been checkpointed prior tothe auto-inhibit of GOAL when the exception was sent by the FEP.

A. Checkpointed Table LOADS via $CLAI or $TOC.

   The loading of checkpointed FEP Tables which contain Function Designators which have console notification active for Control Logic, EMON or GOAL will re-report

exceptions if the Function Designator is out-of-tolerance at the time the check-pointed tables are loaded. This condition occurs due to the fact that the "last state" fields in the MDT's are not checkpointed, resulting in all MDT "last states" being set to in-tolerance. The microcode will detect all out-of-tolerance conditions the first time the MDT is polled following the re-load of the checkpointed tables, and report all exceptions as if they just occurred, even though the data did not change.

Likewise, the GOAL auto-inhibit field is not checkpointed, resulting in GOAL being active in the checkpointed tables, although it was deactivated in the FEP when the first exception was reported. This condition can result in re-reporting of GOAL exceptions when checkpointed tables are loaded, if the MDT data is out-of-tolerance, at the time the tables are reloaded.

E.5 SUGGESTED METHODS FOR MINIMIZING THE IMPACTS OF RE-REPORTED EXCEPTIONS

A. Code Control Logic/GOAL procedures in such a way that a re-invocation of a Control Logic Reactive Sequence or a duplicate interrupt to a GOAL procedure will not produce undesirable results.

Verify that the FD is actually out-of-tolerance (Control Logic only), through the use of an exception status option of a "VERIFY PREFIX" Control Logic statement.

Set flags to skip processing of the exception (within the procedure) if the exception has already been processed.

B. When activating console notification on a Function Designator:

Ensure that processing and polling functions are active on the Function Designator by reading pseudo Function Designators for the FEP's data acquisition and processing status, and by use of the "READ FEP STATUS" GOAL statement for the Function Designator's data acquisition and processing status. If any of the functions are inhibited, do not activate notification on the Function Designator.

C. When re-loading checkpointed tables:

If possible, terminate all GOAL procedures which could receive exceptions from the FEP being reloaded.

Be prepared to see EMON out-of-tolerance exceptions reported on the EMON page(s) (The FEP's should not re-report any EMON intolerance exceptions as a result of a checkpoint table reload).

Take defensive action to prepare for any duplicate Control Logic exceptions which could invoke Control Logic Reactive sequences.

INDEX

INDEX (Continued)

INDEX (Continued)

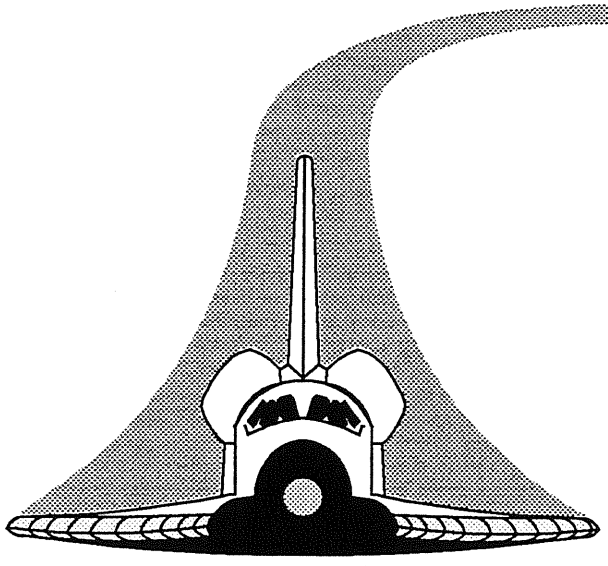
INDEX (Continued)

INDEX (Continued)

THIS PAGE INTENTIONALLY LEFT BLANK.

KSC-LPS-OP-033-03
June 1995
Baseline
S E3 Release

LPS CCMS GOAL LANGUAGE QUICK REFERENCE GUIDE Part 2



LPS SYSTEM
SOFTWARE
DOCUMENTATION

Prepared By:

Lockheed Martin Space Operations
Integrated Data Systems
LPS Software Quality and Data Analysis
LPS Documentation

Contract NAS 10-10900

Send corrections to this document to: Lockheed Martin Space Operations, ATTN: LPS Documentation, LSO-252,
1100 Lockheed Way, Titusville, FL 32780

UPDATE HISTORY

Baseline S E3	June 1995
There were no technical updates incorporated into this document for the S E3 release. However, this document is being issued and baselined under the S E3 release level as a result of format changes driven by the requirement to convert to FrameMaker publication software.	

THIS PAGE INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK.

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

THIS PAGE INTENTIONALLY LEFT BLANK.

1. GOAL SYNTAX DIAGRAM INDEX

THIS PAGE INTENTIONALLY LEFT BLANK.

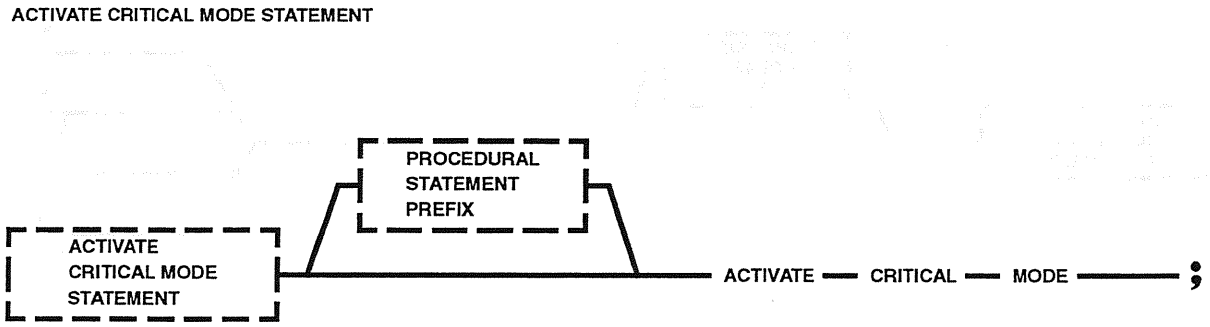


Figure 1-1 Activate Critical Mode Statement

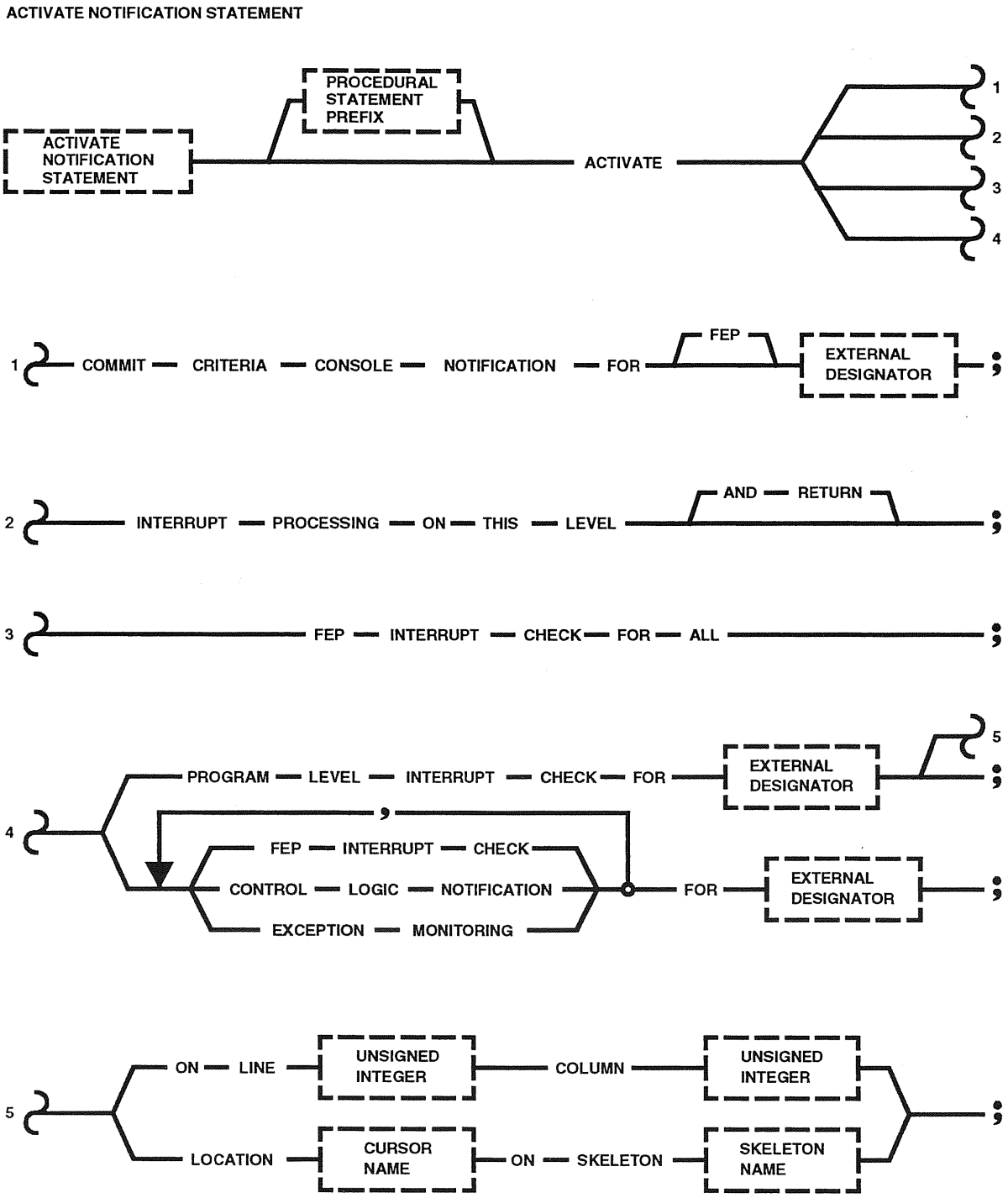


Figure 1-2 Activate Notification Statement

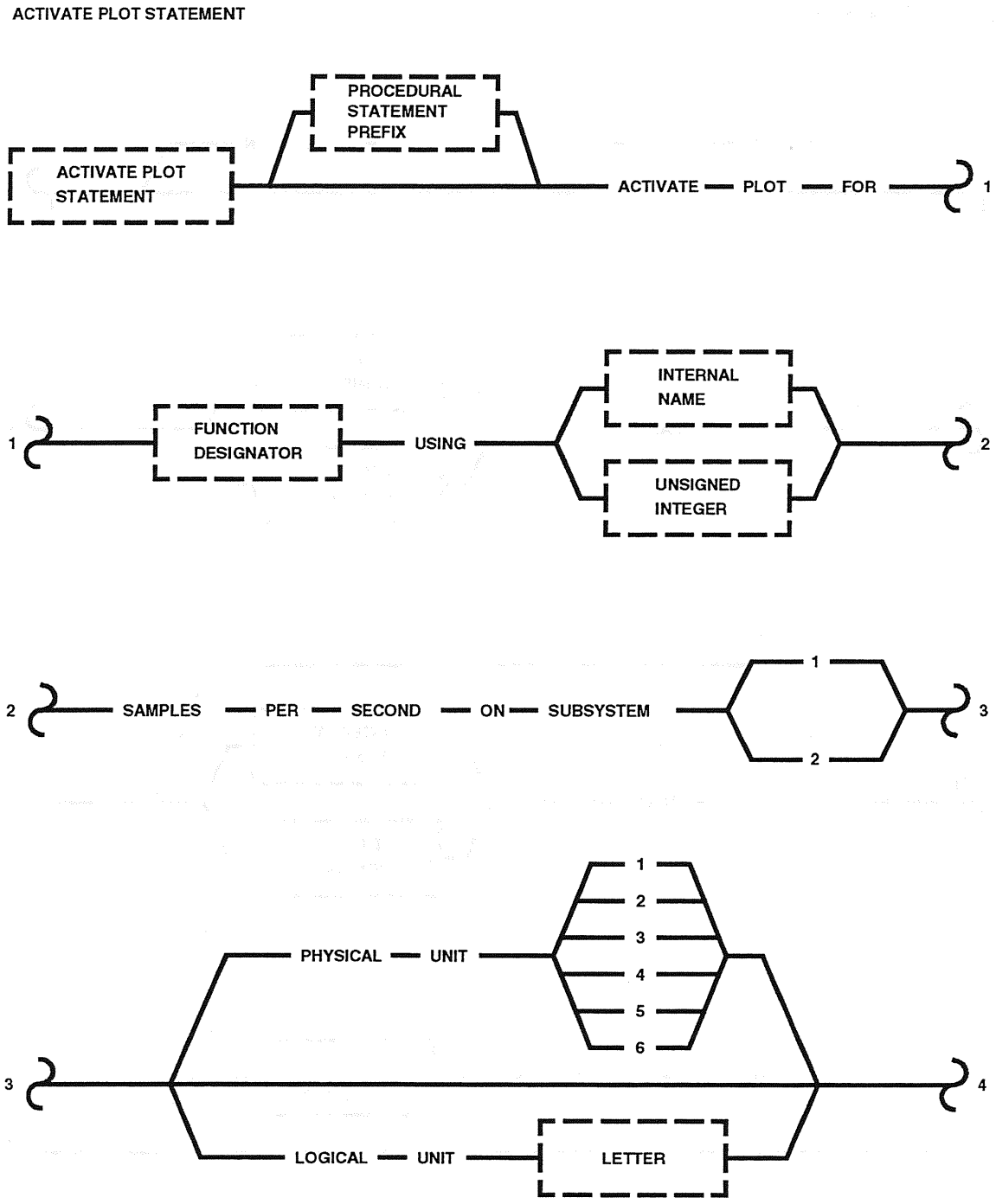


Figure 1-3 Activate Plot Statement (1 of 2)

ACTIVATE PLOT STATEMENT

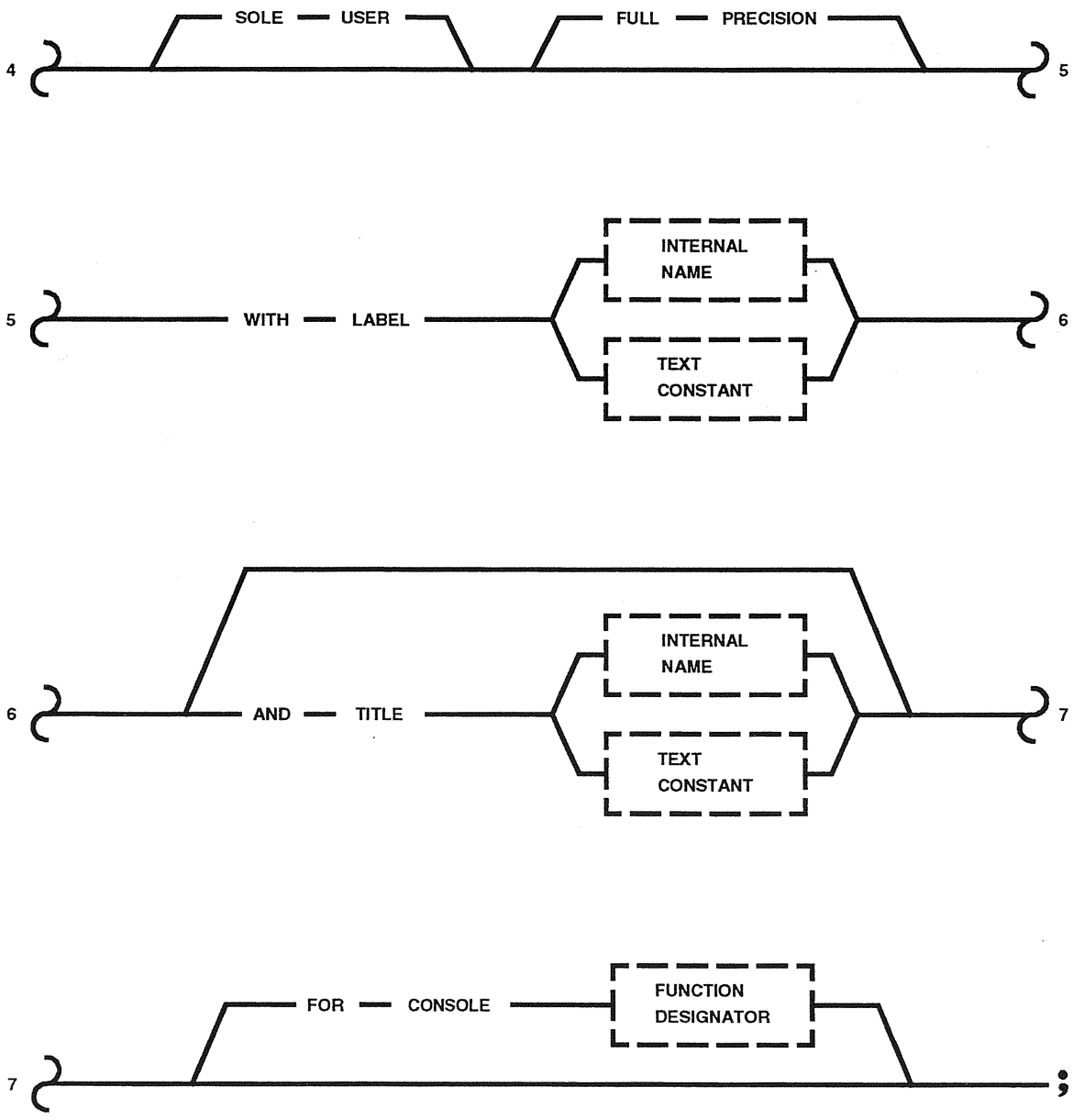


Figure 1-3 Activate Plot Statement (2 of 2)

ACTIVATE SYSTEM STATEMENT

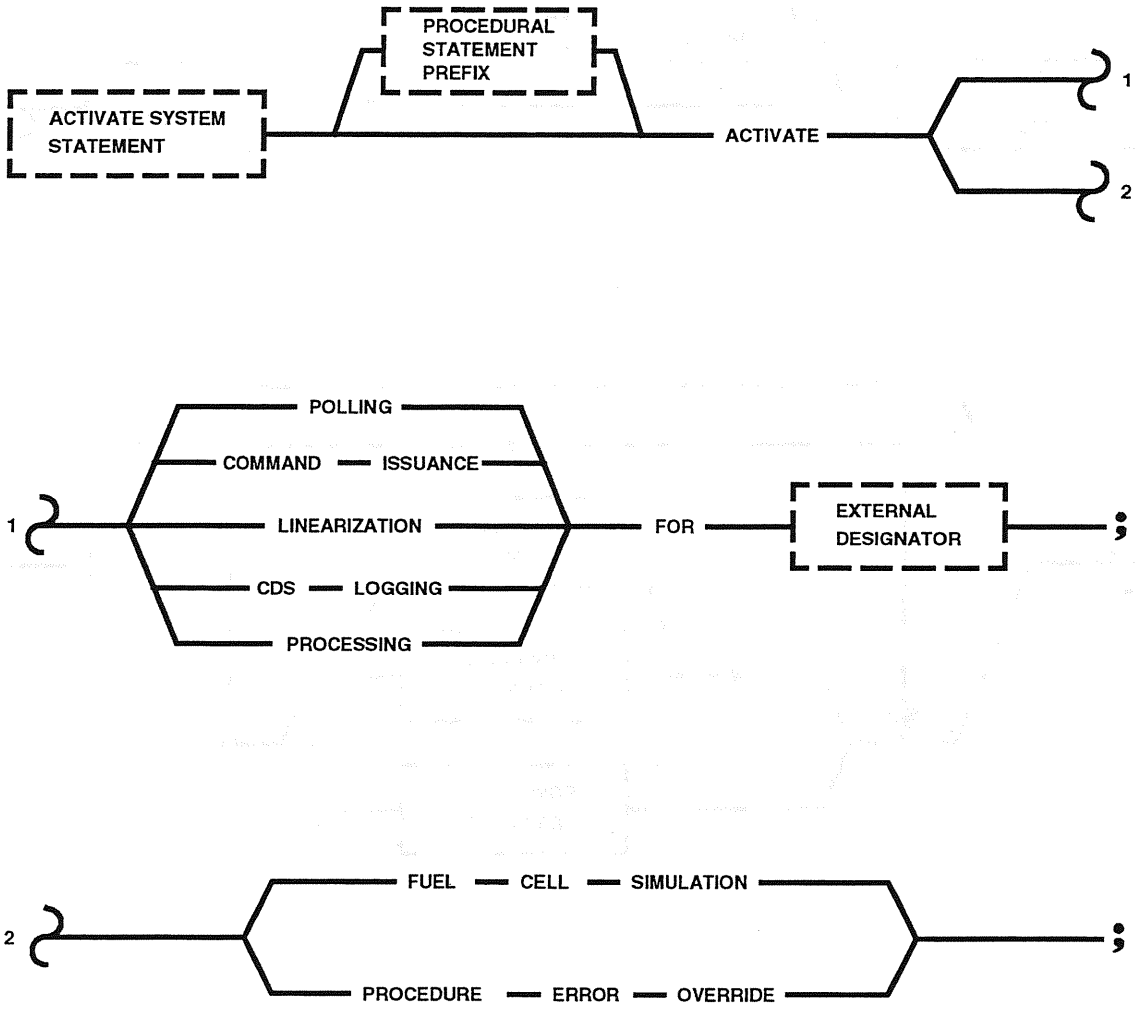


Figure 1-4 Activate System Statement

ACTIVATE TABLE STATEMENT

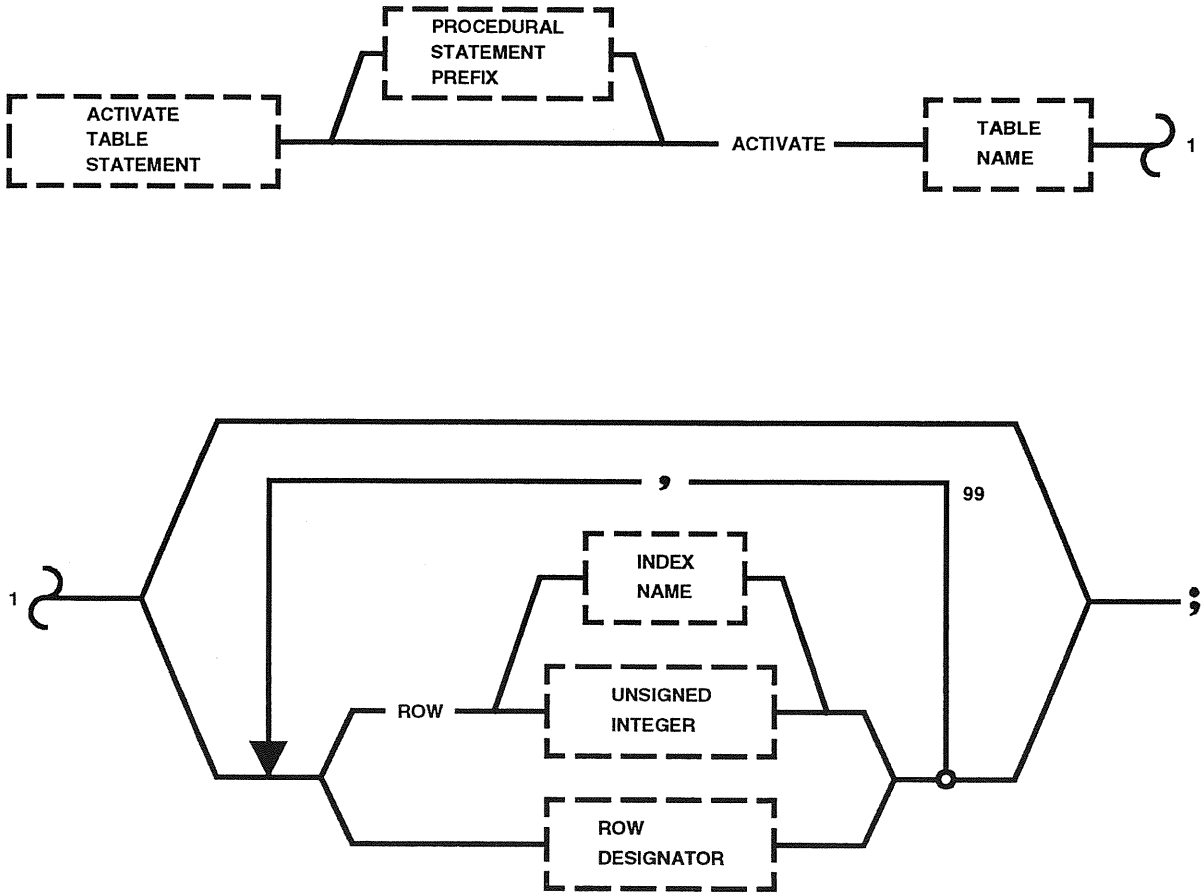


Figure 1-5 Activate Table Statement

ALLOCATE DISK FILE STATEMENT

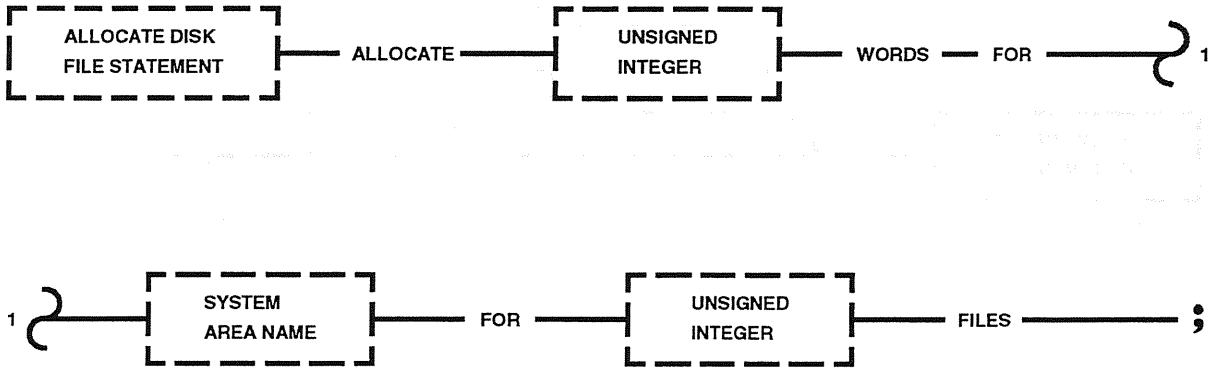


Figure 1-6 Allocate Disk File Statement

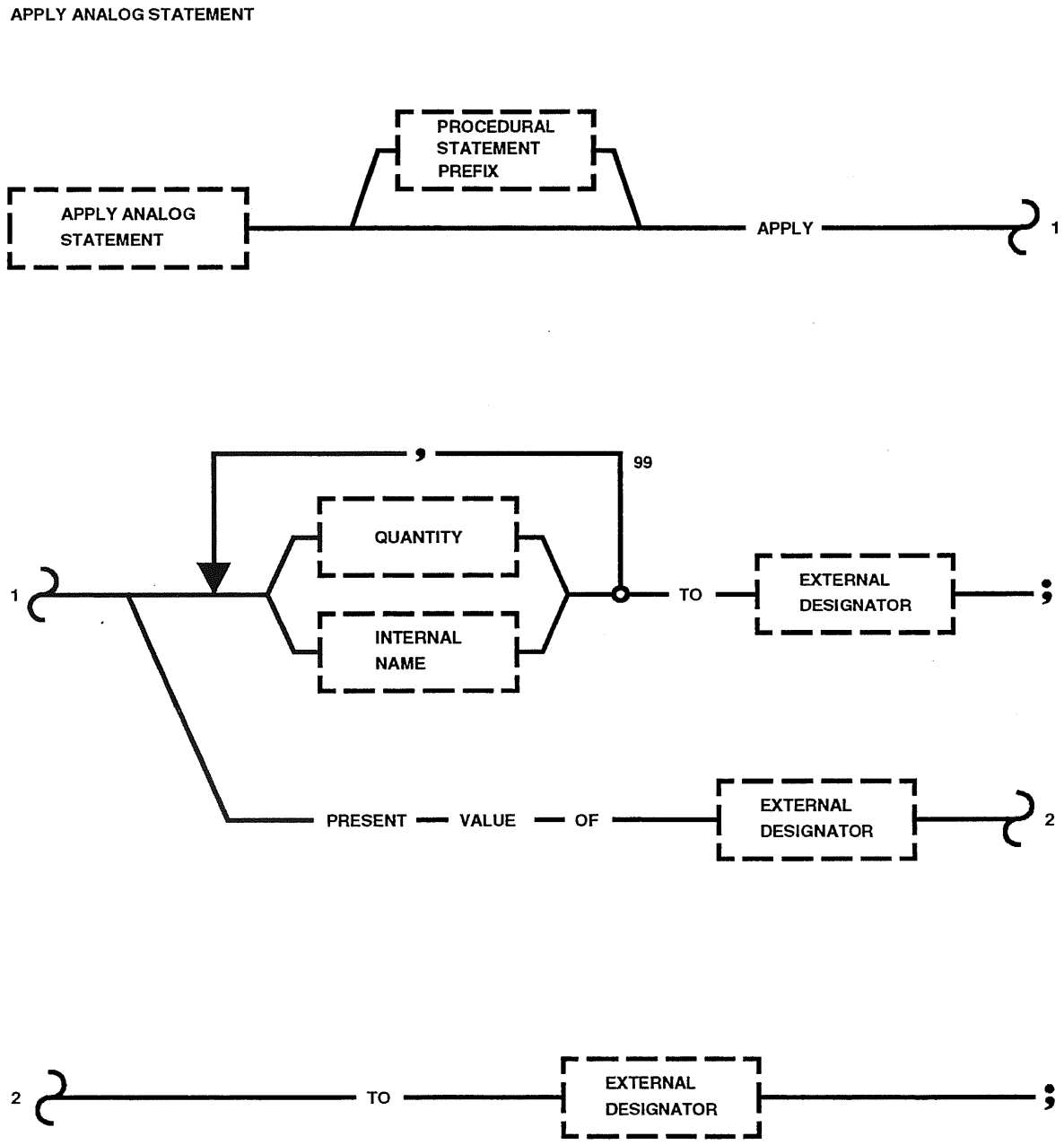


Figure 1-7 Apply Analog Statement

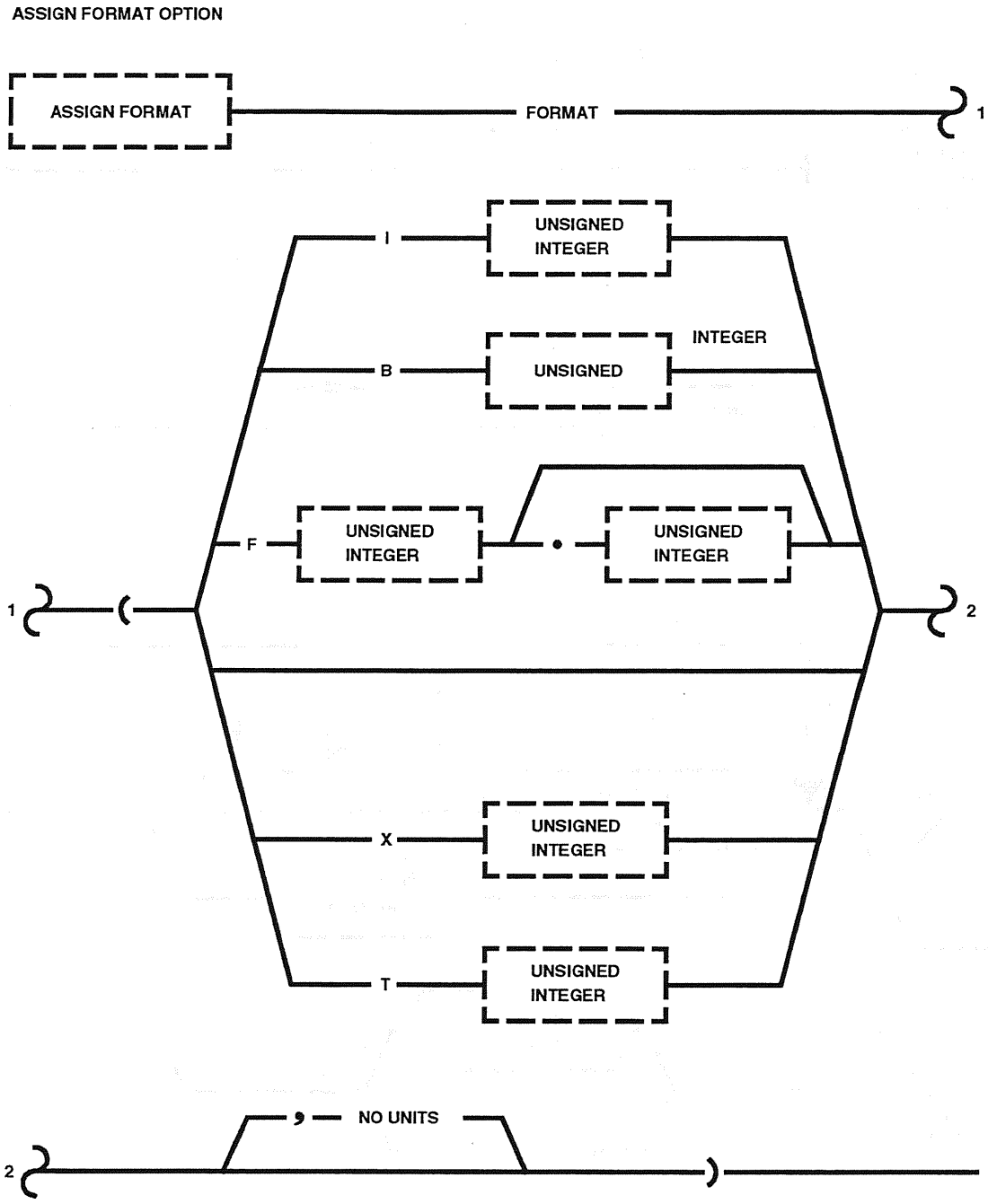


Figure 1-8 Assign Format Option

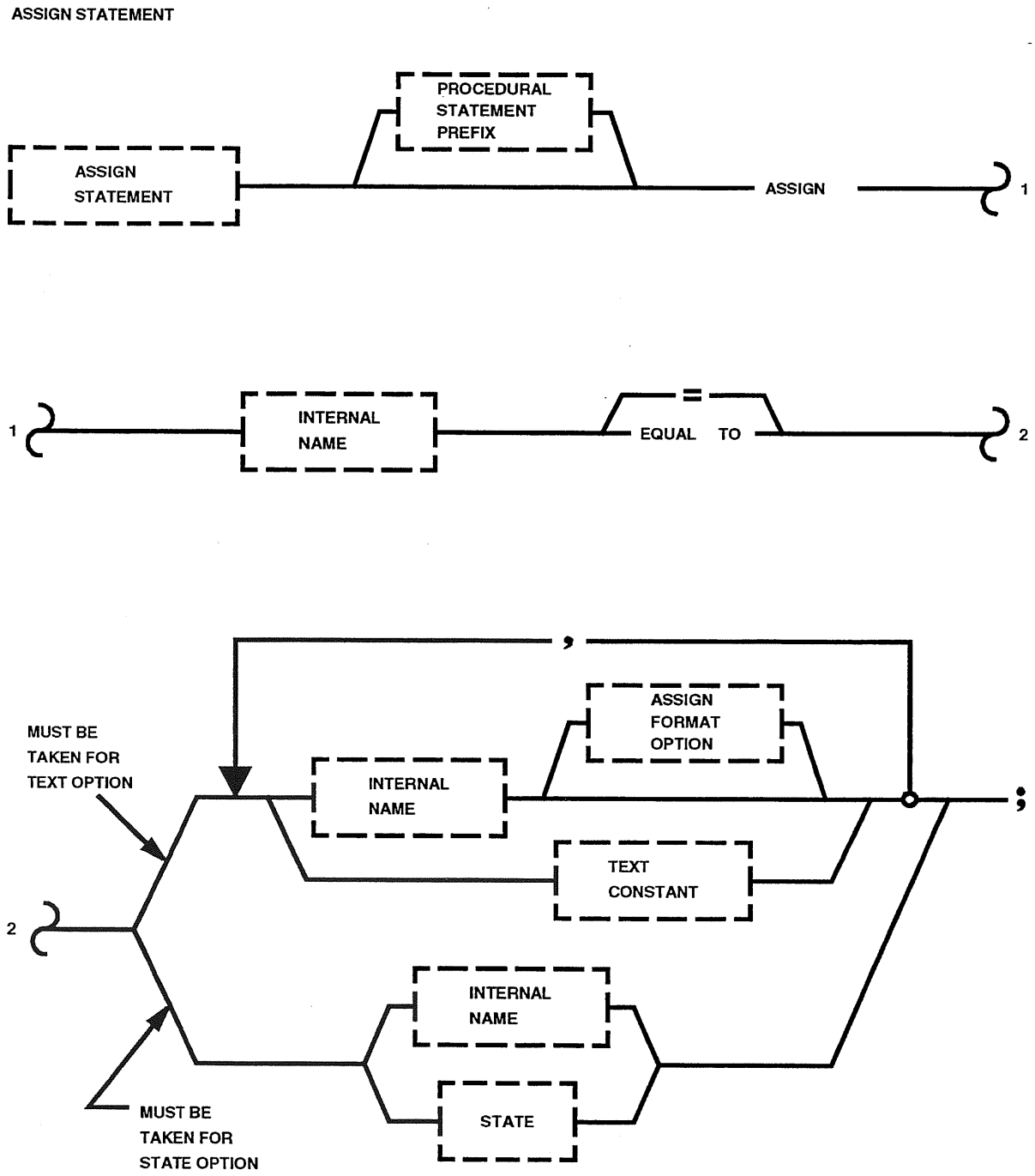


Figure 1-9 Assign Statement

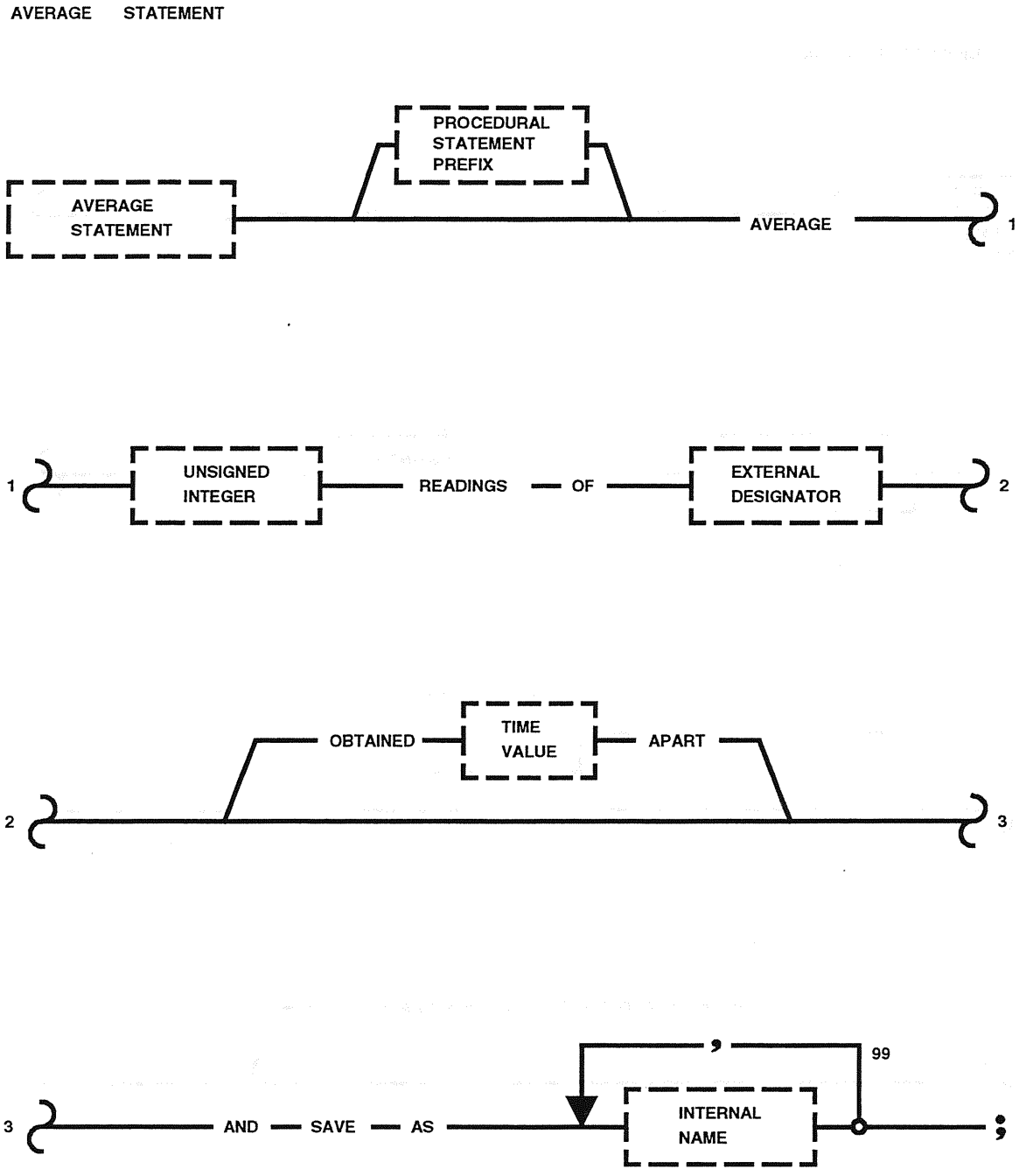


Figure 1-10 Average Statement

BEGIN DISK FILE DEFINITION STATEMENT

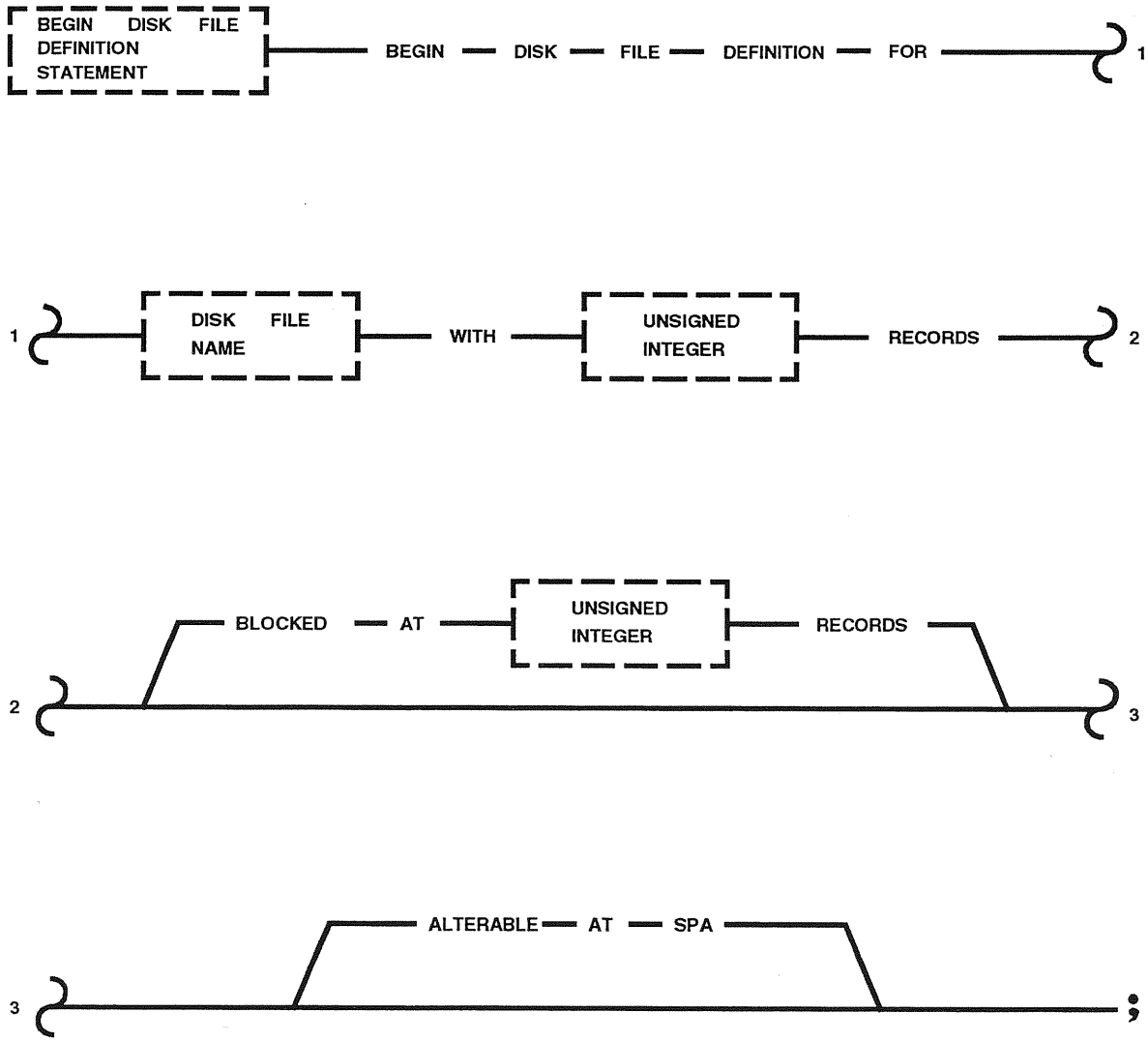


Figure 1-11 Begin Disk File Definition Statement

BEGIN DISK FILE RECORD STRUCTURE DEFINITION STATEMENT

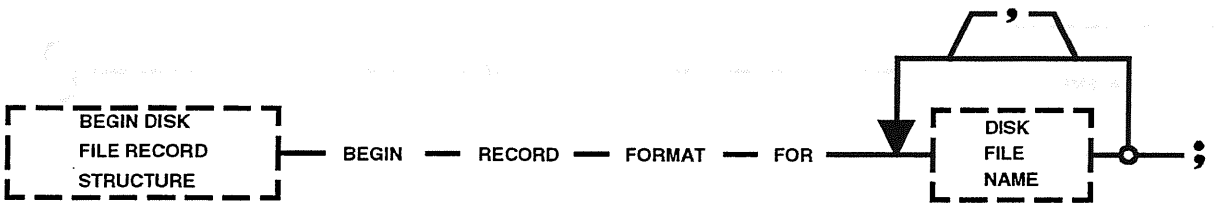


Figure 1-12 Begin Disk File Record Structure Definition Statement

BEGIN MACRO STATEMENT

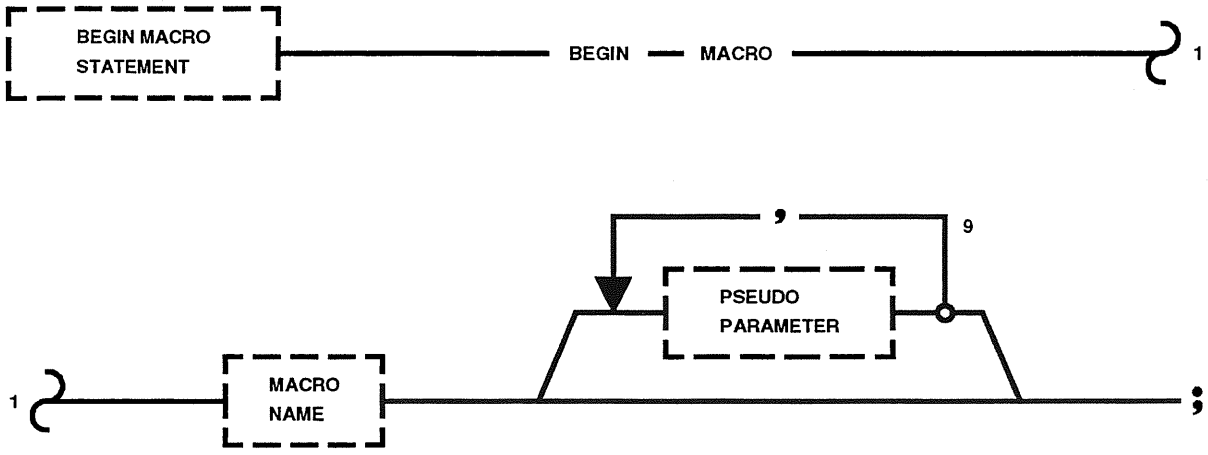


Figure 1-13 Begin Macro Statement

BEGIN PROGRAM STATEMENT

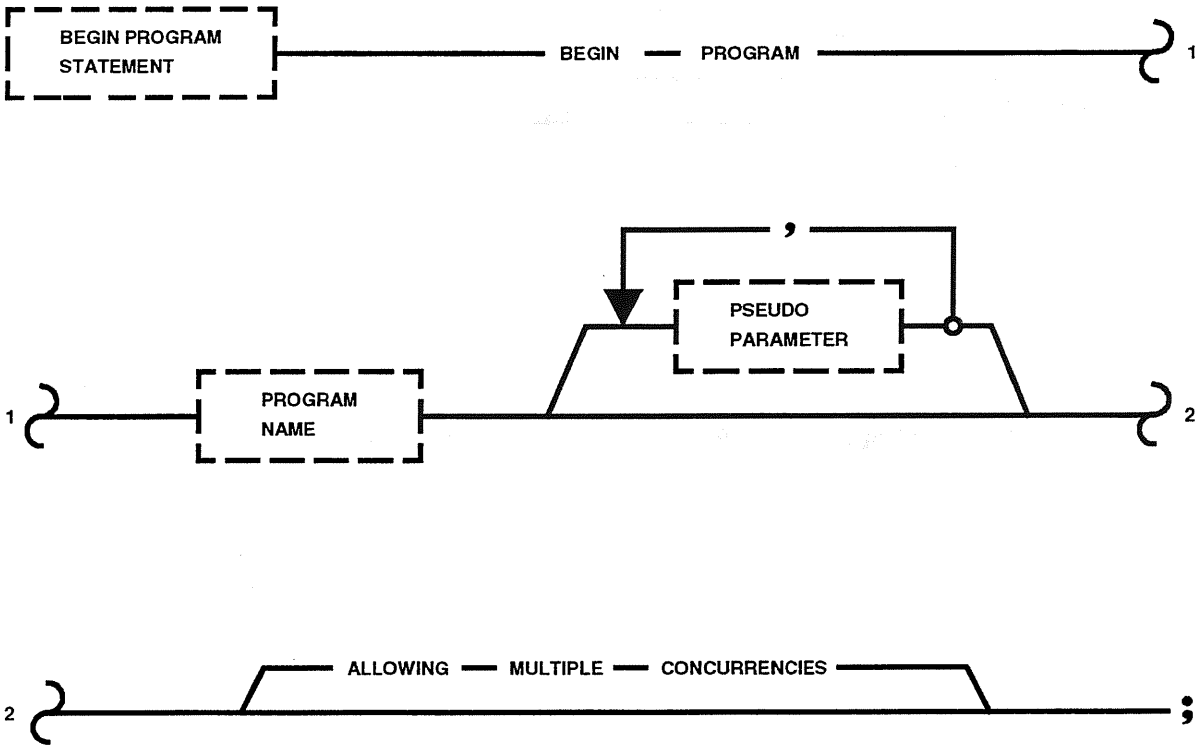


Figure 1-14 Begin Program Statement

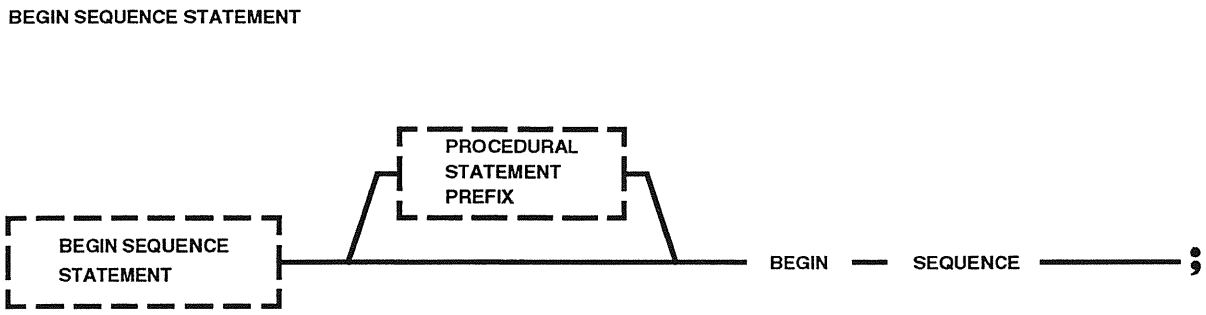


Figure 1-15 Begin Sequence Statement

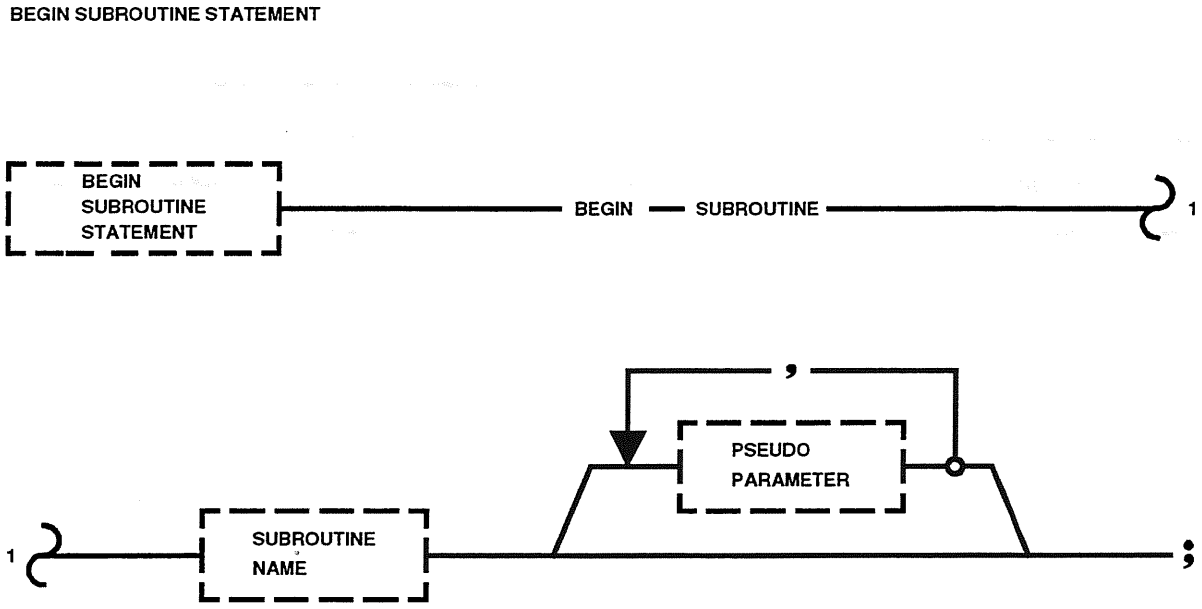


Figure 1-16 Begin Subroutine Statement

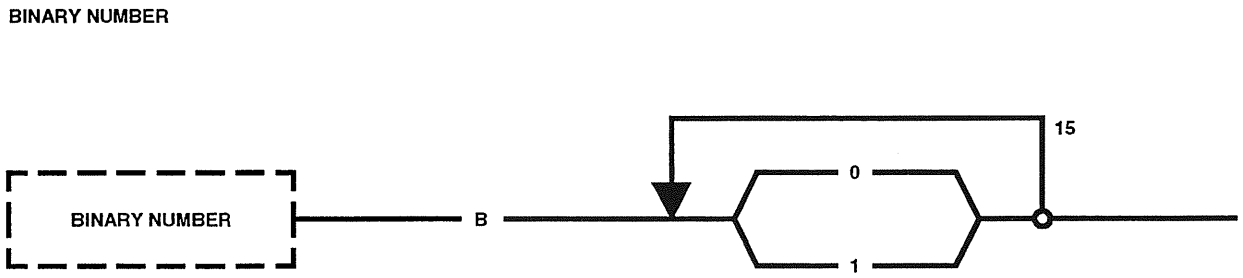


Figure 1-17 Binary Number

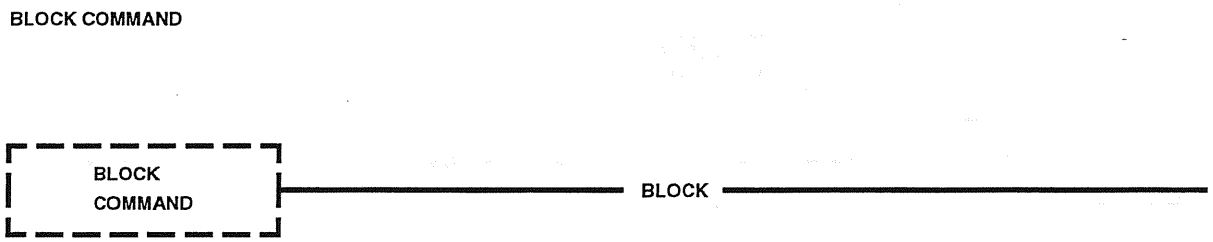
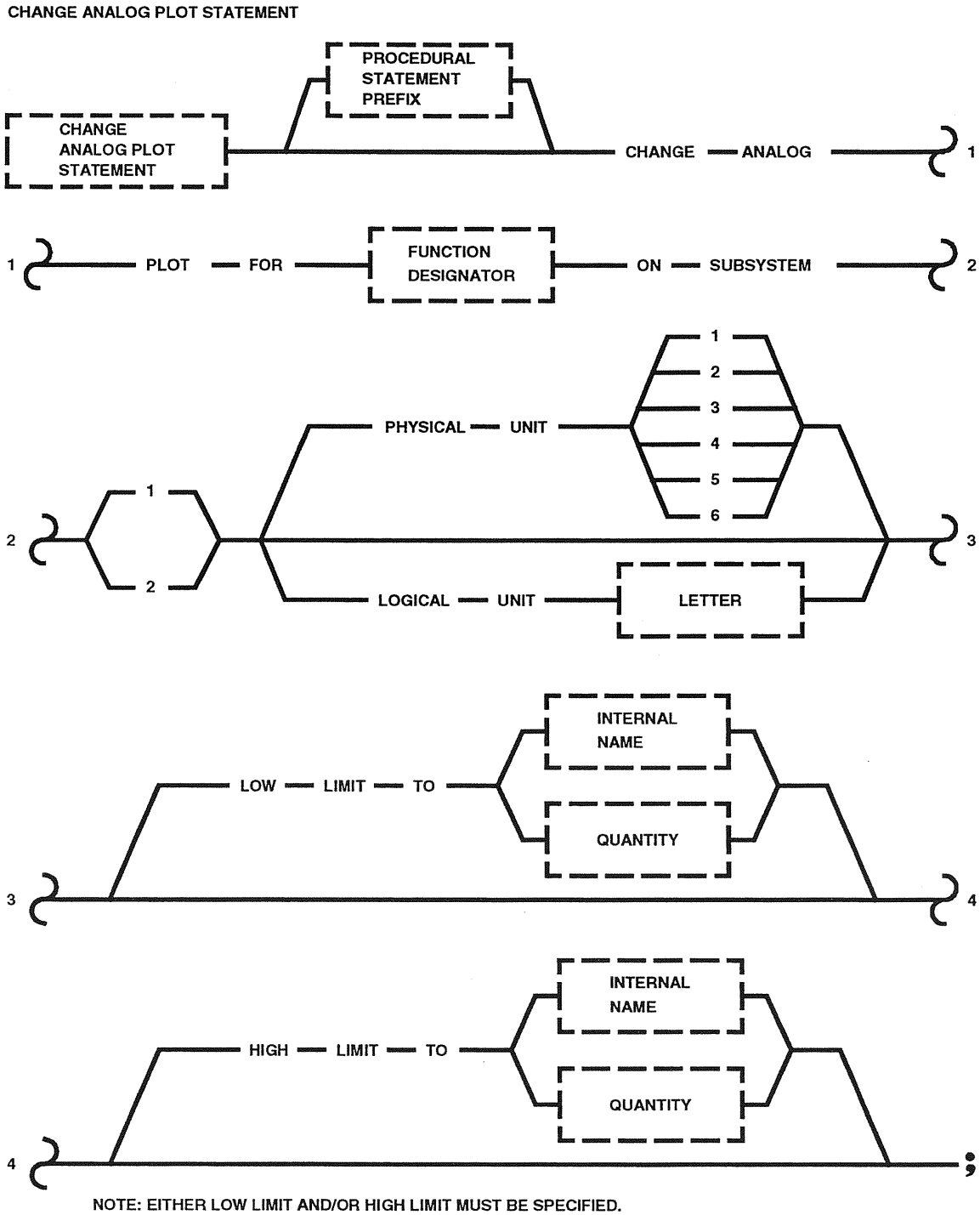


Figure 1-18 Block Command



NOTE: EITHER LOW LIMIT AND/OR HIGH LIMIT MUST BE SPECIFIED.

Figure 1-19 Change Analog Plot Statement

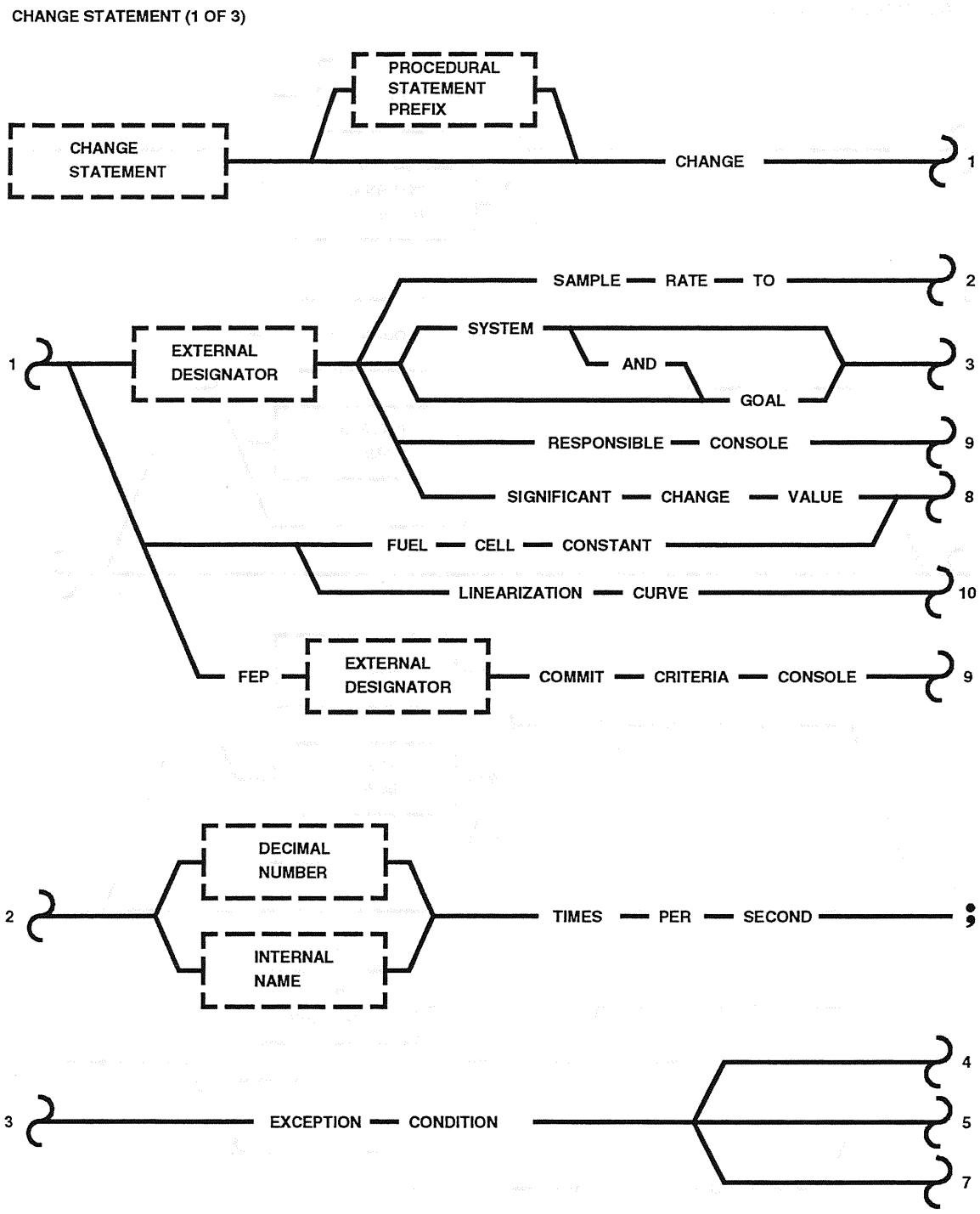


Figure 1-20 Change Statement (1 of 3)

CHANGE STATEMENT (2 OF 3)

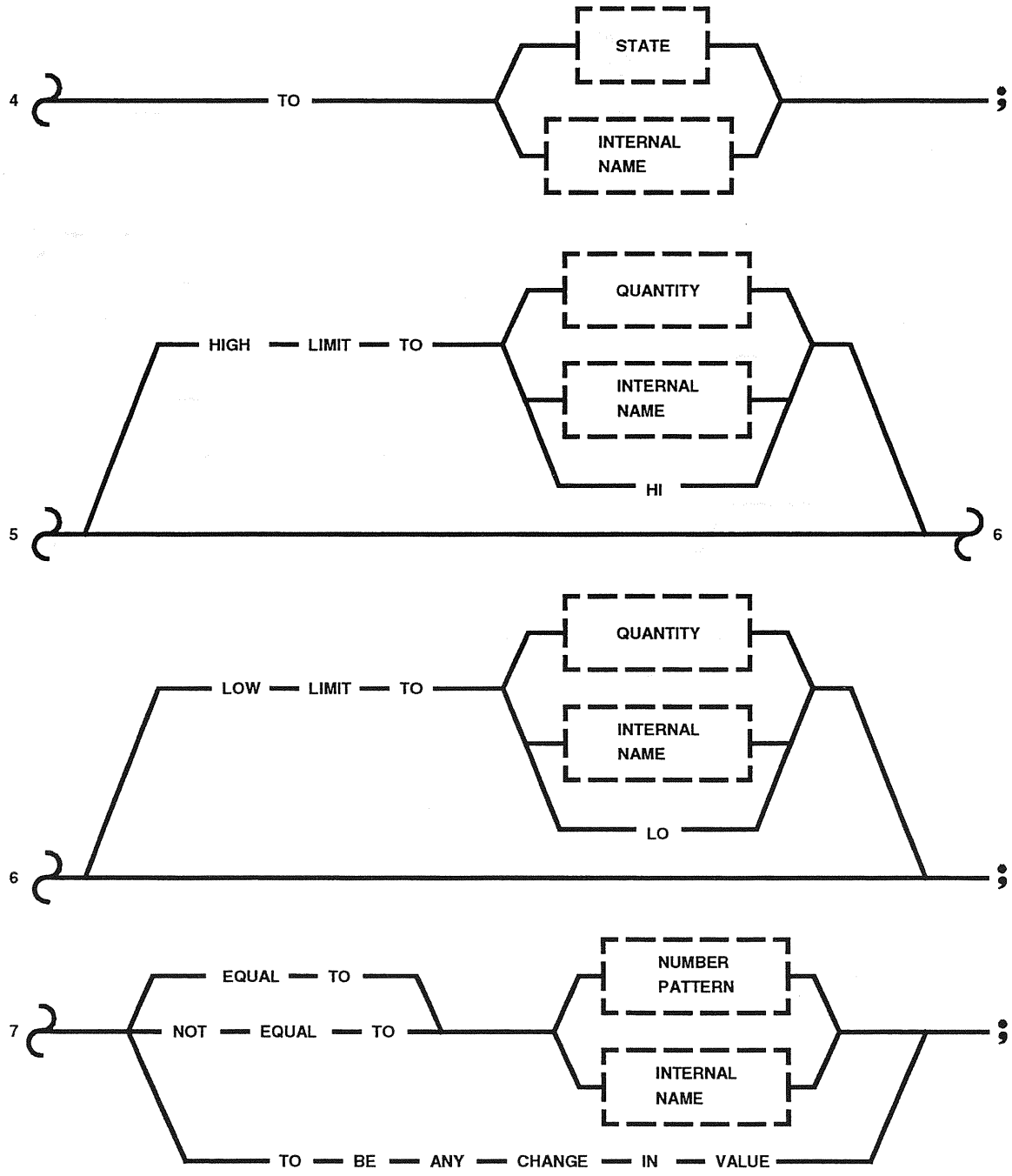


Figure 1-20 Change Statement (2 of 3)

CHANGE STATEMENT (3 OF 3)

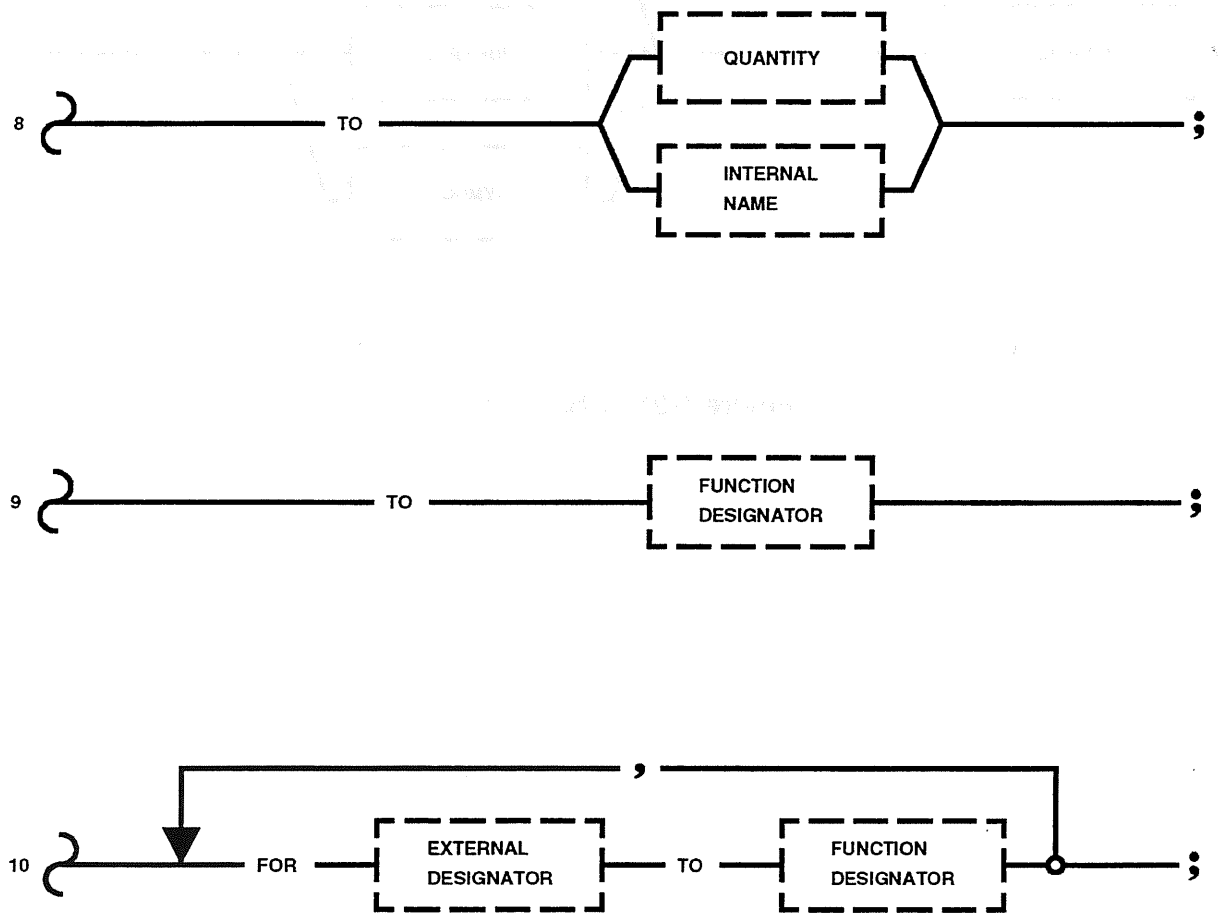


Figure 1-20 Change Statement (3 of 3)

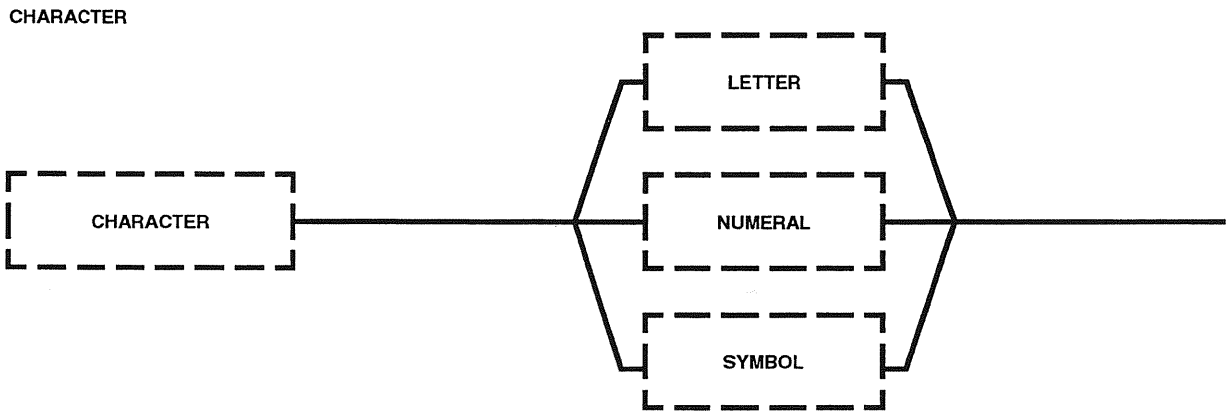


Figure 1-21 Character

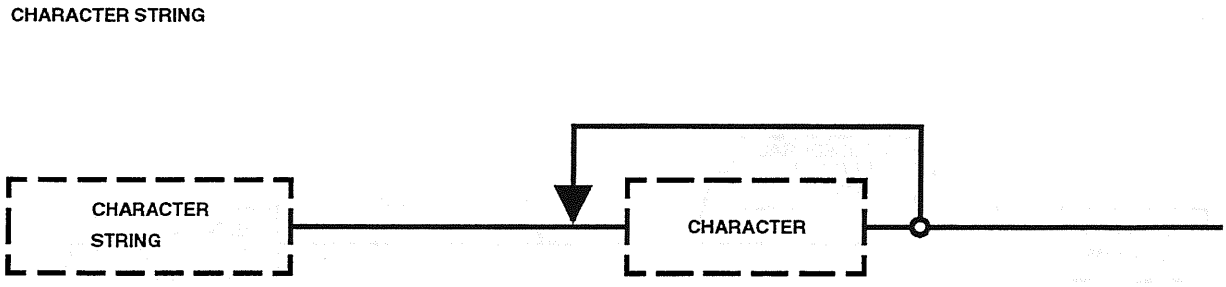


Figure 1-22 Character String

CLEAR DISPLAY STATEMENT

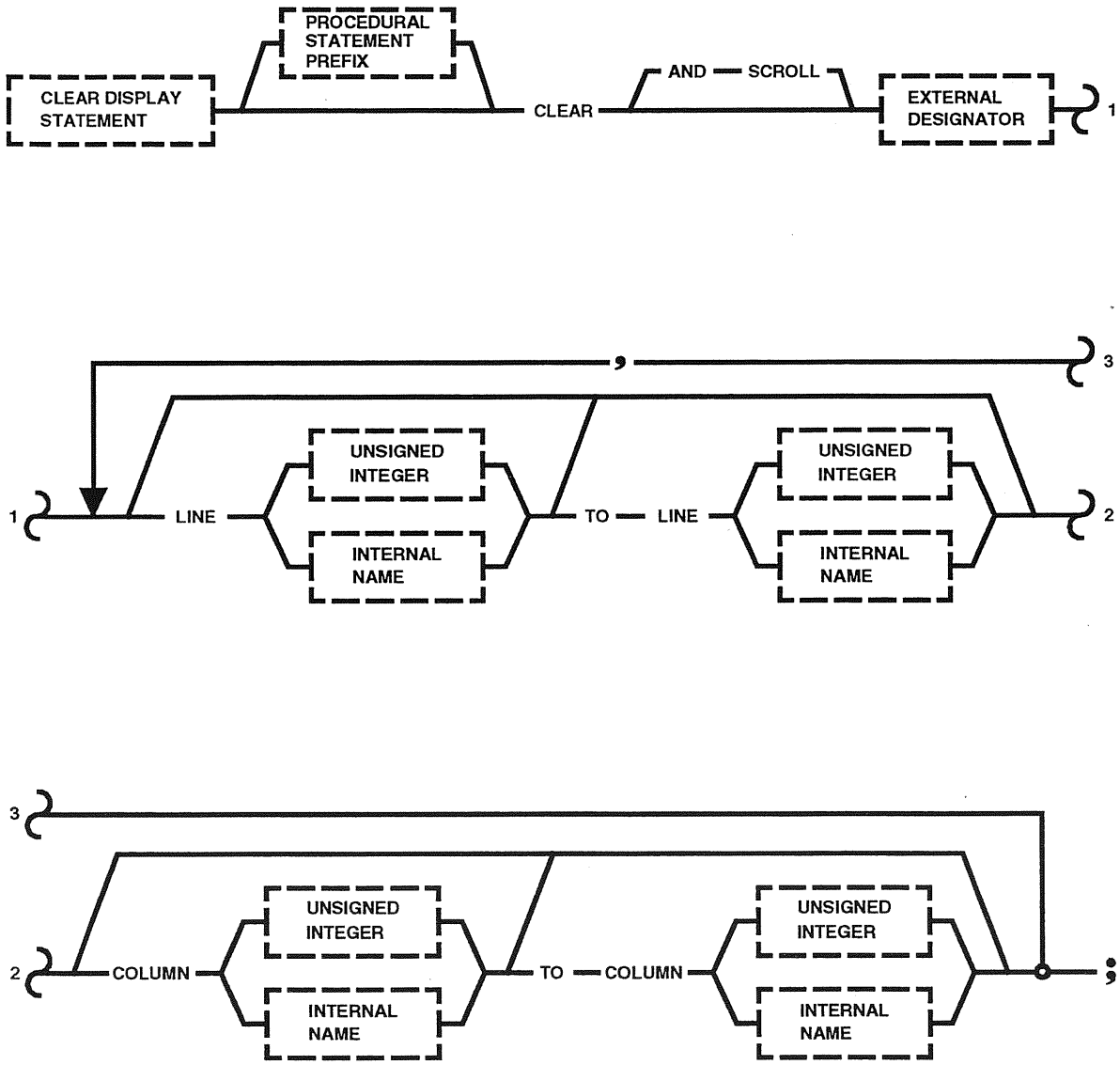


Figure 1-23 Clear Display Statement

CLOSE DISK FILE STATEMENT

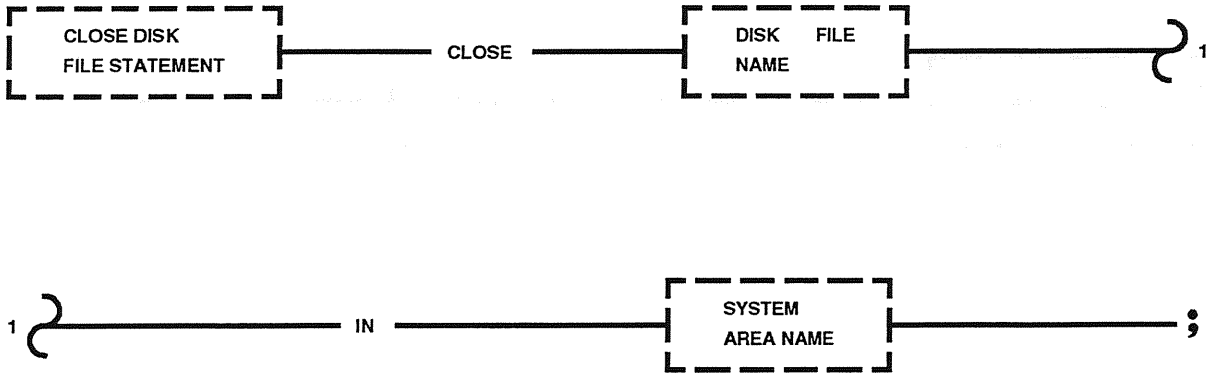


Figure 1-24 Close Disk File Statement

COLUMN NAME

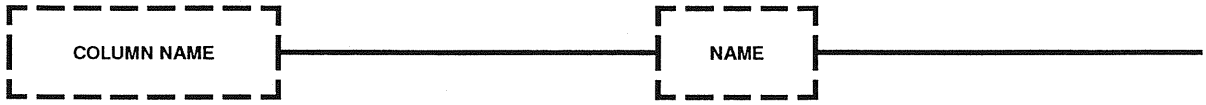


Figure 1-25 Column Name

COMMENT STATEMENT

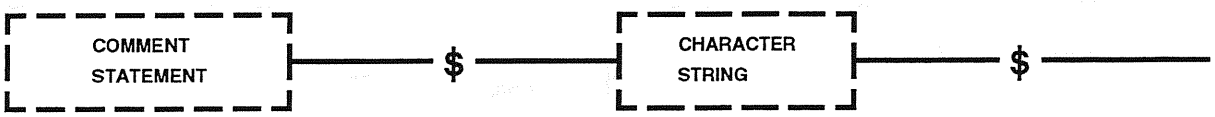


Figure 1-26 Comment Statement

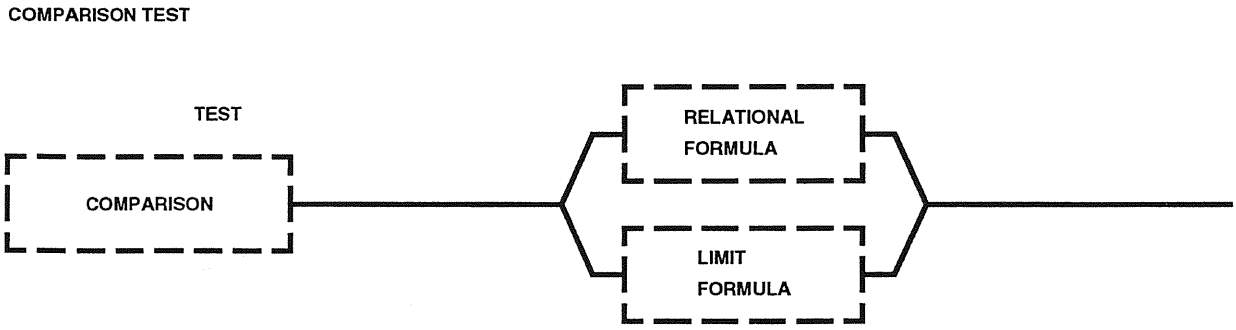


Figure 1-27 Comparison Test

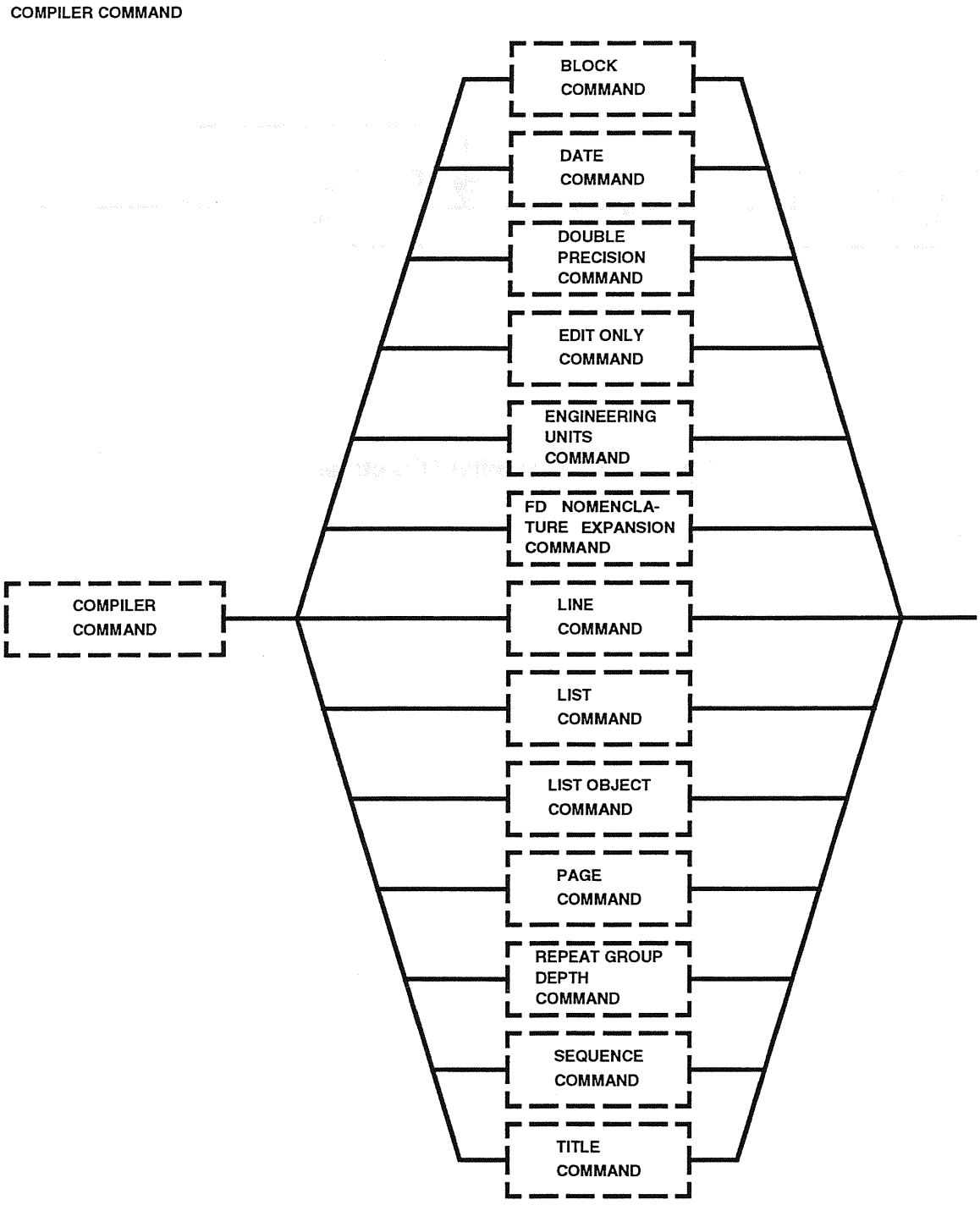


Figure 1-28 Compiler Command

COMPILER DIRECTIVES

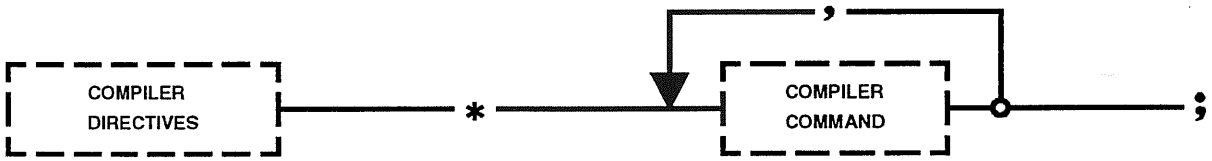


Figure 1-29 Compiler Directives

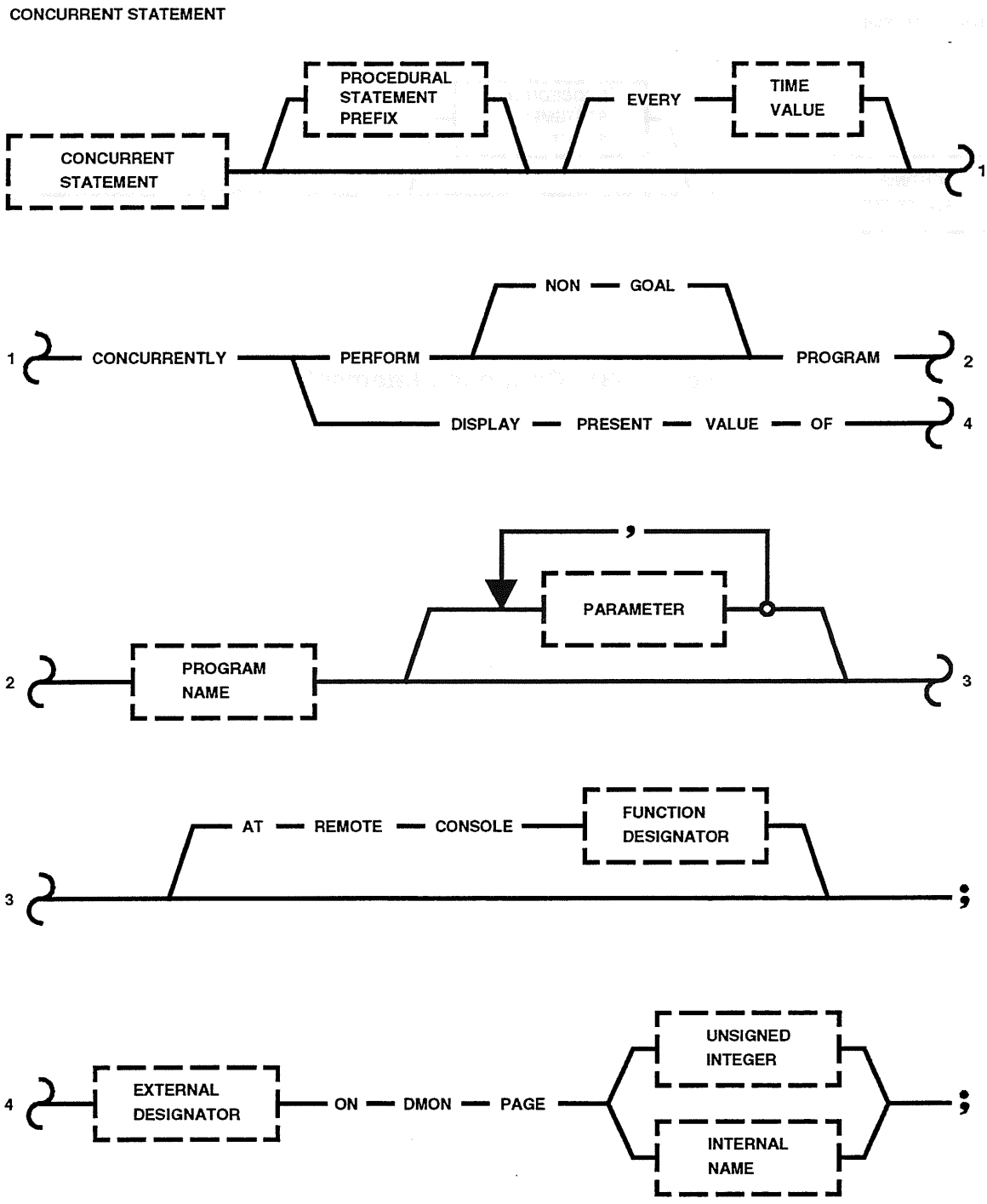


Figure 1-30 Concurrent Statement

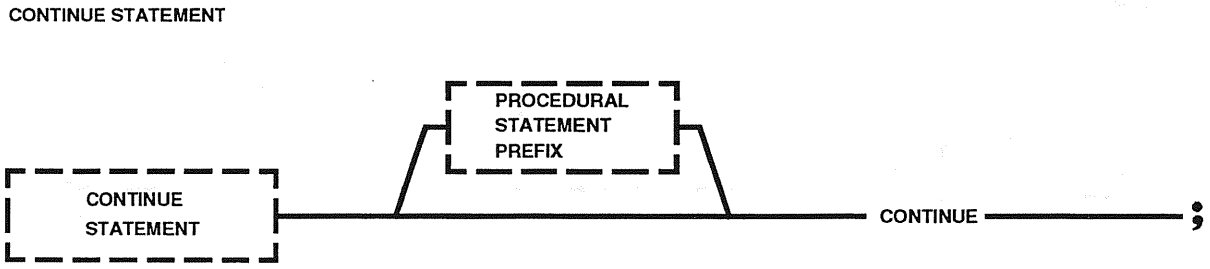


Figure 1-31 Continue Statement

CURSOR NAME

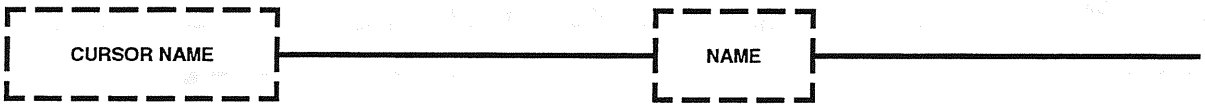


Figure 1-32 Cursor Name

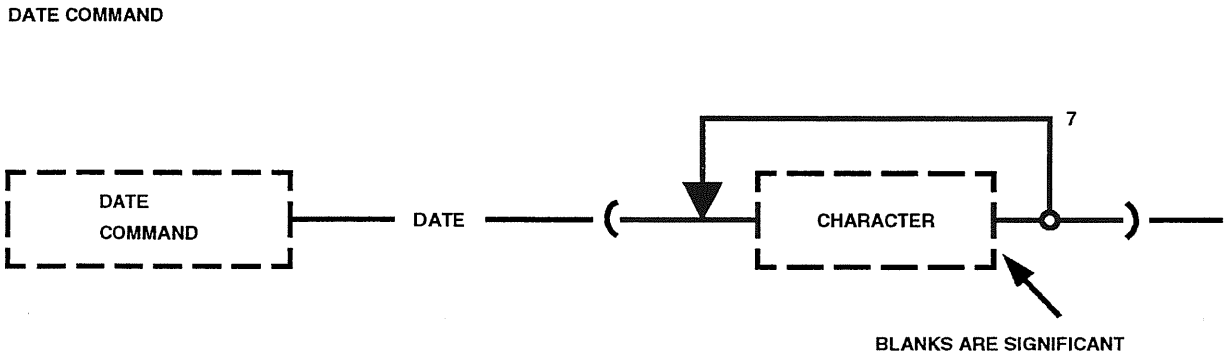


Figure 1-33 Date Command

DEALLOCATE DISK FILE STATEMENT

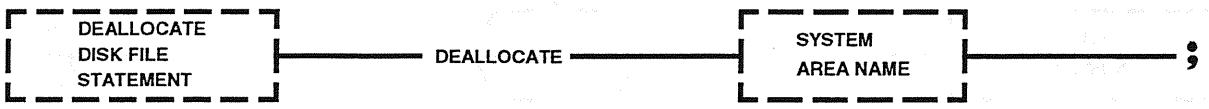


Figure 1-34 Deallocate Disk File Statement

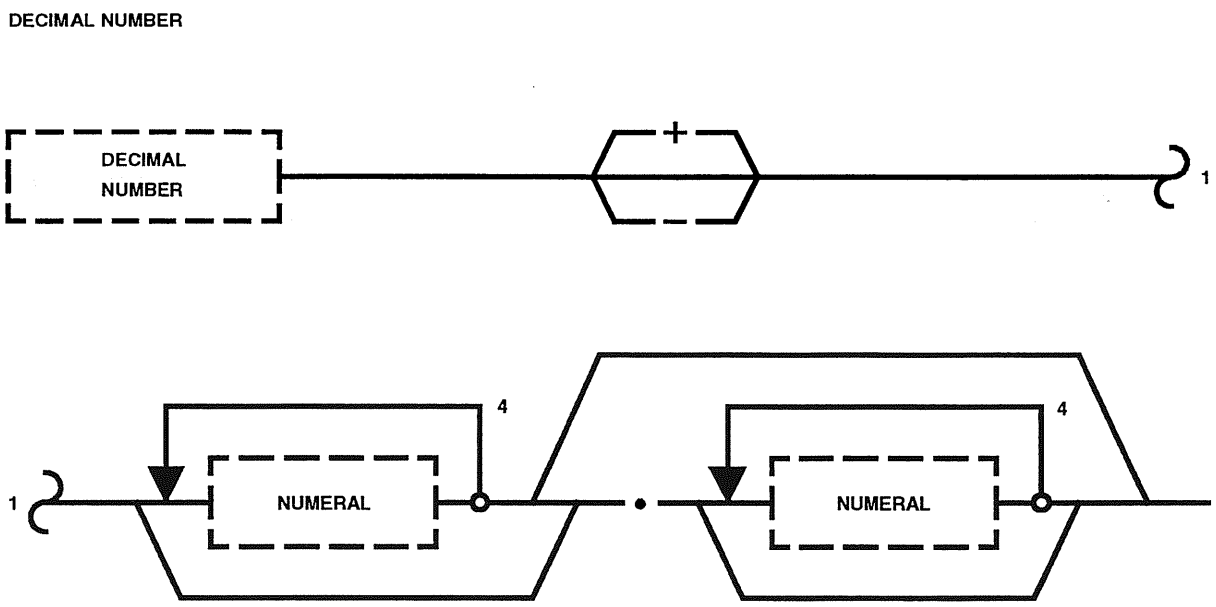


Figure 1-35 Decimal Number

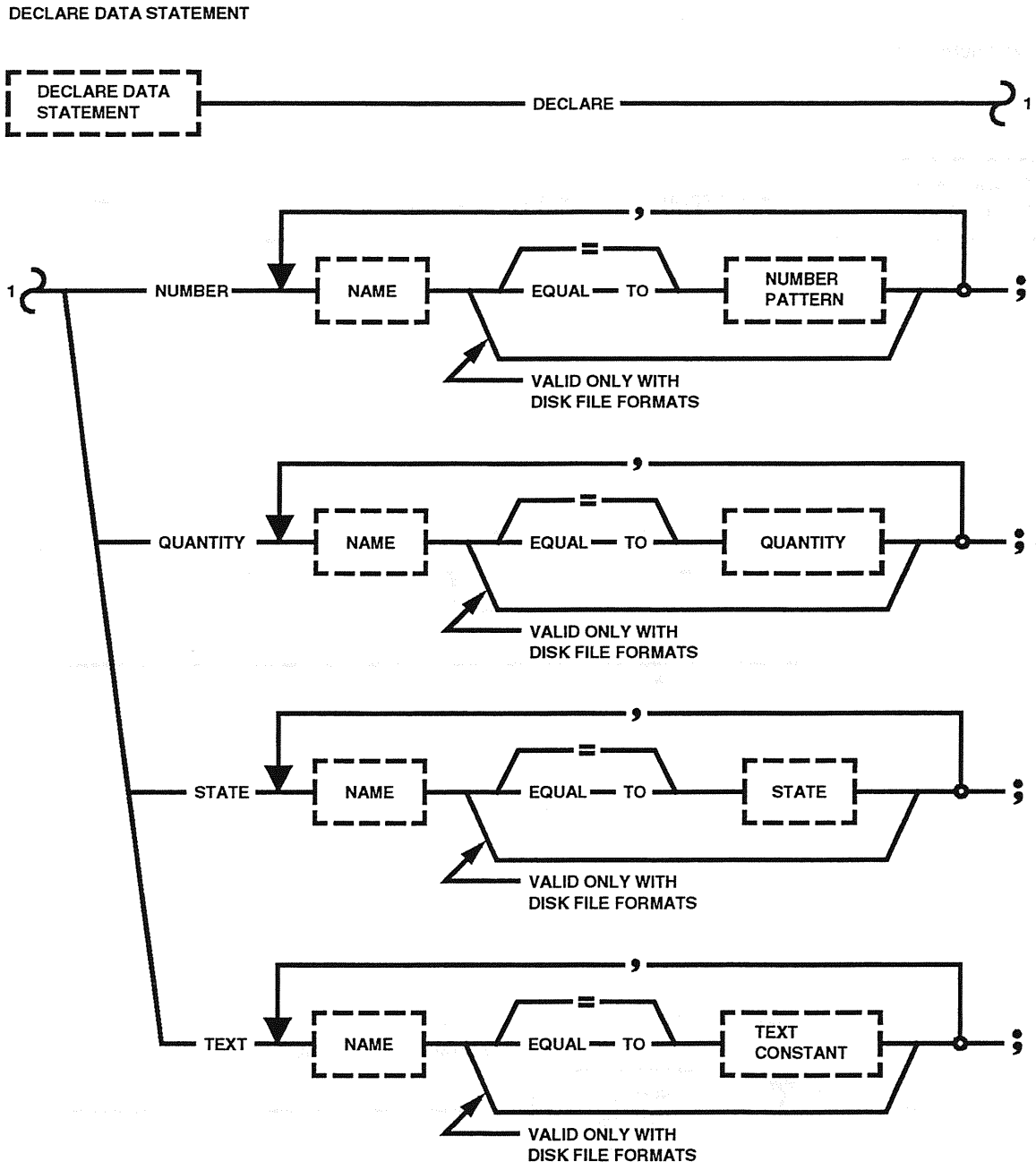


Figure 1-36 Declare Data Statement

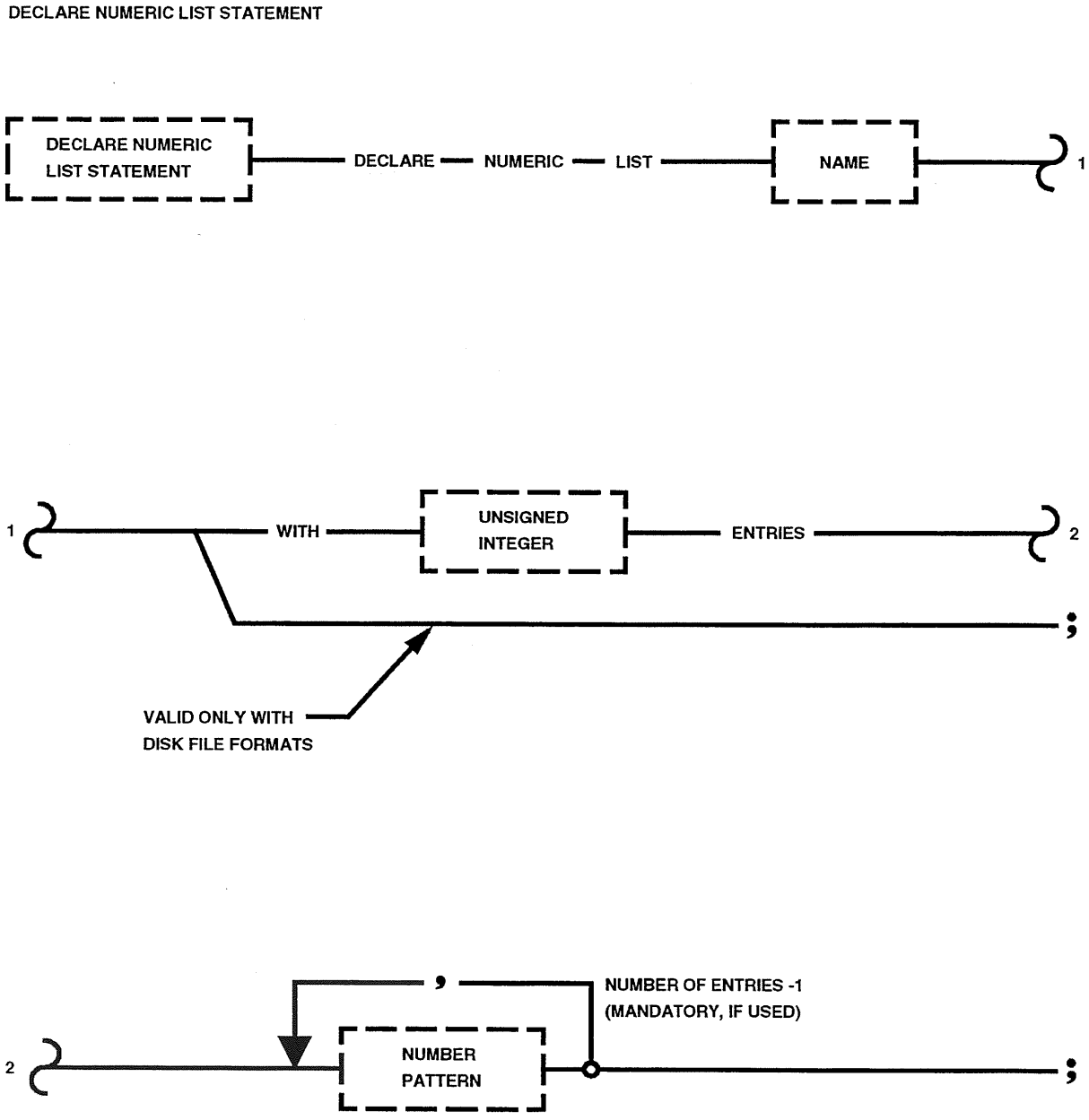


Figure 1-37 Declare Numeric List Statement

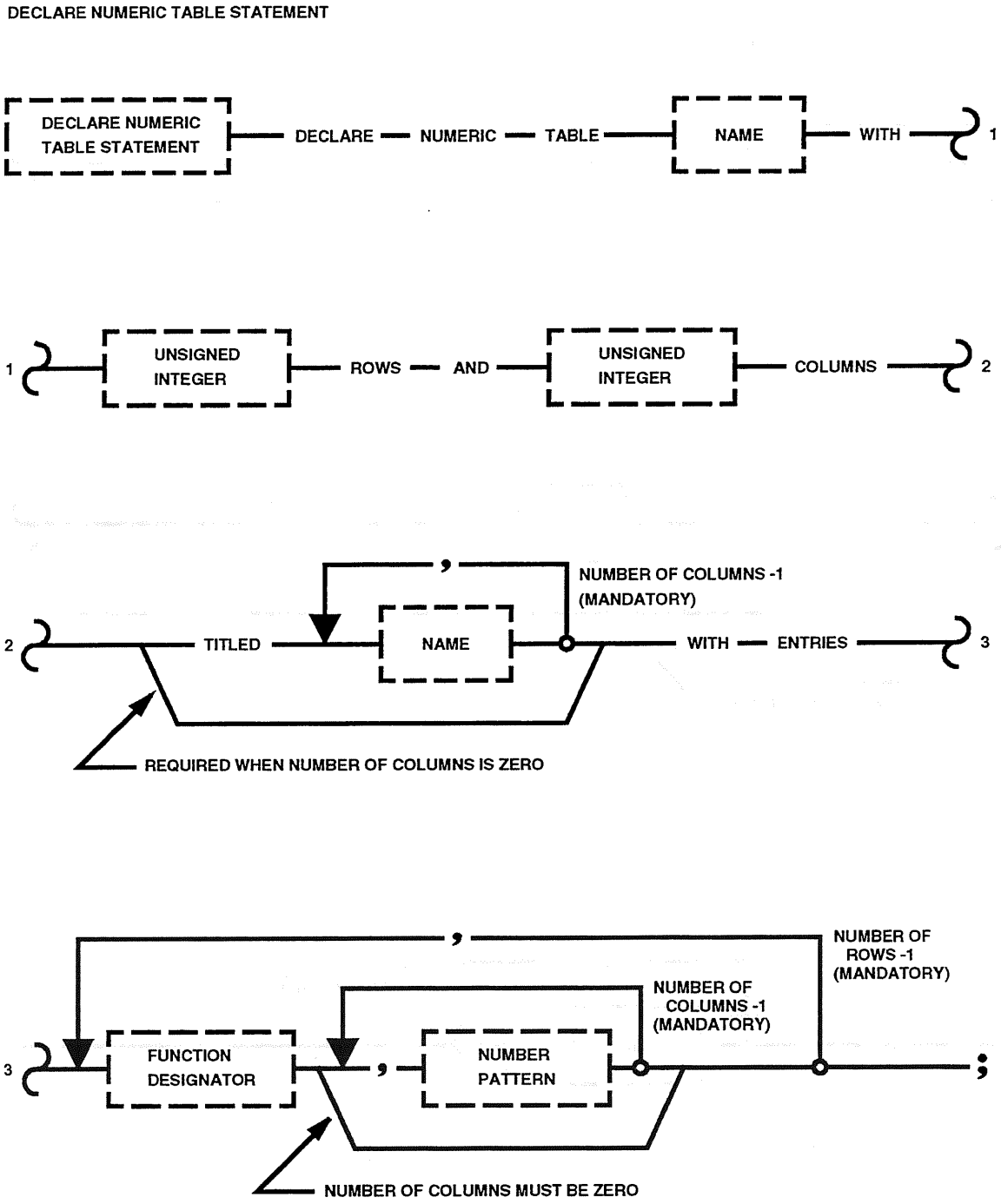


Figure 1-38 Declare Numeric Table Statement

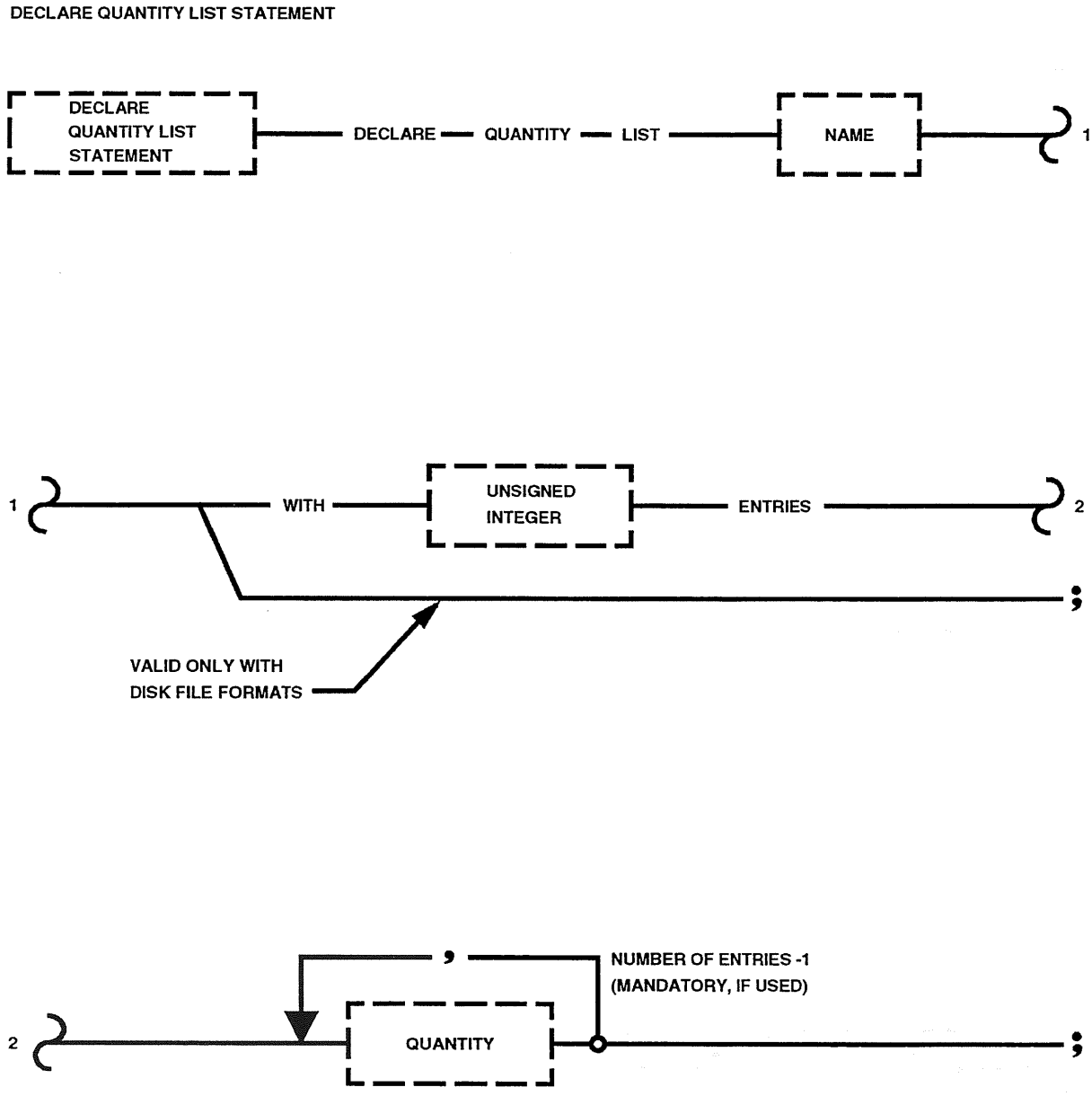


Figure 1-39 Declare Quantity List Statement

DECLARE QUANTITY TABLE STATEMENT

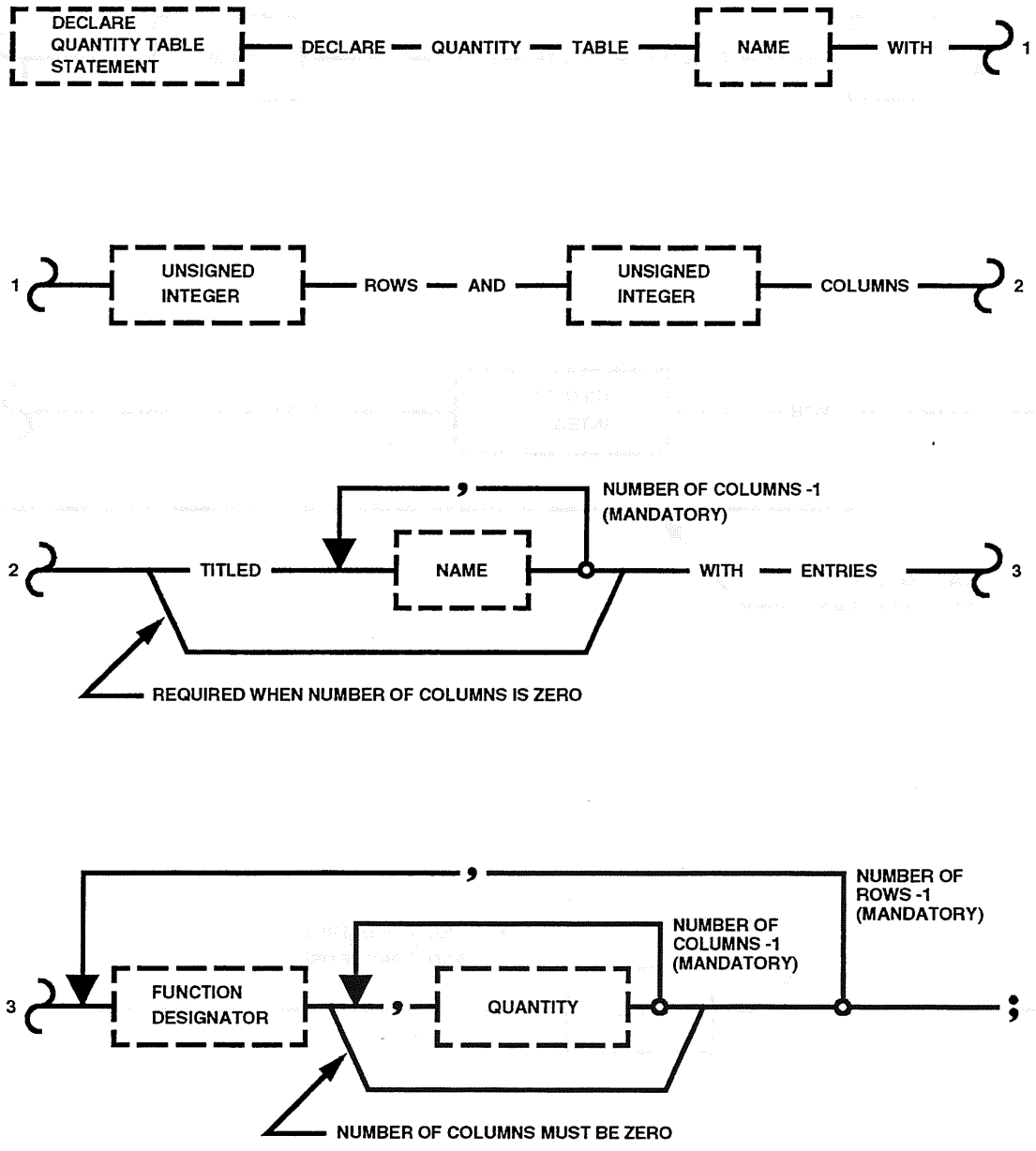


Figure 1-40 Declare Quantity Table Statement

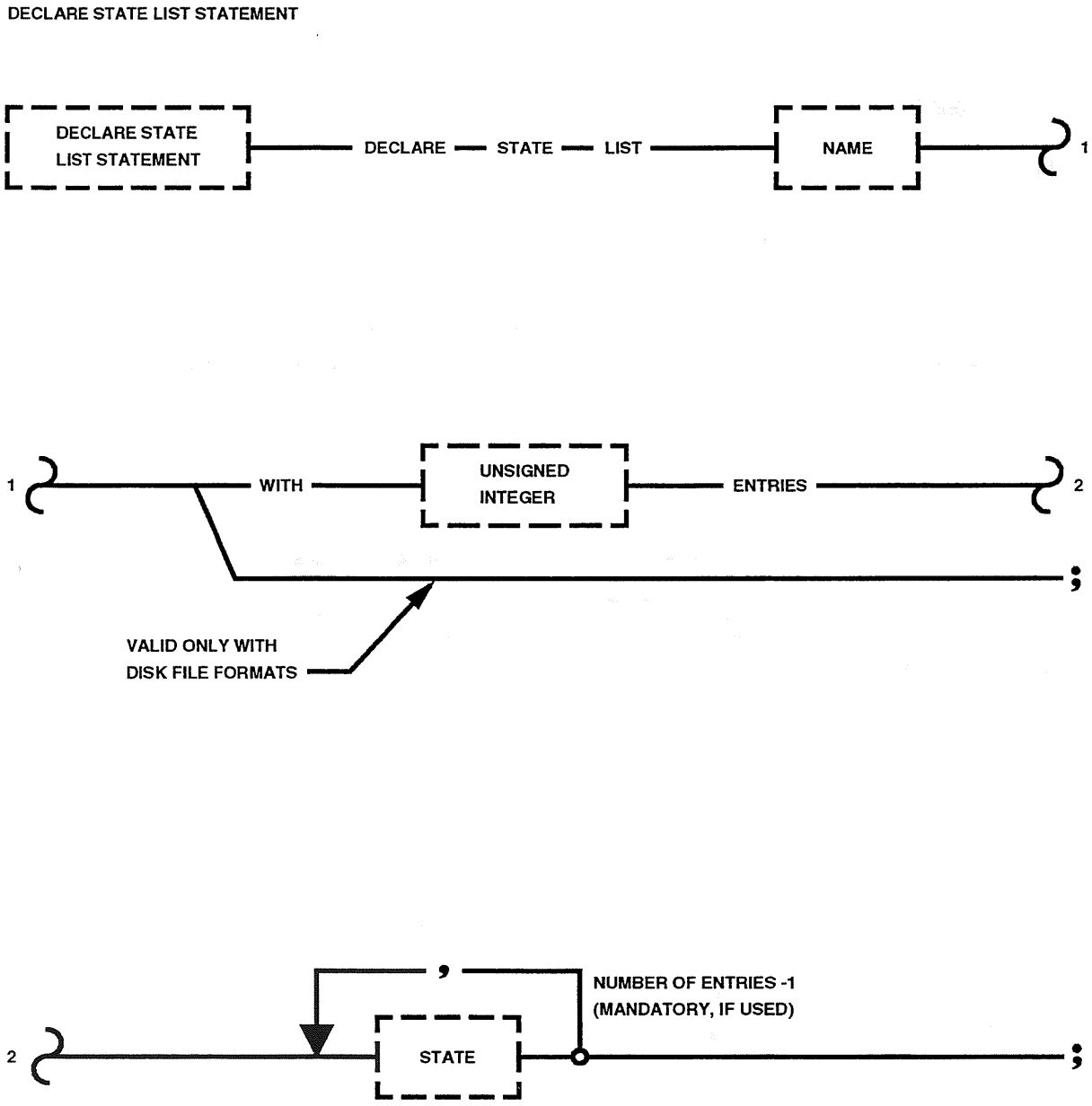


Figure 1-41 Declare State List Statement

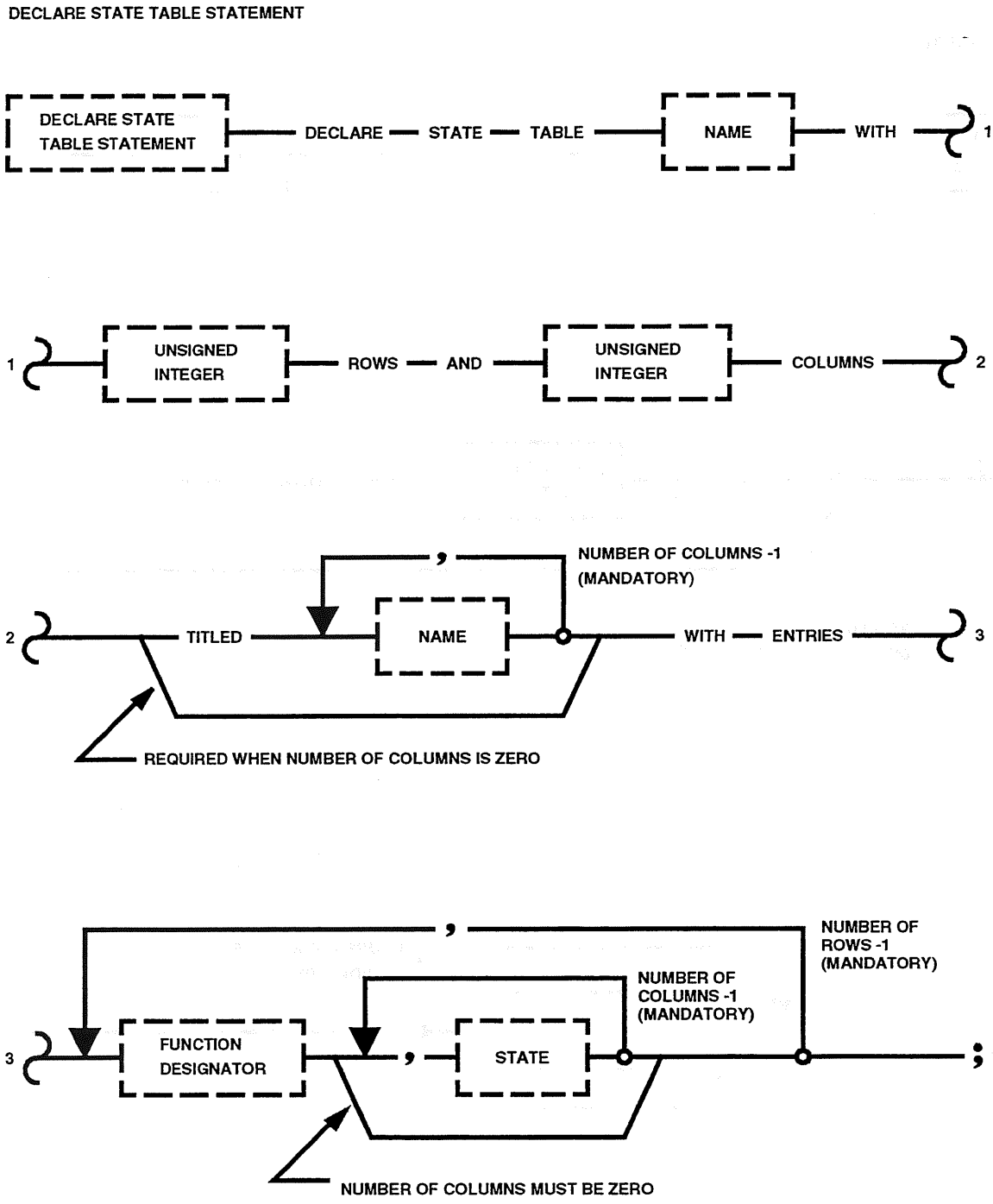


Figure 1-42 Declare State Table Statement

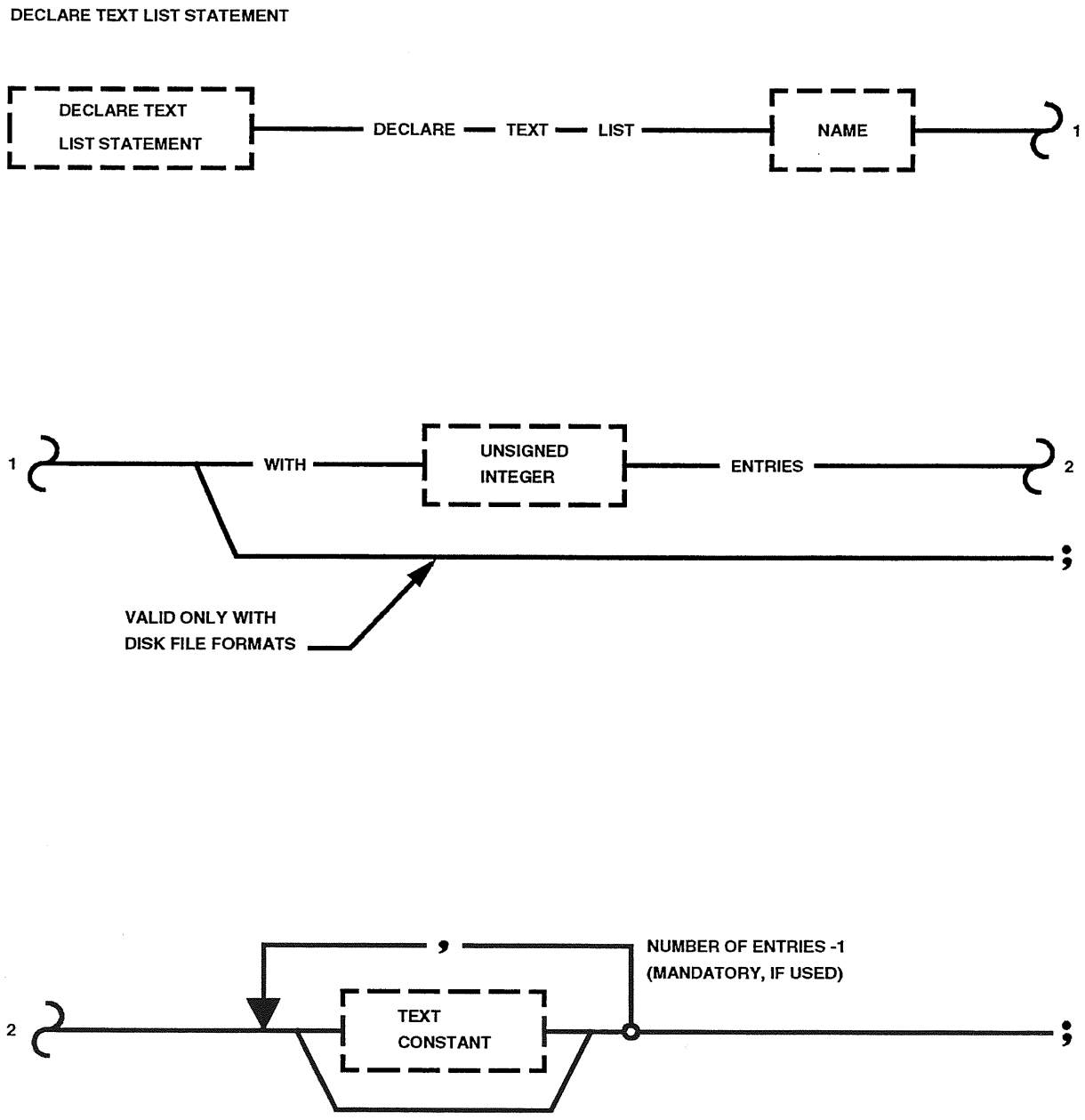


Figure 1-43 Declare Text List Statement

DECLARE TEXT TABLE STATEMENT

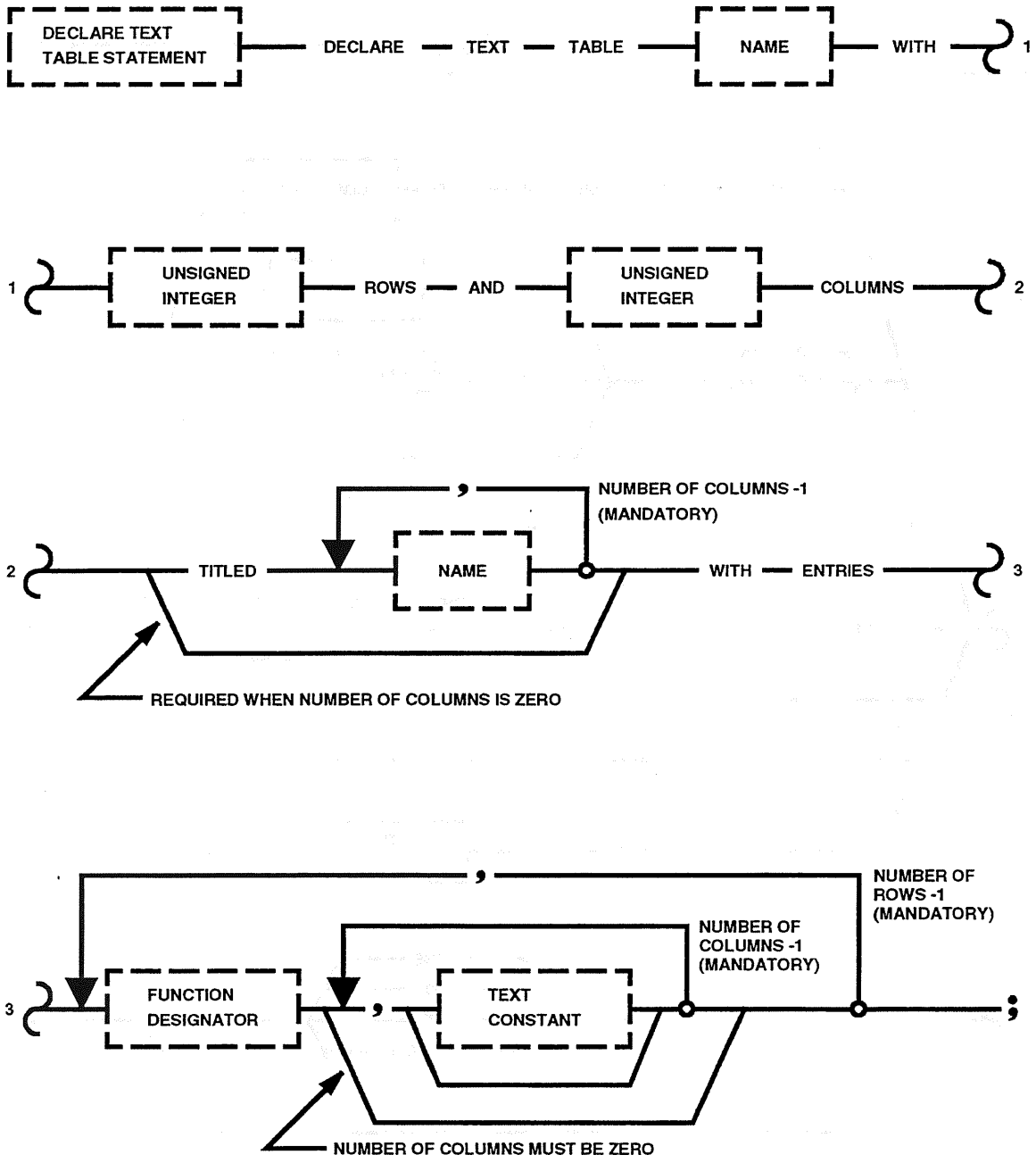


Figure 1-44 Declare Text Table Statement

DEFINE STATEMENT

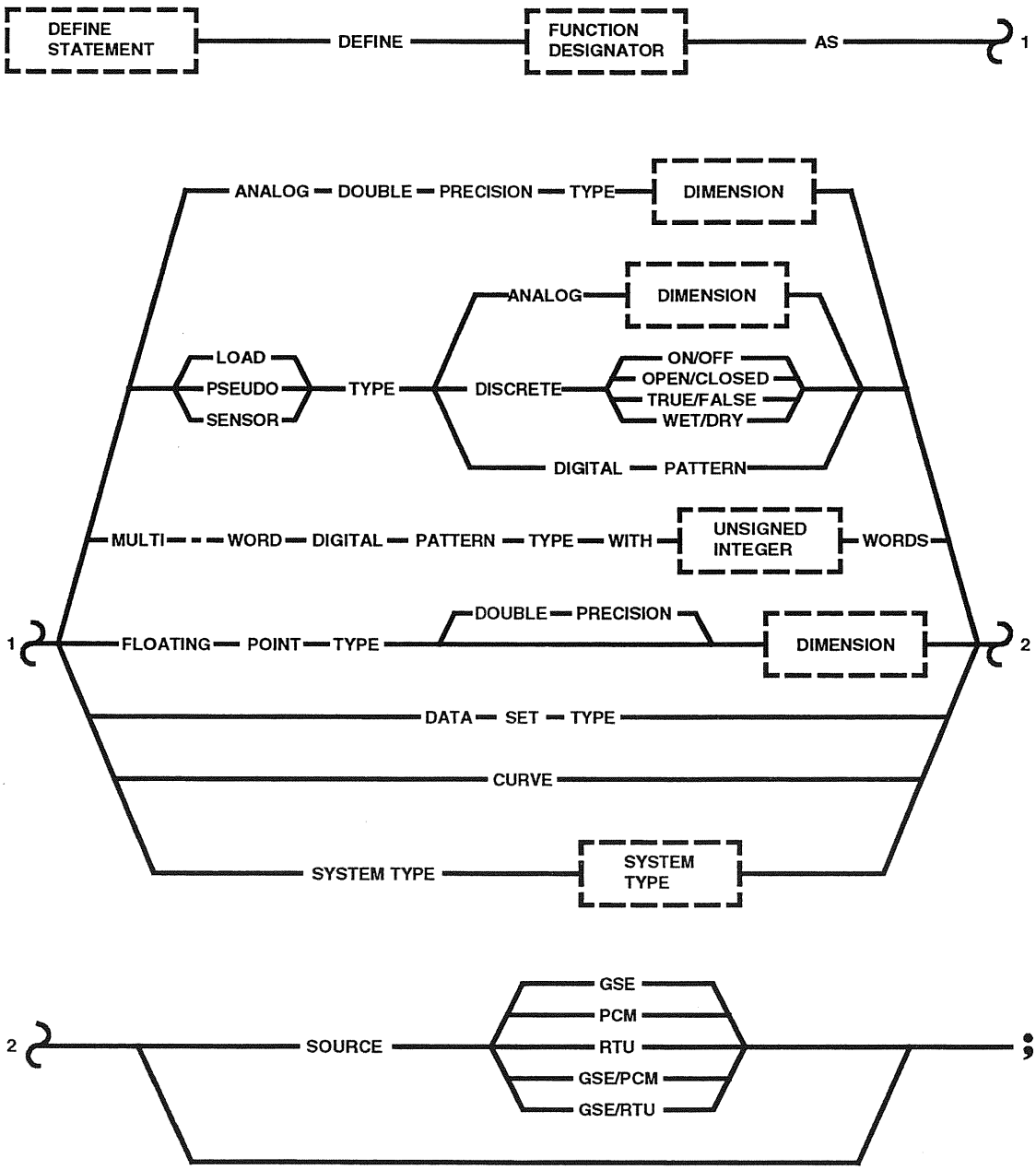


Figure 1-45 Define Statement

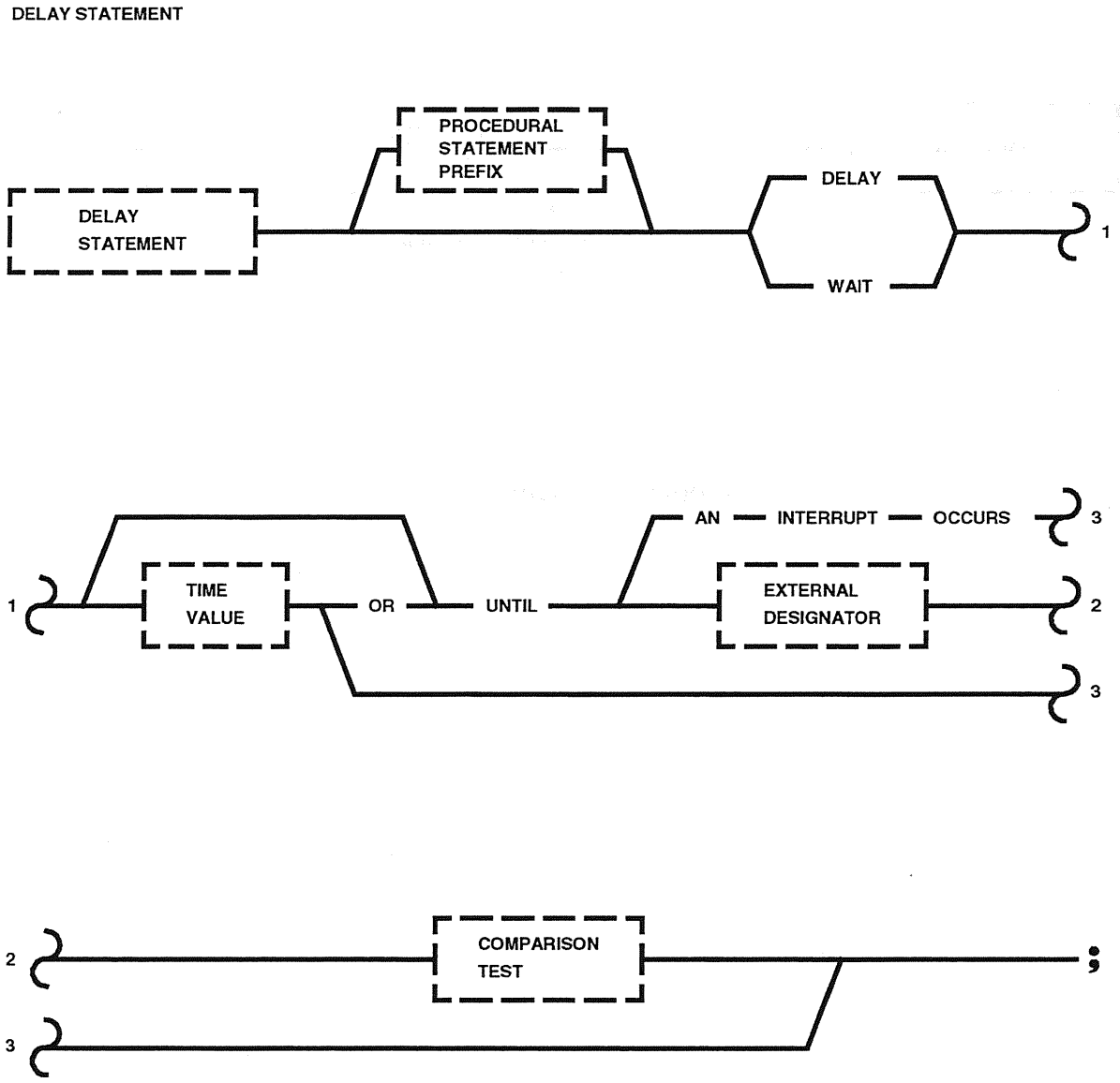


Figure 1-46 Delay Statement

DIMENSION

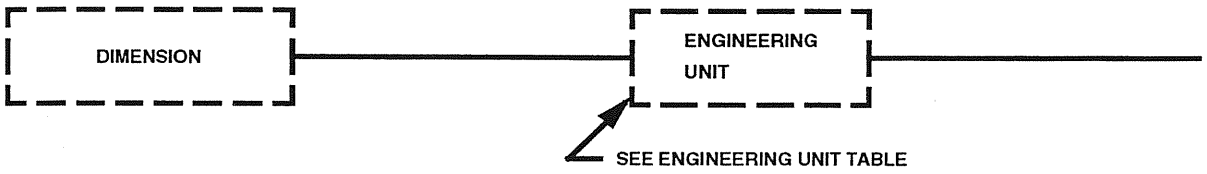


Figure 1-47 Dimension

DISK FILE NAME

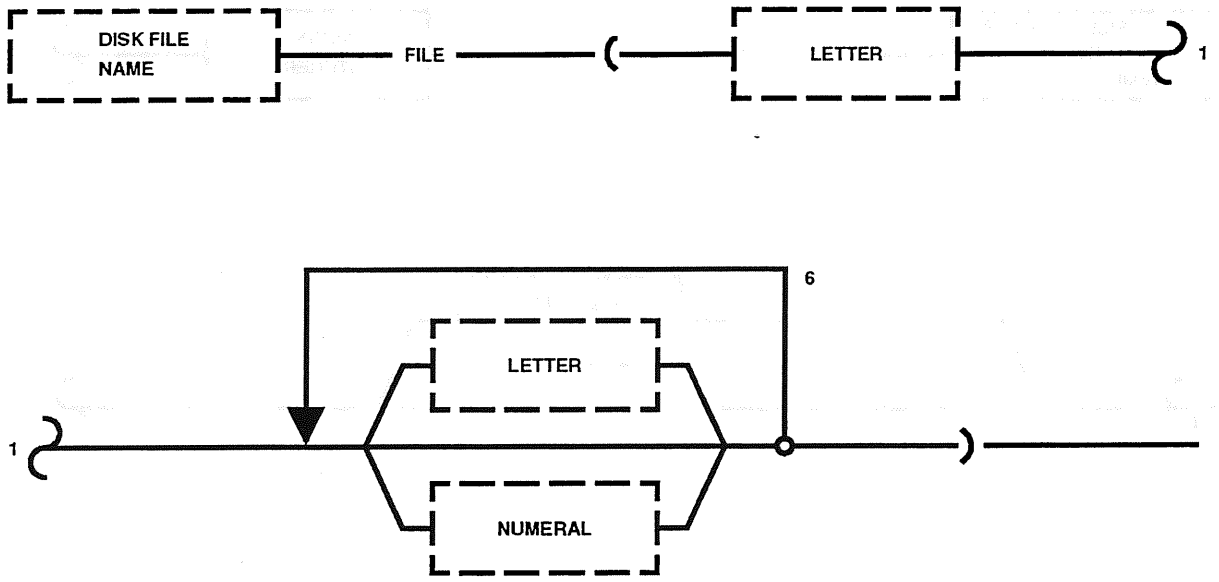


Figure 1-48 Disk File Name

DISK FILE RECORD INITIALIZATION STATEMENT

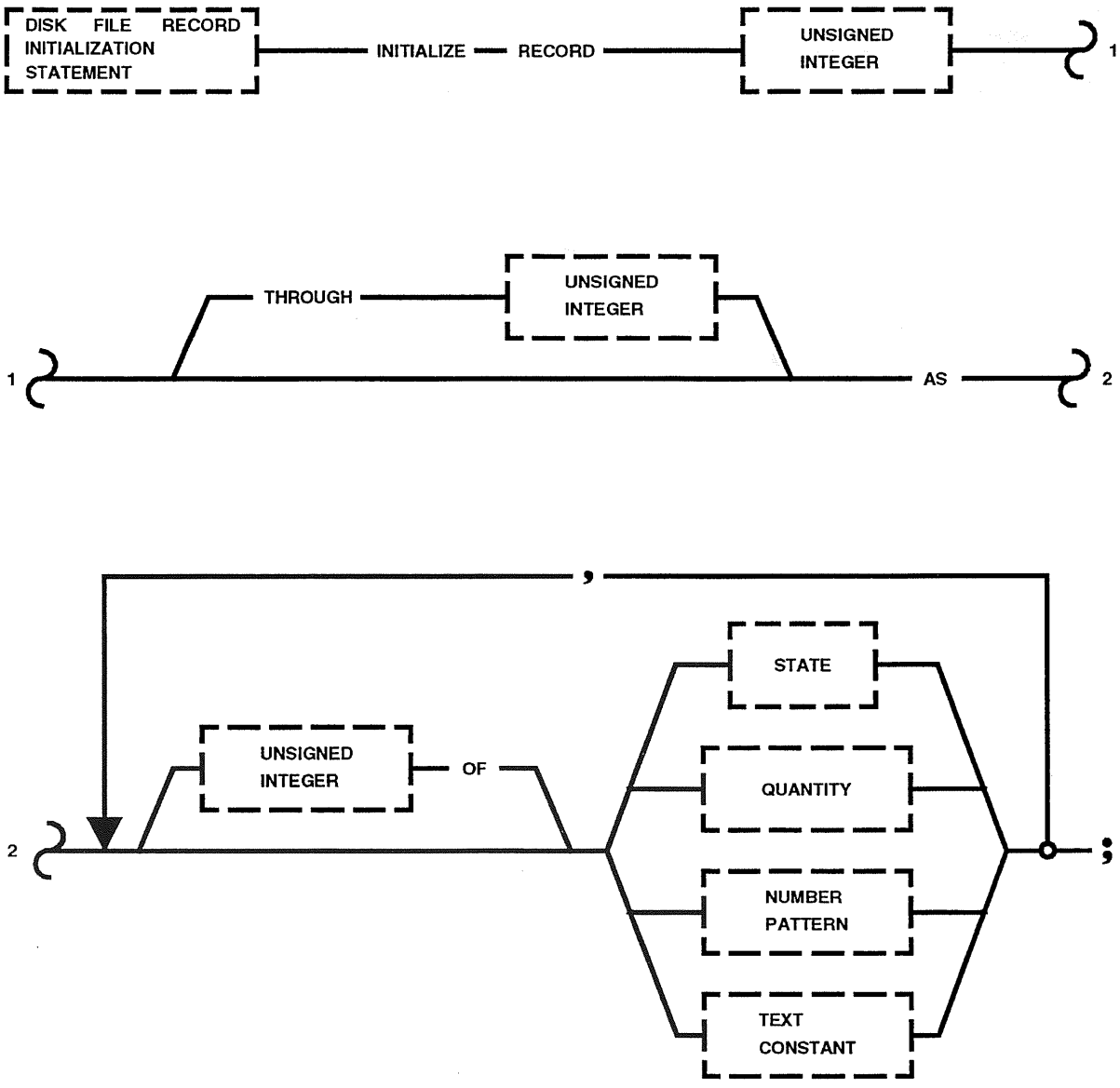


Figure 1-49 Disk File Record Initialization Statement

DISPLAY SKELETON STATEMENT

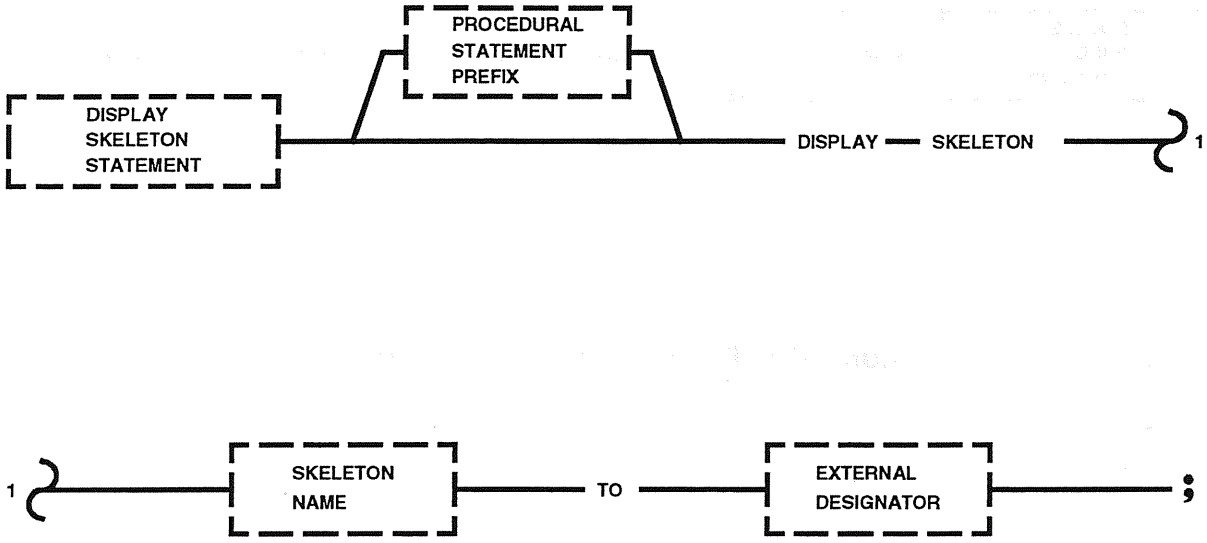


Figure 1-50 Display Skeleton Statement

DOUBLE PRECISION COMMAND

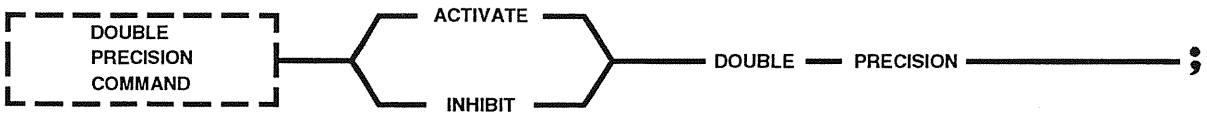


Figure 1-51 Double Precision Command

EDIT ONLY COMMAND

Figure 1-52 Edit Only Command

END DISK FILE DEFINITION STATEMENT

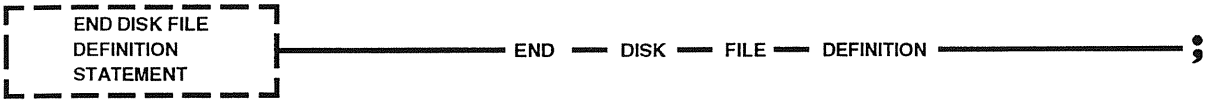


Figure 1-53 End Disk File Definition Statement

END DISK FILE RECORD STRUCTURE DEFINITION STATEMENT

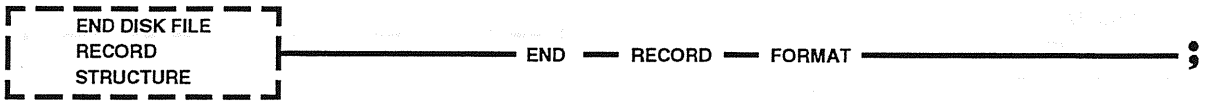


Figure 1-54 End Disk File Record Structure Definition Statement

END MACRO STATEMENT

Figure 1-55 End Macro Statement

END PROGRAM STATEMENT

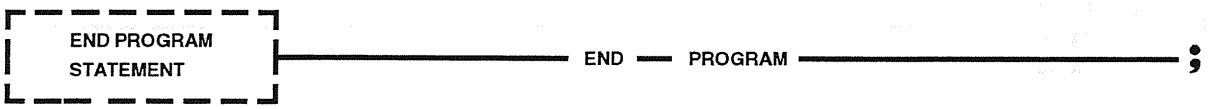


Figure 1-56 End Program Statement

END SEQUENCE STATEMENT

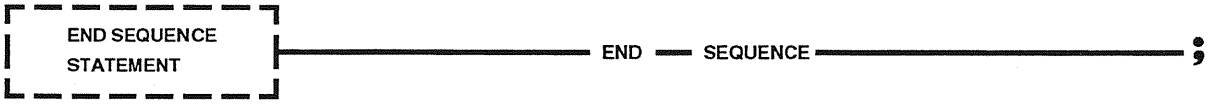


Figure 1-57 End Sequence Statement

END STATEMENT GROUPS STATEMENT

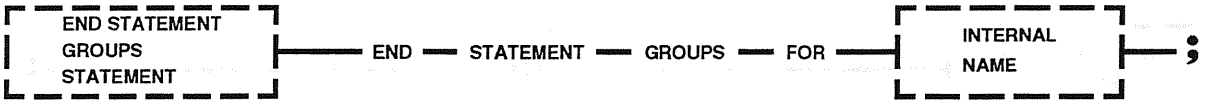


Figure 1-58 End Statement Groups Statement

END SUBROUTINE STATEMENT

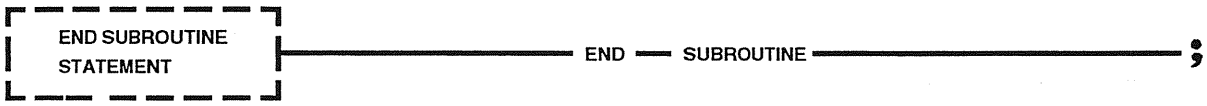


Figure 1-59 End Subroutine Statement

ENGINEERING UNITS COMMAND

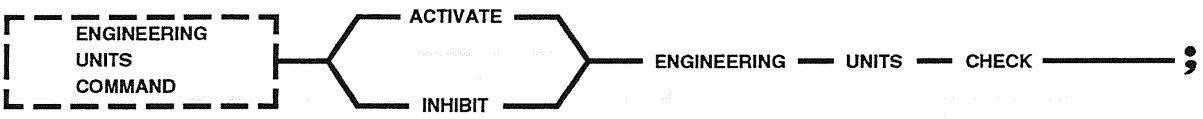


Figure 1-60 Engineering Units Command

EXPAND MACRO STATEMENT

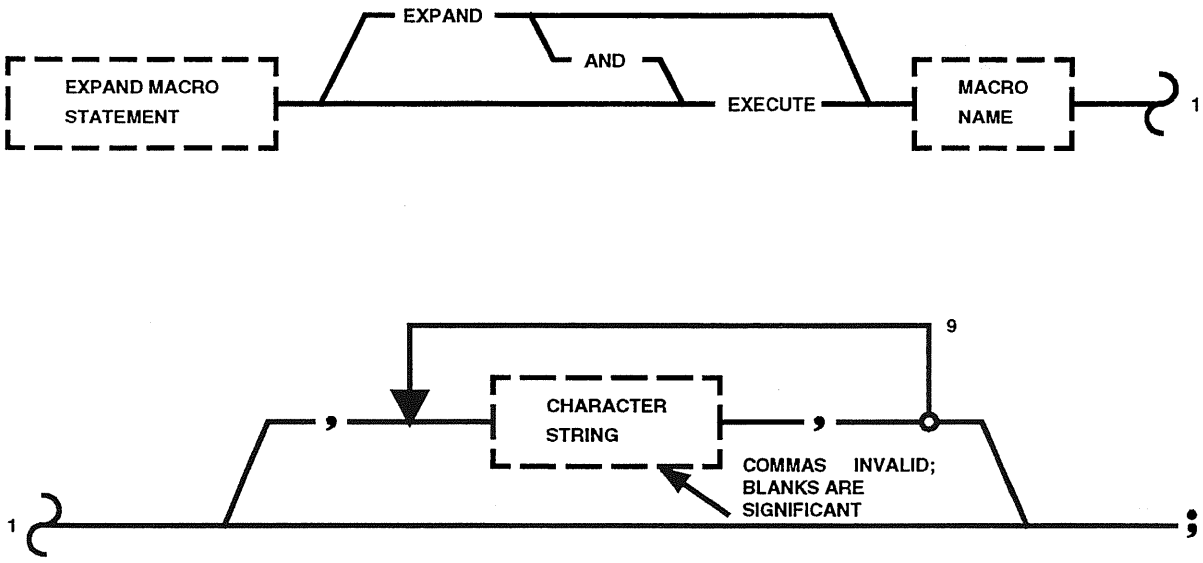


Figure 1-61 Expand Macro Statement

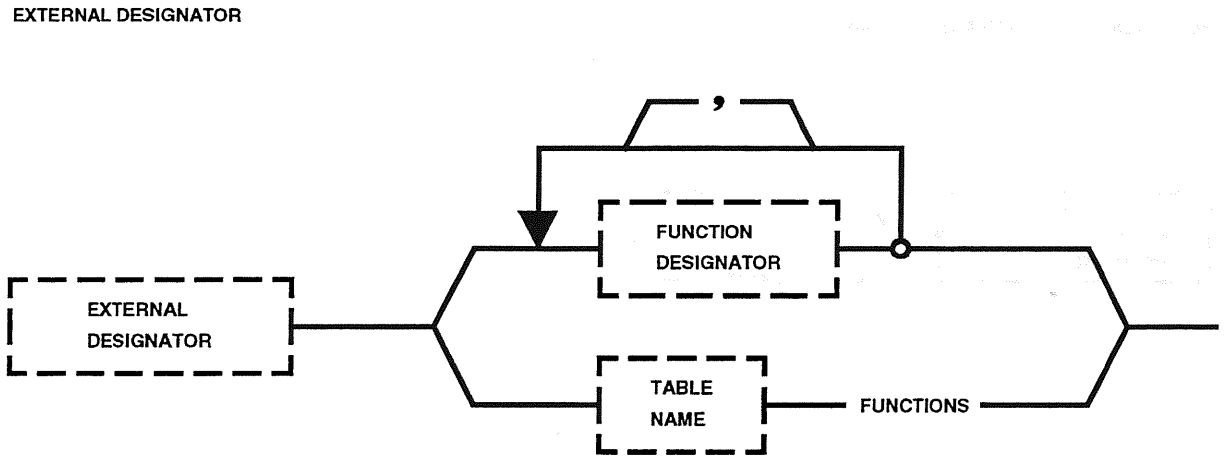


Figure 1-62 External Designator

FD NOMENCLATURE EXPANSION COMMAND

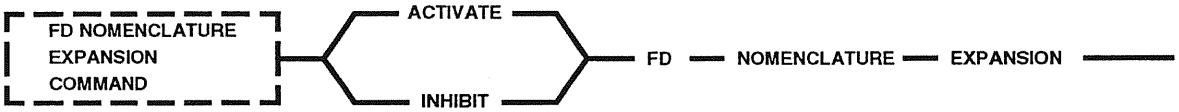


Figure 1-63 FD Nomenclature Expansion Command

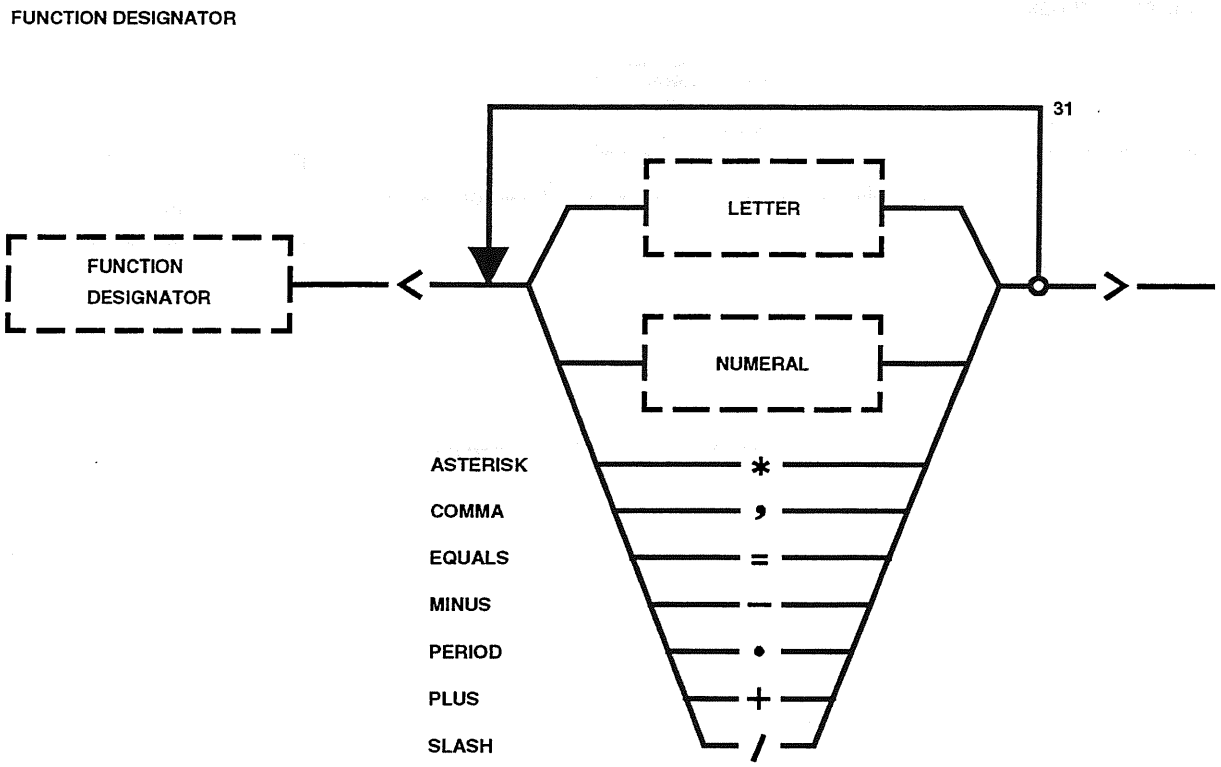


Figure 1-64 Function Designator

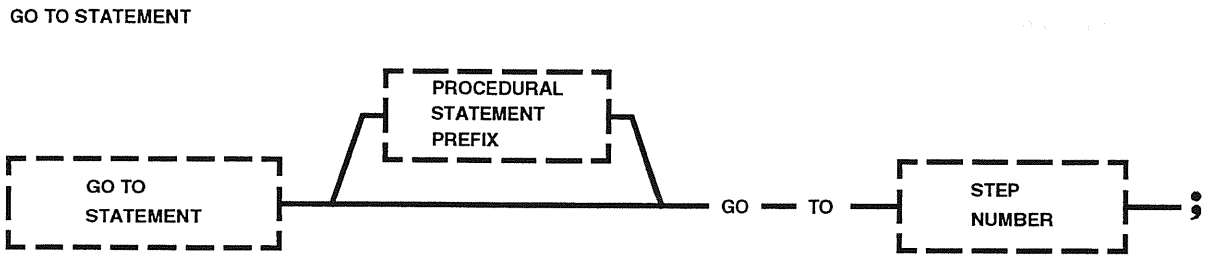


Figure 1-65 Go To Statement

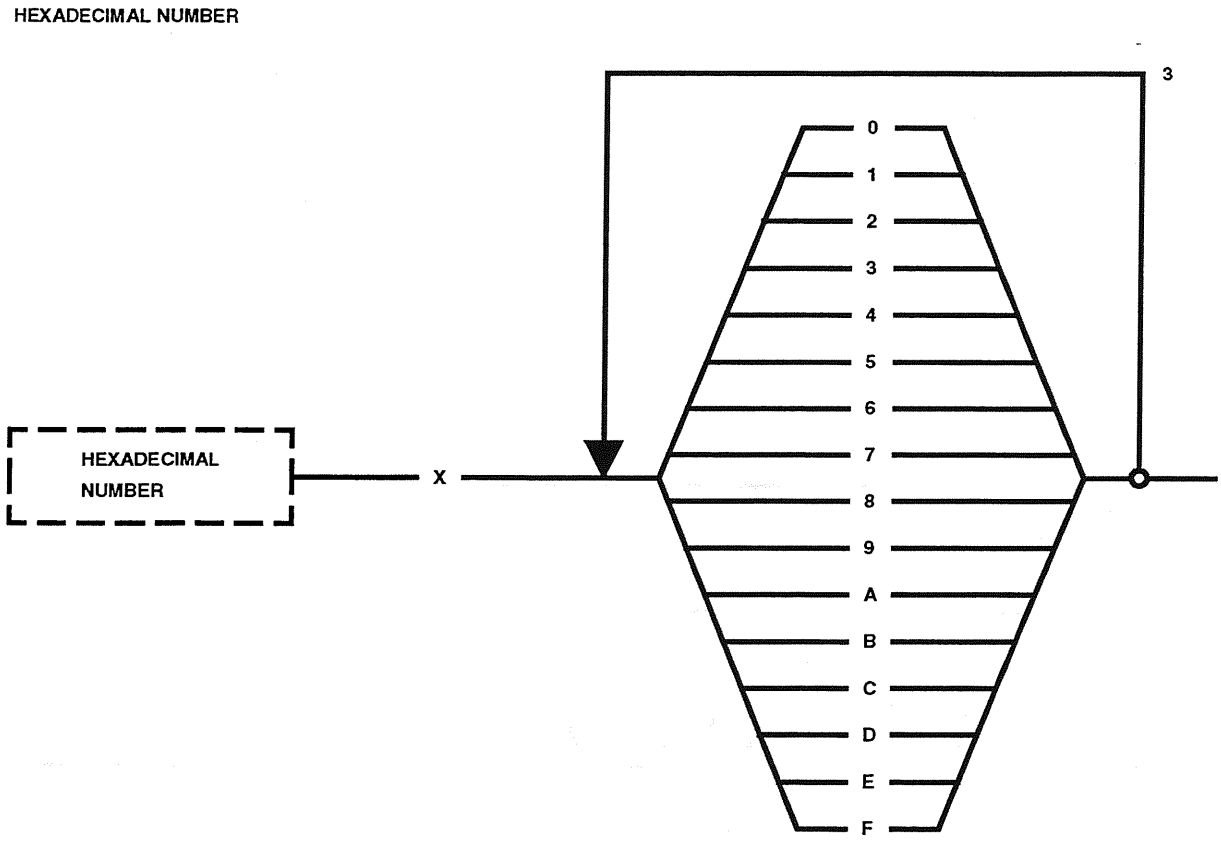


Figure 1-66 Hexadecimal Number

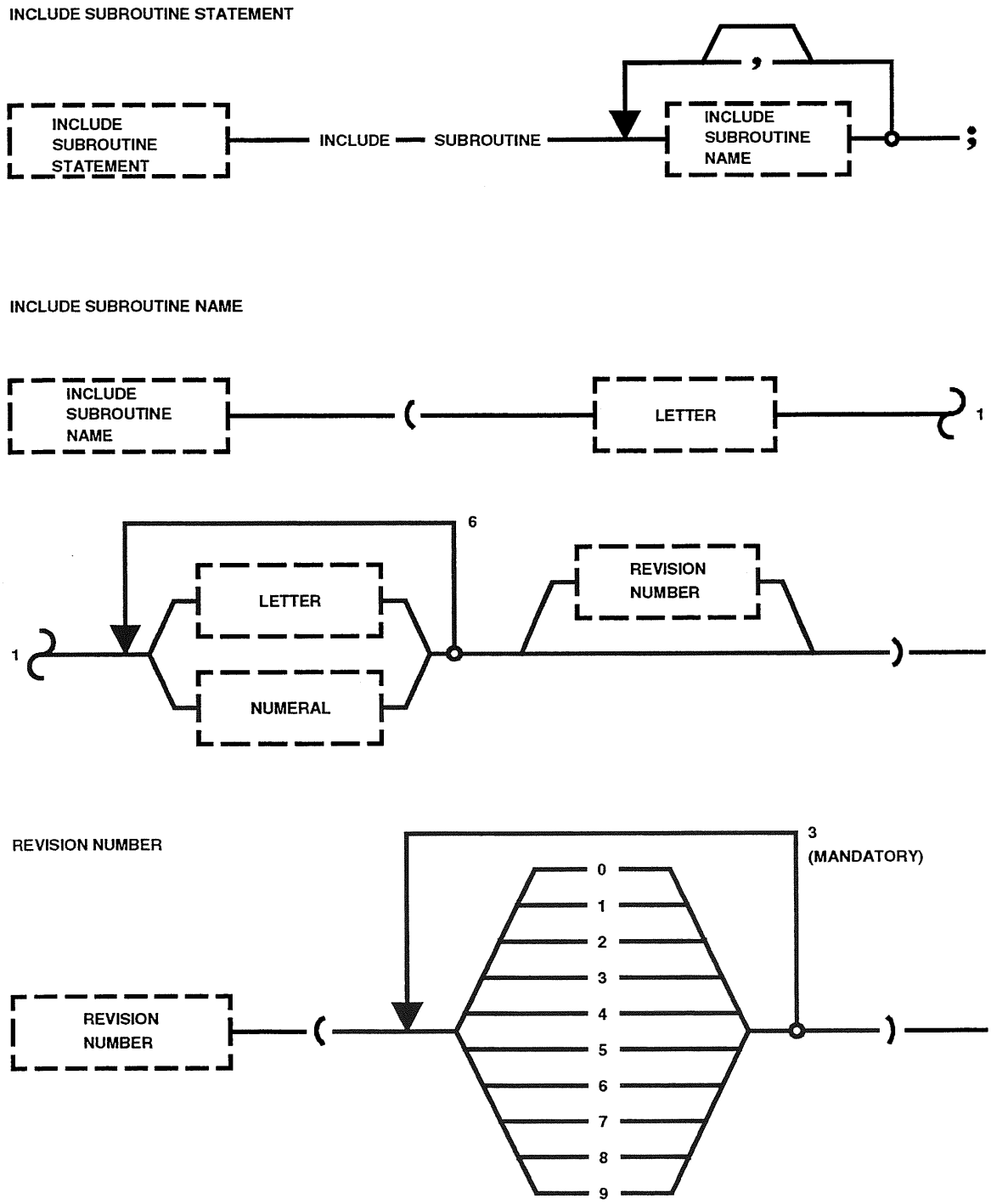


Figure 1-67 Include Subroutine Statement

INDEX NAME

Figure 1-68 Index Name

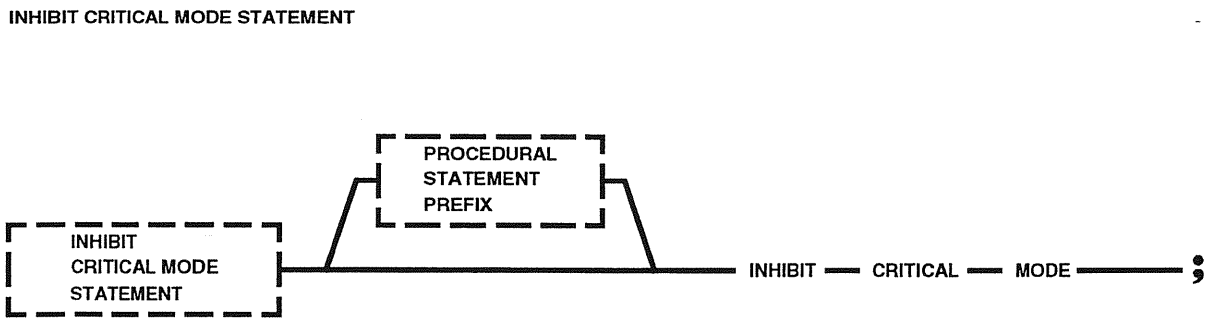


Figure 1-69 Inhibit Critical Mode Statement

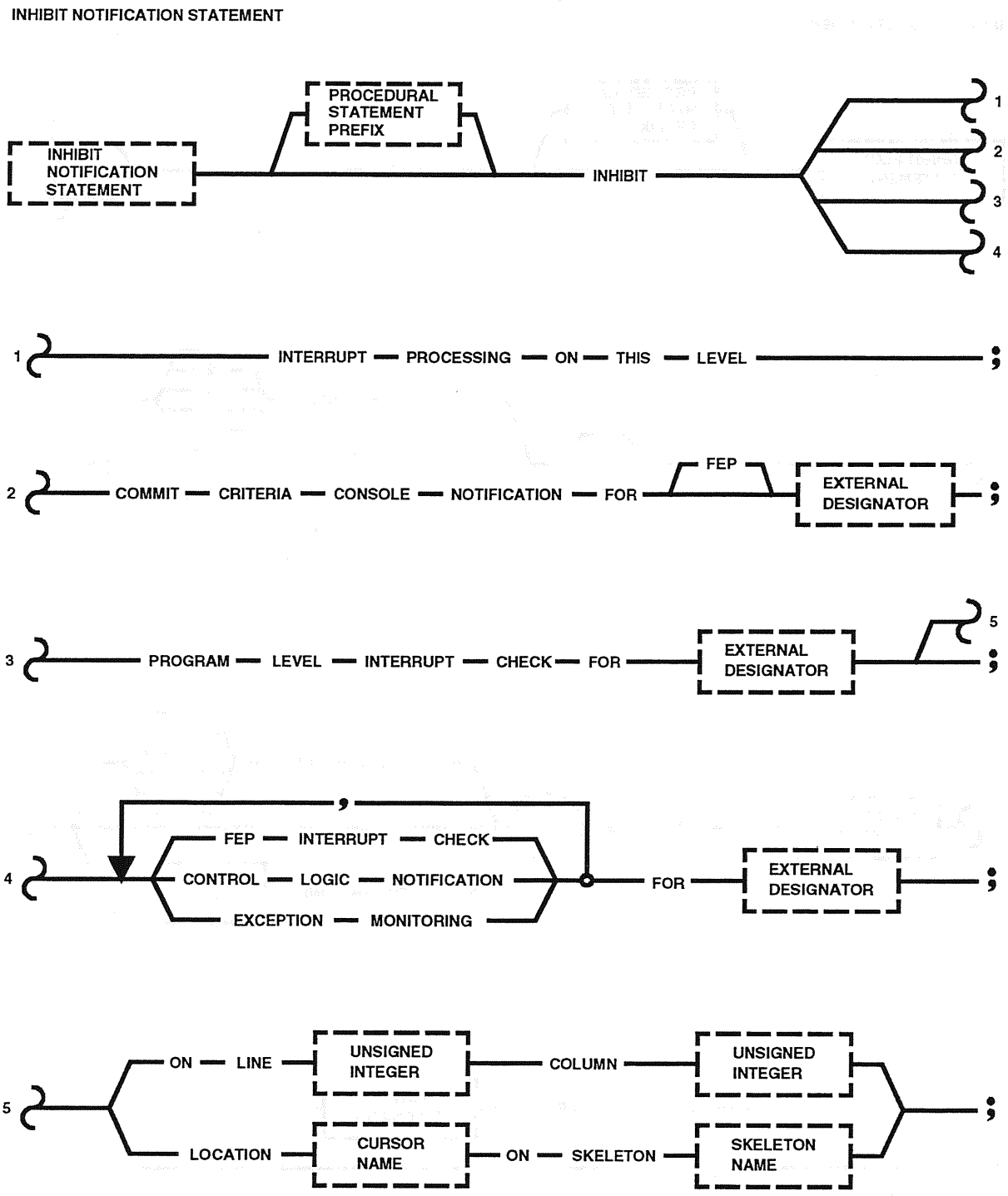


Figure 1-70 Inhibit Notification Statement

INHIBIT PLOT STATEMENT

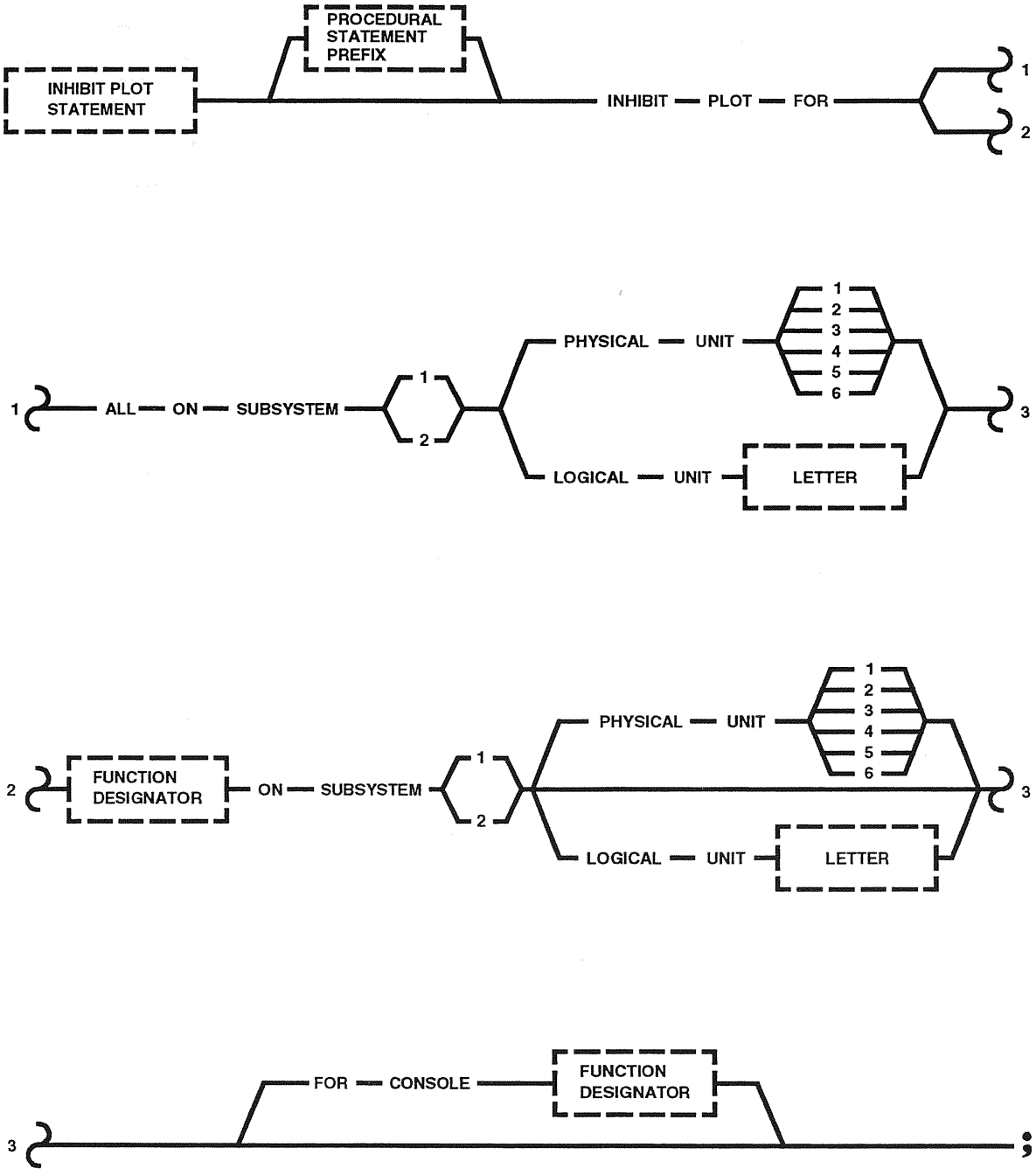


Figure 1-71 Inhibit Plot Statement

INHIBIT SYSTEM STATEMENT

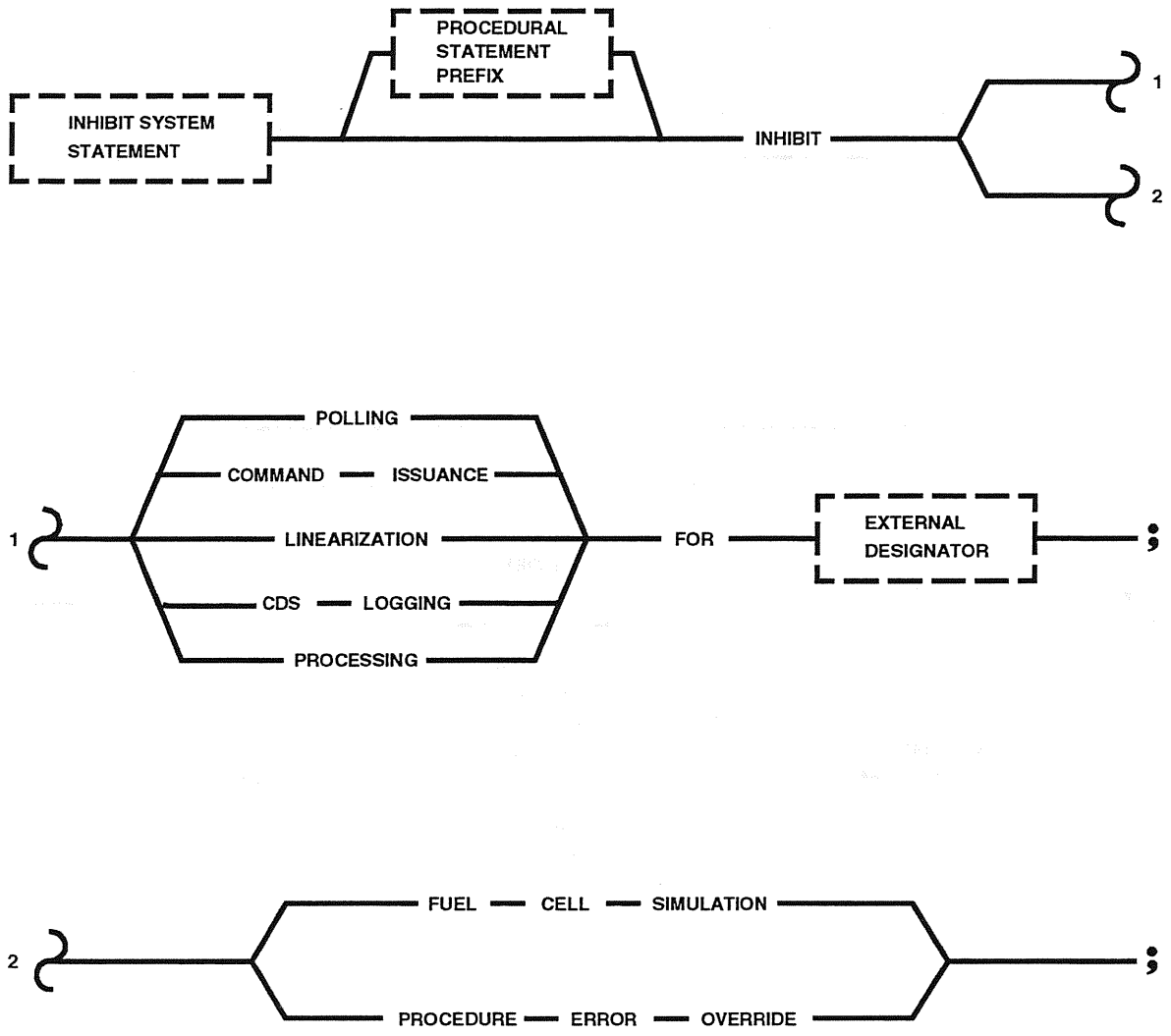


Figure 1-72 Inhibit System Statement

INHIBIT TABLE STATEMENT

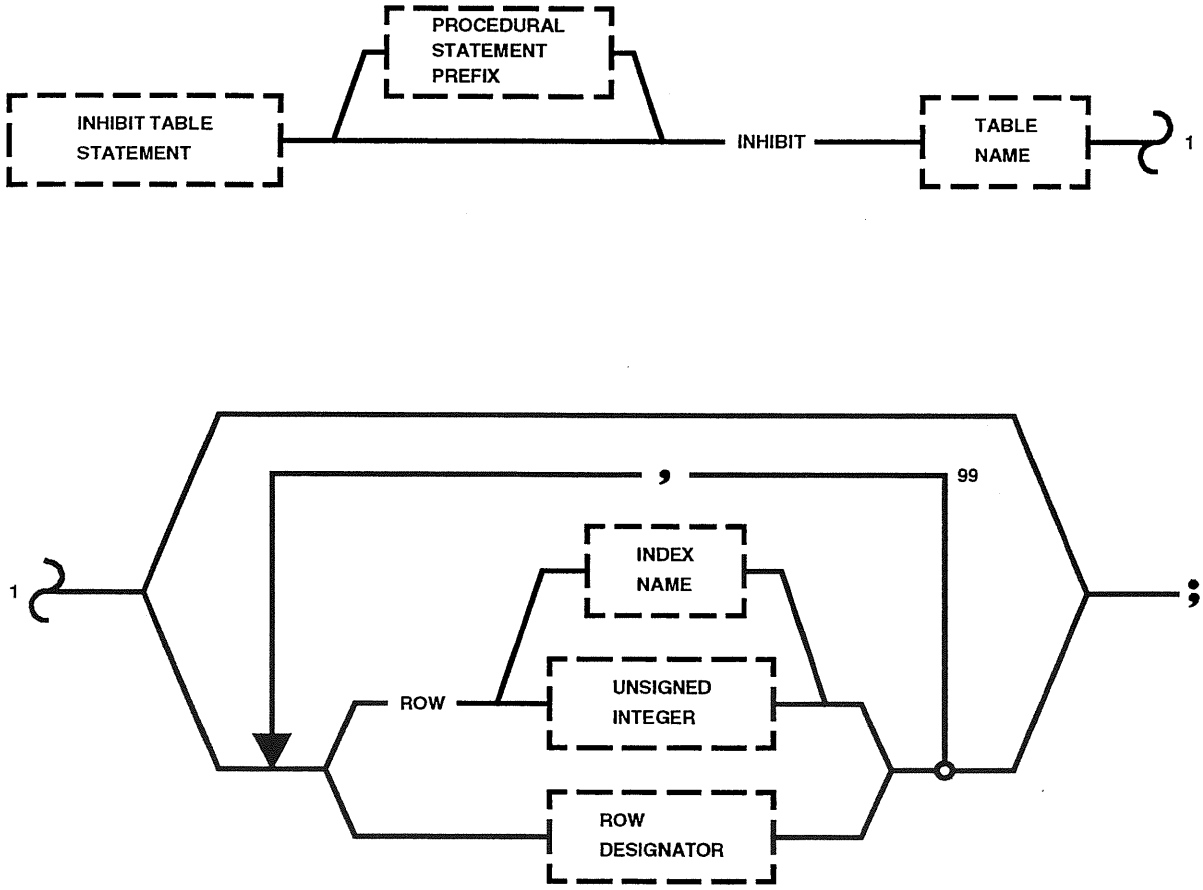


Figure 1-73 Inhibit Table Statement

INTEGER NUMBER

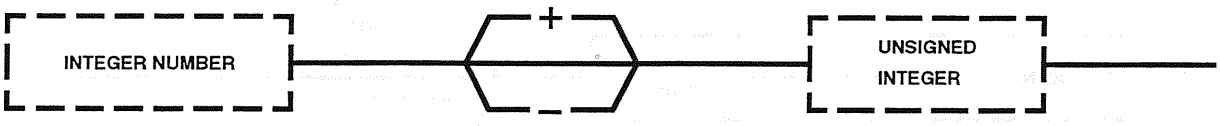


Figure 1-74 Integer Number

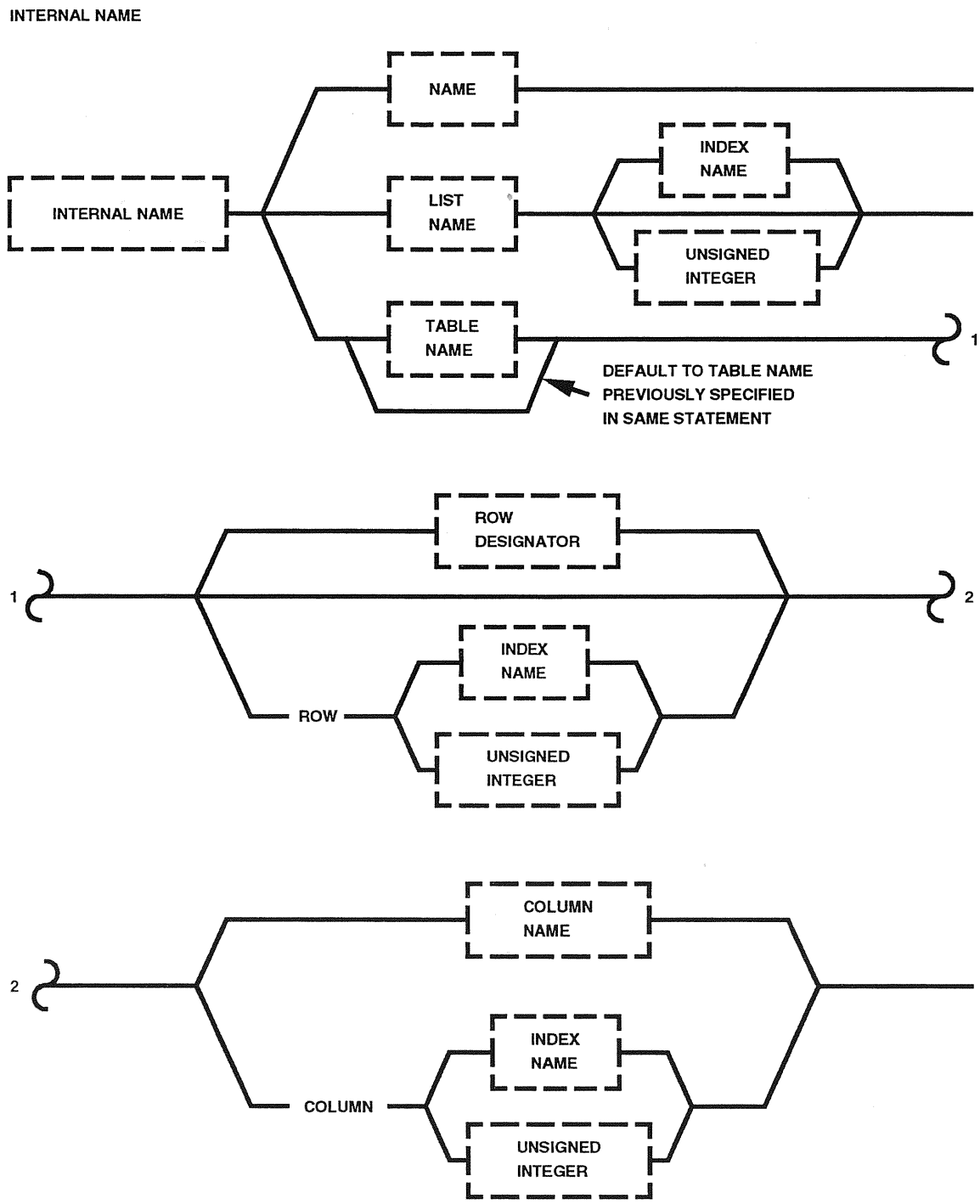


Figure 1-75 Internal Name

ISSUE DIGITAL PATTERN STATEMENT

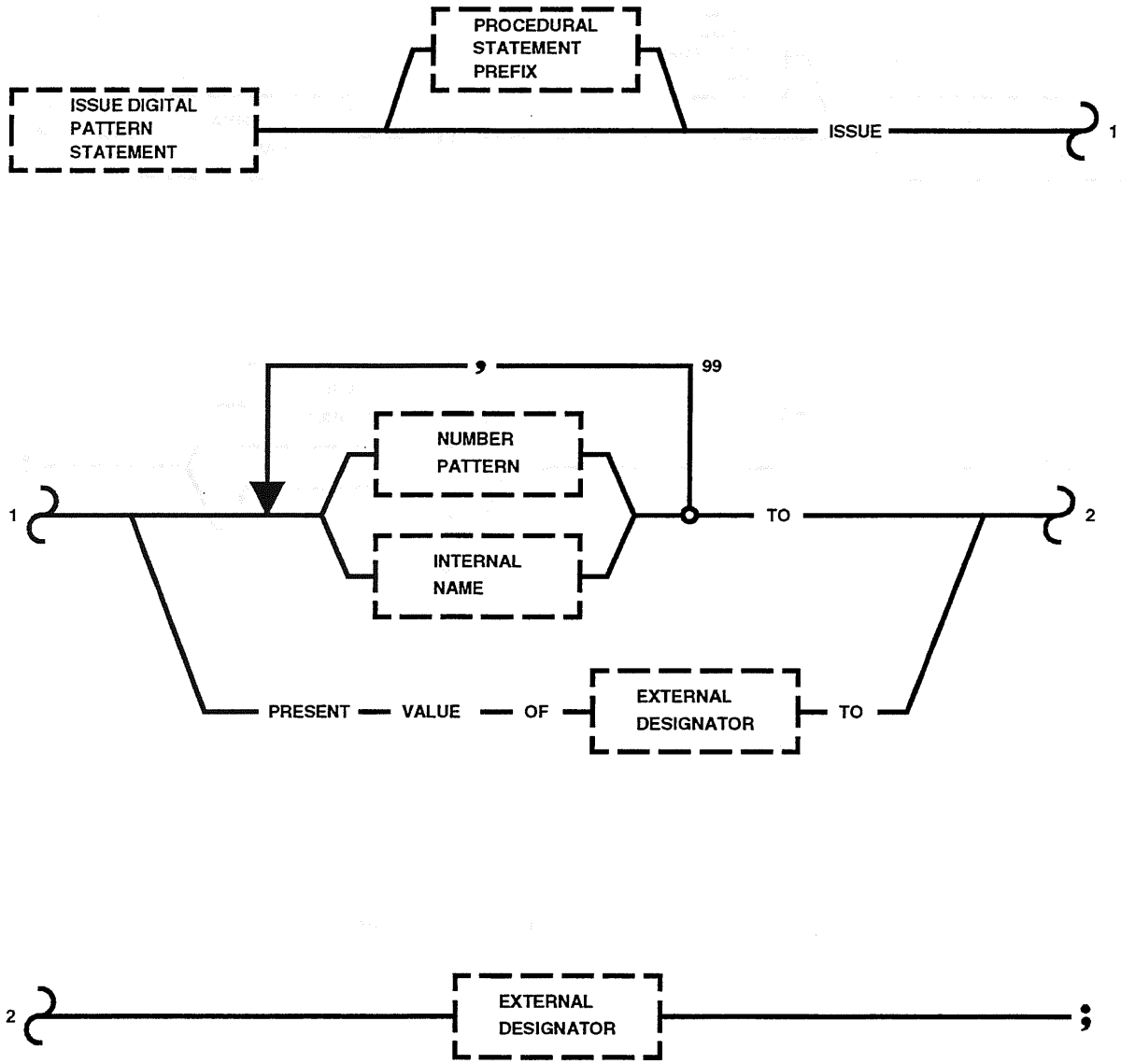


Figure 1-76 Issue Digital Pattern Statement

LET EQUAL STATEMENT

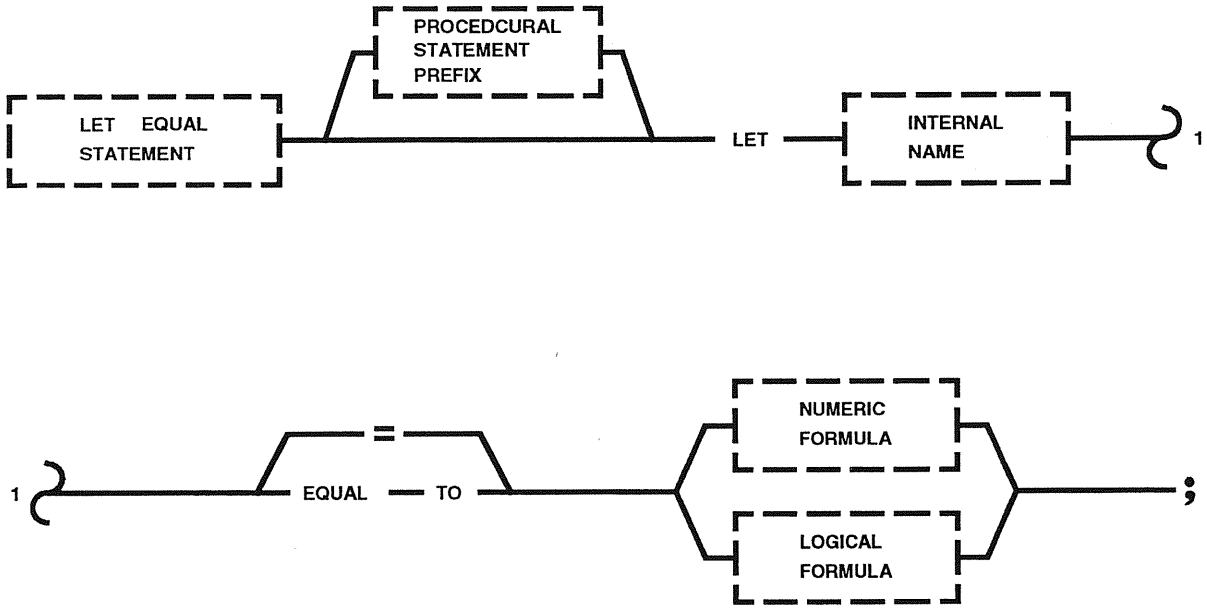


Figure 1-77 Let Equal Statement

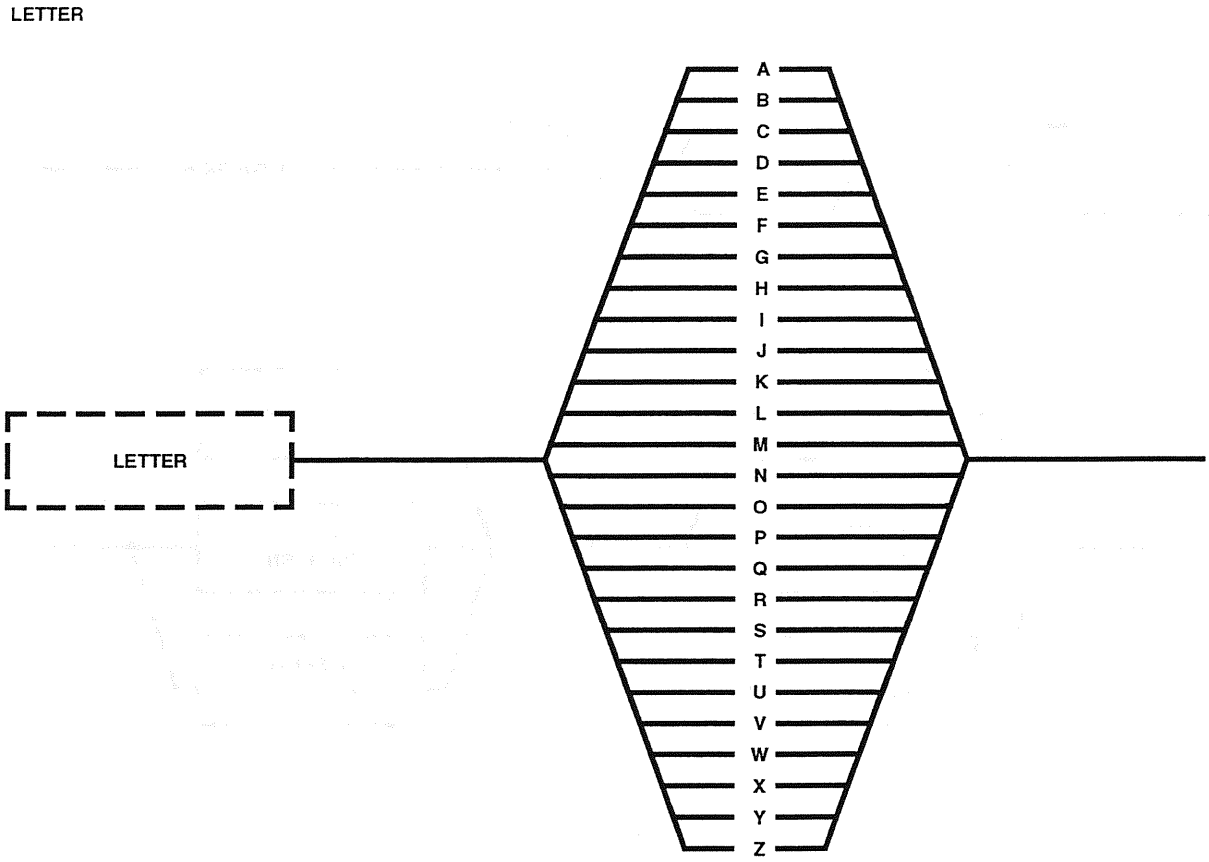


Figure 1-78 Letter

LIMIT FORMULA

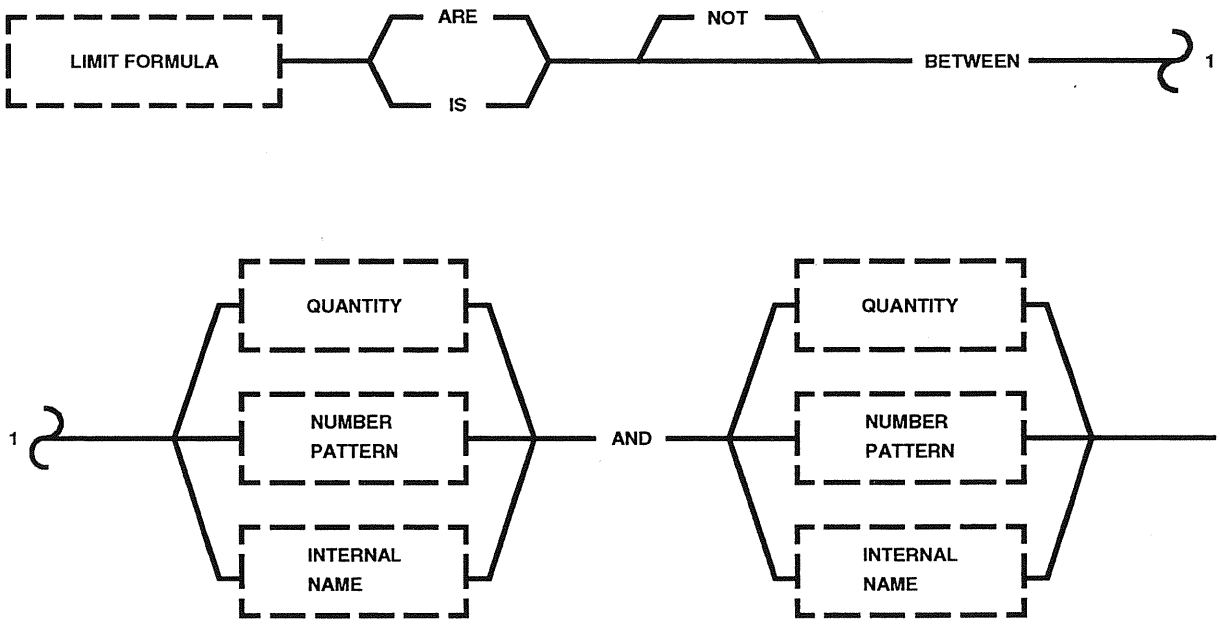


Figure 1-79 Limit Formula

LINE COMMAND

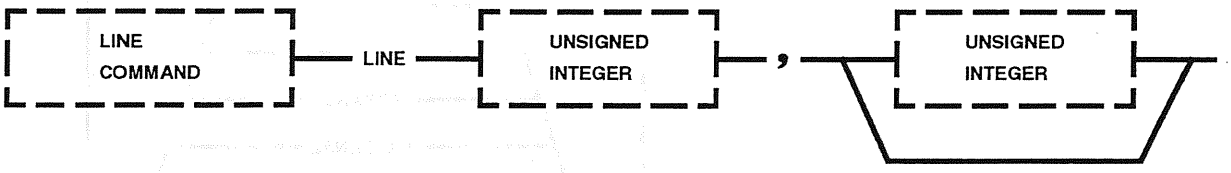


Figure 1-80 Line Command

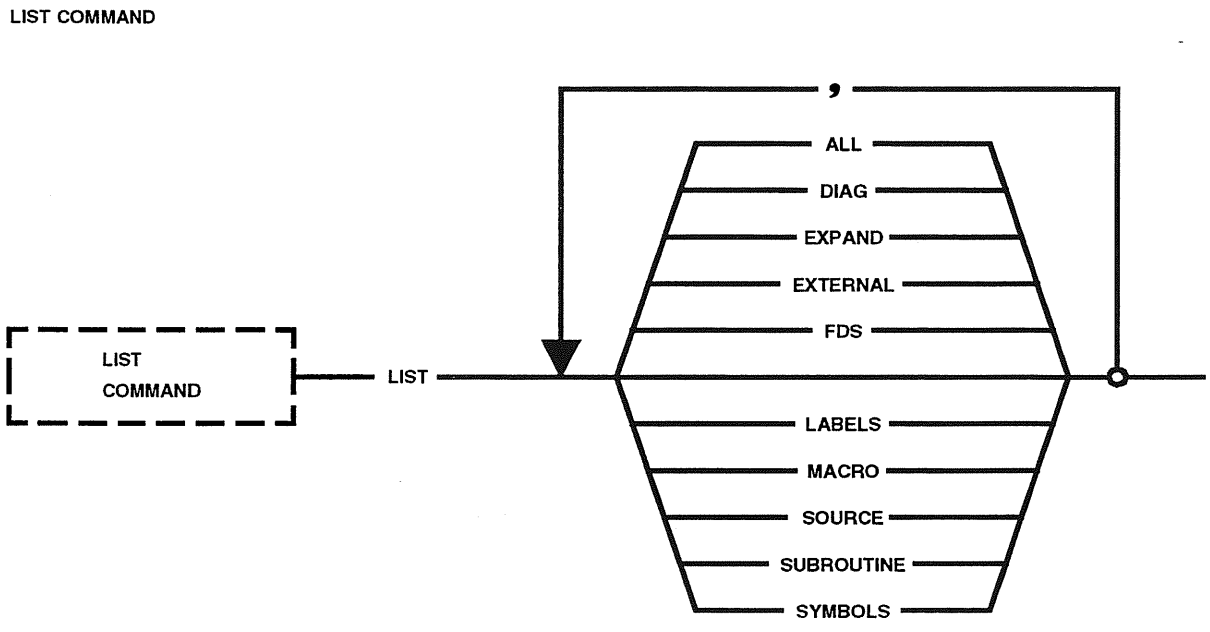


Figure 1-81 List Command

LIST NAME

Figure 1-82 List Name

LIST OBJECT COMMAND

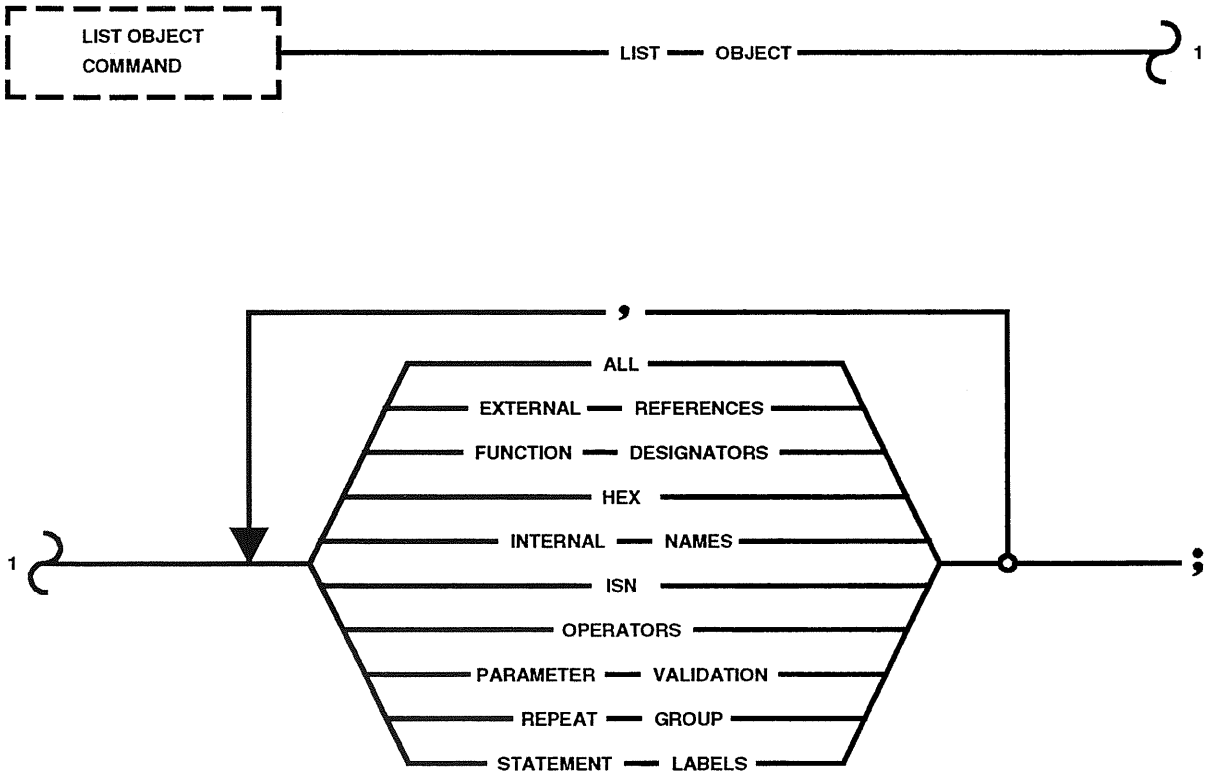


Figure 1-83 List Object Command

LOAD SAFING SEQUENCE STATEMENT

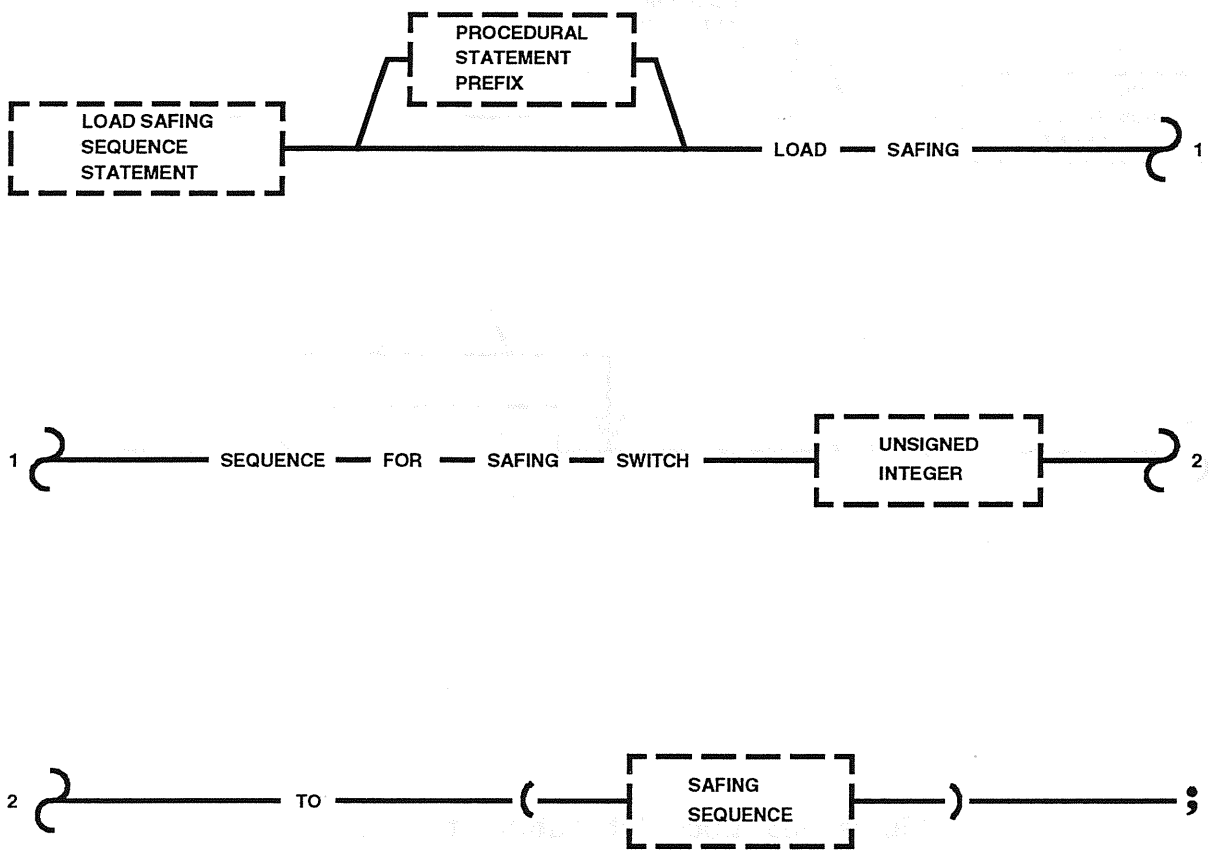


Figure 1-84 Load Safing Sequence Statement

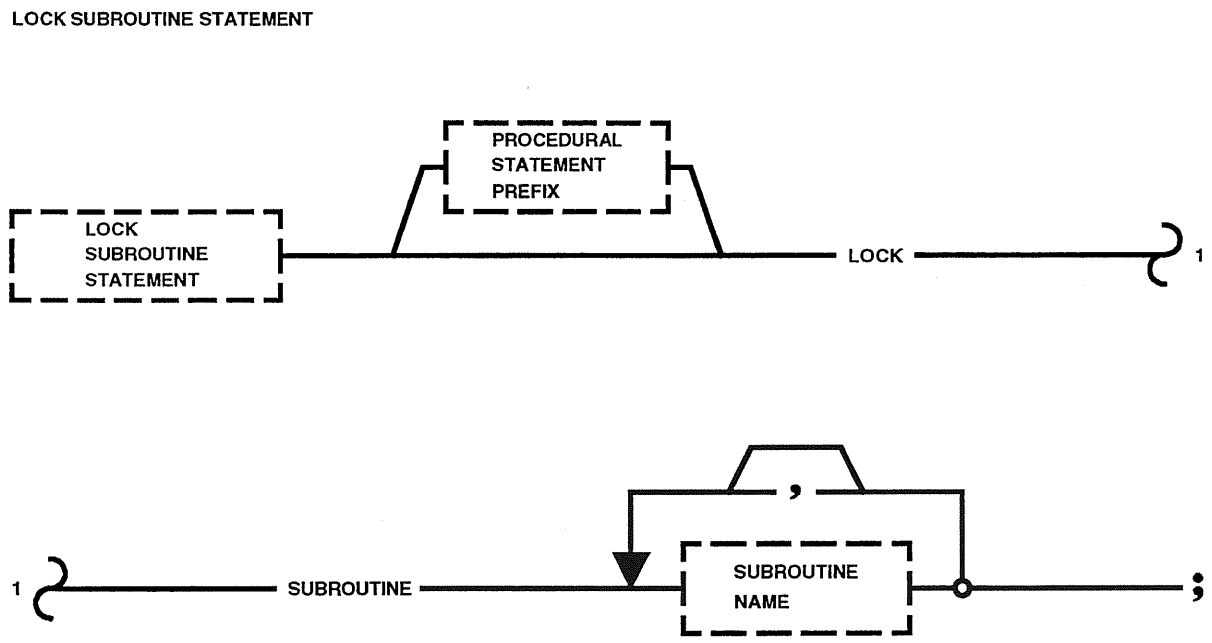


Figure 1-85 Lock Subroutine Statement

LOGICAL FORMULA

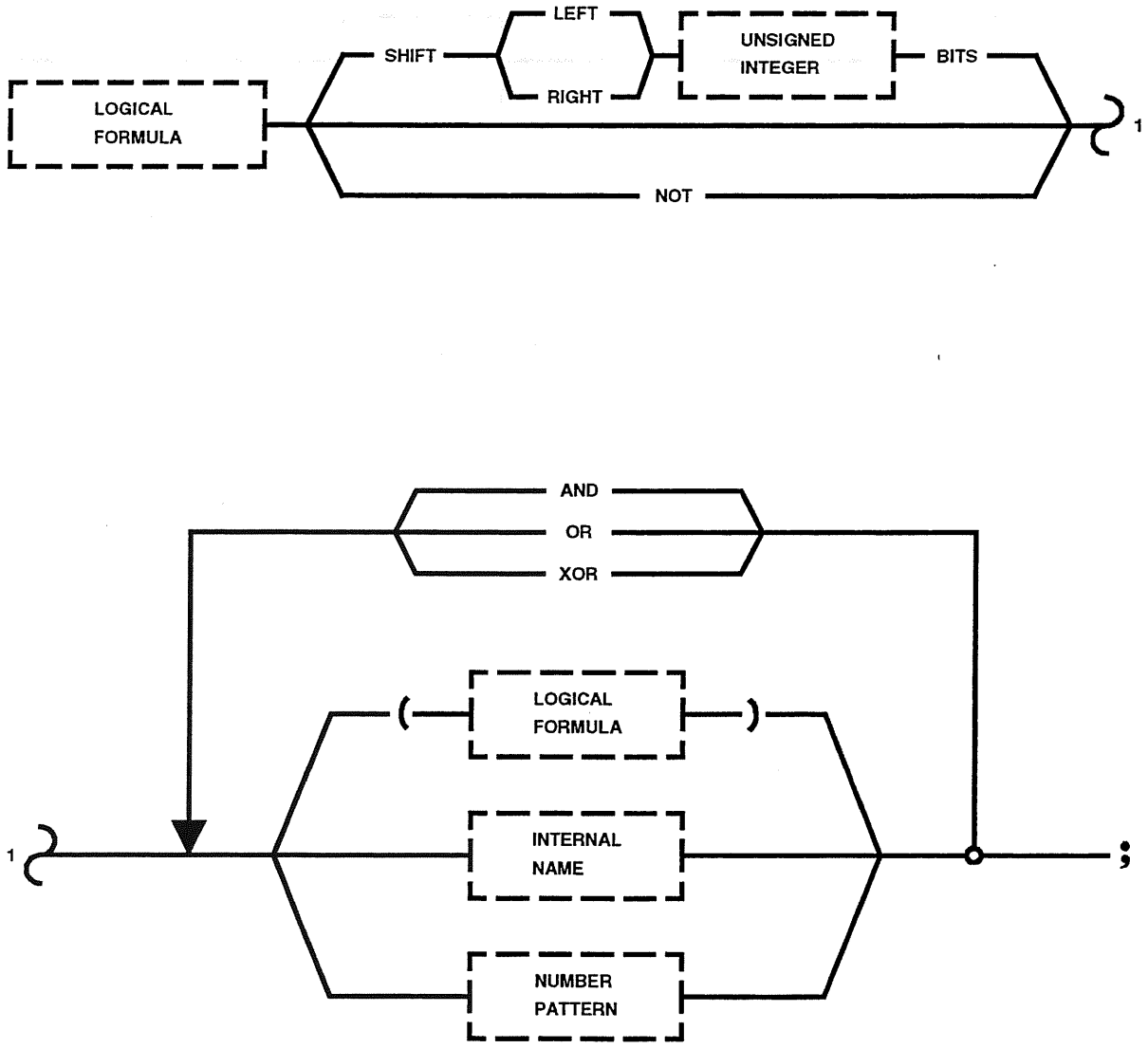


Figure 1-86 Logical Formula

MACRO NAME

Figure 1-87 Macro Name

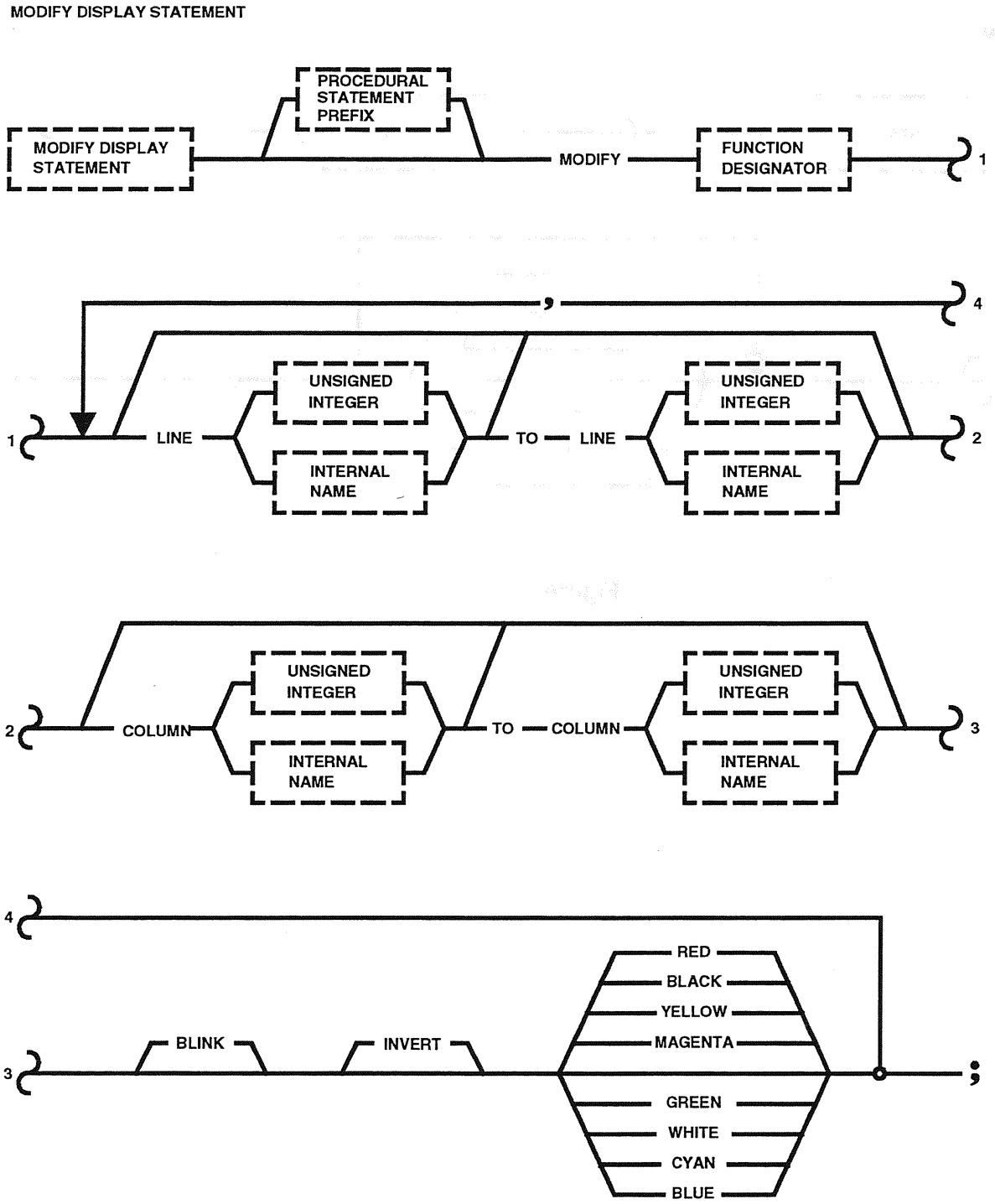


Figure 1-88 Modify Display Statement

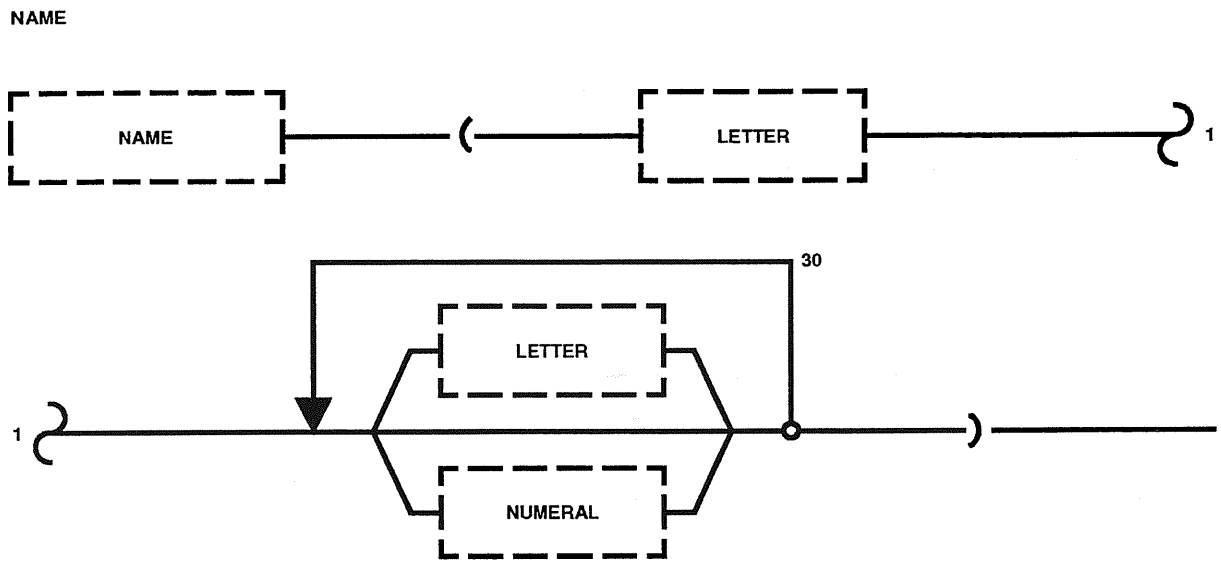


Figure 1-89 Name

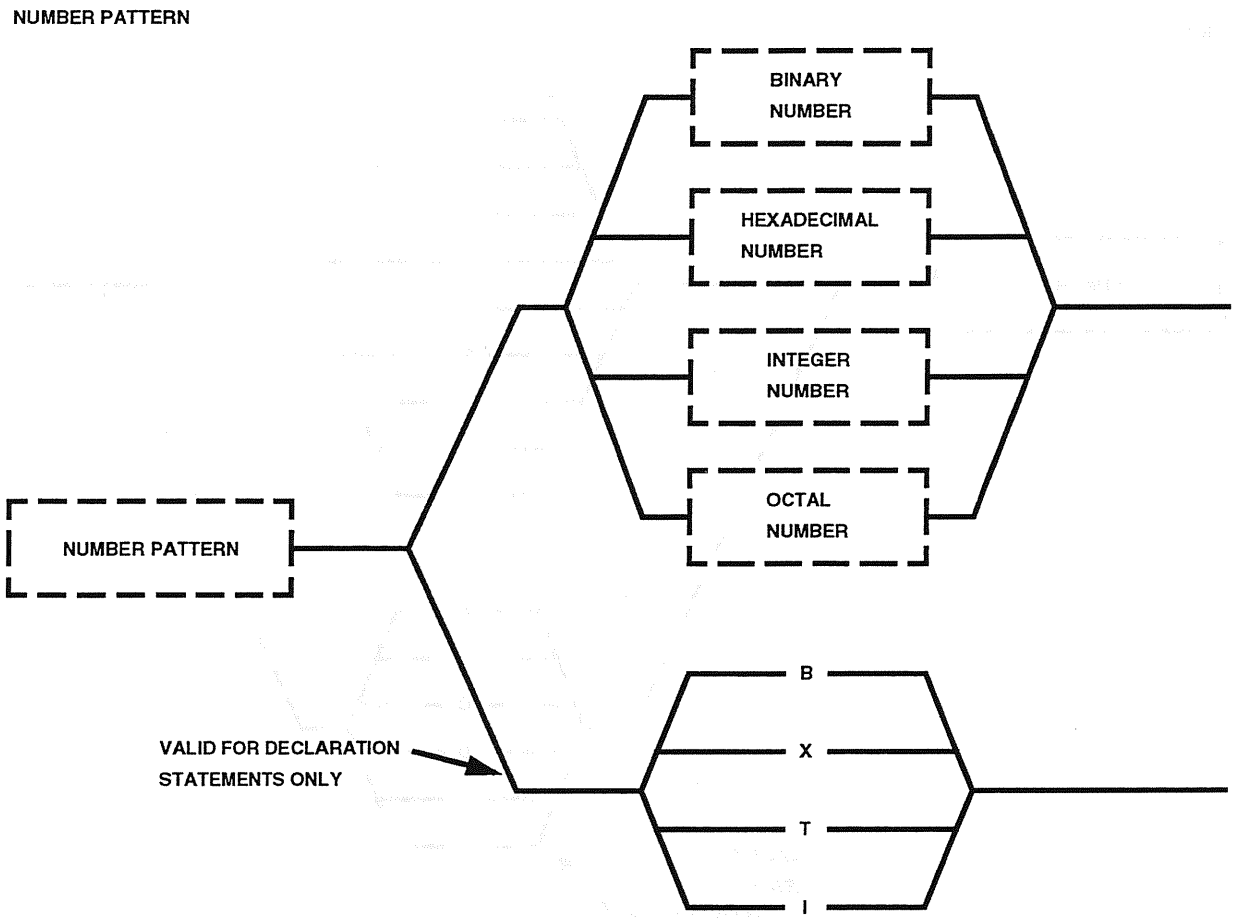


Figure 1-90 Number Pattern

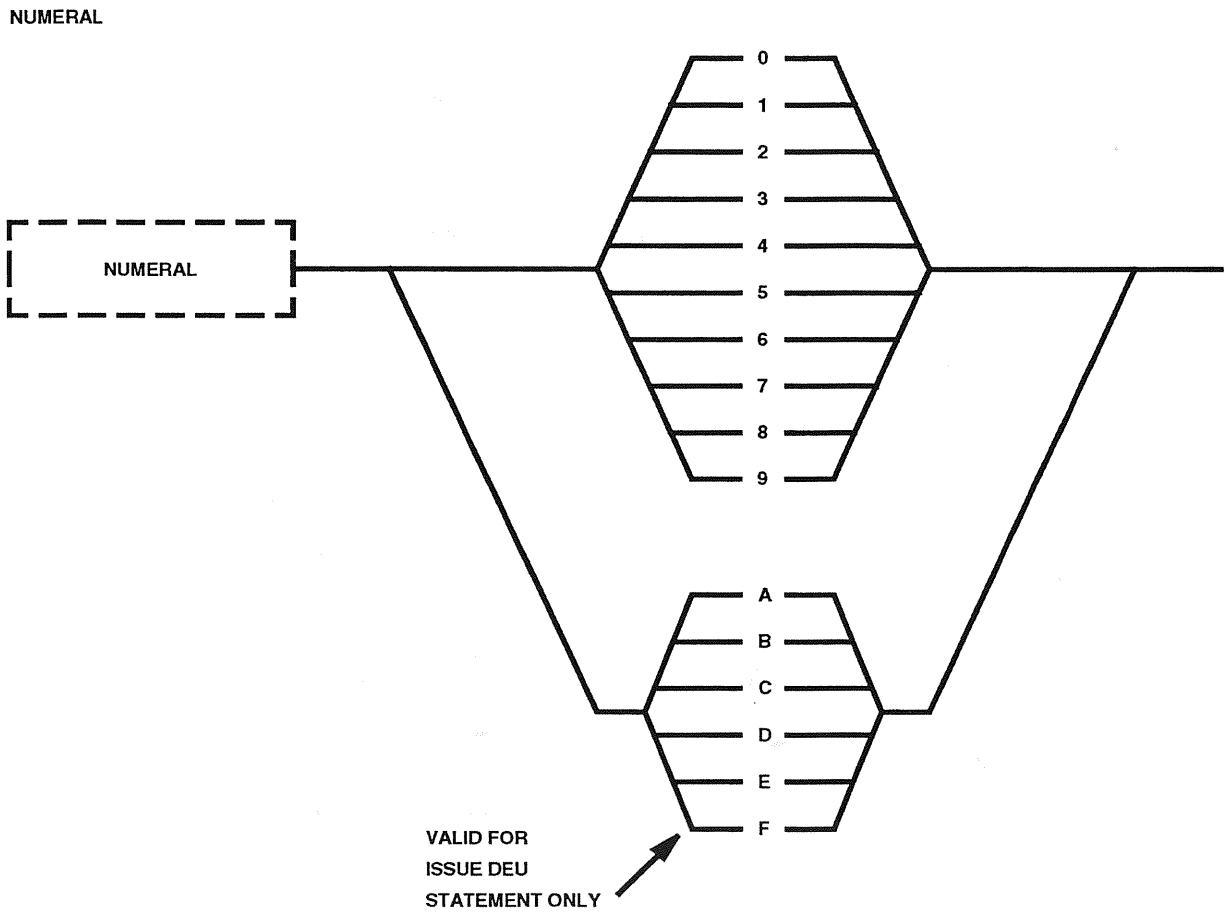


Figure 1-91 Numeral

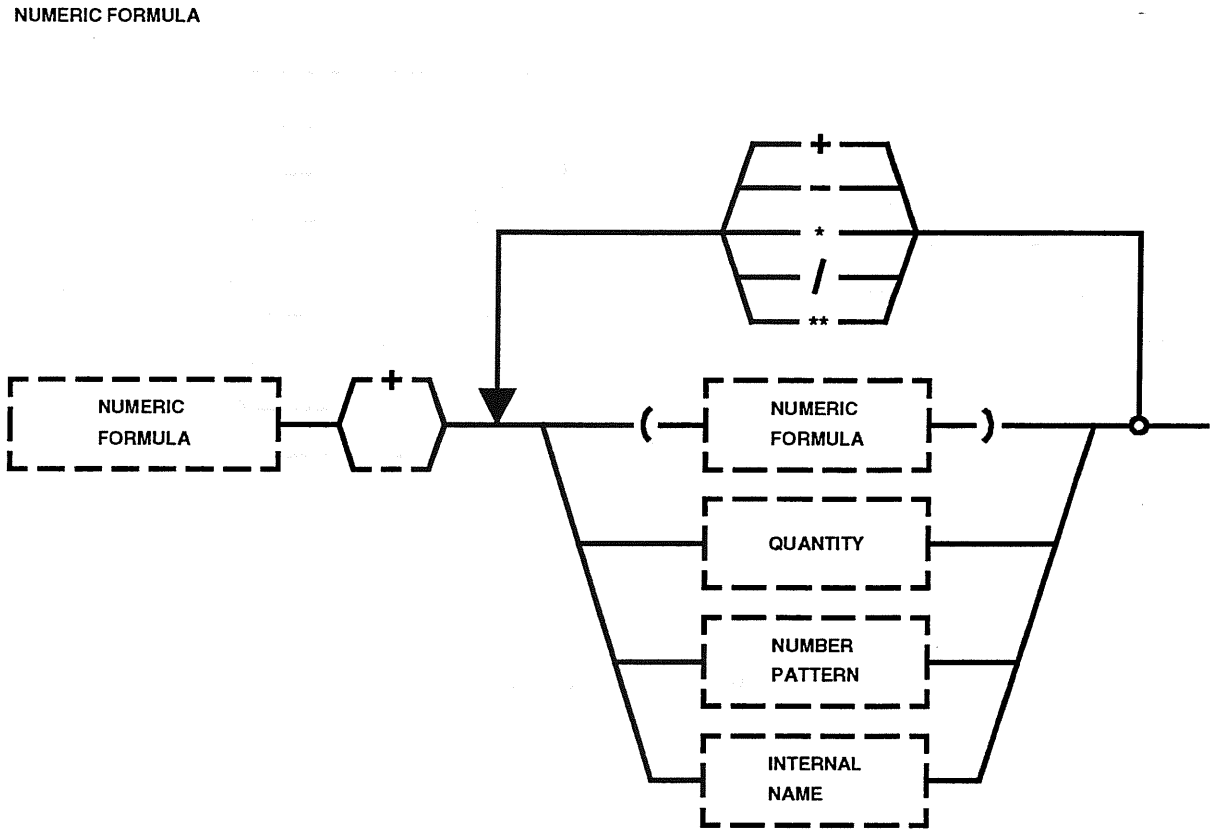


Figure 1-92 Numeric Formula

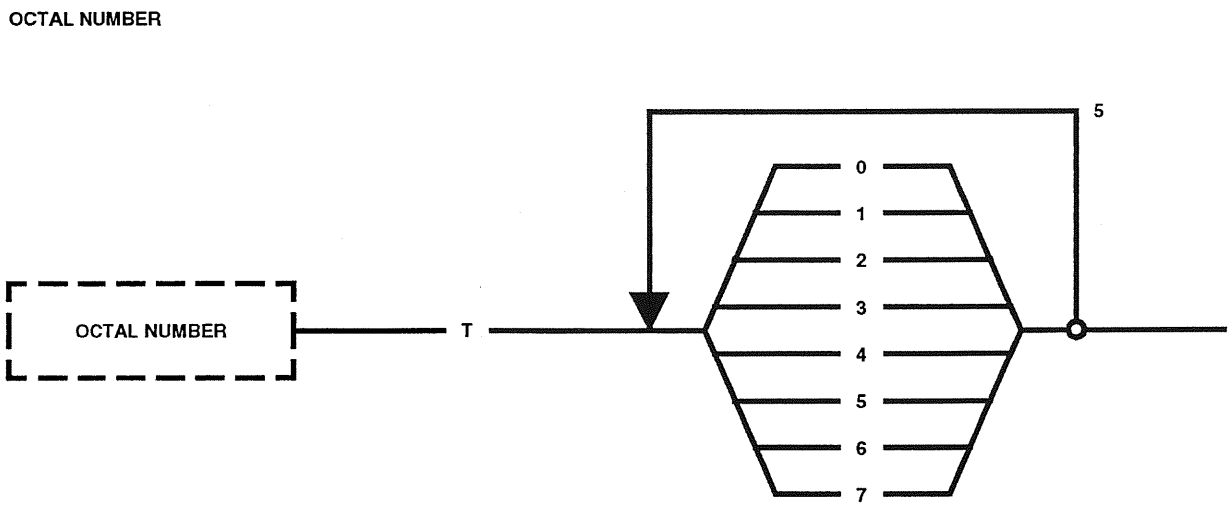


Figure 1-93 Octal Number

OPEN DISK FILE STATEMENT

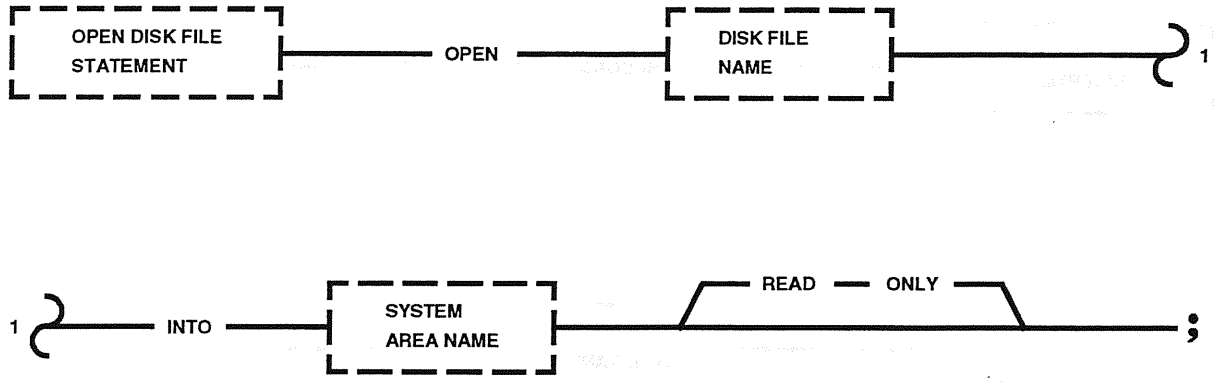


Figure 1-94 Open Disk File Statement

OUTPUT EXCEPTION

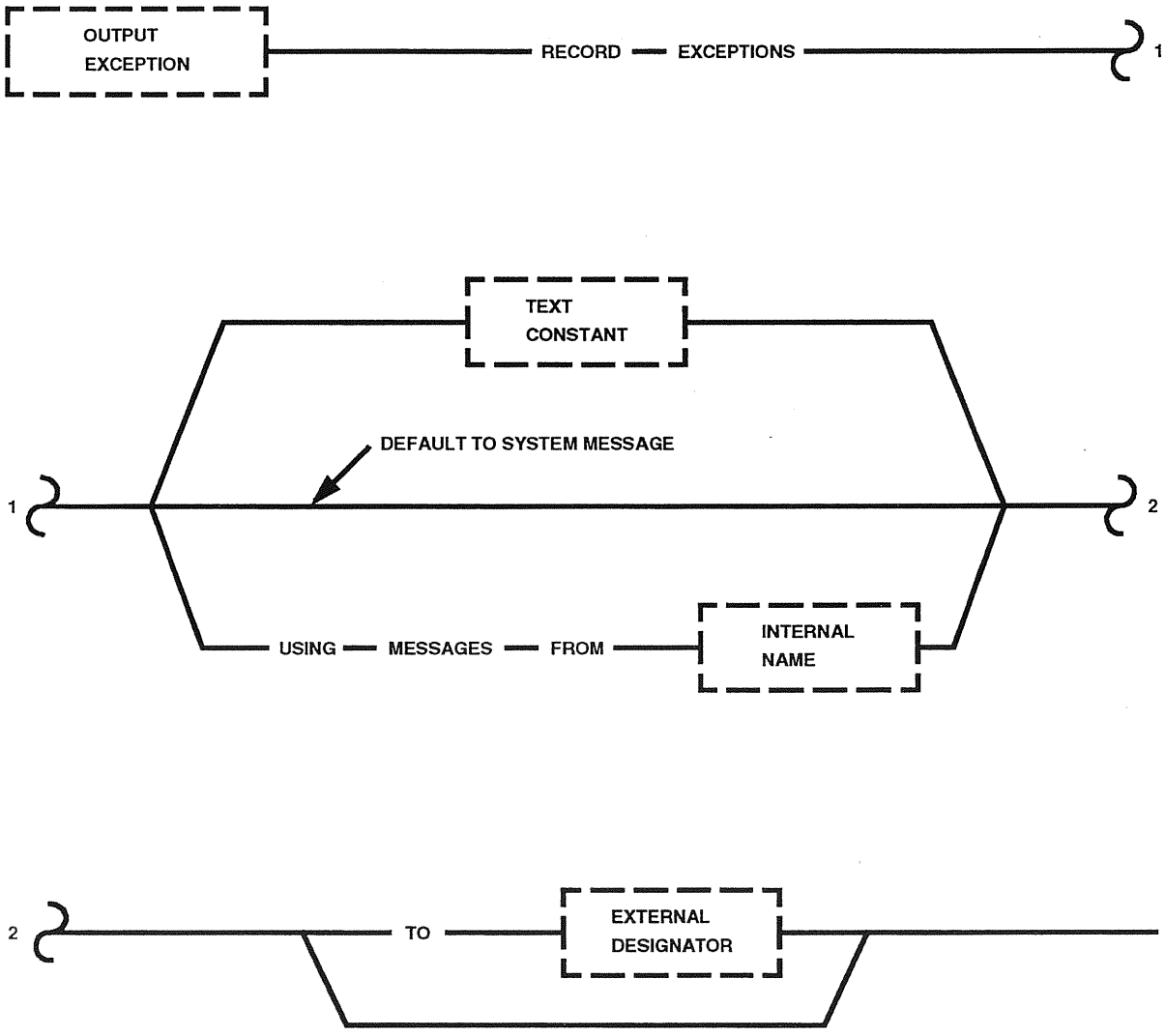


Figure 1-95 Output Exception

PAGE COMMAND

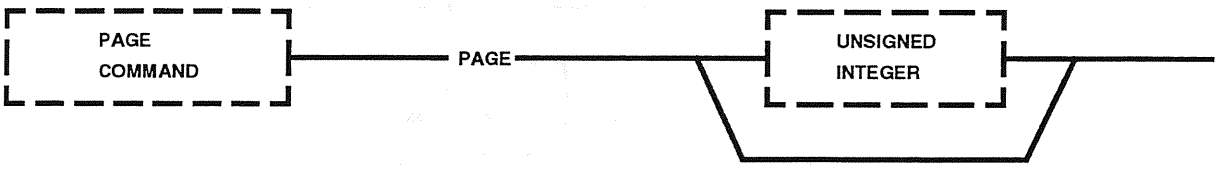


Figure 1-96 Page Command

PARAMETER

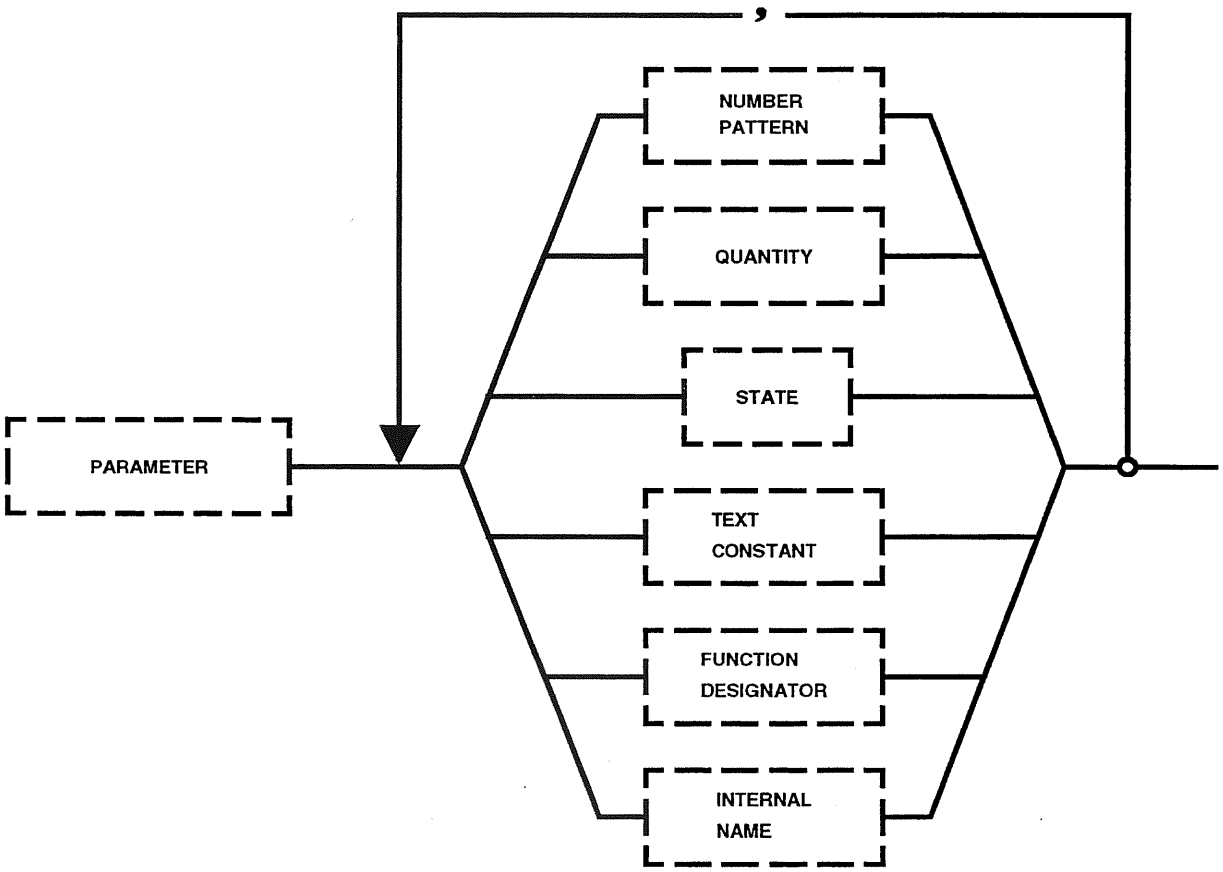


Figure 1-97 Parameter

PERFORM PROGRAM STATEMENT

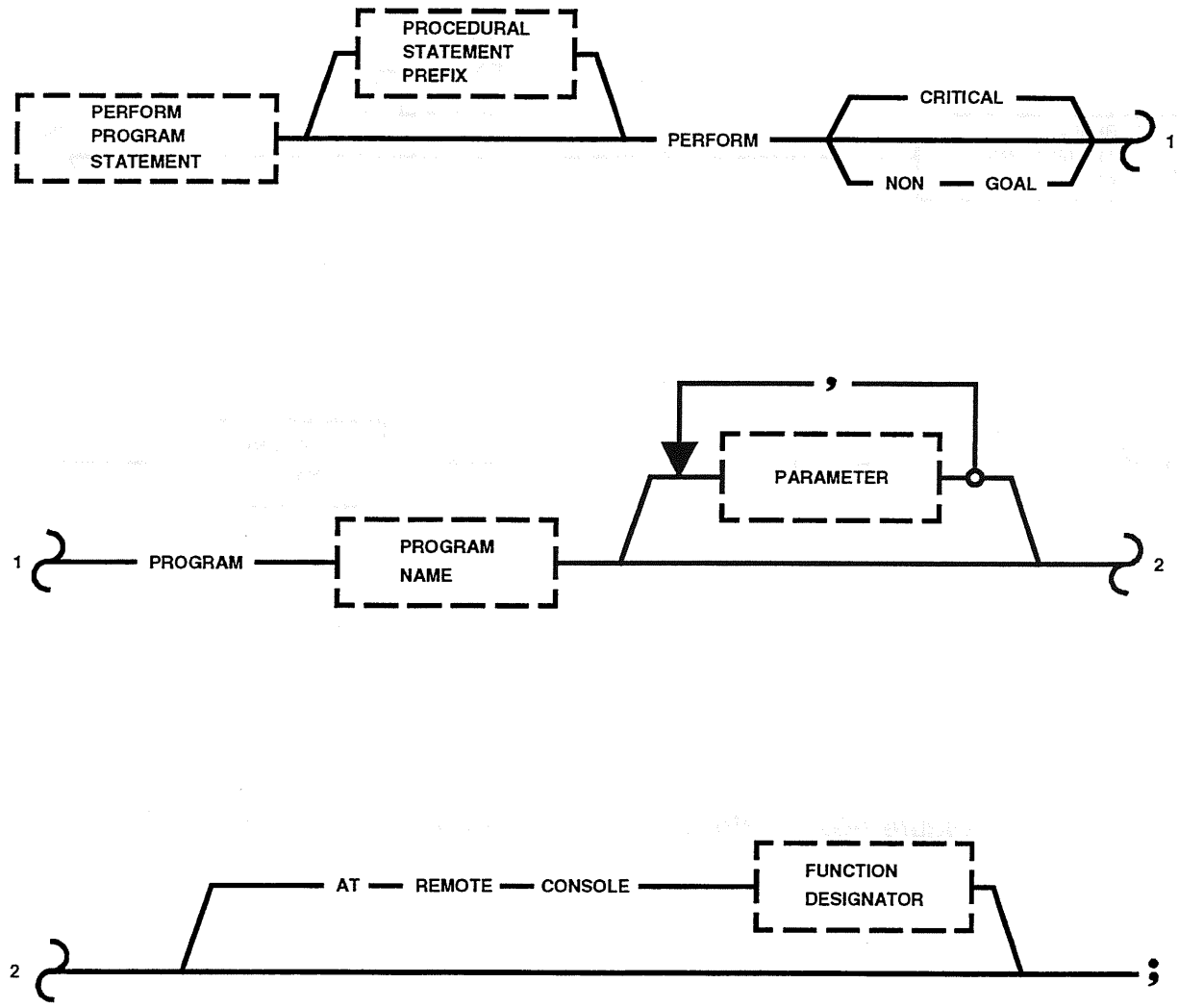


Figure 1-98 Perform Program Statement

PERFORM STATEMENT GROUPS STATEMENT

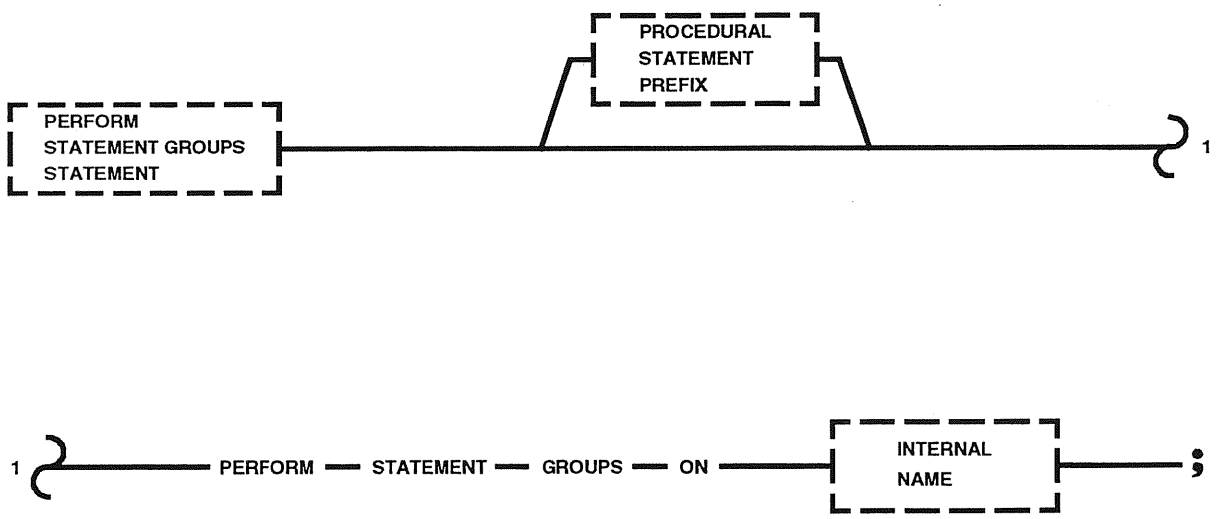


Figure 1-99 Perform Statement Groups Statement

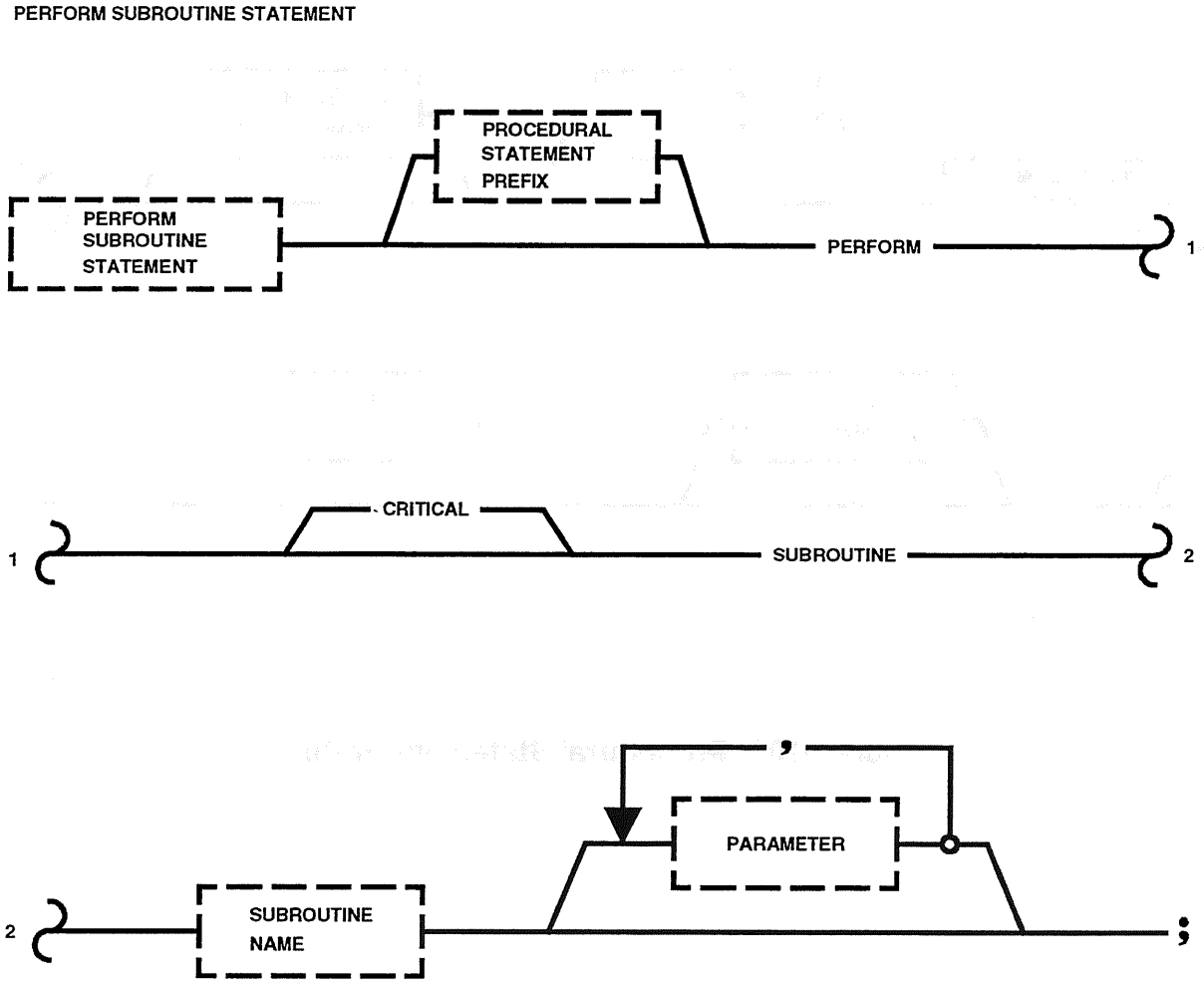


Figure 1-100 Perform Subroutine Statement

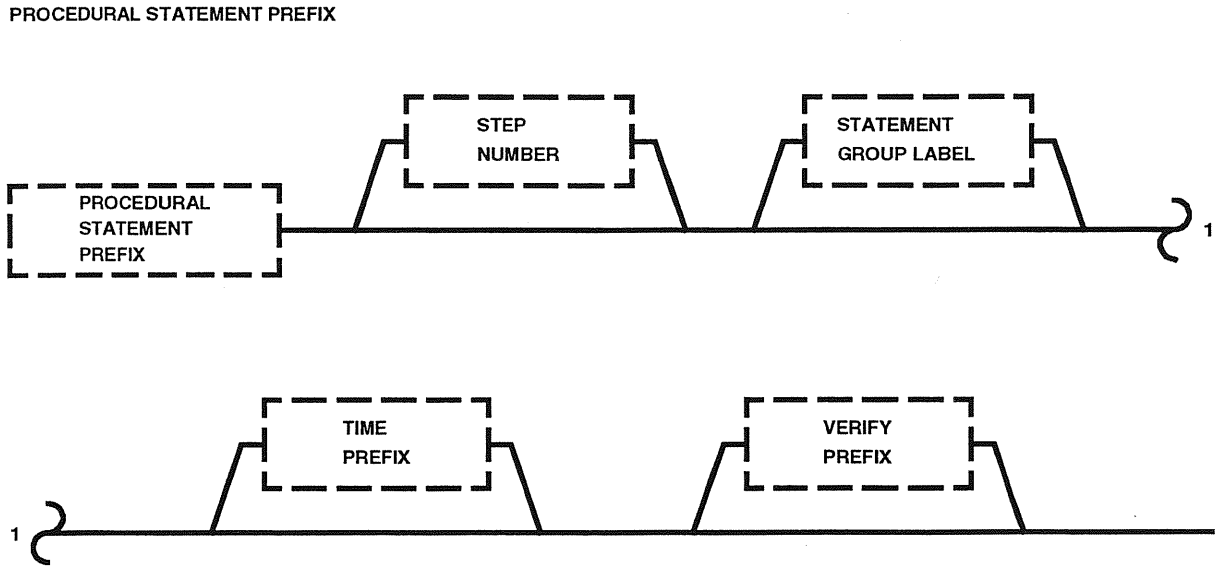


Figure 1-101 Procedural Statement Prefix

PROGRAM NAME

Figure 1-102 Program Name

PSUEDO PARAMETER

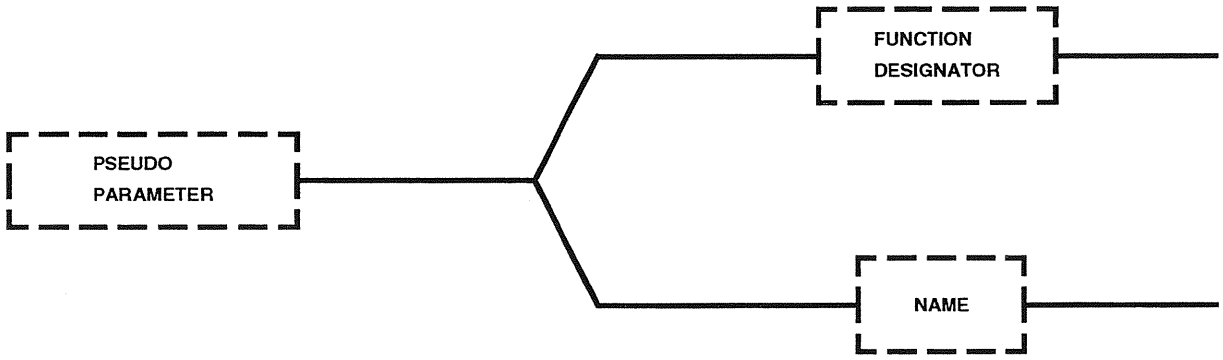


Figure 1-103 Pseudo Parameter

QUANTITY

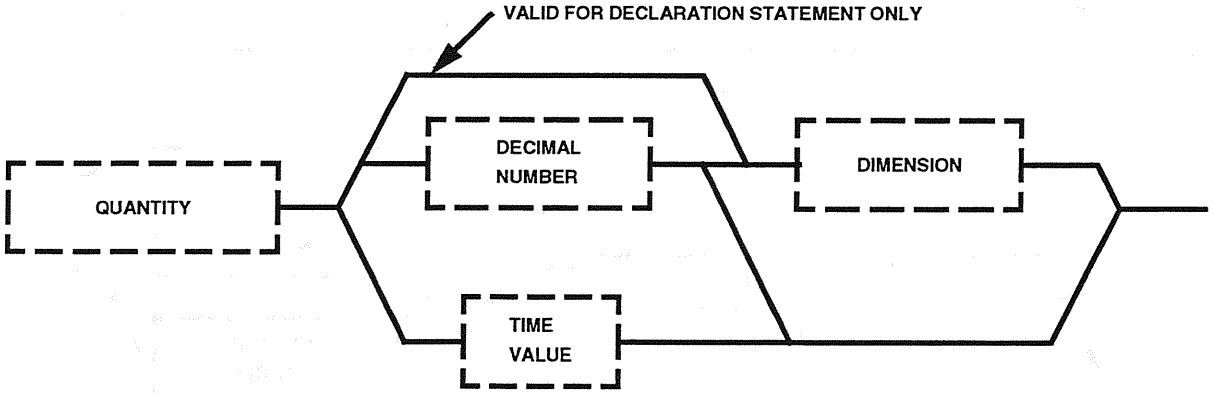


Figure 1-104 Quantity

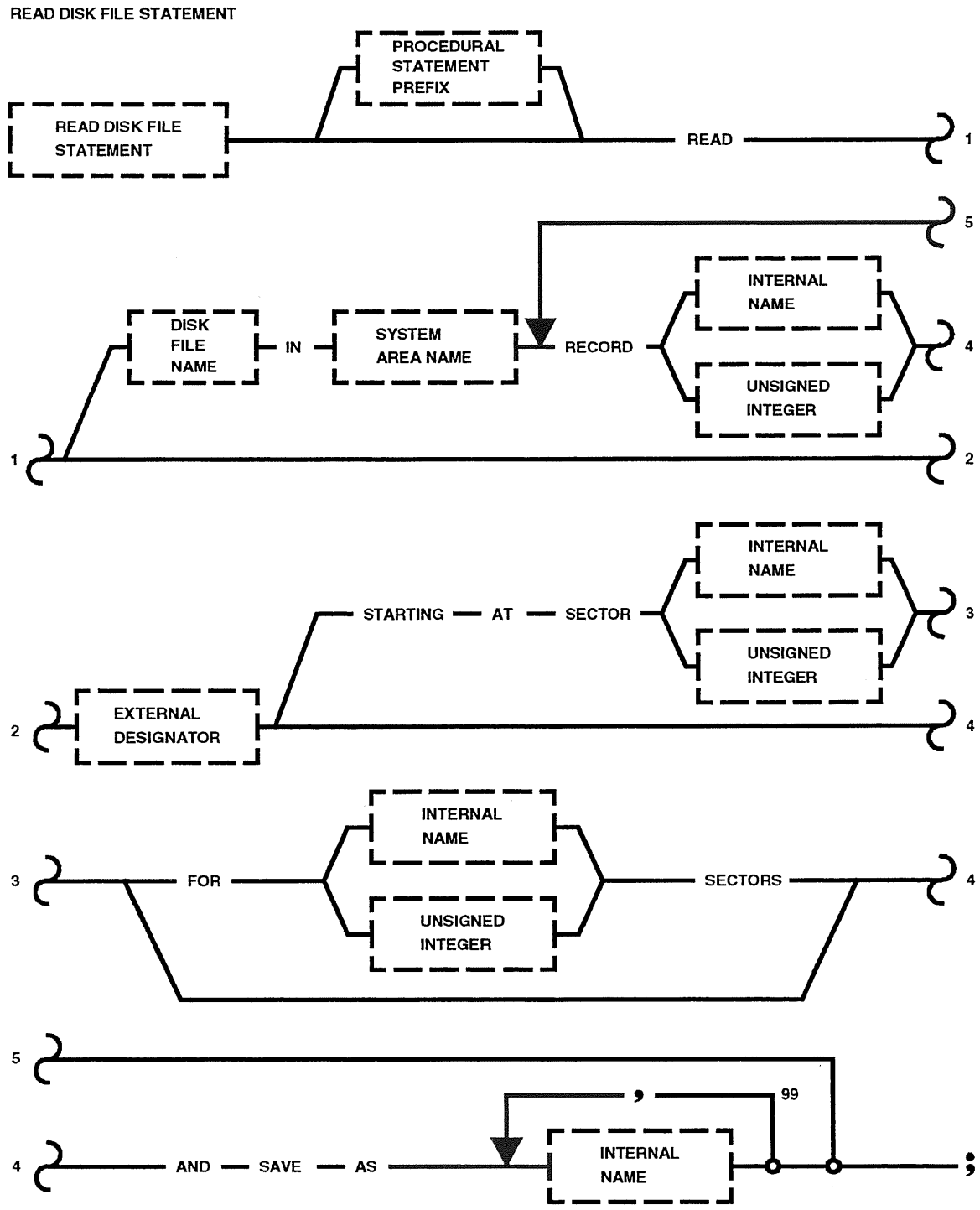


Figure 1-105 Read Disk File Statement

READ FEP STATUS STATEMENT

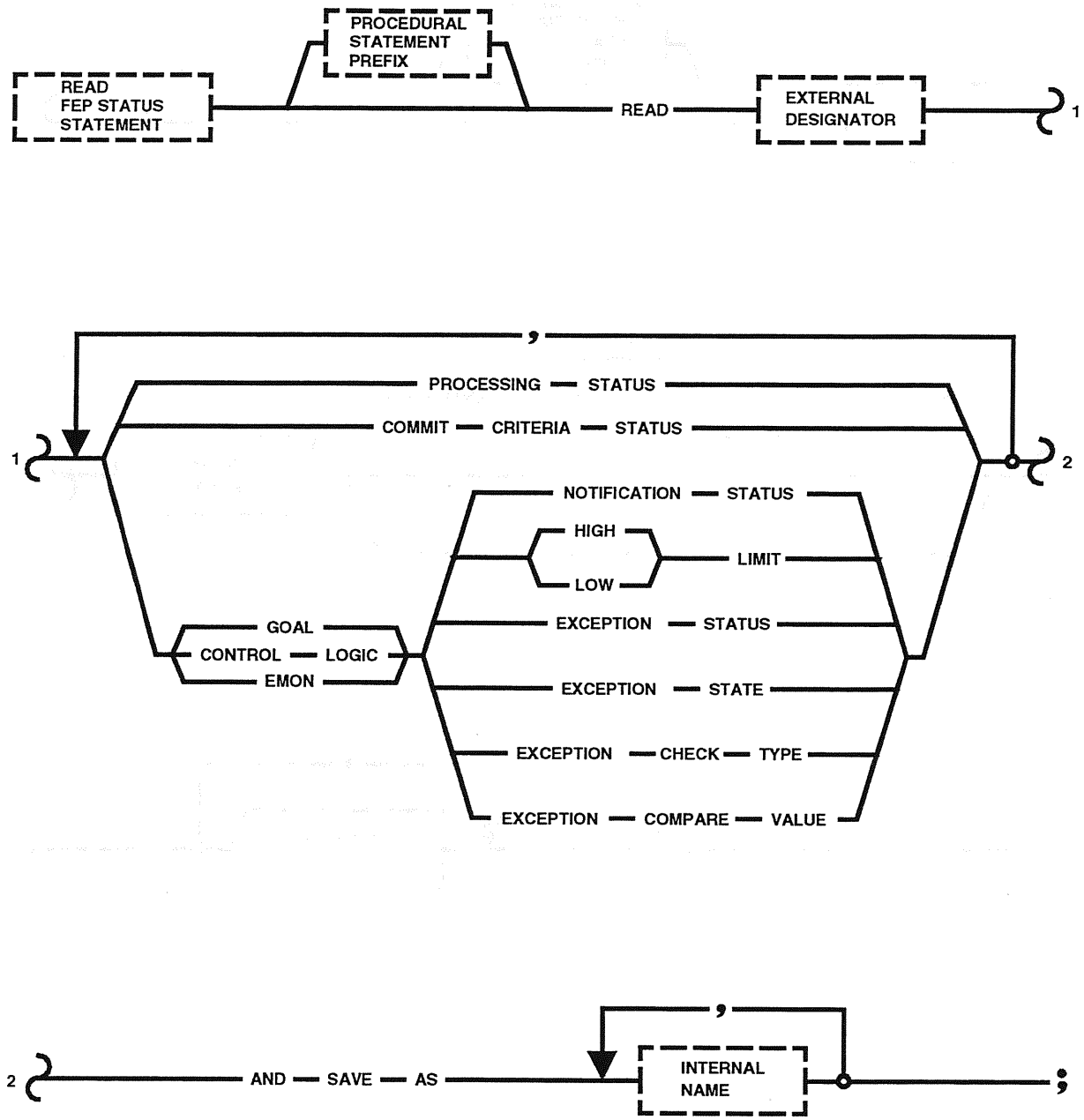


Figure 1-106 Read FEP Status Statement

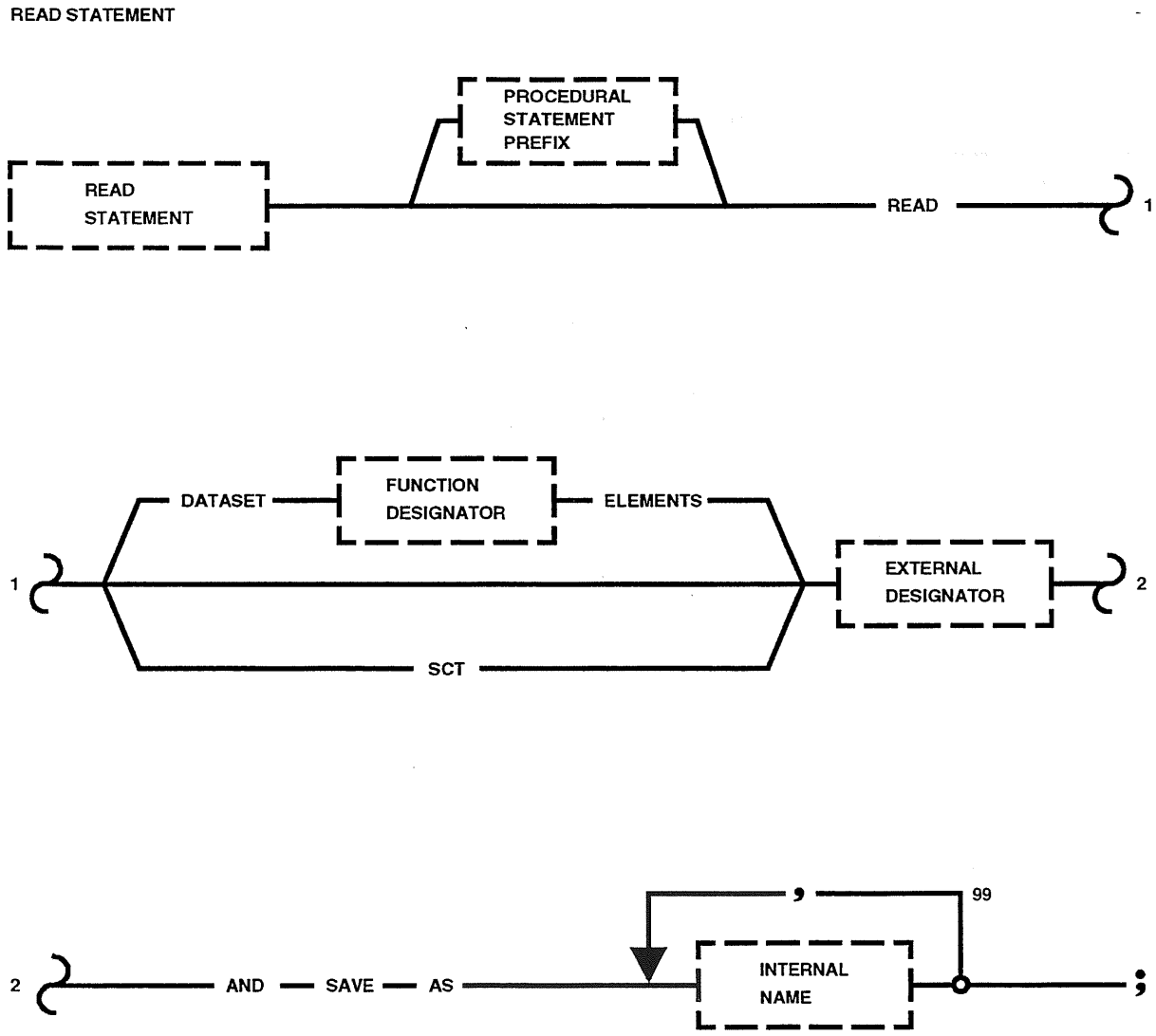


Figure 1-107 Read Statement

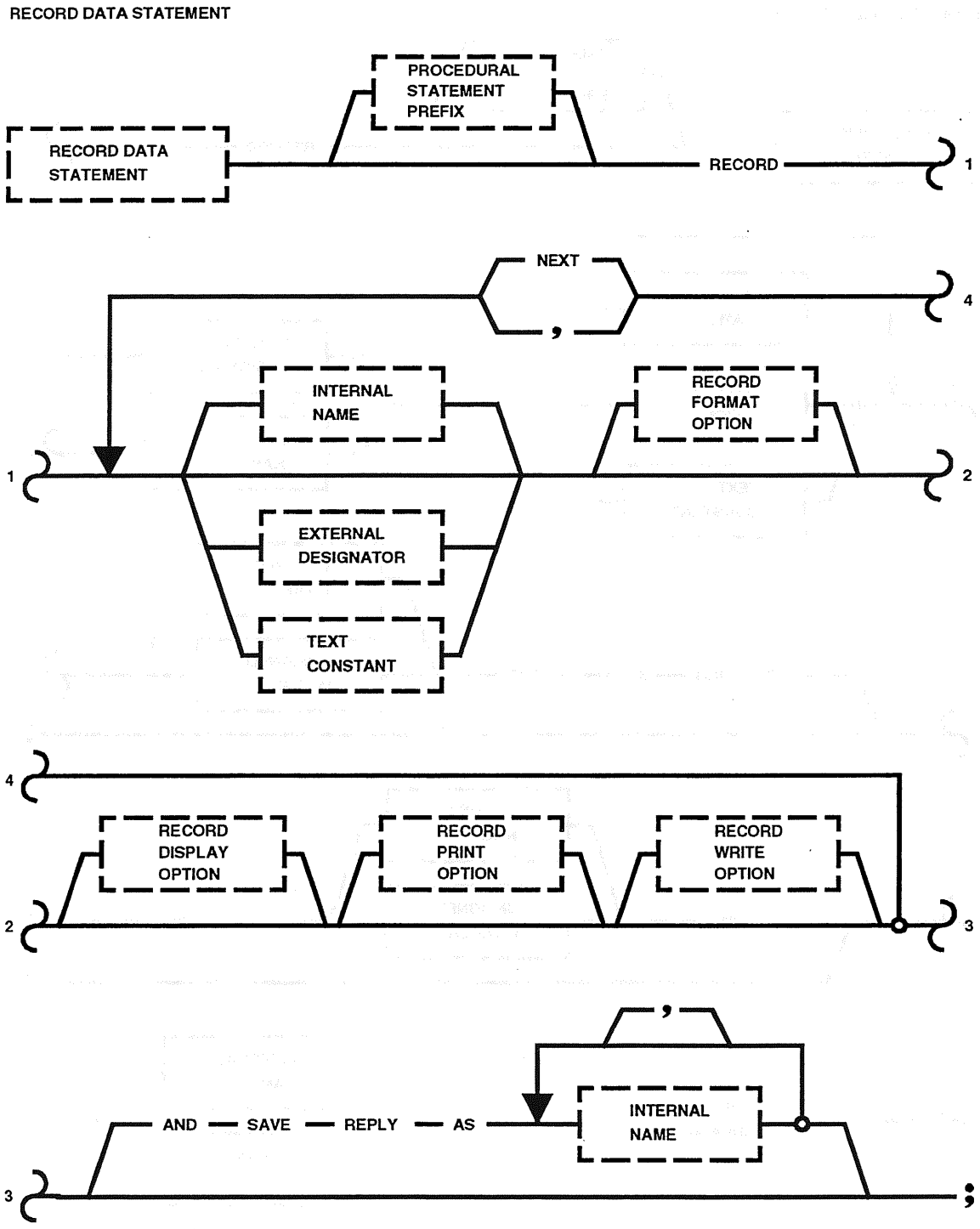


Figure 1-108 Record Data Statement

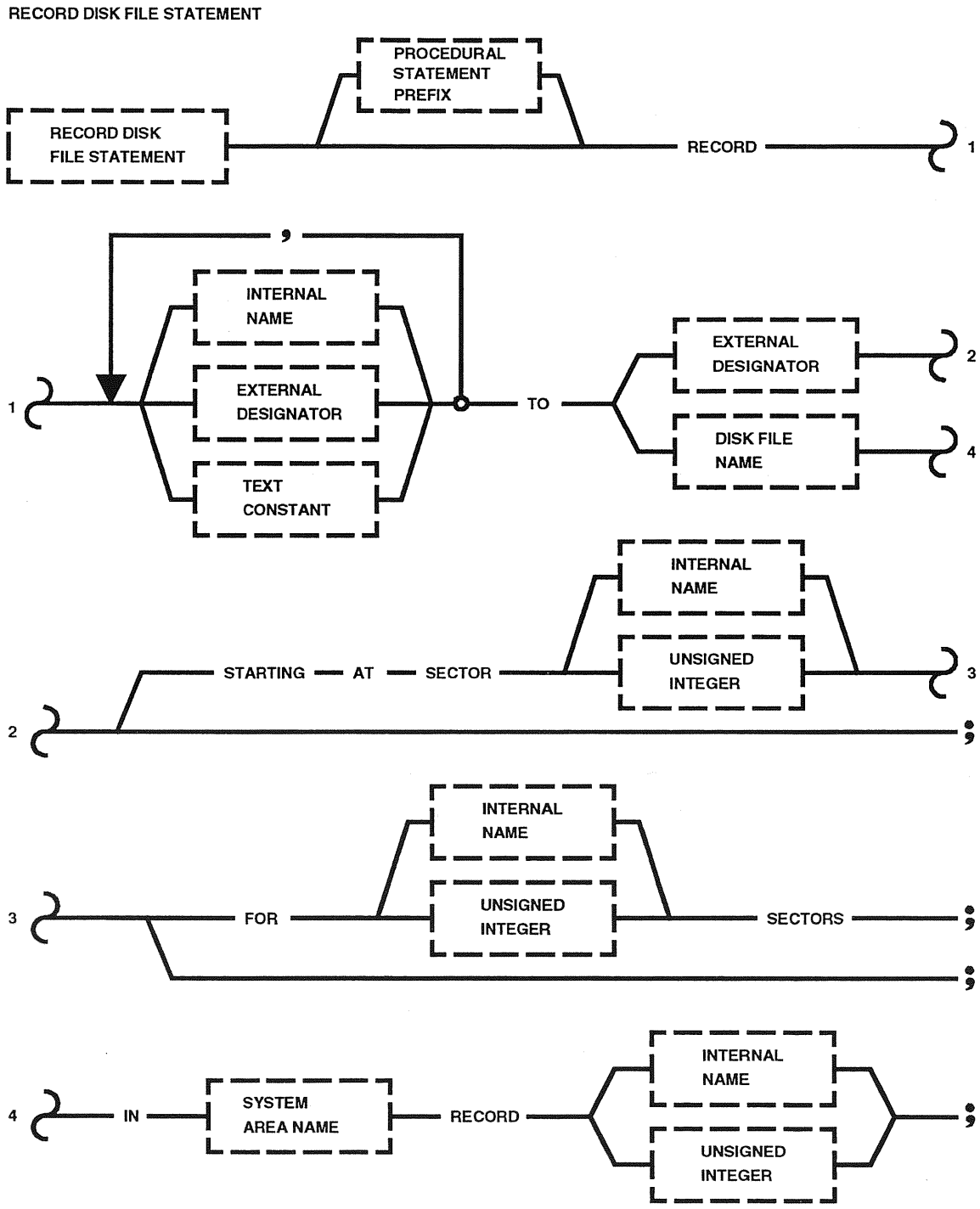


Figure 1-109 Record Disk File Statement

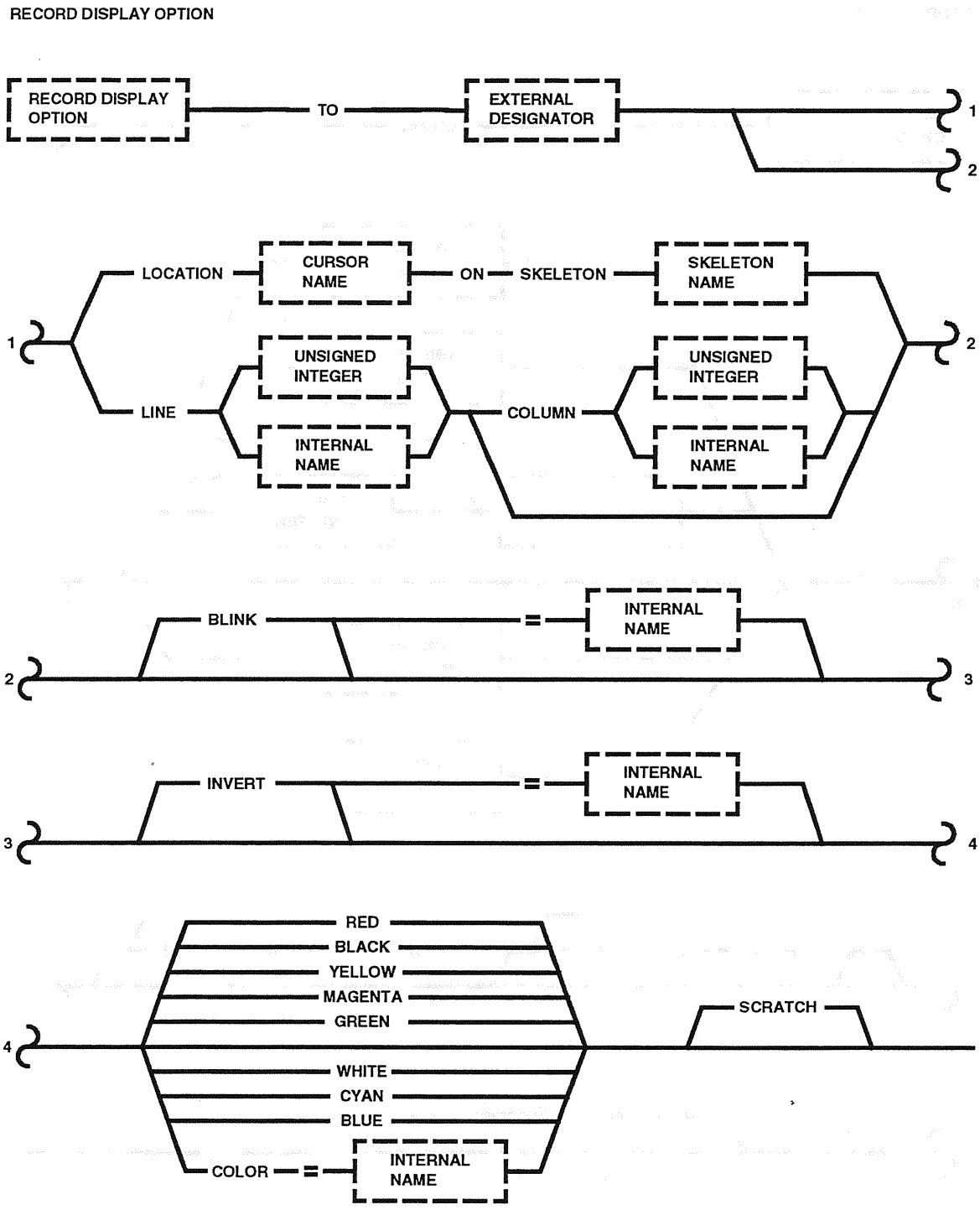


Figure 1-110 Record Display Option

RECORD FORMAT OPTION

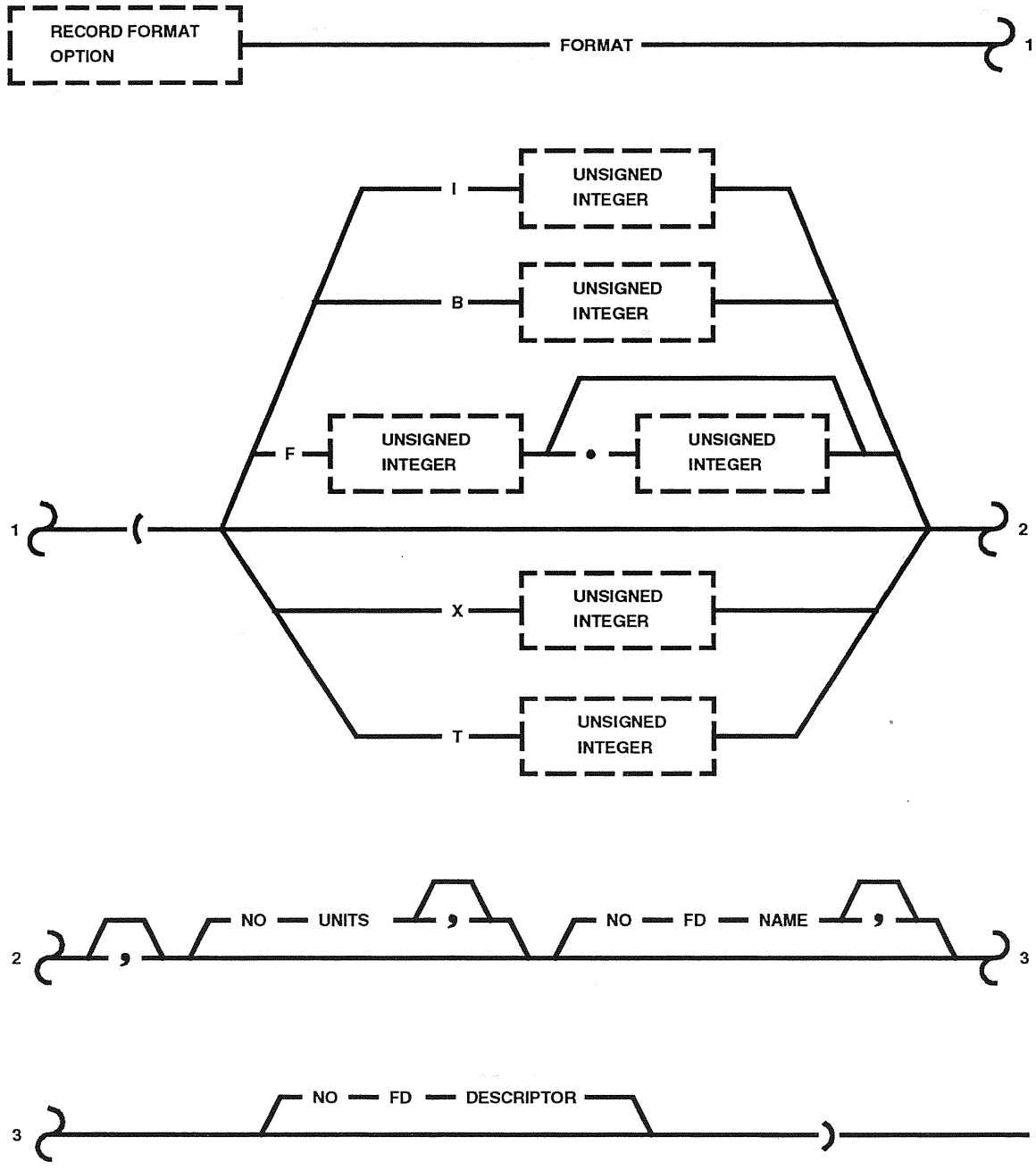


Figure 1-111 Record Format Option

RECORD PRINT OPTION

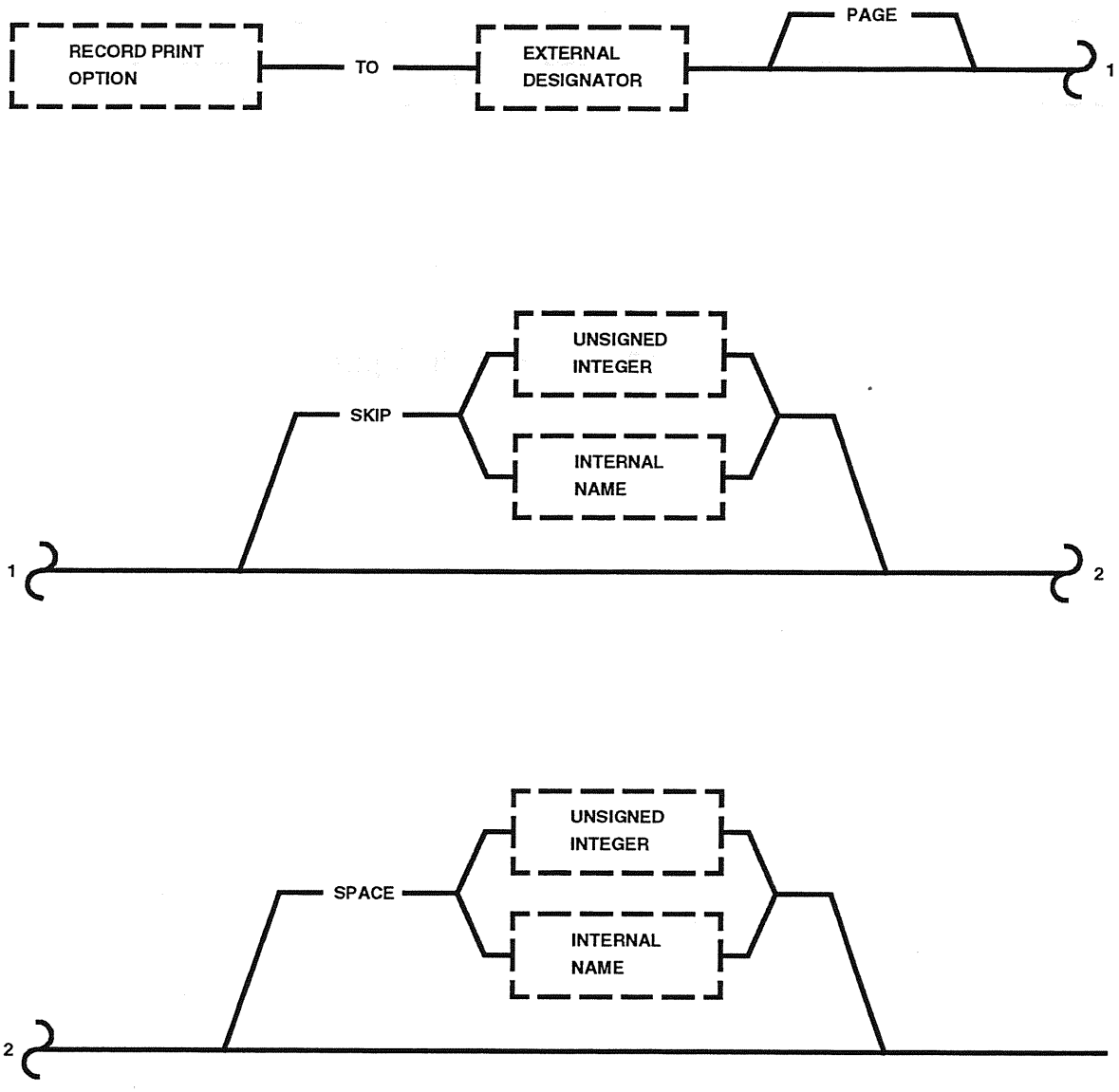


Figure 1-112 Record Print Option

RECORD WRITE OPTION

Figure 1-113 Record Write Option

RELATIONAL FORMULA

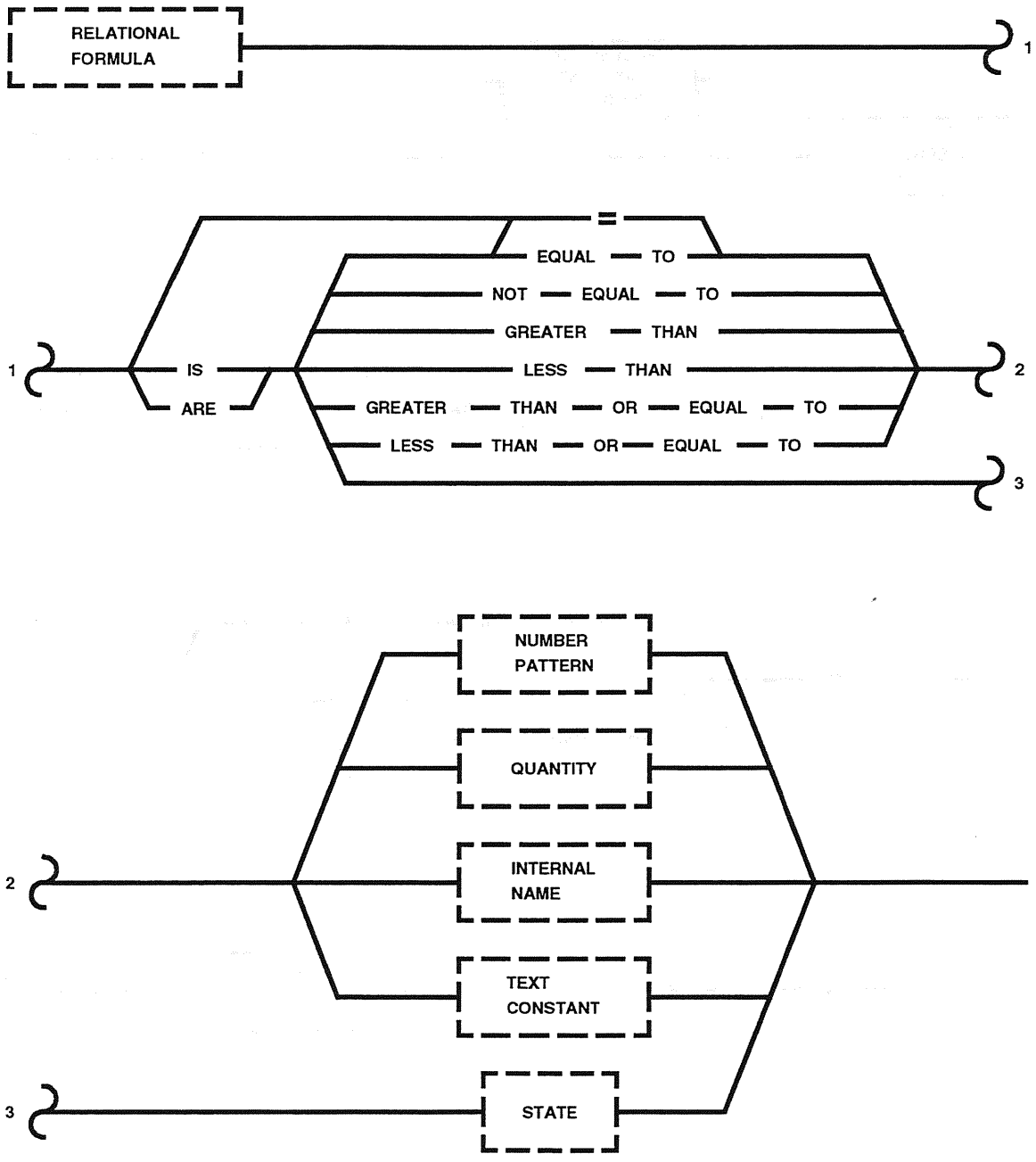


Figure 1-114 Relational Formula

RELAY INTERRUPT STATEMENT

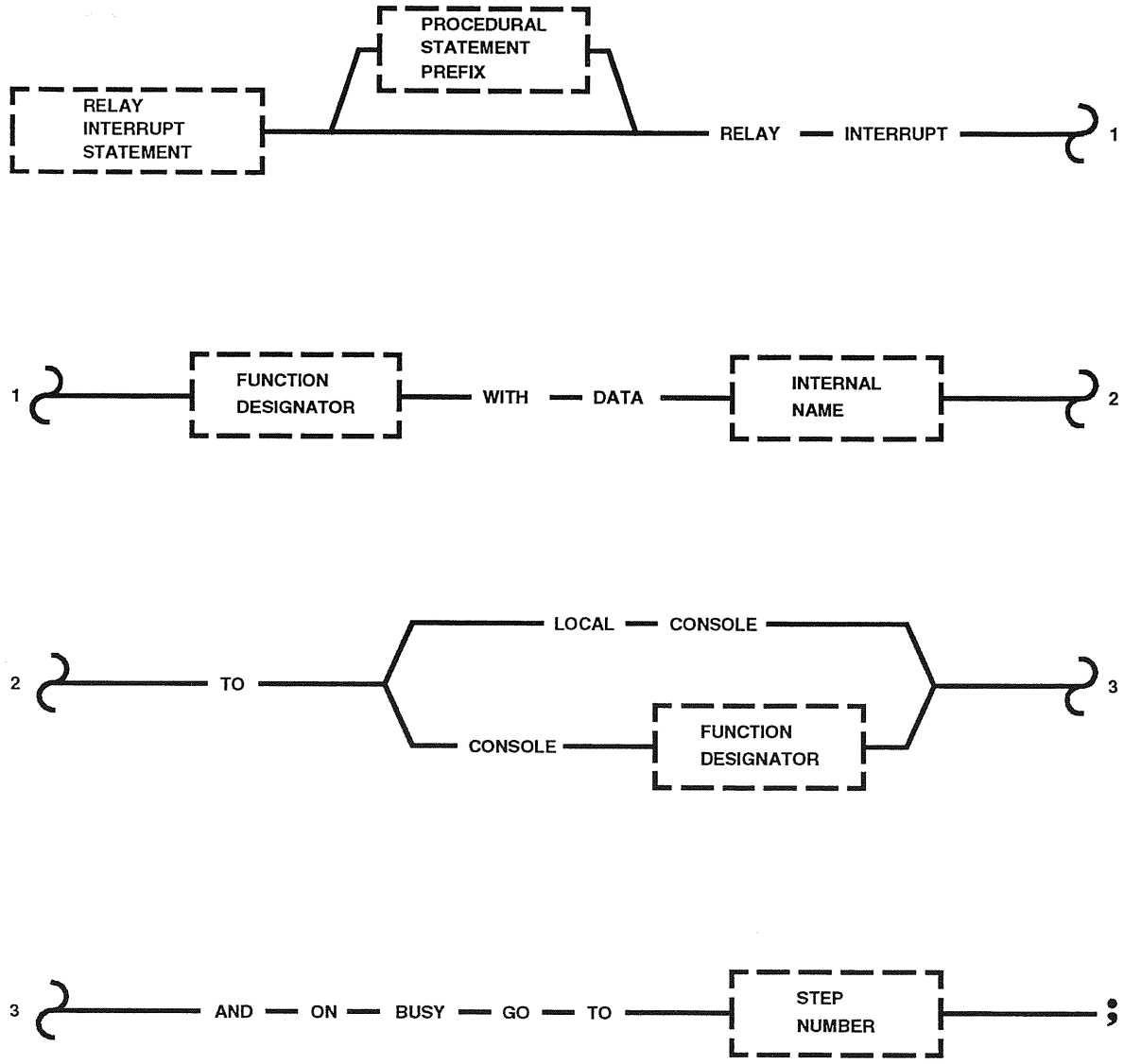
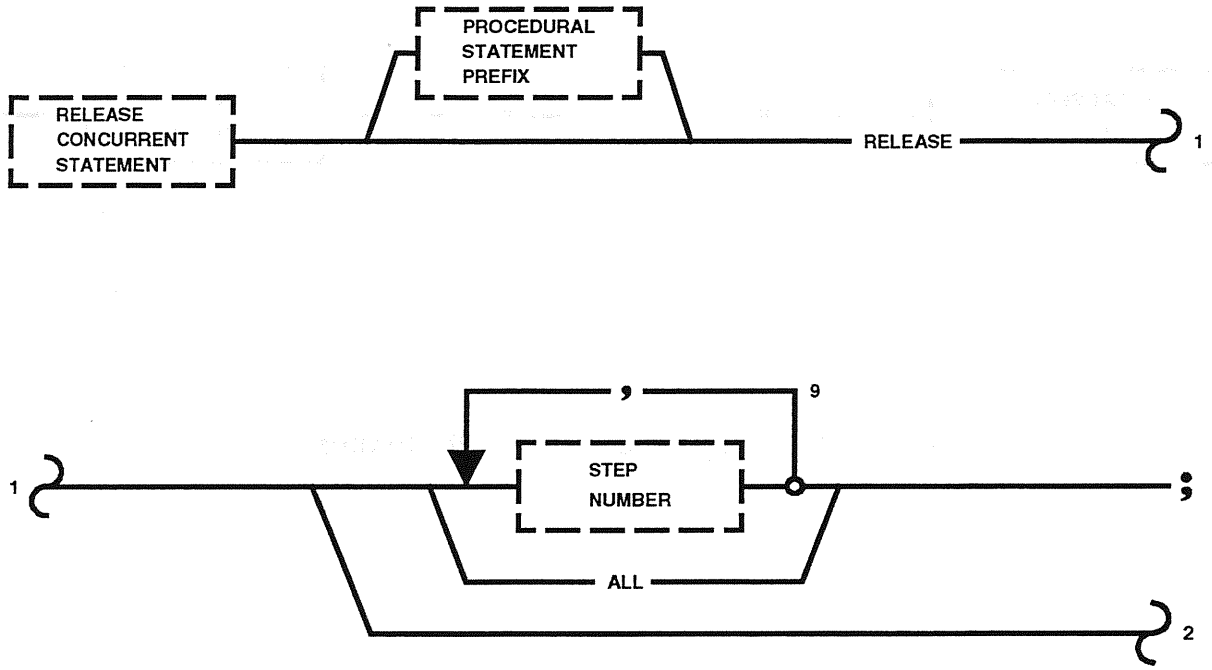


Figure 1-115 Relay Interrupt Statement

RELEASE CONCURRENT STATEMENT



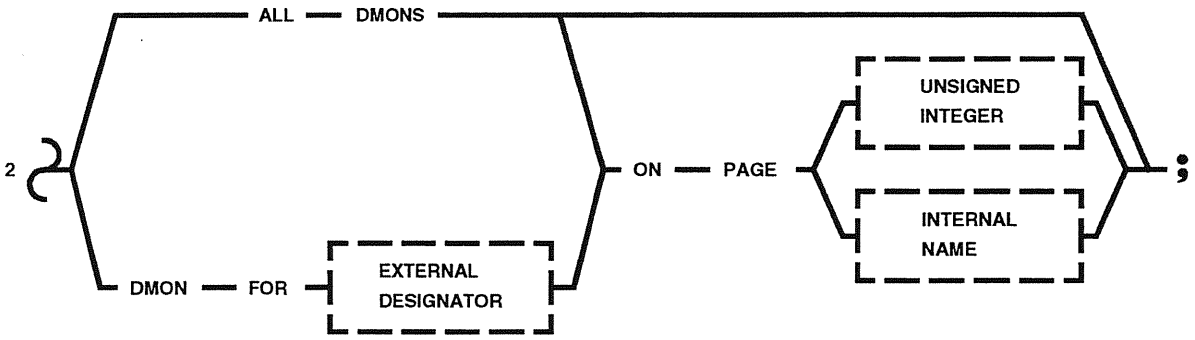


Figure 1-116 Release Concurrent Statement

REPEAT GROUP DEPTH COMMAND

Figure 1-117 Repeat Group Depth Command

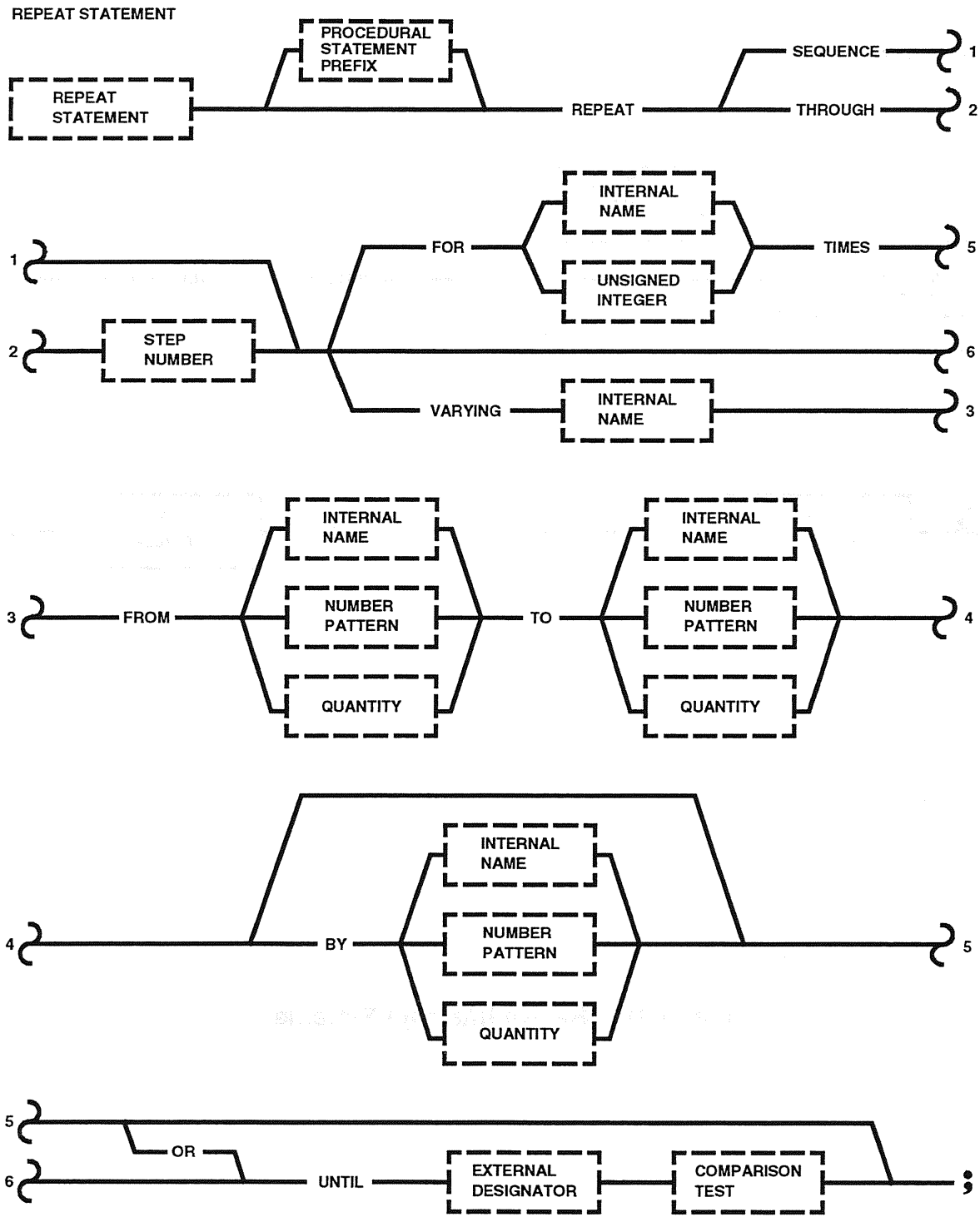


Figure 1-118 Repeat Statement

RETURN INTERRUPT STATEMENT

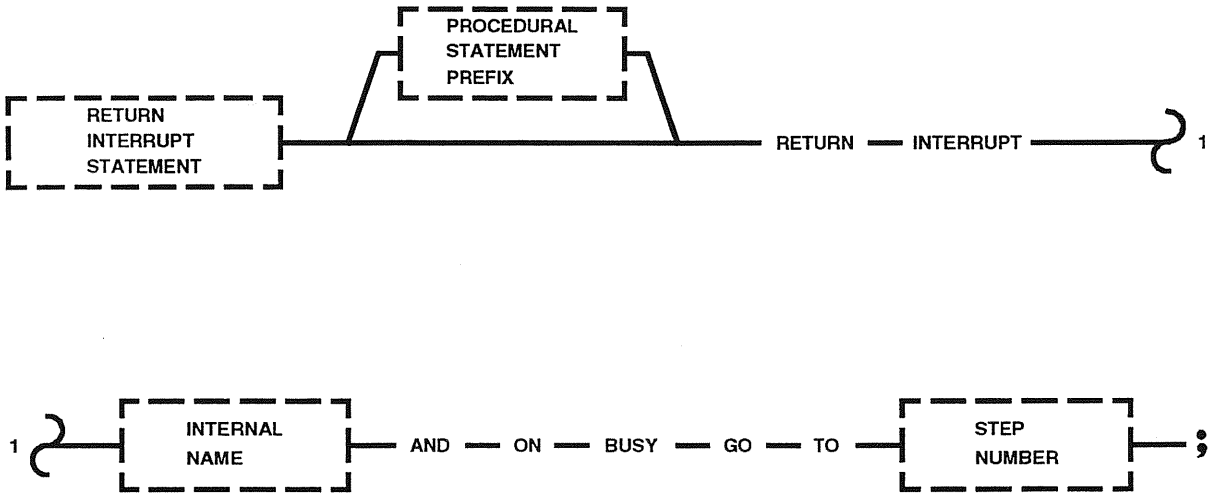


Figure 1-119 Return Interrupt Statement

ROW DESIGNATOR

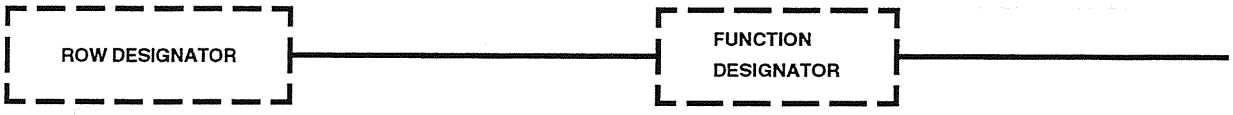


Figure 1-120 Row Designator

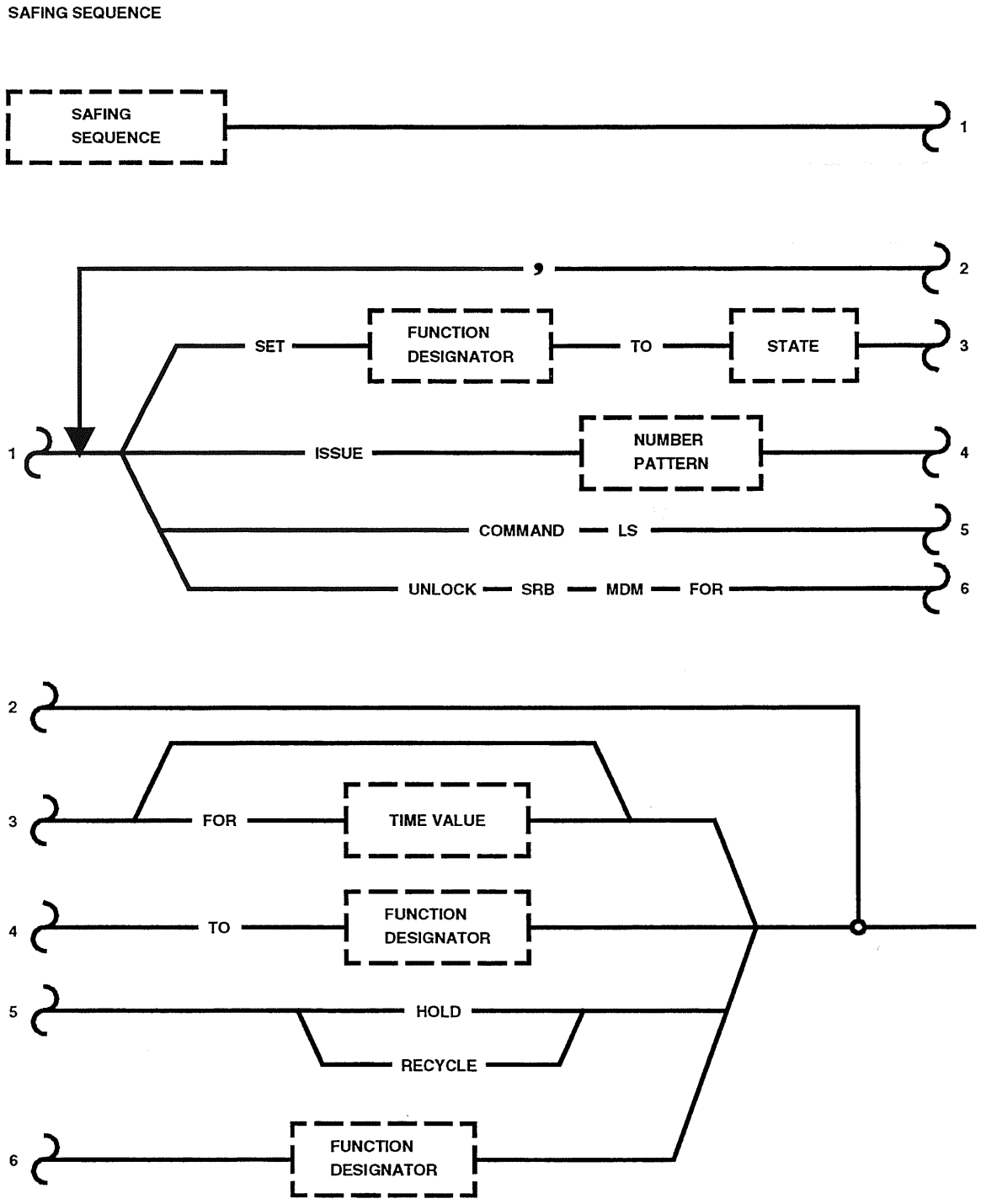


Figure 1-121 Safing Sequence

SEND INTERRUPT STATEMENT (1 OF 2)

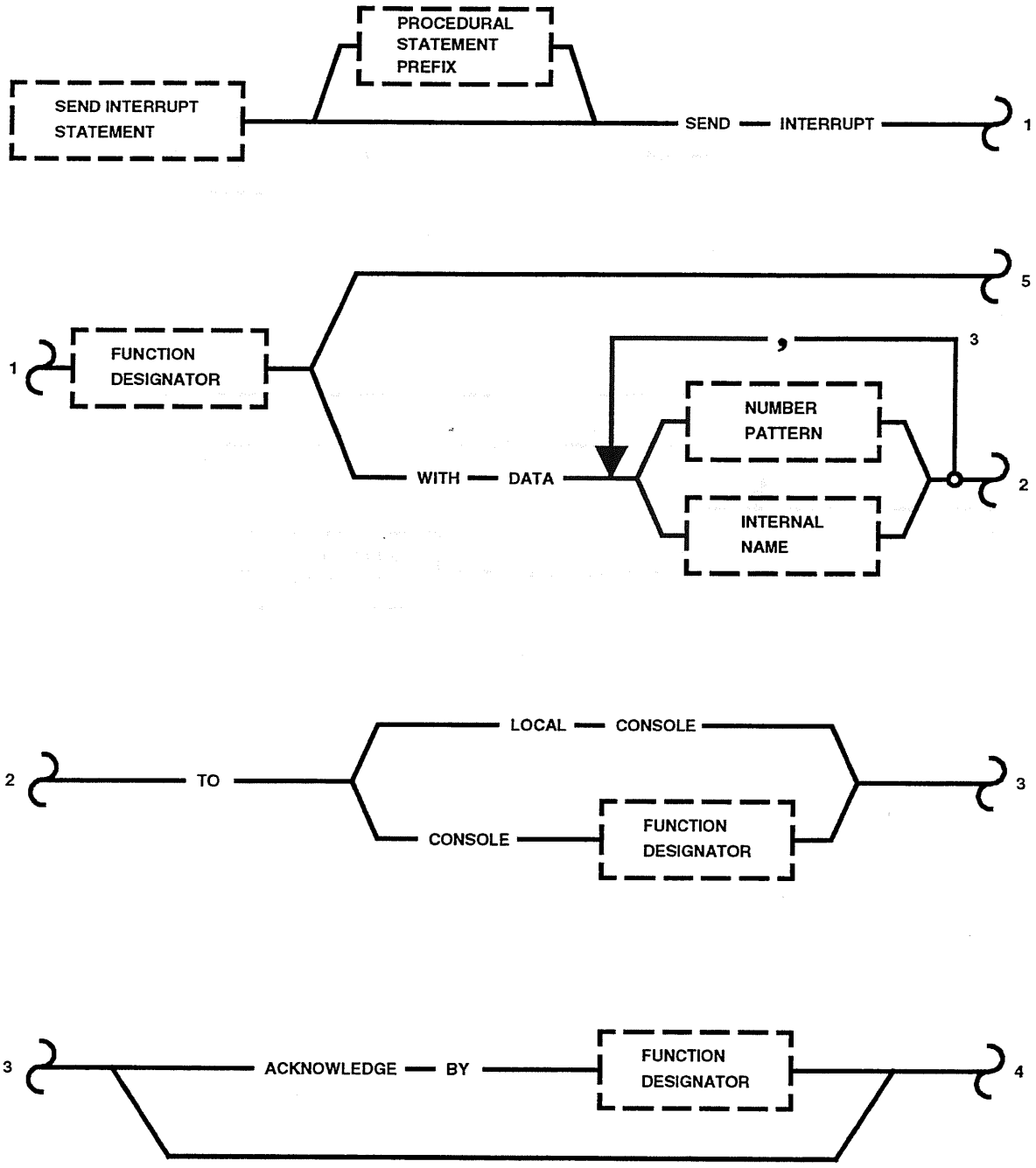


Figure 1-122 Send Interrupt Statement (1 of 2)

SEND INTERRUPT STATEMENT (2 OF 2)

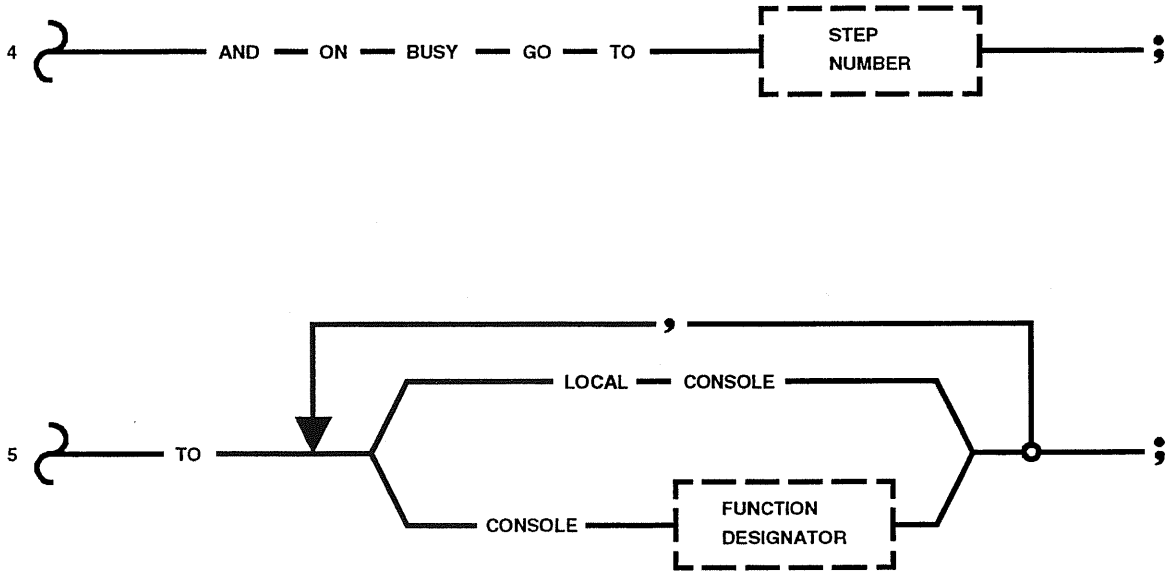


Figure 1-122 Send Interrupt Statement (2 of 2)

SEQUENCE COMMAND

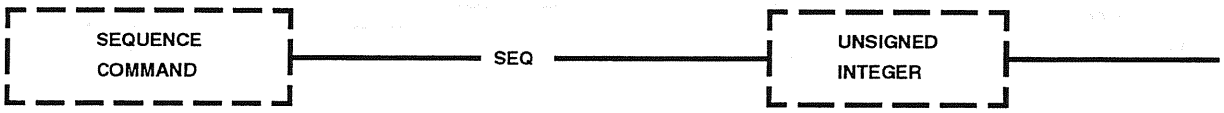


Figure 1-123 Sequence Command

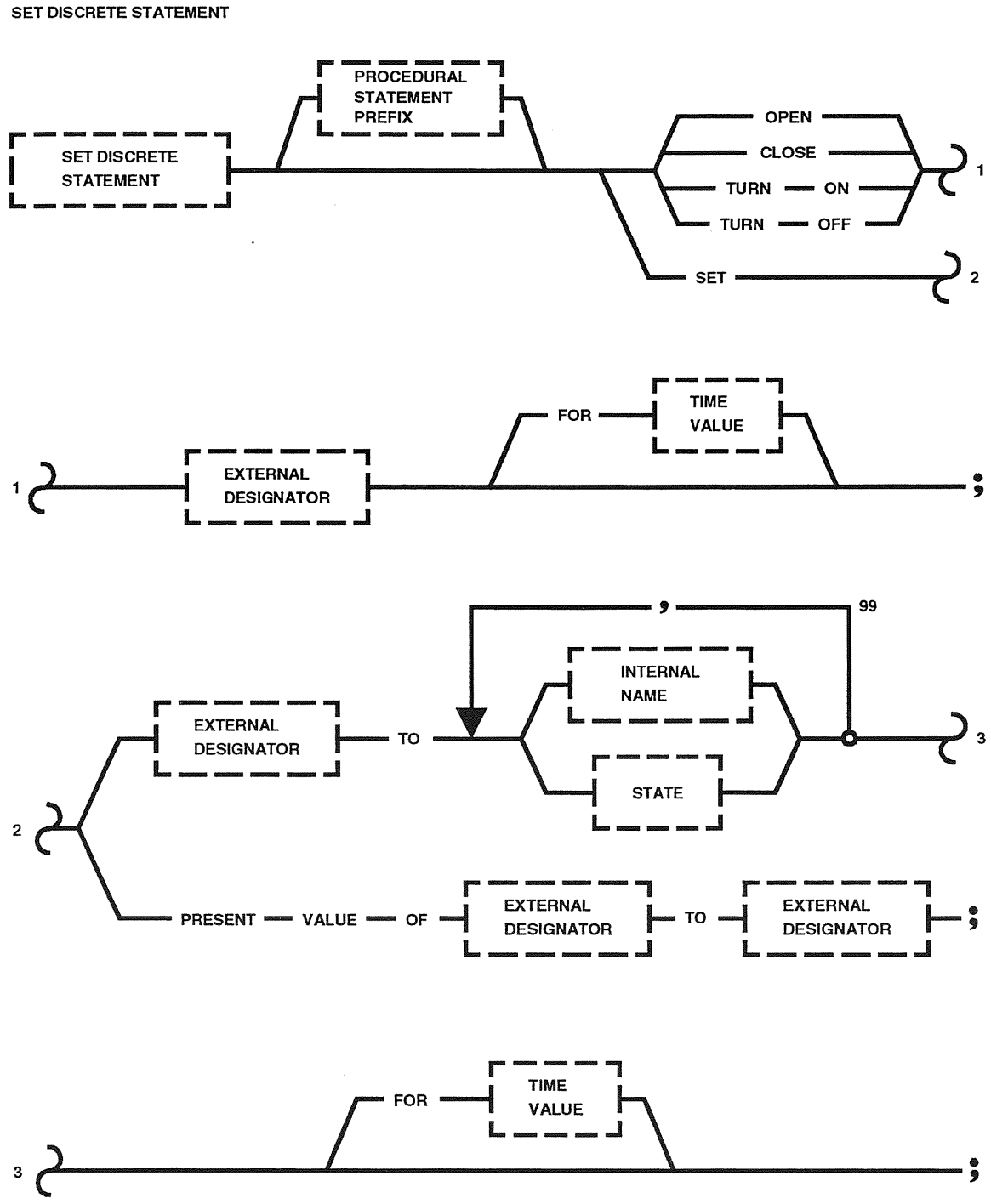


Figure 1-124 Set Discrete Statement

SKELETON NAME

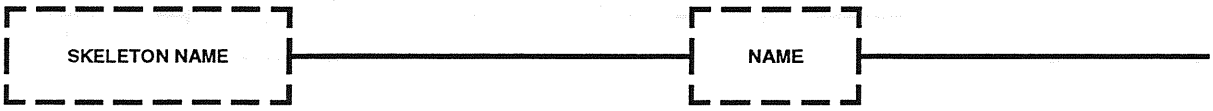


Figure 1-125 Skeleton Name

SPECIFY INTERRUPT STATEMENT

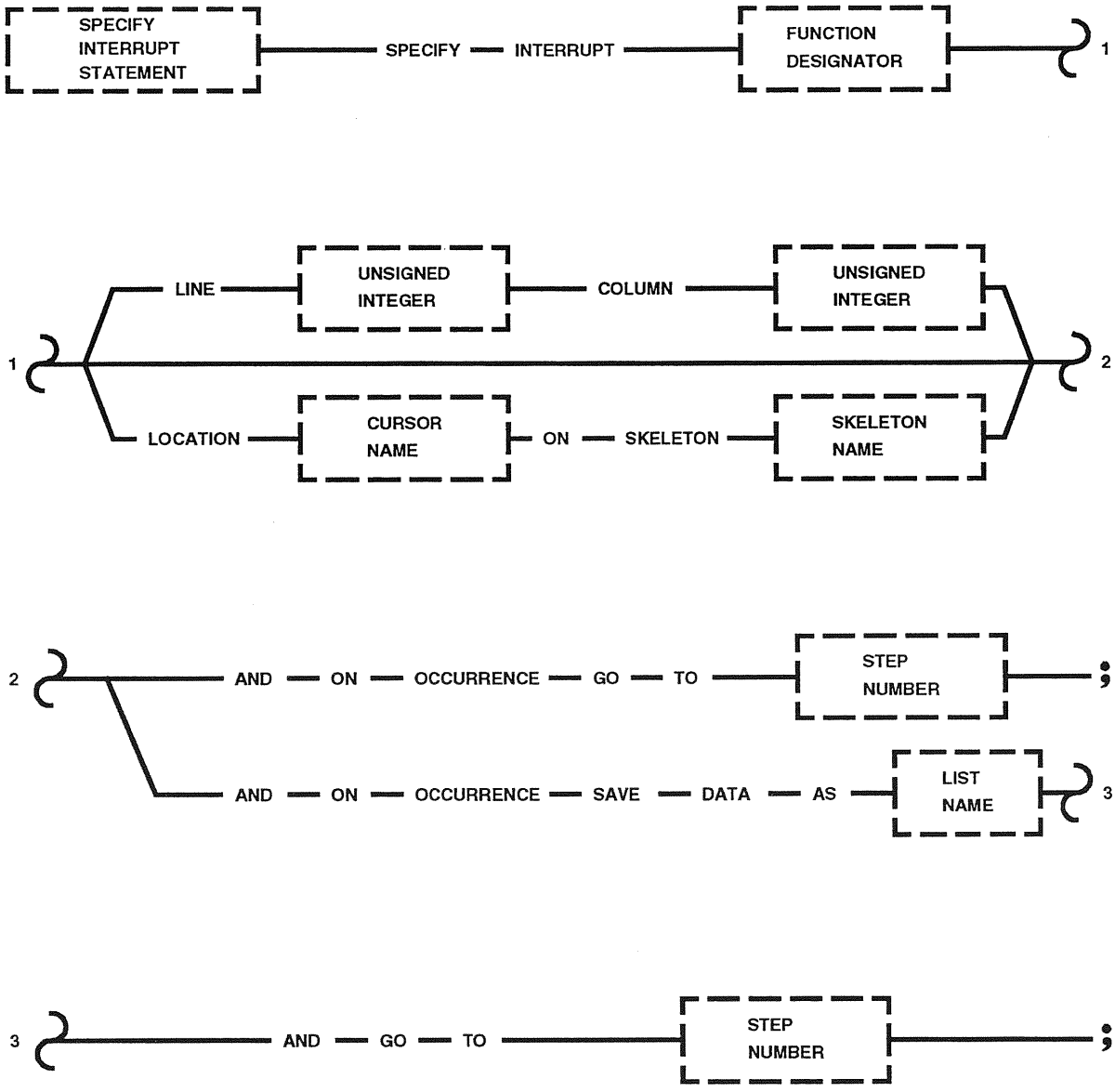


Figure 1-126 Specify Interrupt Statement

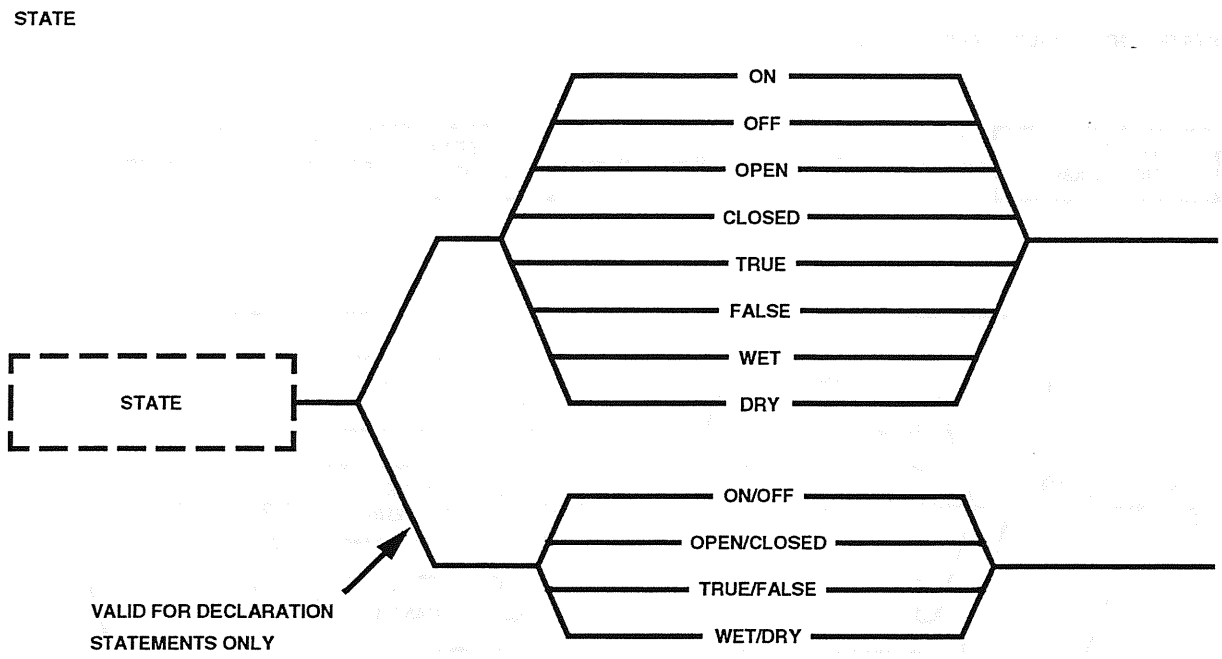


Figure 1-127 State

STATEMENT GROUP LABEL

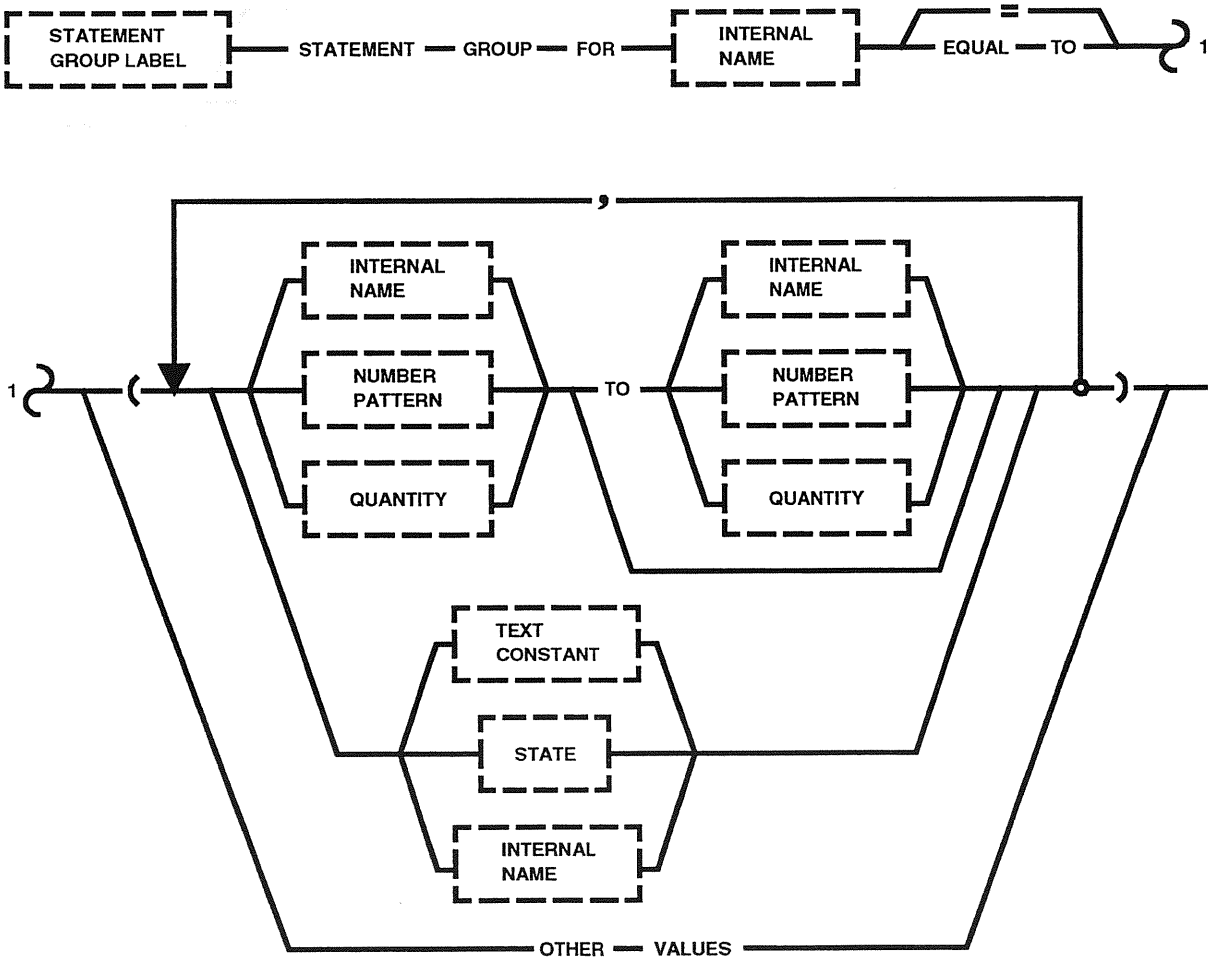


Figure 1-128 Statement Group Label

STEP NUMBER

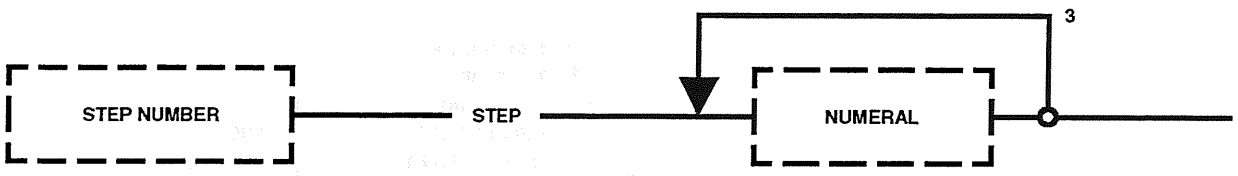


Figure 1-129 Step Number

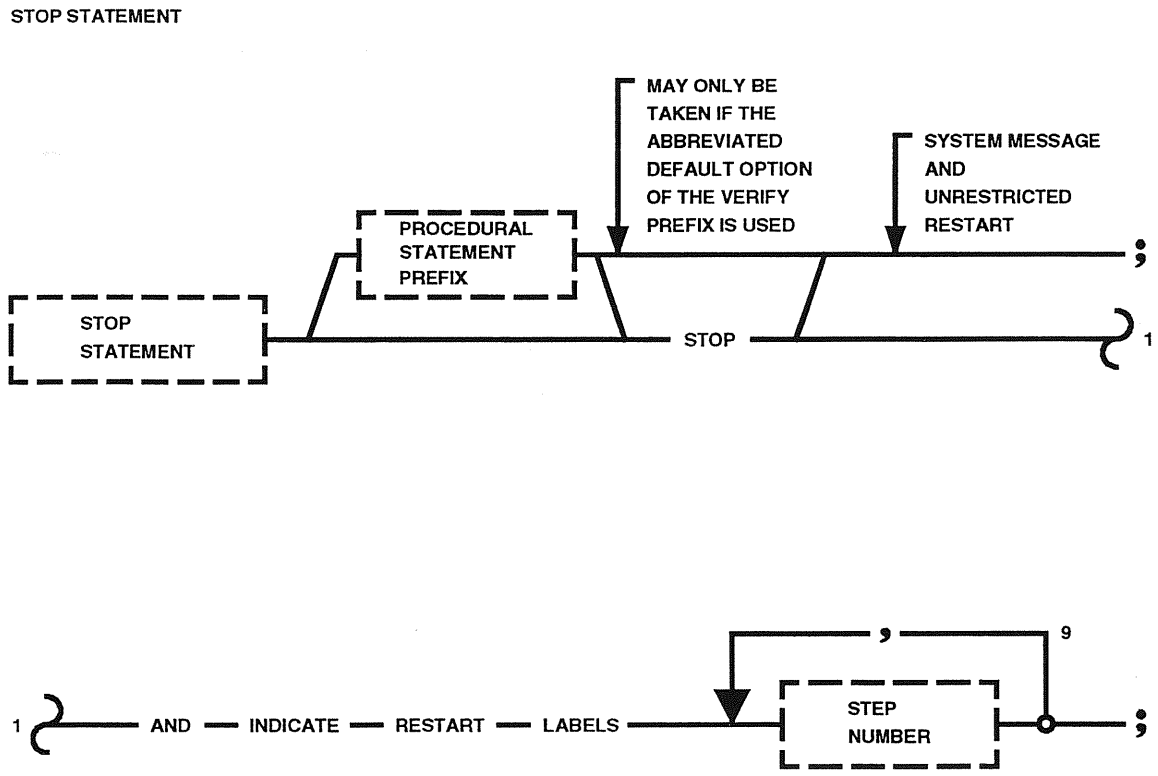


Figure 1-130 Stop Statement

SUBROUTINE NAME

Figure 1-131 Subroutine Name

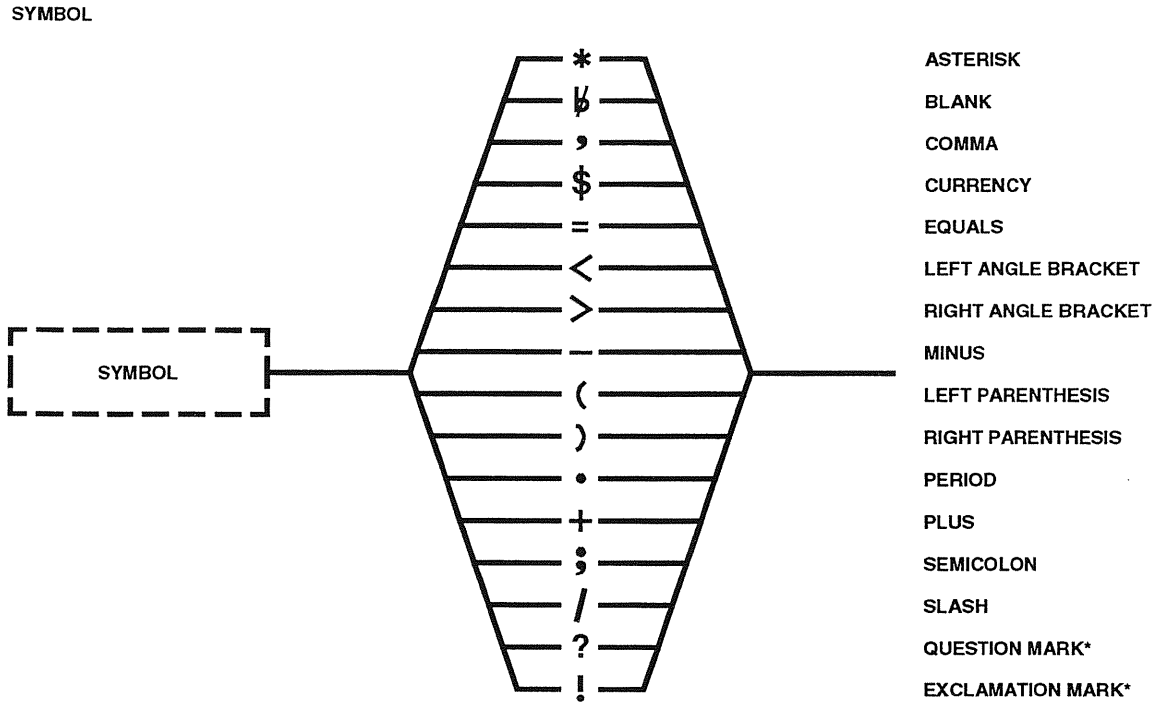


Figure 1-132 Symbol

SYSTEM AREA NAME

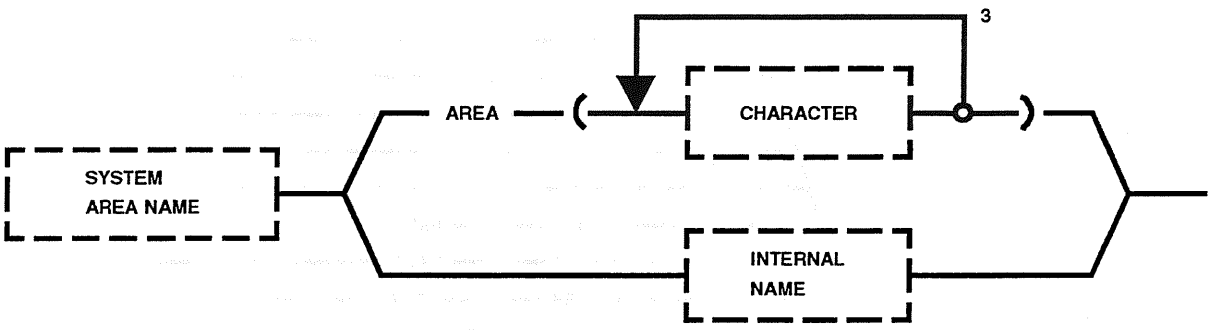


Figure 1-133 System Area Name

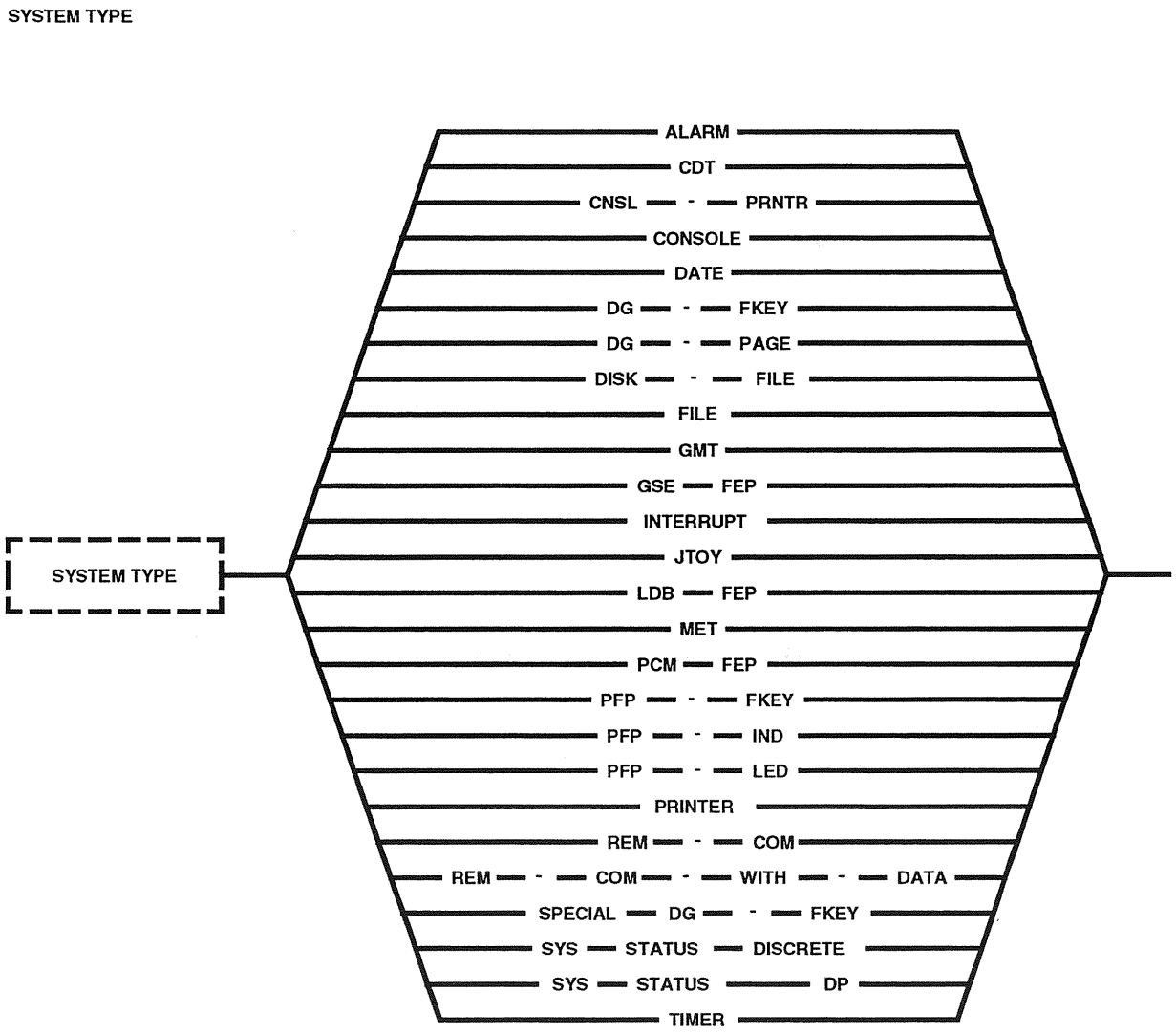


Figure 1-134 System Type

TABLE NAME

Figure 1-135 Table Name

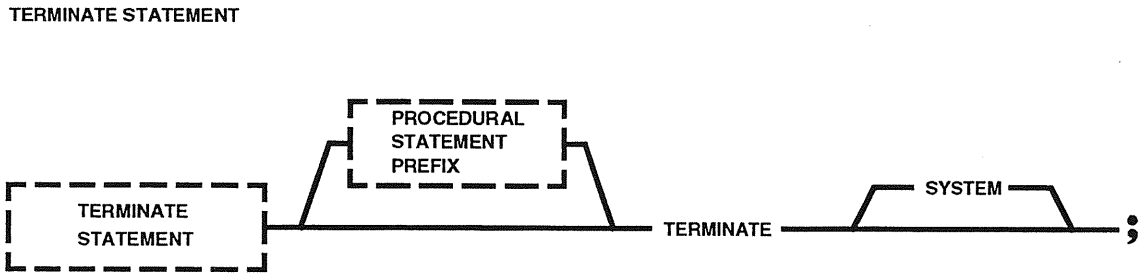


Figure 1-136 Terminate Statement

TERMINATE SUBROUTINE STATEMENT

Figure 1-137 Terminate Subroutine Statement

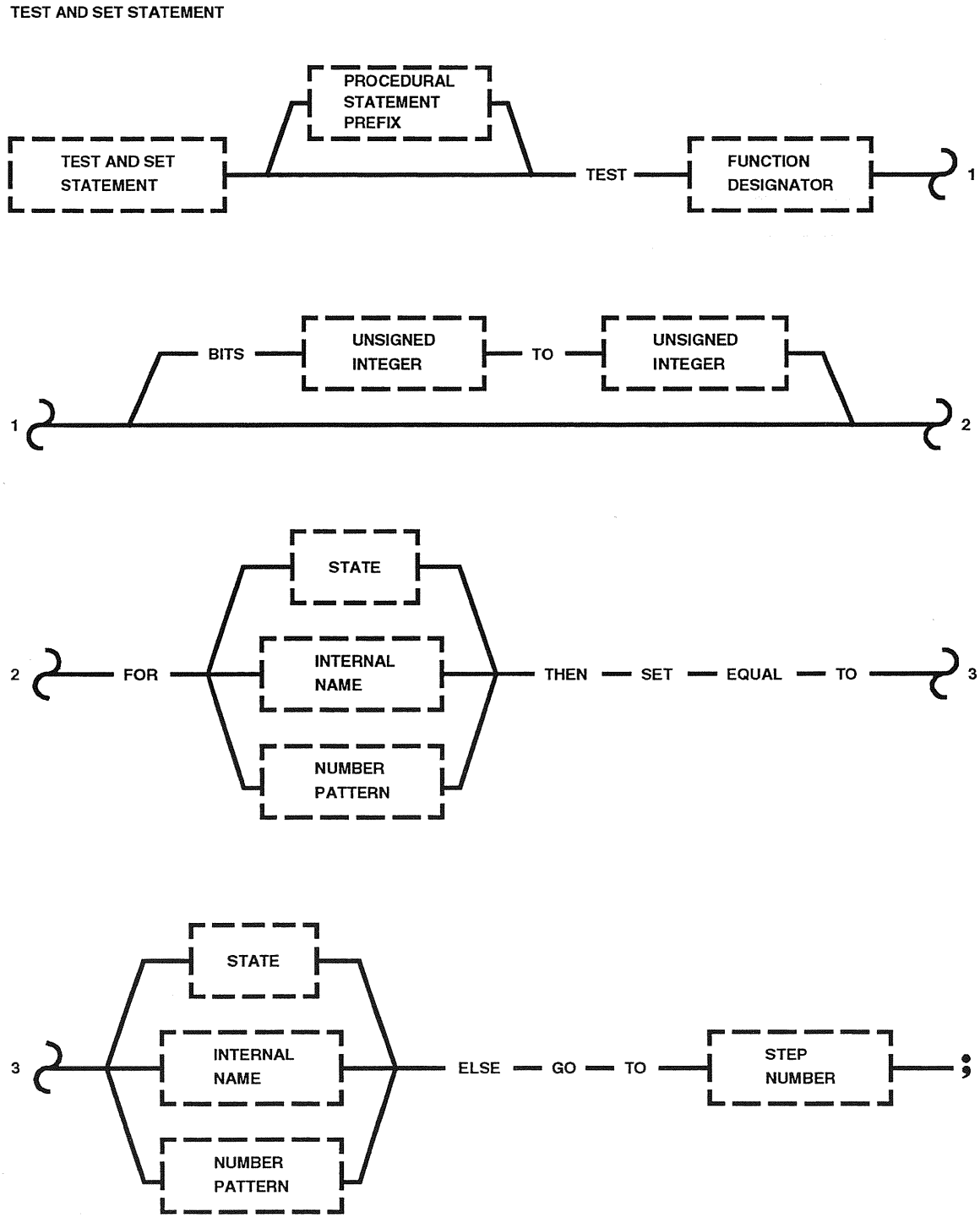


Figure 1-138 Test And Set Statement

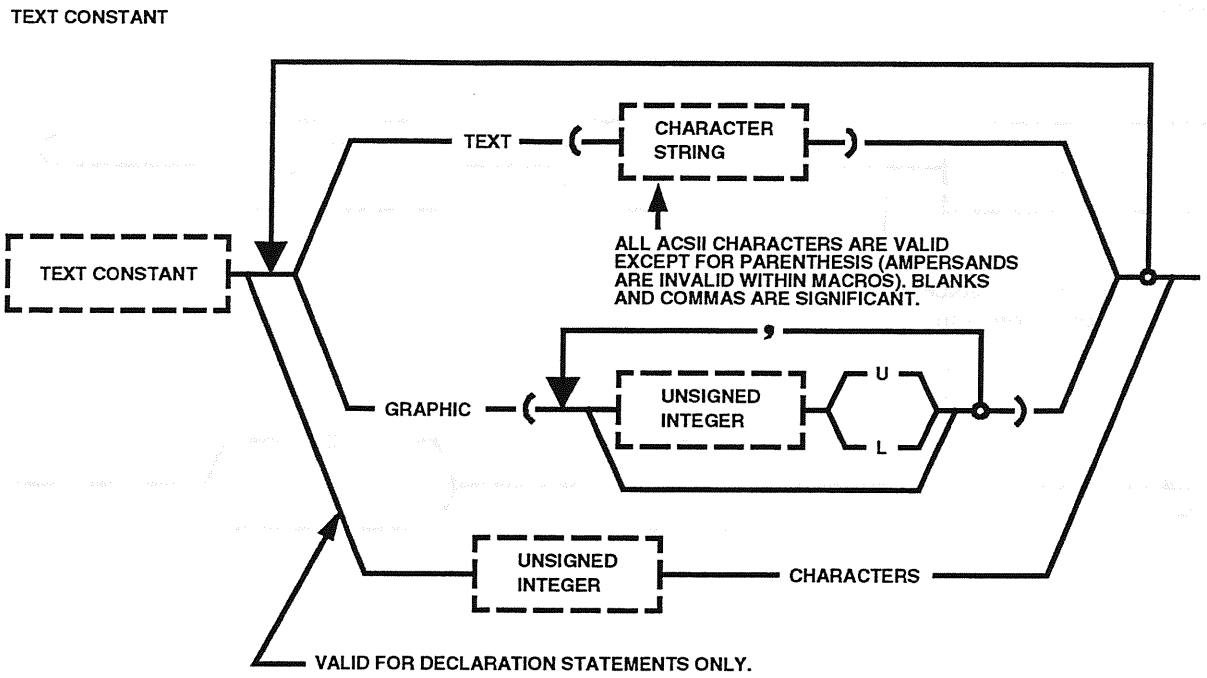


Figure 1-139 Text Constant

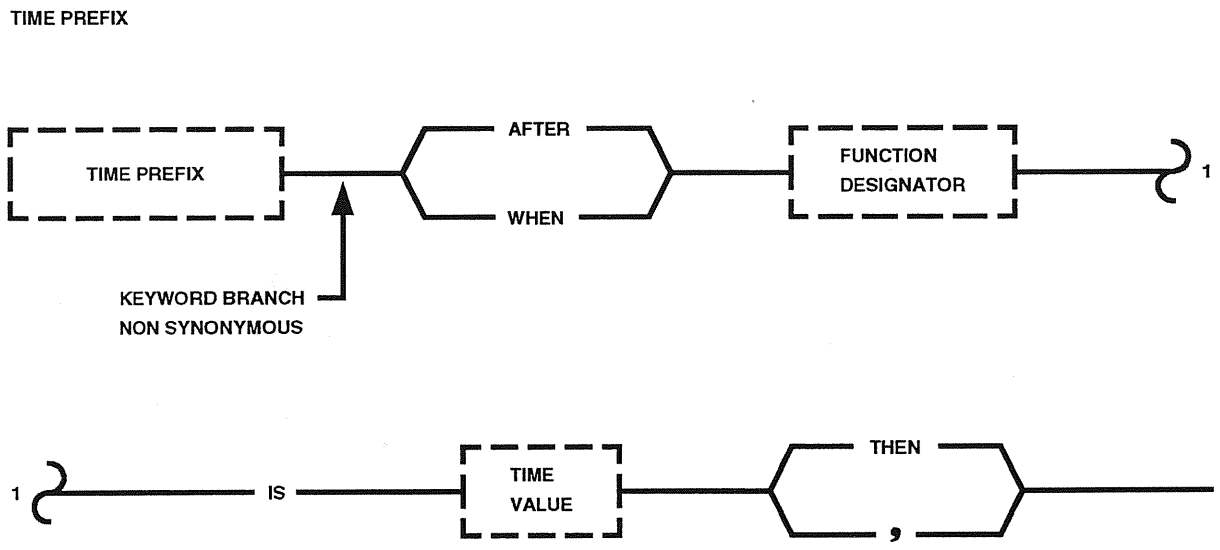


Figure 1-140 Time Prefix

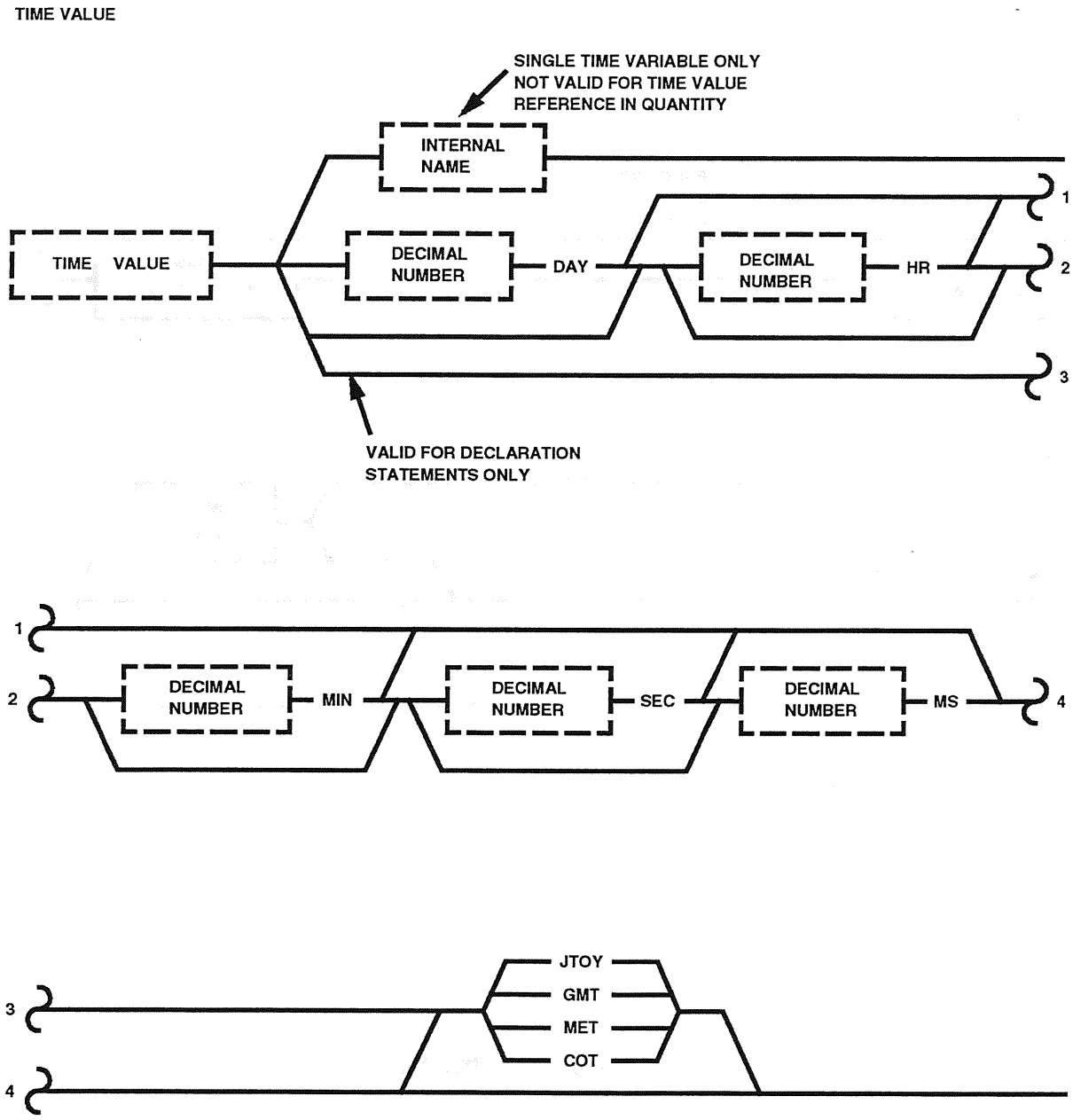


Figure 1-141 Time Value

TIMER CONTROL STATEMENT

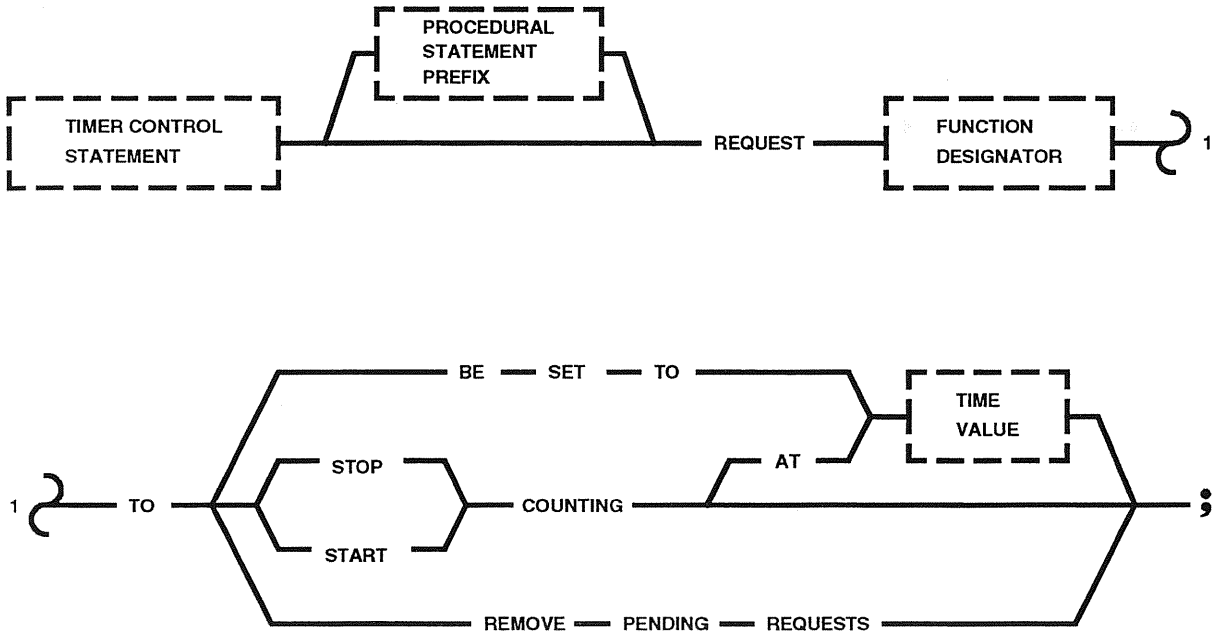


Figure 1-142 Timer Control Statement

TITLE COMMAND

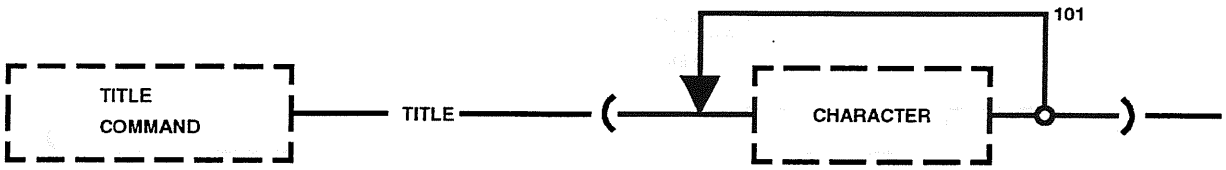


Figure 1-143 Title Command

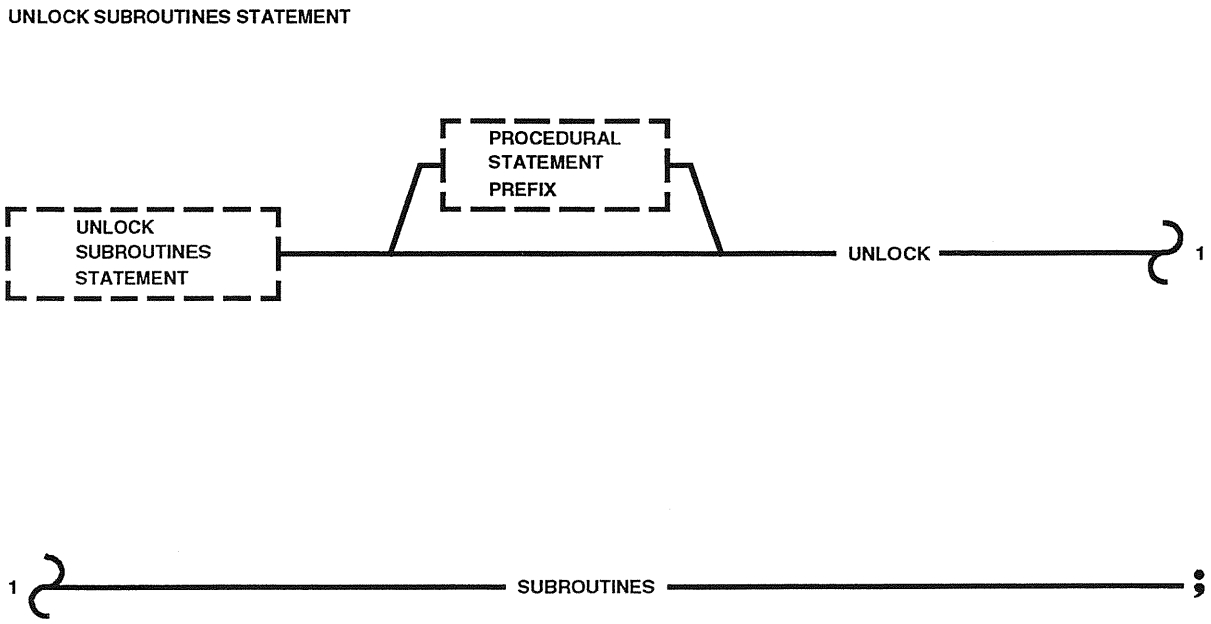


Figure 1-144 Unlock Subroutines Statement

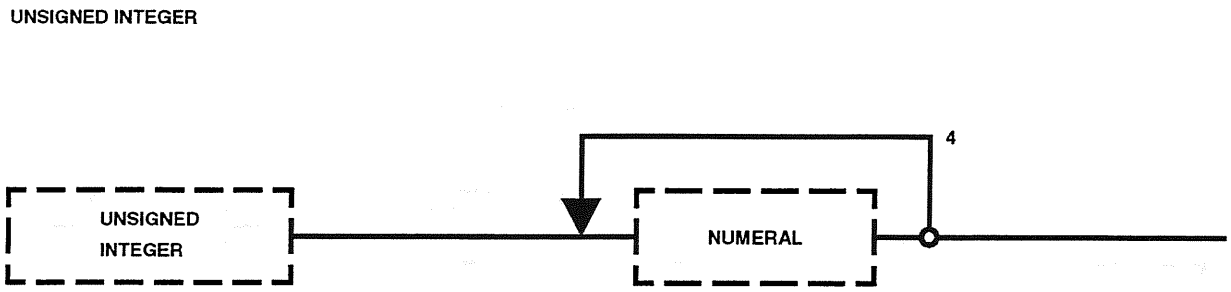


Figure 1-145 Unsigned Integer

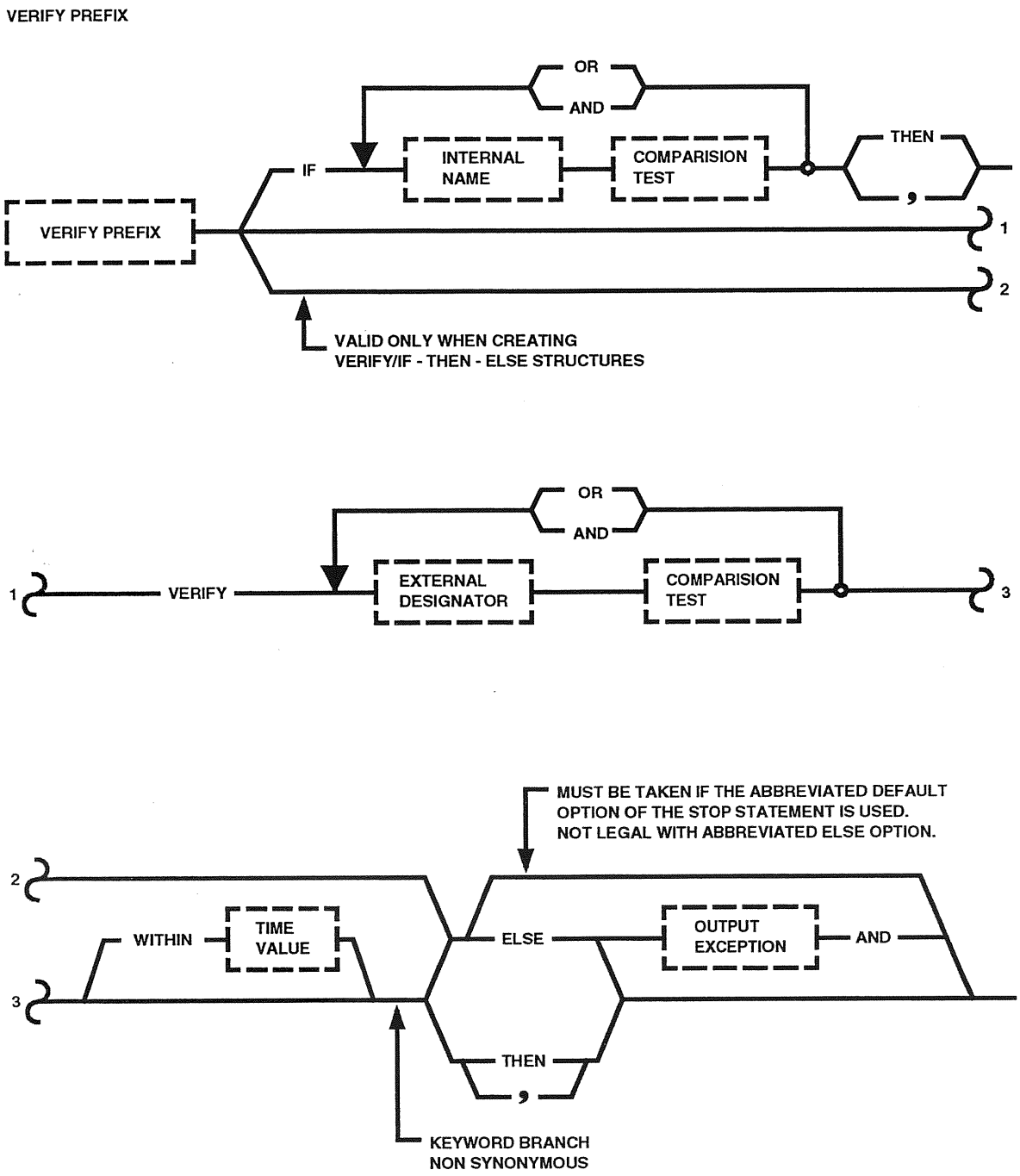


Figure 1-146 Verify Prefix

WHEN INTERRUPT STATEMENT

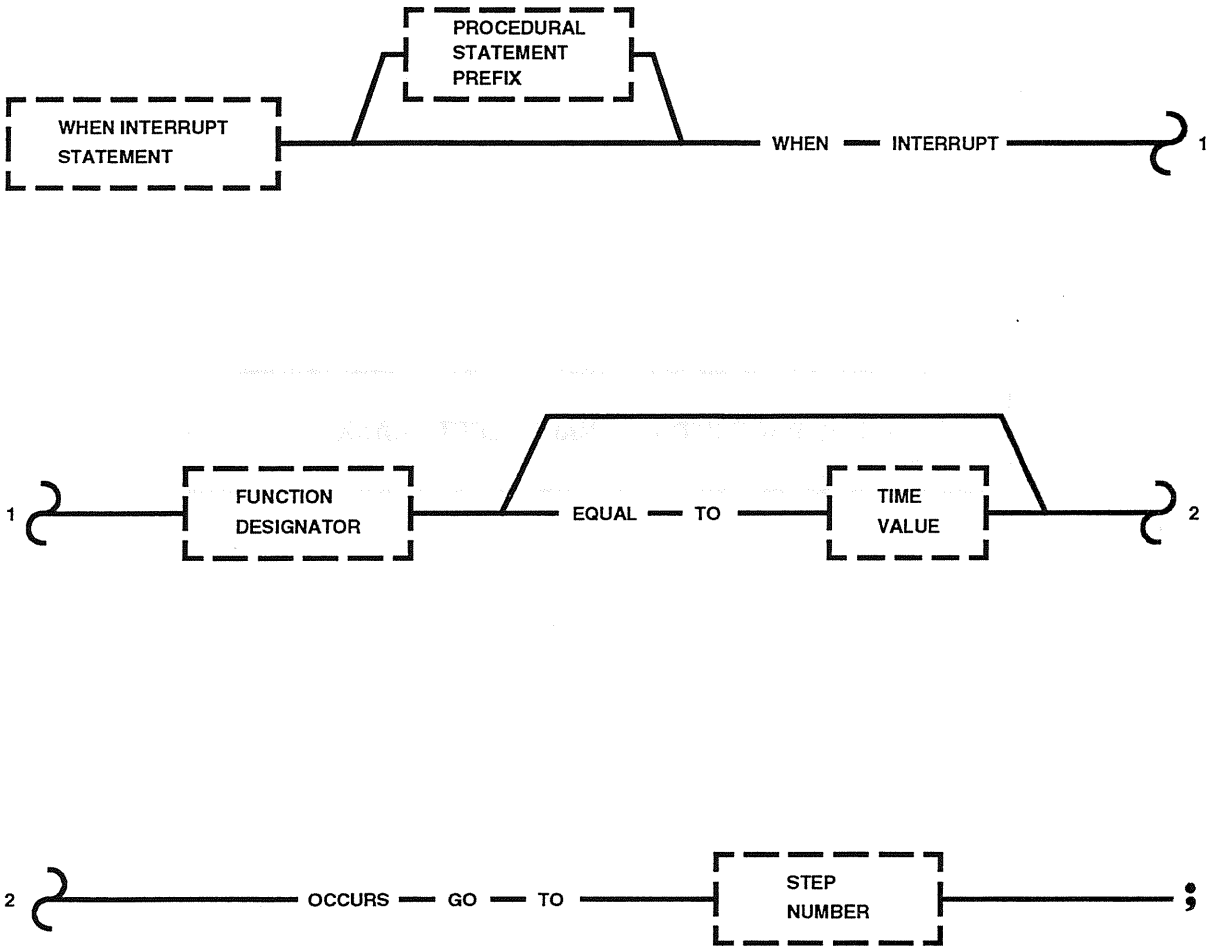


Figure 1-147 When Interrupt Statement

THIS PAGE INTENTIONALLY LEFT BLANK.