Ground Operations Aerospace Language

GOAL

Final Report

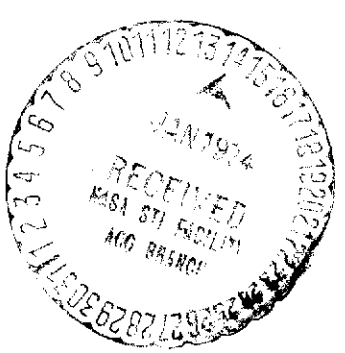
Volume II

Compiler

(NASA-CR-136780) GROUND OPERATIONS AEROSPACE LANGUAGE (GOAL). VOLUME 2: COMPILER Final Report (International Business Machines Corp.) ~~265~~ 264 p HC $15.25 CSCL 09B G3/08

N74-15888

Unclas 28966



31 July 1973

Ground Operations Aerospace Language

GOAL

Final Report

Volume II

Compiler

Contract NAS 10-6900

Approved by:
J. W. Handley, Manager
Systems Programming and
Advanced Programs

IBM

31 July 1973

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

1.0 INTRODUCTION

This volume identifies and describes the principal elements and functions of the GOAL Compiler. It is the result of implementation efforts based on specifications provided by NASA/KSC in publications, TR-1228, Ground Operations Aerospace Language (GOAL) Textbook, and TR-1213, Ground Operations Aerospace Language (GOAL) Syntax Diagrams Handbook, both dated 16 April 1973.

A general description of the system is presented in Section 2, Compiler Specifications. This Section provides an overview of the elements that comprise the GOAL System. It describes the technique used to transcribe the syntax diagrams into machine processable format for use by the parsing routines. An explanation of the parsing technique used to process GOAL source statements is also included. The Compiler diagnostics and the output reports generated during a GOAL compilation are explained.

A detailed description of the GOAL program package is presented in Section 3, Compiler Software Descriptions. This Section includes a write-up for each of the FORTRAN subprograms. In addition, the formats for the Intermediate Text and SYMTAB chain definitions are described. The "common" communication cells required are identified by name, location, and usage.

Appendices are included to provide the user with the cataloged procedures used to generate/maintain the GOAL System on the IBM S/360-40 computer system. In addition, listings of the Syntax Table, FORTRAN programs, and Auto Flow charts are provided. The diagnostic messages which may be encountered during the compilation of a GOAL Procedure are listed by message number.

2.0 COMPILER SPECIFICATIONS

The principal functions of the GOAL Language compiler are:

- o To accept a GOAL program as input on punched cards.
- o To parse GOAL statements according to syntax diagrams.
- o To generate diagnostic messages for statement errors.
- o To generate Intermediate GOAL data.

To support these functions the following principal software elements are provided:

- o Syntax specifications (equations) for GOAL Language.
- o Syntax processor.
- o Compiler input processor.
- o Parsing routines.
- o Diagnostics routine.
- o Output report generator.
- o Intermediate GOAL data generator.

The relationship between these items is shown in Figure 2-1. They are described in further detail in the following sections.

2.1 SYNTAX SPECIFICATIONS

The GOAL Language is specified by a set of syntax diagrams which define all variations of GOAL statements. These diagrams are transcribed into a modified Backus Naur Form. The syntax of this notation is described, using syntax diagram format, in Figure 2-2.

Modified BNF statements are processed by the GOAL SYNTAX GENERATION PROGRAM to generate syntax tables which are subsequently used by the GOAL compiler to parse GOAL language statements.

The set of statements describing the GOAL syntax are arranged as a group. The order is not important, however, the last statement must be 'END'.

Each statement may use up to twenty cards. The end of statement is indicated by the character, ';'.

Each statement, (except 'END'), defines a syntactical element of the GOAL LANGUAGE. This element may be referenced by other statements. The symbolic

name of the syntactical element being defined appears on the left of the character, '=', in the Modified BNF statement. Each element including the 'ROOT' element must be defined ONCE in the input group. The 'ROOT' element is the top of the 'syntax tree'. All valid GOAL statement variations can be derived starting with the 'ROOT' element.

Each definition statement is in either SEQUENTIAL or ALTERNATE form. The sequential form indicates that all items specified on the right of the character '=' must be processed or identified to satisfy the definition of the syntactical element associated with that statement. The alternate form indicates that any item identified on the right of the character, '=', will satisfy the definition. In both cases comparison proceeds from left to right. Items in the alternate form are separated by the character, '|'. Sequential and alternate forms cannot be mixed in the same statement.

Sequential or alternate items may be of the following types:

1. REFERENCE - In this case the item is the name of a syntactical element. Its definition must be satisfied for identification of the item.

2. FIXED TEXT - This item is a quoted string of characters. The string must be found, in the GOAL statement being parsed, in the indicated order for identification of the item.

3. ACTION CALL - This item identifies an action routine to be performed at this point in the parse. Examples would be: Flag statement type, verify labels/symbols, build lists, write out intermediate data.

4. ERROR CHECK - The parsing algorithm will normally 'back up' to try alternatives if a definition is not satisfied. If an error check is processed the parse cannot 'back up' past it. The numbered error message is given in this event.

The items, REFERENCE and FIXED TEXT, may be repeated zero to n times in a GOAL language statement.

Zero repetition would indicate an optional construction. N repetition is used to indicate 'feed back' loops.

Replication is indicated, in Modified BNF, by special characters immediately following the item to which they apply.

Three types of replication are provided:

SPECIAL CHARACTER	DEFINITION
?	item may be absent or used once.
*N	item may be absent or used N times.
+N	item must be used 1 to N times.

N may be a three digit number. If it is exceeded an error message is given. If it is Zero or not specified no limit is used, (N is indefinite).

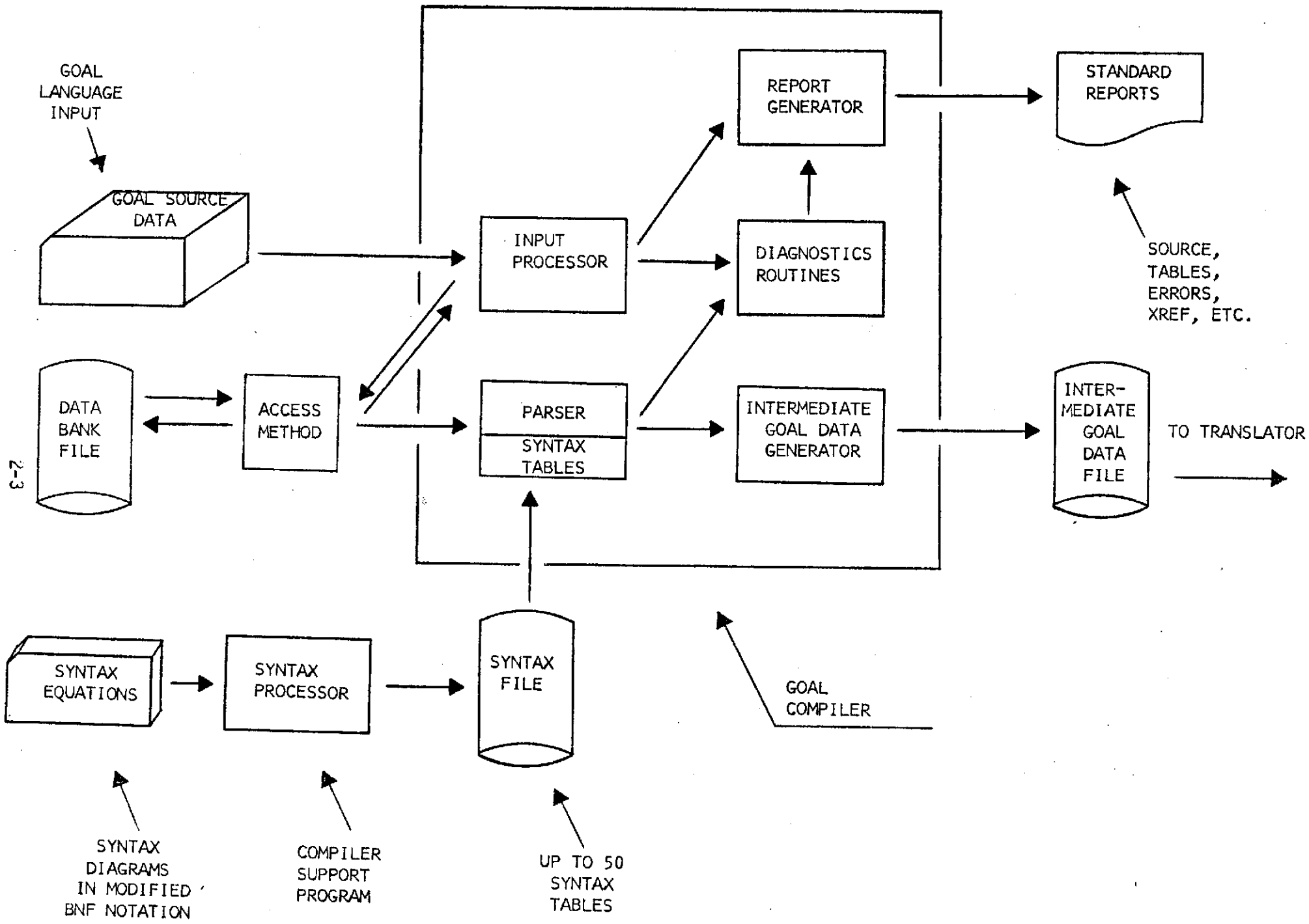

GOAL LANGUAGE INPUT
GOAL SOURCE DATA
INPUT PROCESSOR
REPORT GENERATOR
STANDARD REPORTS
DIAGNOSTICS ROUTINES
SOURCE, TABLES, ERRORS, XREF, ETC.
DATA BANK FILE
ACCESS METHOD
PARSER
SYNTAX TABLES
INTERMEDIATE GOAL DATA GENERATOR
INTER-MEDIATE GOAL DATA FILE
TO TRANSLATOR
SYNTAX EQUATIONS
SYNTAX PROCESSOR
SYNTAX FILE
GOAL COMPILER
SYNTAX DIAGRAMS IN MODIFIED BNF NOTATION
COMPILER SUPPORT PROGRAM
UP TO 50 SYNTAX TABLES

Figure 2-1.

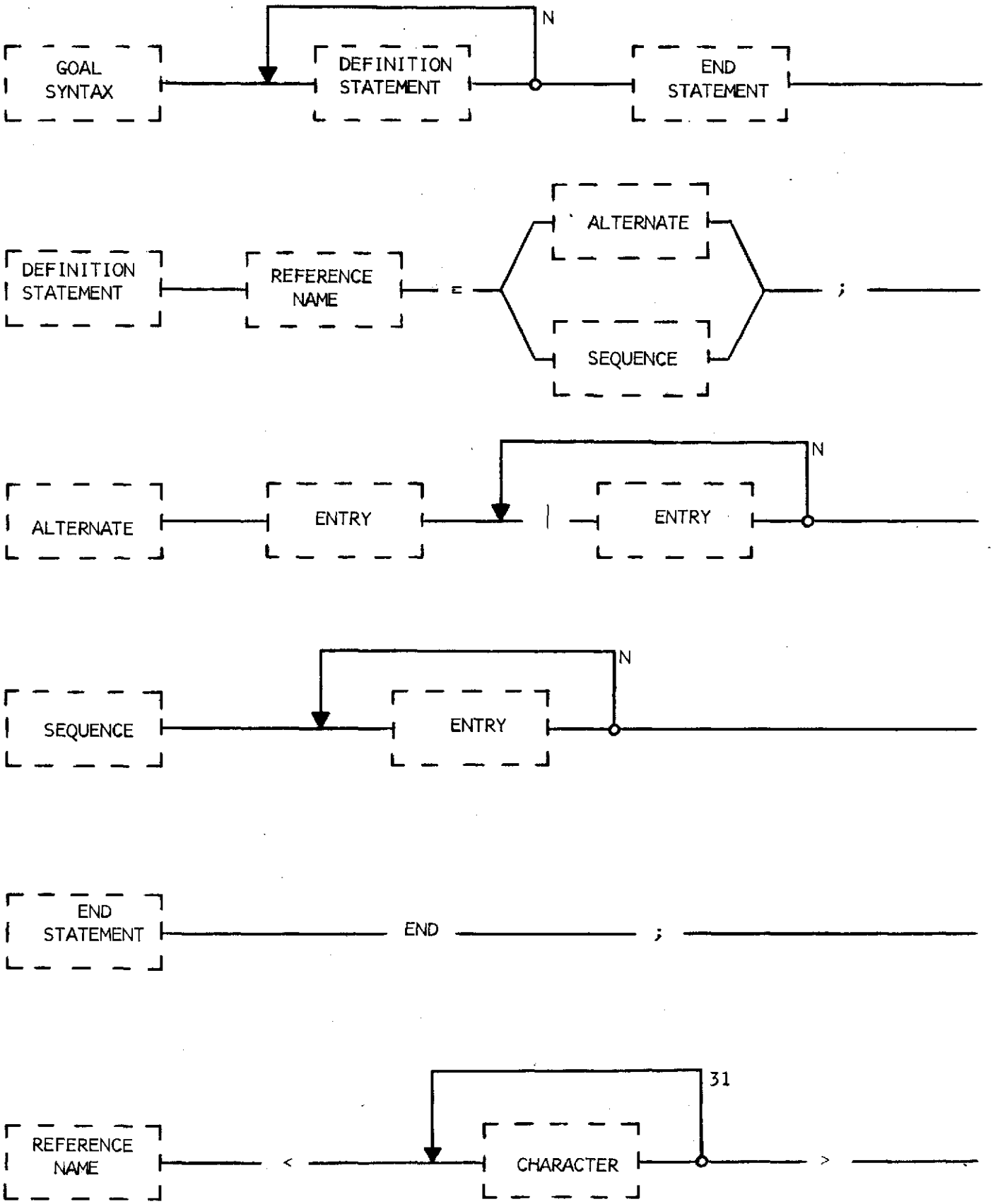


Figure 2-2 (1 of 3)

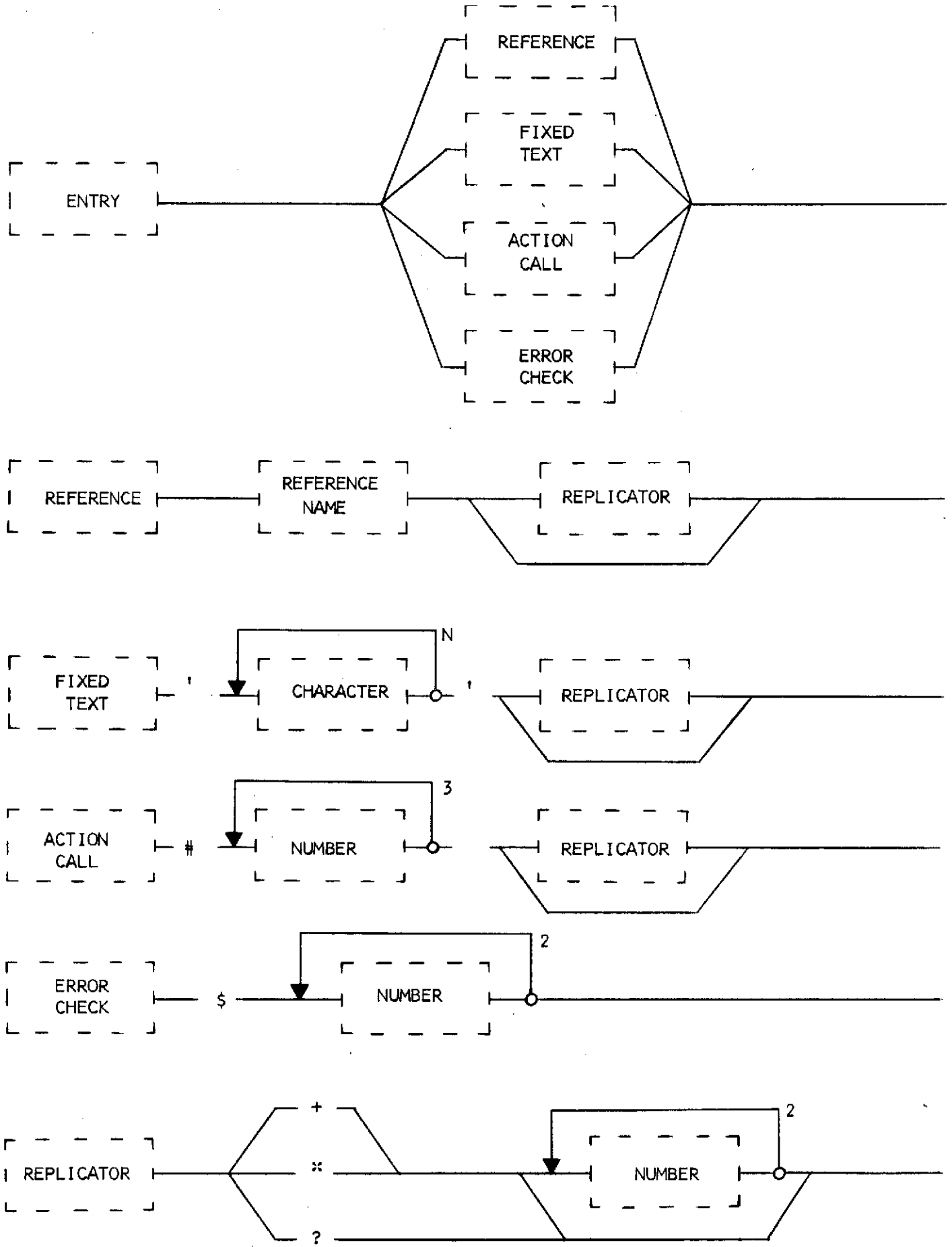


Figure 2-2 (2 of 3)

N IS A LARGE INTEGER. ITS VALUE IS IMPLEMENTATION DEPENDENT.

Figure 2-2 (3 of 3)

2.1.1 Syntax Equation Translation Example

To illustrate the technique for translating syntax diagrams to modified BNF, the BEGIN SUBROUTINE statement, (Figure 2-3), is converted to the syntax equations, (Figure 2-4).

Note that intermediate syntactical elements are defined to represent various paths through the syntax diagram. Equations are also provided to define lower level syntactical elements such as LETTER and NUMERAL. The value of the replicator, K, is not specified, thus no limit is imposed during parsing. The action routines, (#101 to #105), perform the actual compilation process and test for usage errors. The syntax equations (tables) serve to identify the statement format and cue action routines according to the structure of the statement.

2.1.2 Parsing Example

The equations shown in Figure 2-4 were processed by the syntax processor to generate tables which were then used by a 'prototype' parser to process several variations of the BEGIN SUBROUTINE statement.

The results of this run are shown in Figure 2-5. This listing is not an example of compiler output. It is only intended to show the operation of the parser and the relationship between syntax diagrams and the statement being parsed.

2.2 SYNTAX PROCESSOR

The syntax processor is a stand-alone program which accepts input in the form of syntax equations, (described in Section 2.1), and converts this to syntax tables which are stored in the Syntax Table File for subsequent use by the GOAL compiler parsing routines. This relationship is shown in Figure 2-1. The syntax File may contain up to fifty different syntax tables of moderate size. This enables the use of language subsets or experimental versions of a language syntax.

The syntax processor employs a basic parsing routine and a special syntax table which is generated by the syntax file initialization program. In general, the initialization program need only be used once. The initialization program also reads in a character set record which is used by the GOAL compiler to identify all letters, numerals, and symbols used in the GOAL language. This technique enables functional substitution of one character for another, (in compiler input), without regeneration of the GOAL compiler itself.

The operation of the syntax file initialization program and the syntax processor is shown in Figure 2-6.

GOAL SYNTAX DIAGRAM NUMBER 8

BEGIN SUBROUTINE

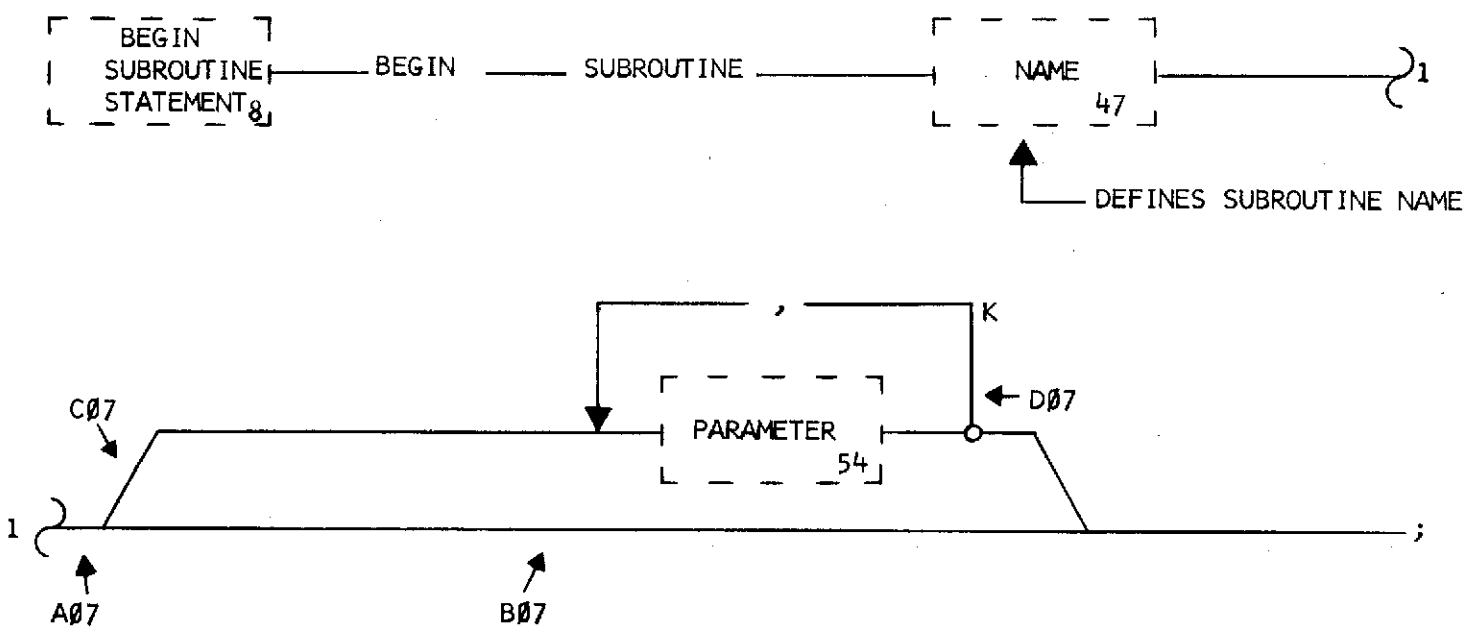


Figure 2-3

```
*
*          GOAL SYNTAX EQUATION FOR BEGIN SUBROUTINE STATEMENT
*
<BEGIN SUBROUTINE STMT> = 'BEGIN SUBROUTINE' #101 $101 <NAME> #102
                $102 <A07> #103 ;
        <A07> = <B07> | <C07> ;
        <B07> = ';' #104 ;
        <C07> = <PARAMETER> #105 <D07>* ';' ;
        <D07> = ',' <PARAMETER> #105 ;
*
*          COMMENTS FOR ACTION AND ERROR ROUTINES
* #101- INITIALIZE LIST FLAGS AND COUNTERS FOR BEGIN SUBROUTINE STMT.
* $101- ERROR - INCORRECT SUBROUTINE NAME. STOP PARSING WITH $101.
* #102- SAVE THE SUBROUTINE NAME.
* $102- ERROR - INCORRECT PARAMETER. STOP PARSING WITH $102.
* #103- PARSE IS OK. PREPARE OUTPUT LIST. RETURN CONTROL TO THE PARSER.
* #104- SET LIST FLAG TO EMPTY. THERE ARE NO PARAMETERS.
* #105- PUT PARAMETER IN THE LIST. INCREMENT LIST COUNTER.
* THE REPLICATOR FOR REFERENCE D07 IS K.
*
*
*
*
<LETTER> =
          'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' |
          'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' |
          'U' | 'V' | 'W' | 'X' | 'Y' | 'Z' ;
*
*
*
<NAME> =
          '(' <LETTER> <NAME 1>*31 ')' ;
<NAME 1> =
          <LETTER> | <NUMERAL> ;
*
*
*
<NUMERAL> =
          '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;
*
*
*
<PARAMETER> =
          <NAME> ;
END ;
```

Figure 2-4

```
BEGIN SUBROUTINE (4XY) (KSC) ;
        ACTION routine 101 called
        ERROR number 101 occurred  - parse terminated

BEGIN SUBROUTINE (XY4) (KSC) ;
        ACTION routine 101 called
        ACTION routine 102 called
        ACTION routine 105 called
        ACTION routine 103 called

BEGIN SUBROUTINE (XY4) (KSC),(KSC1),(KSC2) ;
        ACTION routine 101 called
        ACTION routine 102 called
        ACTION routine 105 called
        ACTION routine 105 called
        ACTION routine 105 called
        ACTION routine 103 called

BEGIN SUBROUTINE (XY4) (KSC),(KSC1),(2KSC) ;
        ACTION routine 101 called
        ACTION routine 102 called
        ACTION routine 105 called
        ACTION routine 105 called
        ERROR number 102 occurred - parse terminated

BEGIN SUBROUTINE (ABC) (DEF) #$/@| ;
        ACTION routine 101 called
        ACTION routine 102 called
        ACTION routine 105 called
        ERROR number 102 occurred - parse terminated

BEGIN SUBROUTINE (ABCD) ;
        ACTION routine 101 called
        ACTION routine 102 called
        ACTION routine 104 called
        ACTION routine 103 called
```

NOTE: Six GOAL Begin Subroutine statements were compiled using the syntax tables derived from the syntax equations. Three of the statements were determined to contain syntax errors and the appropriate error numbers were selected for printing on this listing. The three correct statements compiled and the appropriate action routines were called for processing elements of the statements.

Figure 2-5

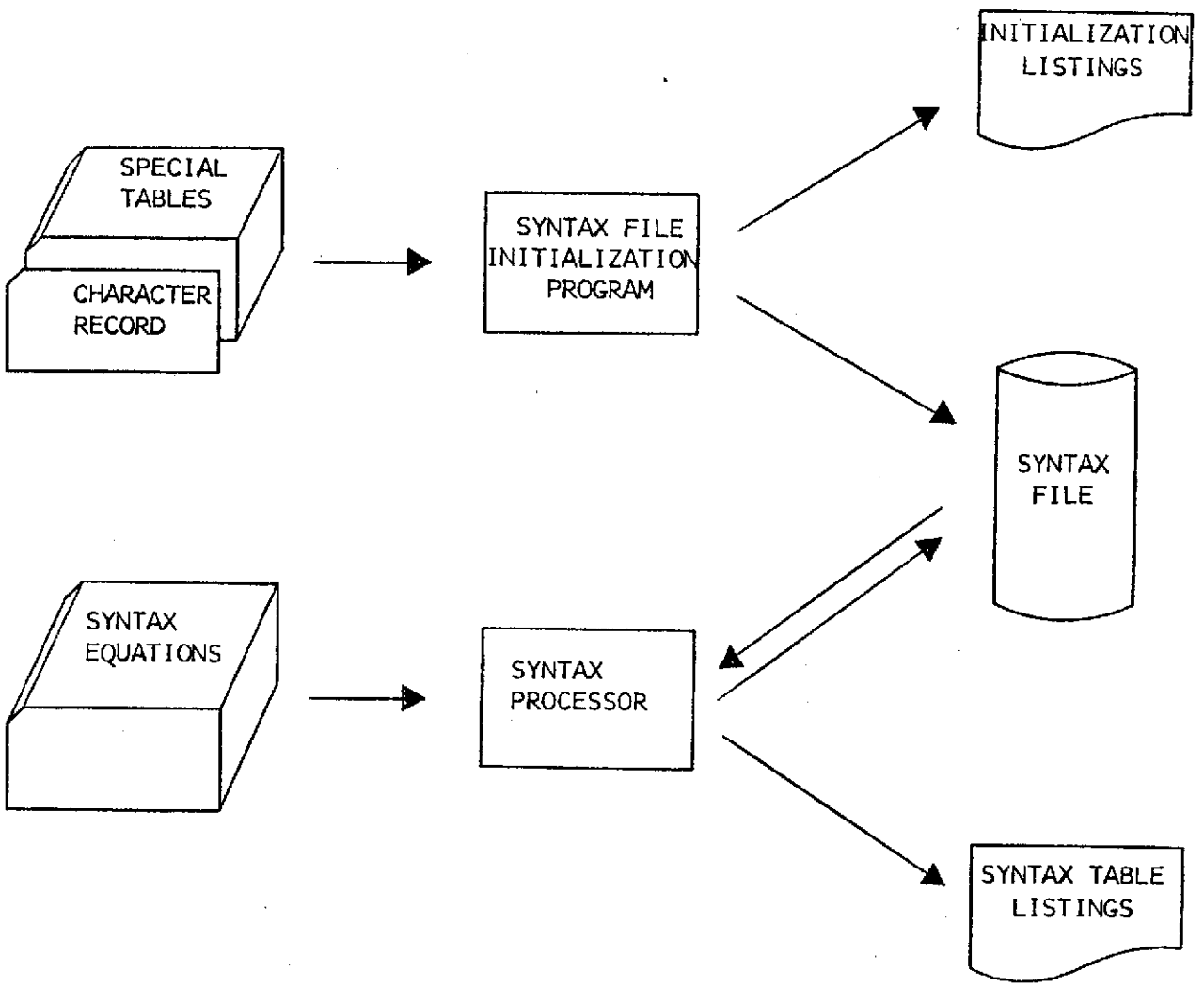

INITIALIZATION
LISTINGS
SPECIAL
TABLES
CHARACTER
RECORD
SYNTAX FILE
INITIALIZATION
PROGRAM
SYNTAX
FILE
SYNTAX
EQUATIONS
SYNTAX
PROCESSOR
SYNTAX TABLE
LISTINGS

Figure 2-6.

2.3 INPUT PROCESSOR

The Input Processor controls all inputs to the GOAL compiler. These inputs come from the following sources:

1. SOURCE DECK - A GOAL program which the user supplies for the GOAL compiler.

2. MACRO FILE - A file consisting of macro 'bodies'. These 'bodies' are placed in the macro file whenever a macro is defined. They are retrieved from the file whenever expansion or execution of one is desired. The Input Processor will determine when records from this file are to be input to the GOAL Compiler for parsing.

3. SUBROUTINE FILE - A file consisting of copies of subroutines which were embedded in a GOAL program. The Input Processor will determine when records from this file are to be input to the GOAL compiler for parsing.

4. DATA BANK - Copies of macro bodies are contained in the Data Bank. The Input Processor will determine when macro records from the Data Bank are to be input to the GOAL compiler for parsing.

The Input Processor also scans all inputs, when applicable, for abbreviations. When an abbreviation is found, the proper substitution is made.

If, at any time during a GOAL compilation the source deck is depleted, prior to encountering an END statement, the Input Processor will log an error to this effect and terminate the compilation.

2.4 PARSING ROUTINES

The GOAL compiler will utilize a table guided top-down parsing algorithm. The tables used by the parser are generated as described in Section 3.2. The parsing routines are of two types:

1. GENERAL PURPOSE - These routines form the basic parser. They are used by all syntax tables. They perform the input statement scan according to the top-down technique, flag unrecognizable constructions, and cue the execution of action routines according to the structure of the input statement.

2. SPECIAL PURPOSE - These are the action routines which provide processing to support recognition and testing of specific syntactical elements such as labels, variables, macros, and subroutines.

2.4.1 General Purpose Parsing Routines

The primary function of these routines is to scan statements in the input stream and recognize permissible constructions according to the contents of the syntax tables. The recognition criterion is simple appearance. As constructions are recognized the parse continues until the entire statement has been processed. If a construction is not recognized, alternate definitions are tested. If none is satisfied, the invalid field is marked and a diagnostic message is given according to the last error checkpoint processed from the syntax tables. These routines also test maximum repetition counts when specified in the syntax equations. When a construction is successfully identified in the input stream these routines may cue the execution of a special purpose action routine specified in the syntax tables.

2.4.2 Special Purpose Parsing Routines

These are the action routines cued by the general purpose parsing routines. They may perform any of the following types of functions:

1. Specialized compiler support such as macro definition, macro expansion, and subroutine processing.

2. Symbol table operations for definition and reference.

3. Usage validation for any syntactical element of the GOAL Language.

The special purpose parsing routines may signal a no-compare condition to the basic parser. In this case alternate definitions will be tested or a diagnostic message will be given. In this way the special purpose routines may resolve the difference between syntactical elements which have similar appearance but different meanings.

2.5 COMPILER DIAGNOSTICS

Two basic types of errors are recognized in the input statements to the GOAL compiler. These are:

1. Syntax errors - The appearance of the statement does not conform to any permissible variation described in the GOAL Syntax Diagrams. In this case the parse is terminated for the current statement.

2. Usage errors - The statement is syntactically correct, however, some valid construction is incorrectly used. In this case the parse may continue to process the remainder of the current statement.

In both cases the statement is flagged in the expanded statement listing, (see Section 2.6). A mark is placed under the field in which the error occurred and all relevant data is logged for use in the diagnostic summary report.

2.6 COMPILER OUTPUT REPORTS

The following reports will be provided on request, by the GOAL compiler.

1. Source Record Listing
2. Expanded Source Statement Listing
3. Internal Name Cross-Reference Listing
4. Statement Label Cross-Reference Listing
5. Function Designator Cross-Reference Listing

These reports may be requested using compiler directives described in sub-section 2.6.7. The reports are described in greater detail and examples are given in the following sub-sections. The examples are intended to illustrate the contents and organization of the reports. These reports are available for the main GOAL program and GOAL subroutines.

2.6.1 Source Record Listing

This report contains a listing of all source records processed by the GOAL compiler. The records are assigned sequence numbers to facilitate reference from diagnostic messages. An example of this report is given in Figure 2-7.

2.6.2 Expanded Source Statement Listing

This report contains a listing of all GOAL statements processed by the compiler. Each statement is assigned a sequence number for reference in other output reports. This sequence number has a plus '+' sign next to it if the statement was part of a macro body being expanded. When the first valid procedural statement is encountered, a new page is started and the comment

```
$ ********** BEGIN OPERATING STEPS ********** ;
```

is printed as the first line of this page. The beginning of a GOAL statement will start on a new line in this report. If a statement contains an error, the word ** ERROR ** will be placed in the margin preceding the statement number and an asterisk (*) will be placed under the field in error. All abbreviations and replacements will be expanded. Text replacement will be performed subject to the following rules:

1. If the replacement text field is the same size as the original text field, a simple substitution is performed.
2. If the replacement text field is smaller than the original text field, it is inserted left-justified and any remaining original text is replaced by blanks.
3. If the replacement text field is larger than the original text field, the statement is expanded in size to provide space for the replacement text.

4. A record is not expanded in size if it contains blank areas sufficient to contain replacement text. That is, the position of non-replacement text is not affected.

5. If an expanded record cannot be printed on a single line, additional line(s) are used to contain the overflow.

The above rules were chosen to give the user control of the format of this report. When the letter 'S' is used to represent a step number instead of the word 'STEP', this report substitutes the word 'STEP' according to the rules above. An example of this report is given in Figure 2-8.

2.6.3 Internal Name Cross-Reference Listing

This report gives a listing of all internal names defined or referenced in a GOAL program. The names are listed in alphabetical order. The statement numbers refer to the sequencing given in the expanded source statement listing. Type and size attributes are given. Undefined and unreferenced names are flagged. An example of this report is given in Figure 2-9.

2.6.4 Statement Label Cross-Reference Listing

This report gives a listing of all statement labels defined or referenced in a GOAL program. The labels are listed in ascending sequence. The statement numbers refer to the sequencing given in the expanded source statement listing. Undefined and unreferenced statement labels are flagged. An example of this report is given in Figure 2-10.

2.6.5 Function Designator Cross-Reference Listing

This report produces two listings. The first listing contains all of the Data Bank names with their revision labels and a Data Bank reference number. These are listed in alphabetical order.

The second listing contains all of the Function Designators referenced in a GOAL program. These are listed in alphabetical order. Relevant information such as type, address and Data Bank number is listed. The Data Bank number which is listed corresponds to the Data Bank number in the listing of Data Bank names. This enables the user to identify the name of the Data Bank from which a given Function Designator was retrieved. Statement numbers refer to sequencing given in the expanded source statement listing. Undefined Function Designators are flagged. An example of this report is given in Figure 2-11.

2.6.6 Diagnostics Summary

This report gives a listing of all warnings and errors detected in a GOAL program. The following warnings are generated if they exist:

1. Unreferenced Internal Names

2. Unreferenced Step Numbers

The following errors are generated if they exist:

1. Parsing errors
2. Undefined internal names
3. Undefined step numbers
4. Undefined Function Designators
5. Step numbers referenced on a Disable Statement, but not defined on a when Interrupt Statement
6. Step numbers referenced on a Release Statement, but not defined on a Concurrent Statement

Parsing errors are recognized during a GOAL compilation and are flagged and marked in the expanded source statement listing. All other warnings and errors are determined after the compilation and may not be flagged in the expanded source statement listing. An example of this report is given in Figure 2-12.

2.6.7 Compiler Directives

The following GOAL statements are provided to enable the user to control compiler options. None of these statements is mandatory. They are individually described in the following sections. The default action is explained, if applicable, when they are not used. Syntax equations for these statements are shown in Figure 2-13.

```
*SEQ  n  ;
```

This statement is used to specify the size of the sequencing field of the input records. This field is taken to be the last (n) characters of each record. The sequencing field is ignored by the GOAL compiler. If this statement is not used the entire input record is processed. $0<n\leq 10$.

```
*EDIT  ONLY  ;
```

This statement is used to suppress generation of intermediate GOAL (object) data. This expedites compilation for the purpose of obtaining listings and error checks. If this statement is not used the intermediate GOAL data is generated.

```
*LIST   option,  option, ... option  ;
```

This statement is used to request generation of specific GOAL output reports. The options may be:

SOURCE, EXPAND, LABELS, SYMBOLS, FDS, DIAG

These correspond to the reports described earlier in this section. If the word list appears with no options, then no reports will be generated. The diagnostic summary will always be generated when errors or warnings are detected during a compilation. If this statement is not used all of the above reports will be generated.

The following compiler directives affect only the format of the expanded statement listing.

```
*TITLE   ( ... TEXT CONSTANT ... )  ;
```

The text constant is printed on the top line of each page. If this statement is not used this portion of the top line is left blank. The length of the text constant must not exceed 100 characters.

```
*DATE ( ... TEXT CONSTANT ... )     ;
```

The specified date is printed on the top line of each page. If this statement is not used this field is left blank. The length of the text constant must not exceed 8 characters.

```
*LINE   a,b   ;
```

(a) and (b) are integers indicating the number of lines per page and the number of characters per line respectively. After printing (a) lines on a given page, a new page will be started. After filling (b) characters in a given line, a new line will be started. The value of (a) must be between 1-32767 and the value of (b) must be between 80 and 110. If this statement is not given, the value for (a) is taken to be 50 and the value for (b) is 100. Note that the (b) value only applies when the expanded statement cannot be printed on a single line.

```
*PAGE   n   ;
```

This statement is used to begin a new page and set the page counter to (n). If (n) is not specified the page counter is not affected. The page number is given on the top line of the listing.

```
*CONVERT  ;
```

This statement has two functions:

1. If encountered while the compiler is in normal processing mode, this statement will cause all short form GOAL statements following it to be expanded to long form GOAL statements.

2. If encountered while the compiler is in the convert and punch deck mode, this statement will cause the punch deck option to be terminated and short form conversion will continue.

```
*CONVERT DECK  ;
```

This statement has two functions:

1. If encountered while the compiler is in the convert mode, this statement will cause a source deck to be punched.

2. If encountered while the compiler is in normal processing mode, this statement will cause conversion of short form to long form to start and also cause a source deck to be punched.

Note: This statement causes the line size of the expanded source statement listing to be set to 80 if not already there. This causes the card images being punched to correspond with the expanded source statement listing. The line size compiler directive can appear while the compiler is in this mode, but the line size specified will not take effect until this mode is terminated. If no line size compiler directives are encountered while in this mode, then the line size will be restored to the value it was when the convert deck option was encountered.

```
*CONVERT OFF  ;
```

This statement causes all options of the convert statement to be terminated. When this statement is encountered, the compiler will return to normal processing mode.

GOAL COMPILER SOURCE RECORD LISTING

```
RECORD     SOURCE RECORD

     1         * TITLE (GOAL LISTING EXAMPLE), DATE (5NOV73) ;
     2         BEGIN PROGRAM (ERROR TEST) REVISION 1 ;
     3         USE (TERMINALS) ;
     4         BEGIN MACRO XXX (TITLE), (NAME), (DATE) ;
     5            DISPLAY TEXT (TITLE) TO <CRT1-0> ;
     6            DISPLAY TEXT (NAME) TO <CRT1-2> ;
     7            DISPLAY TEXT (DATE) TO <CRT1-3> ;
     8         END MACRO ;
     9         DECLARE NUMBER (NBR) = 100 ;
    10         DECLARE NUMBER (NB1) = 10 ;
    11         GO TO S 100 ;
    12    S 200 WAIT 5 SECS ;
    13         EXPAND AND EXECUTE XXX ,(GOAL PROCEDURE),(VATC),(5 NOV 1973), ;
    14         VERIFY <UNDEFINED FUNCTION DESIGNATOR> IS ON, GO TO S 300 ;
    15         (NBR) = 10 ;
    16         LET (NB1) = 20 ;
    17    S 300 SET (UNDEFINED NAME) FUNCTIONS TO (ON) ;
    18         DISPLAY TEXT (GOAL EXAMPLE) TO <CRT1-0> ;
    19         DISABLE S 300 ;
    20         DISABLE S 800 ;
    21         RELEASE S 300 ;
    22         RELEASE S 900 ;
    23         DSP TXT (REDUNDANT PRESS REG ON COMMAND),
    24             TXT (HAS BEEN ISSUED), TO <CRT2> ;
    25    S    800 WTE 5 SECS ;
    26         * CONVERT ;
    27         DSP TXT (PRESENT VALUE OF STAGE INLET PRESS),
    28                 TO <CRT2-10> ;
    29    S    900 DLY 5 SECS ;
    30         * CONVERT OFF ;
    31         END PROGRAM ;
```


Figure 2-7.

GOAL COMPILER EXPANDED SOURCE STATEMENT LISTING

```
STMT      EXPANDED SOURCE STATEMENT

   1        BEGIN PROGRAM (ERROR TEST) REVISION 1 ;
   2        USE (TERMINALS) ;
   3        BEGIN MACRO XXX (TITLE), (NAME), (DATE) ;
               DISPLAY TEXT &1&&&&  TO <CRT1-0> ;
               DISPLAY TEXT &2&&&  TO <CRT1-2> ;
               DISPLAY TEXT &3&&&  TO <CRT1-3> ;
   4        END MACRO ;
   5        DECLARE NUMBER (NBR) = 100 ;
   6        DECLARE NUMBER (NB1) = 10 ;
```


Figure 2-8. (1 of 2)

GOAL COMPILER EXPANDED SOURCE STATEMENT LISTING

```
            STMT      EXPANDED SOURCE STATEMENT

                                  $ ********** BEGIN OPERATING STEPS ********** ;

              7         GO TO STEP 100 ;
              8   STEP 200 WAIT 5 SECS ;
              9   XXX ,(GOAL PROCEDURE),(VATC),(5 NOV 1973), ;
             10+           DISPLAY TEXT (GOAL PROCEDURE) TO <CRT1-0> ;
             11+           DISPLAY TEXT (VATC) TO <CRT1-2> ;
             12+           DISPLAY TEXT (5 NOV 1973) TO <CRT1-3> ;
** ERROR **   13          VERIFY <UNDEFINED FUNCTION DESIGNATOR> IS ON, GO TO S 300 ;
                                 *
** ERROR **   14          (NBR) = 10 ;
                           *
             15          LET (NB1) = 20 ;
** ERROR **   16   STEP 300 SET (UNDEFINED NAME) FUNCTIONS TO (ON) ;
                                 *
             17          DISPLAY TEXT (GOAL EXAMPLE) TO <CRT1-0> ;
             18          DISABLE STEP 300 ;
             19          DISABLE STEP 800 ;
             20          RELEASE STEP 300 ;
             21          RELEASE STEP 900 ;
             22          DSP TXT (REDUNDANT PRESS REG ON COMMAND),
                             TXT (HAS BEEN ISSUED), TO <CRT2> ;
             23   STEP 800 WTE 5 SECS ;
             24          DISPLAY TEXT (PRESENT VALUE OF STAGE INLET PRESS),
                               TO <CRT2-10> ;
             25   STEP 900 DELAY 5 SECS ;
             26          END PROGRAM ;
```


Figure 2-8. (2 of 2)

INTERNAL NAME CROSS-REFERENCE LISTING

INTERNAL NAME	TYPE	SIZE	DEFINED AT	REFERENCED AT
(NBR)	NUMERIC	00001	0005	** UNREFERENCED **
(NB1)	NUMERIC	00001	0006	0015
(UNDEFINEDNAME)			** UNDEFINED **	0016

Figure 2-9.

STATEMENT LABEL CROSS-REFERENCE LISTING

```
 LABEL      DEFINED AT       REFERENCED AT

S 0100   ** UNDEFINED **   0007
S 0200         0008        ** UNREFERENCED **
S 0300         0016        0018 0020
S 0800         0023        0019
S 0900         0025        0021
```


Figure 2-10.

FUNCTION DESIGNATOR CROSS-REFERENCE LISTING

DATA BANK NAME	DATA BANK REVISION LABEL	DATA BANK NUMBER
(TERMINALS)		0001

Figure 2-11. (1 of 2)

FUNCTION DESIGNATOR CROSS-REFERENCE LISTING

FUNCTION DESIGNATOR	TYPE	ADDRS	DATA BNK	REFERENCED AT
<CRT1-0>	SYSTEM I/O	00100	0001	0010 0017
<CRT1-2>	SYSTEM I/O	00102	0001	0011
<CRT1-3>	SYSTEM I/O	00103	0001	0012
<CRT2>	SYSTEM I/O	00002	0001	0022
<CRT2-10>	SYSTEM I/O	00210	0001	0024
<UNDEFINEDFUNCTIONDESIGNATOR>		** UNDEFINED **		0013

Figure 2-11. (2 of 2)

GOAL COMPILER DIAGNOSTIC SUMMARY

WARNINGS.

THE FOLLOWING INTERNAL NAMES WERE UNREFERENCED :

(NBR)

THE FOLLOWING STEP NUMBERS WERE UNREFERENCED :

S 0200

ERRORS.

ERROR NUMBER	STMT NUMBER	STMT POSITION	SOURCE RECORD NUMBER	ERROR DESCRIPTION
806	13	8	14	INVALID OR MISSING EXTERNAL DESIGNATOR.
995	14	1	15	THIS STATEMENT IS NOT RECOGNIZED AS A GOAL STATEMENT .
806	16	14	17	INVALID OR MISSING EXTERNAL DESIGNATOR.

THE FOLLOWING INTERNAL NAMES WERE UNDEFINED :

(UNDEFINEDNAME)

THE FOLLOWING STEP NUMBERS WERE UNDEFINED :

S 0100

THE FOLLOWING FUNCTION DESIGNATORS WERE UNDEFINED :

<UNDEFINEDFUNCTIONDESIGNATOR>

THE FOLLOWING STEP NUMBERS WERE REFERENCED ON A DISABLE STATEMENT BUT NOT DEFINED ON A WHEN INTERRUPT STATEMENT :

S 0300 S 0800

THE FOLLOWING STEP NUMBERS WERE REFERENCED ON A RELEASE STATEMENT BUT NOT DEFINED ON A CONCURRENT STATEMENT :

S 0300 S 0900

END OF DIAGNOSTICS.

TOTAL NUMBER OF SOURCE RECORDS: 31
TOTAL NUMBER OF STATEMENTS: 26
TOTAL NUMBER OF WARNINGS: 2
TOTAL NUMBER OF ERRORS : 10
HIGHEST CONDITION CODE WAS 8

Figure 2-12.

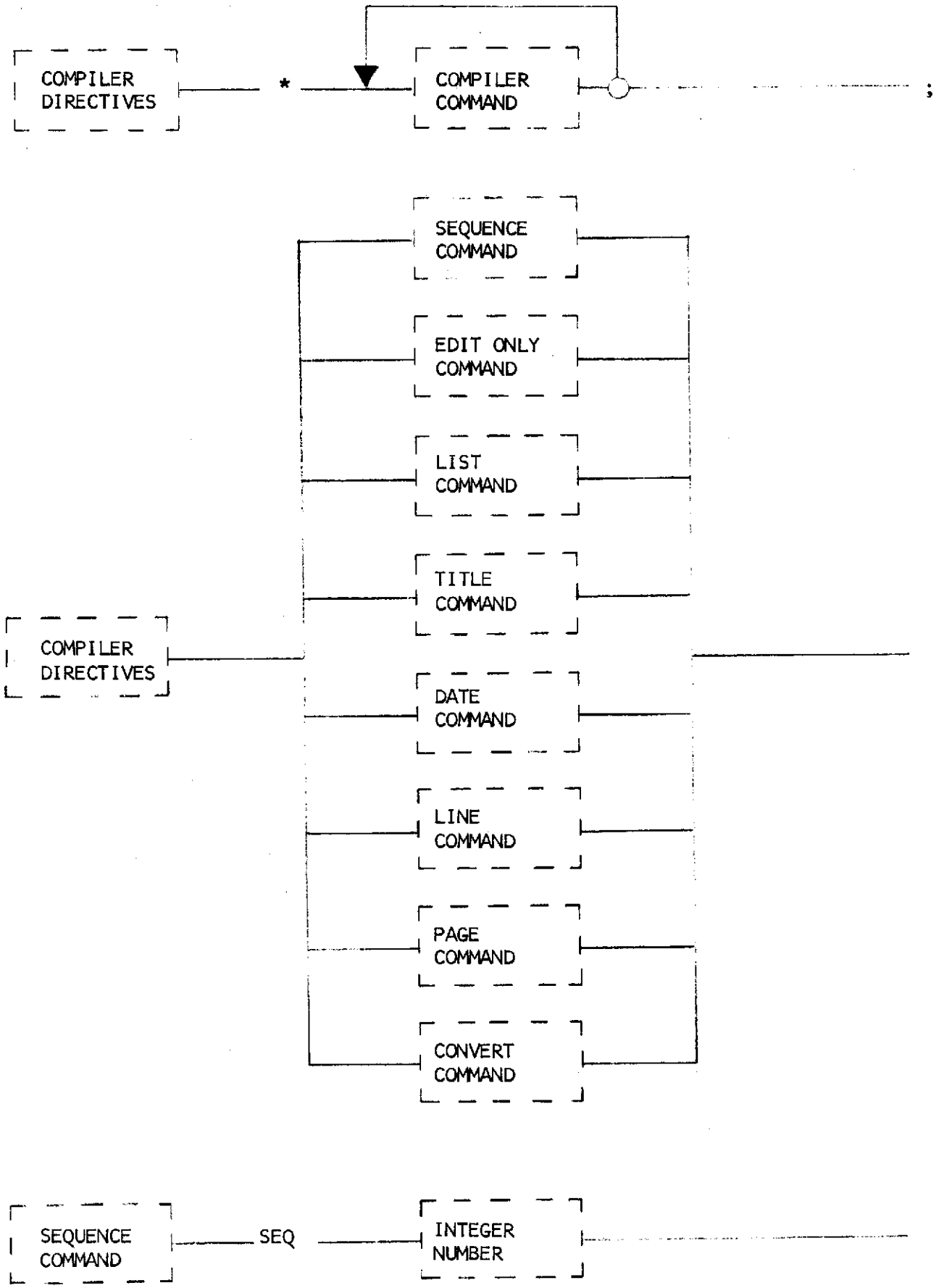


Figure 2-13 (1 of 2)

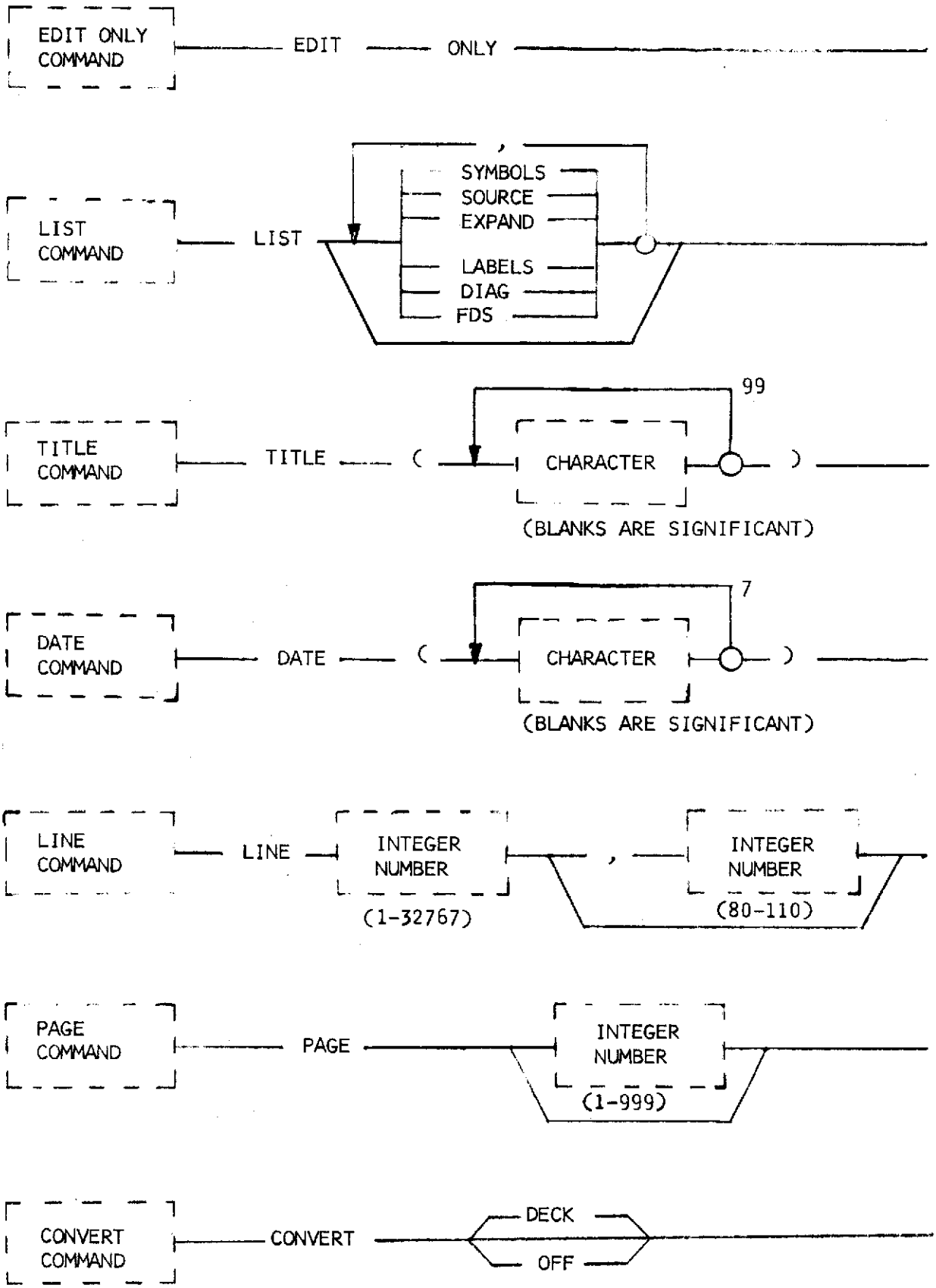


Figure 2-13 (2 of 2)

2.7 INTERMEDIATE GOAL DATA GENERATOR

The intermediate text output from the compiler is a data set which represents, in tabular fashion, all of the informational content of the GOAL source program. This data set is sequentially processable and contains logical records of varying length. Each logical record consists of a fixed header portion followed by a varying length data portion. Total record content can be read, written, and processed in a FORTRAN array of the INTEGER type. Each element of the array is capable of storing a signed number or a single character.

The fixed header portion of each logical record contains the following information:

- Intermediate Text Record Number - Each output text record is numbered in an ascending sequence.
- Record Type - A single number to indicate the format of the data portion of the record.
- GOAL Statement Number - If the intermediate text record resulted from a specific GOAL statement, the sequence number (sequence number printed on the listing by the compiler) is contained in this field.
- GOAL Statement Label - If the record resulted from a specific GOAL statement, and that statement was labeled in the source program, the label (numeric representation) will be contained in this field.
- Record Length - The actual length of this logical record.

Section 3.2.3 illustrates the logical content of the intermediate text data set.

3.0 COMPILER SOFTWARE DESCRIPTIONS

This section provides a more detailed description of the software elements of the Syntax Processor and the GOAL Compiler which were identified in Section 2.

3.1 SYNTAX PROCESSOR

The following Section is a detailed descirption of the Syntax Generation program described in Section 2.2.

The principal functions of the Syntax Generation program are:

- o To accept a set of syntax equations as input on punched cards.
- o To parse these equations according to syntax diagrams (Figure 2-2).
- o To generate diagnostic messages for equation errors.
- o To generate a syntax table.

To support these functions, the Syntax Generation program contains the following software elements:

- o Initialization Section
- o Input Section
- o Parser Section
- o Action Routines
- o Subroutines

Each individual element of the Syntax Generation program is described in Sub-sections 3.1.1 - 3.1.5.

A brief description of all variables used is in Sub-section 3.1.6.

A graphic description of the syntax table generated by the program is given in Sub-section 3.1.7.

3.1.1 Initialization Section

All variables used by the program are declared with their initial values.

The character table to be used is read into the variable CHRTAB.

A control card is read which contains the following information:

1. The number of the syntax table to be used by the program for parsing.

2. The number to be assigned to the syntax table being created by the program.

3. A flag which indicates whether or not a trace of the program's operation is desired.

4. A flag which indicates whether the syntax table being created is to replace an existing syntax table in the syntax file or to be added to the syntax file.

The syntax table to be used by the program is read into the variable STTAB.

3.1.2 Input Section

This Section is repeated each time a statement is completed. Its purpose is to fill the input buffer with a statement, excluding comment cards. If the input deck should become exhausted prior to encountering an end statement, a warning message will be printed and an END statement will be inserted.

3.1.3 Parser Section

This Section of the program functions the same as the Parser for the GOAL compiler. According to the syntax table being used, it executes sections of the program which make syntactical analysis of the input stream. The Parser consists of the following parts:

- Initialization
- Entry Pointer Advancement
- Reference Routine
- Text Routine
- Subroutine Routine
- Error Checkpoint Routine
- Terminal Routine

3.1.3.1 Initialization. This part initializes entry into the syntax table used by the program. It also initializes pointers used by the other parts in processing the syntax table.

3.1.3.2 Entry Pointer Advancement. This part advances the initialized entry pointer to the next entry in the syntax block. A branch is made to either the reference routine, text routine, subroutine routine, or error checkpoint routine, according to the type code found in the syntax table.

3.1.3.3 Reference Routine. This part is entered only if the syntax table contains a reference to another block. It adjusts all of the pointers and saves 'backup' information in case an error occurs. When all adjustments have been made it returns to the entry pointer advancement part.

3.1.3.4 Text Routine. This part is entered only if the syntax table contains a reference to a text constant. This routine determines if the text in the input stream is the same as the text constant in the syntax table.

3.1.3.5 Subroutine Routine. This part is entered only if the syntax table contains a reference to an action routine. This routine will branch to the specified action routine.

3.1.3.6 Error Checkpoint Routine. This part is entered only if the syntax table contains a reference to an error checkpoint. This routine places the error number in the header of the syntax block currently being processed. If an error is detected after this point, the parse will terminate and the error number will be logged.

3.1.3.7 Terminal Routine. This part is entered after the text routine, subroutine routine, or error checkpoint routine have finished. This routine determines if a syntactical error is valid (an alternative exists). It also determines if parsing should terminate because of an error checkpoint or completion of the syntax table for an equation.

3.1.4 Action Routines

The following sub-sections describe the functions of each of the action routines used by the syntax generator program.

3.1.4.1 Action Routine #1. This routine looks for a name in the input buffer. If a name is found, it is validated and added to the symbol table if necessary. A return code of 0 is passed back to the Parser to signal successful completion of this action routine. A return code of -1 is passed back if this action routine fails to find a name or finds an invalid name.

Note: This action routine uses a subroutine to add the name to the symbol table (Sub-section 3.1.5.1).

3.1.4.2 Action Routine #2. This routine looks for an argument (name to the right of the equal sign) in the input buffer. If an argument is found, it is validated and added to the symbol table if necessary. A flag is set to distinguish this element from the ROOT element. This is done because this is a reference to an element and the ROOT is not referenced. The type of argument is stored in the syntax table being built. A return code of 0 is passed back to the Parser. If an invalid argument is detected, a return code of -1 is passed back to the Parser.

Note: This action routine uses a subroutine to add the argument to the symbol table (Sub-section 3.1.5.1).

3.1.4.3 Action Routine #3. This routine looks for a text constant in the input buffer. If a valid text constant is found, it is placed in the text table and the correct type and location in the text table is placed in the syntax table being built. A return code of 0 is passed back to the Parser. If an invalid text constant is found, a return code of -1 is passed back to the Parser.

Note: The text table being built by this routine is added to the end of the syntax table after the program is completed. All references to text constants are updated. All text constants in this table are unique. Multiple references to a text constant do not cause multiple copies to be inserted in the table.

3.1.4.4 Action Routine #4. This routine looks for action routine references in the input buffer. If a valid reference to an action routine is found, its correct type code and action routine number are placed in the syntax table being built. A return code of 0 is passed back to the Parser. If an invalid action routine reference is made, a return code of -1 is passed back to the Parser.

Note: This routine uses a subroutine to find a valid action routine number (Sub-section 3.1.5.2).

3.1.4.5 Action Routine #5. This routine looks for error number references in the input buffer. If a valid reference to an error number is found, the correct type code and error number are placed in the syntax table being built. A return code of 0 is passed back to the Parser. If an invalid reference to an error number is made, a return code of -1 is passed back to the Parser.

Note: This routine uses a subroutine to find a valid error number (Sub-section 3.1.5.2).

3.1.4.6 Action Routine #6. This routine sets the second location of the current header being built to two. This signifies that the syntax equation being built is an alternative. A return code of 0 is passed back to the Parser.

3.1.4.7 Action Routine #7. This routine is entered when a semicolon is Parsed. It places a -1 in the syntax table being built signifying the end of an equation. This routine also reads another statement into the input buffer.

3.1.4.8 Action Routine #8. This routine is entered when all syntax equations being input to the new program have been processed. It resolves all references in the new syntax table. It writes the symbol table in the output report, flagging all undefined symbols. It scans the table and determines which equation is the ROOT equation. It writes the new syntax table into the syntax file, if no errors were detected, and terminates the program.

3.1.5 Subroutines

The following Sub-sections describe the functions of each of the subroutines used by the Syntax Generation program.

3.1.5.1 Symbol Table Subroutine. This subroutine maintains the symbol table used in generating a syntax table. It scans the input buffer searching for a name. When a name is found, it adds it to the symbol table if it is missing. It adjusts all pointers which relate to the symbol table and tests for unrecoverable errors, i.e., symbol table overflow, etc.

3.1.5.2 Number Look-Up Subroutine. This routine scans the input buffer searching for a number. If a valid number is found, an error message is printed and parsing is terminated.

3.1.5.3 GETCHR. This subroutine reads a character table into a buffer in the calling program. The character table read is the one used to generate the syntax table which the program is using. This way the correct interpretation of codes in the syntax table can be made.

3.1.5.4 GETSTX. This subroutine reads a syntax table into a buffer in the calling program. The table number, buffer location, buffer size, ROOT and return code are specified via parameters. The desired table is loaded if adequate space is available. If space is not available or the table does not exist, the appropriate return code is passed back to the calling program. This subroutine also produces a listing of the table.

3.1.5.5 PUTSTX. This subroutine writes a syntax table into the syntax file from a buffer in the calling program. The syntax table number, syntax table buffer location, text table buffer location, text table size, syntax table size, ROOT and return code are specified via parameters. The text table is placed at the end of the syntax table and all text references are updated. This new table (combination syntax and text table) is written into the syntax file and assigned the number passed as a parameter. This subroutine also produces a listing of the table.

3.1.6 Variable Descriptions

This section contains a brief description of all variables used in the syntax generation program.

ALPHA	Array, in CHRTAB - Defines the sections of the character table consisting of the alphabetic characters.
AND	Integer, in SPCHAR - Defines the special character '&'.
APFLAG	Logical Flag - Used to signal a text constant in the input buffer. Syntactical processing will continue when this flag is turned off.
APOST	Integer, in SPCHAR - Defines the special character (').
BLKHDR	Integer - Index of header of block currently being processed.
BLNK	Integer, in SPCHAR - Defines the special character ' ' (blank).
CHRTAB	Array - Used to contain the character table read during initialization.
COL	Integer, in SPCHAR - Defines the special character ':'.
COMMA	Integer, in SPCHAR - Defines the special character ','.
DIGIT	Array, in CHRTAB - Defines the section of the character table consisting of the numeric characters.
DOLLAR	Integer, in SPCHAR - Defines the special character '$'.
EQ	Integer, in SPCHAR - Defines the special character '='.
ERRNO	Integer - Contains the error number to be printed in the listing when an error occurs.
GET	Integer - Contains the number of the syntax table to be used by the syntax generation program for parsing.
GT	Integer, in SPCHAR - Defines the special character '>'.
I	Integer - Used as pointer in the input buffer to determine where the card being input is to be placed.
IK	Integer - Used as multi-purpose counter.
IM	Integer - Used as multi-purpose counter.
IN	Integer - Used as multi-purpose counter.

INCARD	Array - Used as the input buffer.
IN1	Integer - Used as integer value of a number in number look-up subroutine.
ISN	Integer - Used as return variable for computed go to statement in number look-up subroutine.
ISR	Integer - Used as return variable for computed go to statement in symbol table subroutine.
IS1	Integer - Used as variable in symbol table subroutine.
IS2	Integer - Used as variable in symbol table subroutine.
I1	Integer - Used as multi-purpose variable.
I3	Integer - Used in action routine #3 as a variable.
I4	Integer - Used in action routine #4 as a variable.
I5	Integer - Used in action routine #5 as a variable.
J	Integer - Used as a pointer in STTAB.
JE	Integer - Used as return variable for computed go to statement in action routine #8.
JJ	Integer - Used to save value of a pointer for the syntax table.
JJ8	Integer - Used as a multi-purpose variable.
J3	Integer - Used in action routine #3 as a variable.
K	Integer - Used as pointer in the input buffer.
KK	Integer - Used to save the value of K for temporary processing.
KS	Integer - Used to save the value of K for temporary processing.
K3	Integer - Used in action routine #3 as a variable.
L	Integer - Used as a pointer in STTAB.
LBS	Integer, in SPCHAR - Defines the special character '#'.
LPAR	Integer, in SPCHAR - Defines the special character '('.
LT	Integer, in SPCHAR - Defines the special character '<'.
L3	Integer - Used in action routine #3 as a variable.

M	Integer - Used as a pointer in STTAB.
MINUS	Integer, in SPCHAR - Defines the special character '-'.
N	Integer - Used as a pointer in STTAB.
NOT	Integer, in SPCHAR - Defines the special character '¬'.
OR	Integer, in SPCHAR - Defines the special character '|'.
OUTSVE	Integer - Used to save the contents of OUTTAB (STMIN) for temporary processing.
OUTTAB	Array - Used to build the new syntax table.
PERIOD	Integer, in SPCHAR - Defines the special character '.'.
PLUS	Integer, in SPCHAR - Defines the special character '+'.
PUT	Integer - Contains the number to be assigned to the new syntax table when written into the syntax file.
QUEST	Integer, in SPCHAR - Defines the special character '?'.
RC	Integer - Contains the return code passed from the action routines to the Parser. Also acts as the return code from the subroutines which use one.
RCSAVE	Integer - Contains the return code which was read off of the control card. Given to PUTSTX subroutine.
REPCNT	Integer - Used to count the number of replications of an argument.
ROOT	Integer - When used with GETSTX, contains the location of the syntax table ROOT element. When used with PUTSTX, it tells the subroutine which element of the new table is the ROOT.
RPAR	Integer, in SPCHAR - Defines the special character ')'.
SEMI	Integer, SPCHAR - Defines the special character ';'.
SFFLAG	Logical Flag - Used to signal symbol table is full so no more entries will be allowed.
SLASH	Integer, in SPCHAR - Defines the special character '/'.
SPCHAR	Array, in CHRTAB - Defines the section of the character table consisting of the special characters.
SPLAT	Integer, in SPCHAR - Defines the special character '*'.

STMAX Integer - Contains the maximum size of the output syntax table OUTTAB.

STMIN Integer - Contains the current size of the output syntax table OUTTAB.

STSAVE Array - Contains the value of STMIN when an alternative equation is entered. STMIN will be restored to this value if an invalid alternative is detected.

STTAB Array - Contains the syntax table the syntax generation program uses.

SUBNUM Integer - Contains the integer equivalent of the character number found in the input buffer by the number look-up subroutine.

SYFLAG Logical Flag - Used to signal the output syntax table OUTTAB is full. No more entries will be allowed.

SYMAX Integer - Contains the maximum size of the symbol table.

SYMIN Integer - Contains the current size of the symbol table.

SYMTAB Array - Used as the symbol table. It contains all symbols used by the program.

TABMAX Maximum size of the combination syntax and text table to be read into STTAB by GETSTX.

TRACE Integer - Used as a flag to determine if a trace of the syntax generator program activities is desired.

TXMAX Integer - Contains the maximum size of the output text table TXTAB.

TXMIN Integer - Contains the current size of the output text table TXTAB.

TXSAVE Array - Contains the value of TXMIN when an alternative equation is entered. TXMIN will be restored to this value if an invalid alternative is detected.

TXTAB Array - Used to build the new text table.

3.1.7 Syntax Table Definition

The Syntax Table generated by the syntax processor (Figure 3-1) is a combination of two tables; a Syntax Equation Table (Figure 3-2) and a Text Table (Figure 3-3).

The Syntax Equation Table (Figure 3-2) is comprised of blocks of half word integers. Each block contains a header, a variable length list of sequential or alternative entries, and a uniquely recognizable marker.

The header of each block consists of six half words. The only half word set when the table is built is the second one (Type of Block). The other five half words are used when the table is processed.

Each entry following the header is comprised of two half words. The first half word is the type of entry. The second half word is the corresponding argument for the type of entry. A description of the different type codes and their arguments is shown in (Figure 3-4).

The uniquely recognizable marker (-1) is used to terminate the block.

The Text Table (Figure 3-3) is comprised of text constants encountered during processing. Each entry is variable in length. The first half word is the length of the text constant. The half words following the length are the text constant.

When a set of syntax equations has been processed, the subroutine PUTSTX is called. This subroutine will combine the two tables by placing the Text Table at the end of the Syntax Equation Table and updating all references to text constants in the Syntax Equation Table.

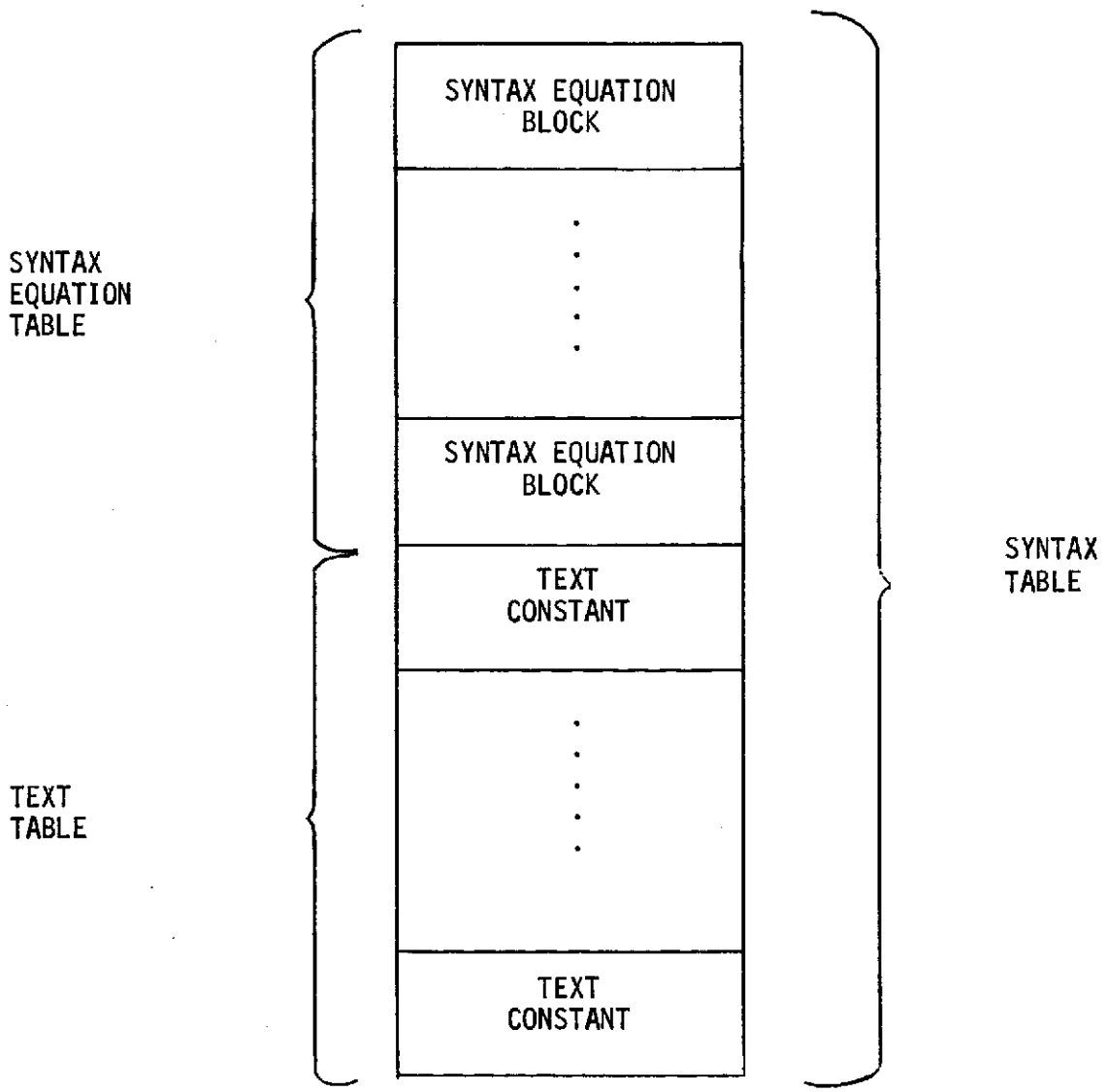


Figure 3-1. SYNTAX TABLE

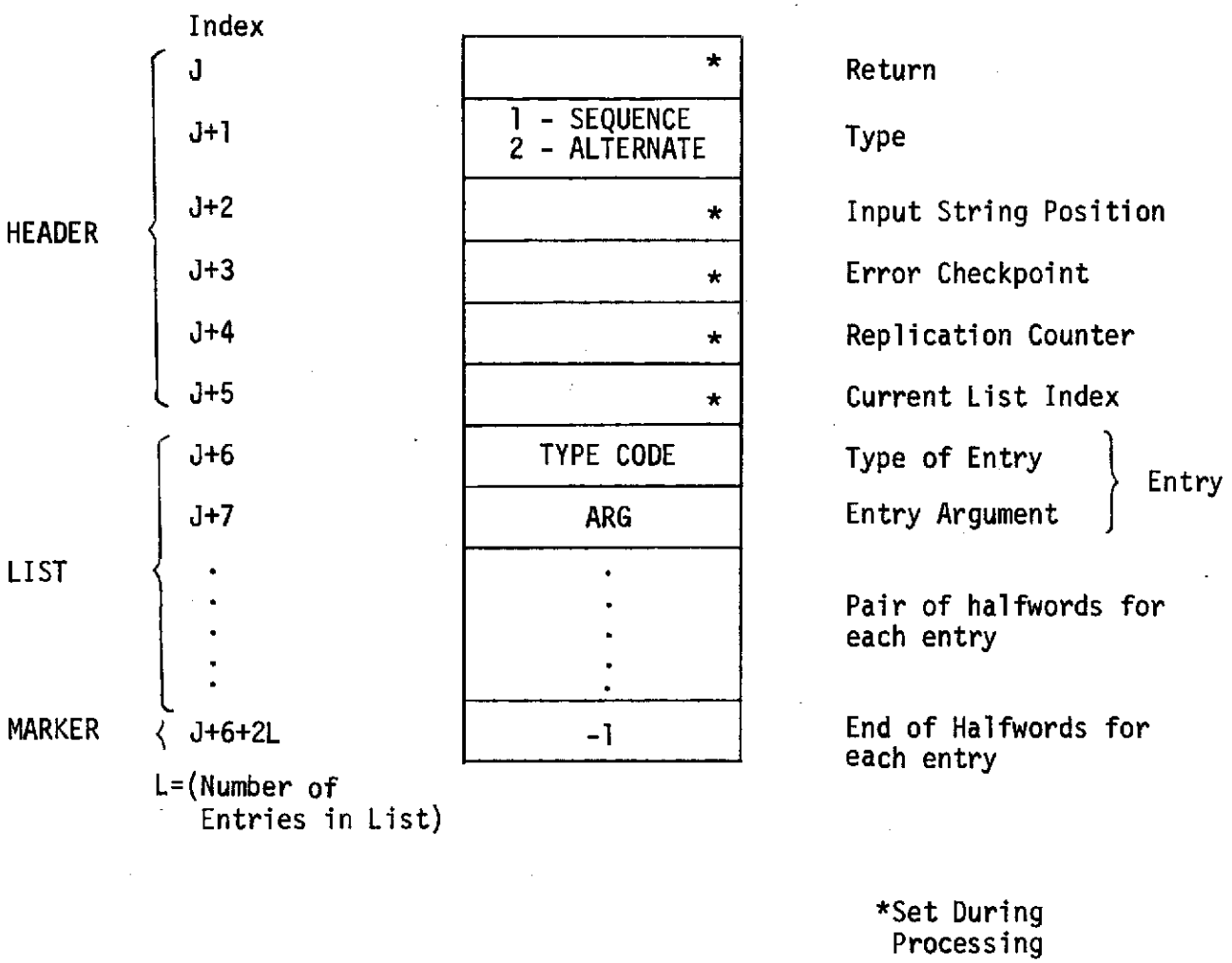


Figure 3-2. SYNTAX EQUATION TABLE

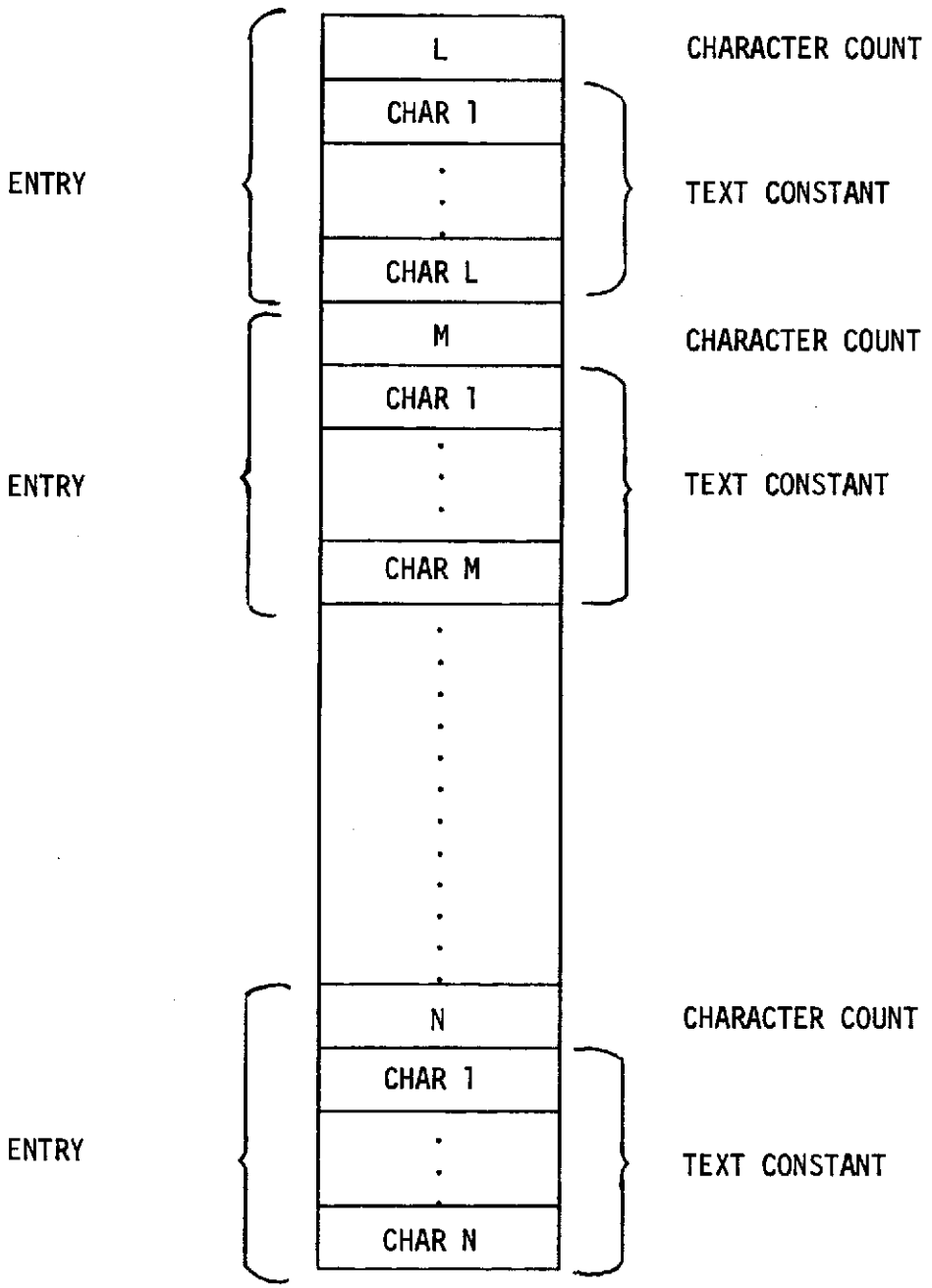


Figure 3-3. TEXT TABLE

TYPE OF ELEMENT	ELEMENT IN SYNTAX EQUATION	TYPE CODE	ARGUMENT
REFERENCE	<REF>	1	LOCATION IN SYNTAX TABLE WHERE REFERENCED BLOCK BEGINS.
	<REF>?	2	
	<REF>+N	3	
	<REF>*N	4	
TEXT CONSTANT	'TEXT'	5	LOCATION IN SYNTAX TABLE WHERE LENGTH OF TEXT CONSTANT IS LOCATED. TEXT CONSTANT FOLLOWS LENGTH.
	'TEXT'?	6	
	'TEXT'+N	7	
	'TEXT'*N	8	
ACTION ROUTINE NUMBER	#9999	9	ACTION ROUTINE NUMBER
	#9999?	10	
	#9999+N	11	
	#9999*N	12	
ERROR NUMBER	$999	13	ERROR NUMBER

NOTE - N is optional. If N is present, the type code is (TYPE CODE) X (1024) + N. This procedure places the TYPE CODE in the upper half of the HALFWORD and N in the lower half.

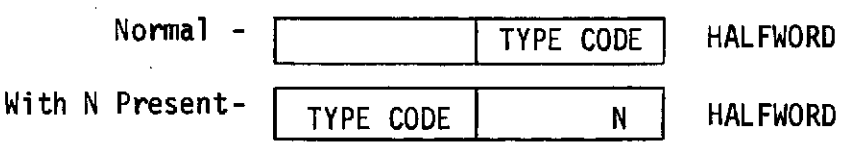


Figure 3-4. ENTRY TYPE CODES AND ARGUMENTS

3.1.8 Diagnostics

When an error in a syntax equation is encountered during processing, the syntax table generator outputs a message containing an error number. The following is a list of these error numbers and a brief description of the error.

ERROR NUMBER	ERROR DESCRIPTION
1	GETSTX routine called and return code received from routine was not ZERO.
2	Statement being processed exceeds 20 cards in length. 20 cards have been read and no semi-colon was found.
3	Syntax table full. An addition to the output syntax table cannot be made because it would require altering locations beyond the end of the table.
4	Illegal name. Program expected '<' symbol but some other symbol was found while looking for name of syntax element.
5	Multi-defined symbol. Symbol used as name to define syntax elements has been used previously to define another block of elements.
6	Symbol table full. An addition to the symbol table canot be made because it would require altering locations beyond the end of the table.
7	Symbol too long. Symbol being processed has more than 32 characters (including blanks) between '<' symbol and '>' symbol.
8	No characters in symbol. Symbol being processed has no characters between '<' symbol and '>' symbol.
9	Text table full. An addition to the text table cannot be made because it would require altering locations beyond the end of the table.
10	Null text string. Text string being processed has no characters (text) between beginning symbol (') and ending symbol (').

ERROR NUMBER	ERROR DESCRIPTION
11	Illegal subroutine number. A number other than one from 1 through 9999 has been designated as a subroutine number.
12	Illegal error (diagnostic) number. A number other than one from 1 through 999 has been designated as an error (diagnostic) number.
13	PUTSTX routine called and return code received from routine was not zero.
14	Two roots found. A search through the symbol table for the root symbol has produced more than one unreferenced symbol. Therefore, the correct root symbol cannot be determined.
15	No root found. A search through the symbol table for the root symbol has produced no unreferenced symbols, therefore, a root symbol cannot be determined.
500	Illegal statement. The statement being parsed is not recognized as a syntax equation.

3.2 COMPILER

3.2.1 Mainline Programs

The Mainline Programs contained within the compiler are identified and described in the following pages.

GOAL COMPILER ROUTINE

NAME – ACTERR

FUNCTION – This routine is called if an invalid ACTION ROUTINE number is encountered during parsing. A message is given and the run terminated.

CALLED BY – ANY ACTION ROUTINE

SUBROUTINES CALLED – SYSERR, (FORTRAN I/O)

DESCRIPTION – The error message indicates the ACTION ROUTINE overlay number and action number. RETURN is made through SYSERR to terminate the run.

GOAL COMPILER ROUTINE

NAME – ACTION

FUNCTION – This routine performs 'resident' action functions and invokes overlay action routines to support parsing of specific statement types.

CALLED BY – PARSER

SUBROUTINES CALLED – ACTERR, NEXTCR, ERROR, LOOKUP, TXTOUT, INPUT, SYSERR, and SUB01 ... SUB52

DESCRIPTION – The action routine codes are validated and control is given to the appropriate routine. Except for SYSERR and ACTERR, return is always made to the calling program, (PARSER).

The functions of the 'resident' action routines are described separately, even though the codes are physically located in the routine, ACTION.

GOAL COMPILER ROUTINE

NAME – BLDXRF

FUNCTION – This routine is called once, after all GOAL statements have been parsed, to prepare for generation of the cross-reference listings.

CALLED BY – LBLXRF, SYMXRF, or FDXREF

SUBROUTINES CALLED – SYSERR, (FORTRAN I/O)

DESCRIPTION – This routine uses the combined areas of the STXTAB and STMTAB to build XRFTAB which contains a tabulation of all references to symbolic names entered in SYMTAB. This is done by scanning the XREF file and logging all statement numbers which reference each symbolic name. If the size of XRFTAB is exceeded SYSERR is called.

GOAL COMPILER ROUTINE

NAME – DIAGSM

FUNCTION – This routine is called to generate the GOAL diagnostic summary listing.

CALLED BY – MAIN

SUBROUTINES CALLED – RCRETN, (FORTRAN I/O)

DESCRIPTION – The error file and symbol table are scanned to generate the diagnostic summary report. If any errors were detected during the compilation, the full diagnostic summary is generated. If no errors were detected, a small summary message is printed. If any errors had occurred return is made via RCRETN to cancel subsequent translation job steps.

GOAL COMPILER ROUTINE

NAME – ERROR

FUNCTION – This routine is called to log errors detected during compilation in the error file.

CALLED BY – PARSER, any action routine that can detect a recoverable error.

SUBROUTINES CALLED – (FORTRAN I/O)

DESCRIPTION – When this routine is called the error count is incremented by one and a record is written in the error file. This record contains:

1. ERROR type
2. Position in statement
3. Statement number
4. Source record number

Return is made to the calling program.

GOAL COMPILER ROUTINE

NAME – FDLKUP

FUNCTION – This routine verifies function designators used in the GOAL program. It also verifies that a macro exists in the Data Bank(s) being used.

CALLED BY – Most 'ACTION' routines that process function designators and SUB21 (Macro Processing Routine)

SUBROUTINES CALLED – LOOKUP, YEFIND

DESCRIPTION – When this routine is called the following parameters are provided:

1. Function designator name or Macro Label
2. Return parms for type, address, O.K. flag

If the GOAL compiler is in the 'subroutine' mode, a search is made to check if the function designator is a parameter. If not, YEFIND is called to search each data bank currently in use. In all cases, the function designator type and address are returned to the calling program. If not found the type is set to zero. LOOKUP is called to log all function designator references in the symbol table. If this routine is used to verify a macro label, then none of the above actions is taken. The macro label is verified and the proper return code is set.

GOAL COMPILER ROUTINE

NAME – FDXREF

FUNCTION – This routine is called to generate the GOAL function designator cross-reference listing.

CALLED BY – MAIN

SUBROUTINES CALLED – BLDXRF, (FORTRAN I/O)

DESCRIPTION – BLDXRF is called, if required, to build XRFTAB. XRFTAB is then scanned to generate the GOAL function designator cross-reference listing. Undefined function designator names are flagged. A summary of data banks used during the GOAL compilation is provided. Return is made to the calling program.

GOAL COMPILER ROUTINE

NAME – FIXUP

FUNCTION – This routine is called when the parser encounters an invalid GOAL statement. Pointers, etc., are updated to continue compilation.

CALLED BY – MAIN

SUBROUTINES CALLED – NEXTCR

DESCRIPTION – When this routine is entered the statement buffer pointer K, is positioned to some unpredictable character in the current GOAL statement. NEXTCR is then called, as many times as required, to position the pointer past the terminal ';'. The symbol table is then purged of any erroneous definitions entered in parsing the statement. Return is always made to the calling program.

GOAL COMPILER ROUTINE

NAME – GINIT

FUNCTION – This routine initializes common data values and loads the character table, CHRTAB, and the Syntax table, STXTAB

CALLED BY – MAIN

SUBROUTINES CALLED – *GETCHR, *GETSTX, SYSERR, (FORTRAN I/O).

(*) - These are part of GINIT

DESCRIPTION – A control card is read which identifies the syntax table to be loaded. This card may also contain up to 5 patches to the table. Max limits and initial values are then set for COMMON DATA. GETCHR is called to load CHRTAB. GETSTX is called to load STXTAB. If the requested table cannot be loaded, SYSERR is called to terminate the run. Data Bank directories MBLOCK and DBLOCK are then loaded. Any specified patches to STXTAB are made. GINIT then returns to the calling program.

GOAL COMPILER ROUTINE

NAME – EXLIST

FUNCTION – This routine is called to write a record in the GOAL compiler expanded source record listing.

CALLED BY – MAIN

SUBROUTINES CALLED – SYSERR, (FORTRAN I/O)

DESCRIPTION – This routine is called from MAIN for each GOAL statement processed by the compiler. If the expanded listing option is not selected no output is produced. MACRO generated statements are not listed unless the 'expand' option is specified. Page and line counts are maintained. Statement data is contained in the STMTAB portion of the common DATA area. This data is formatted to generate the expanded source listing. Error '*'s are inserted as required. This routine is also used to write MACRO 'body' statements into the MACRO FILE, subroutine records in the subroutine file, and card images in the source deck output file. If any of the file maximums are exceeded, SYSERR is called.

GOAL COMPILER ROUTINE

NAME – INPUT

FUNCTION – This routine provides additional data to the input buffer, STMTAB, each time it is called. This data may come from either the input stream or MACRO file, Data Bank or Subroutine file.

CALLED BY – MAIN, NEXTCR, RESET, or any action routine which processes the input stream directly.

SUBROUTINES CALLED – SYSERR, SRLIST, LOOKUP, (FORTRAN I/O).

DESCRIPTION – In normal operation an input record (card) is read each time this routine is called. In the MACRO mode this record is obtained from the MACRO file or Data Bank. In the subroutine mode this record is obtained from the subroutine file. If no data is available ERROR is called and the run is terminated. SRLIST is called to list all records read from the input stream in the normal mode. These records are then scanned and any abbreviations are expanded. STMTAB pointers are updated as required. STMTAB is rolled in and out when switching modes to avoid loss of data. Normal returns are made to the calling program.

GOAL COMPILER ROUTINE

NAME – LBLXRF

FUNCTION – This routine is called to generate the GOAL statement Label cross-reference listing.

CALLED BY – MAIN

SUBROUTINES CALLED – BLDXRF, (FORTRAN I/O)

DESCRIPTION – BLDXRF is called, if required, to build XRFTAB. XRFTAB is then scanned and GOAL statement Label cross-reference listing is generated. Undefined and unreferenced labels are flagged. Return is made to the calling program.

GOAL COMPILER ROUTINE

NAME – LOOKUP

FUNCTION – This routine will define and/or verify symbolic names in the symbol table, SYMTAB.

CALLED BY – Most GOAL compiler routines.

SUBROUTINES CALLED – SYSERR

DESCRIPTION – Each time this routine is used the calling program provides the following parameters:

1. Symbolic name
2. Type
3. Option
4. Flag

'Option' indicates define or verify
'type' indicates label, symbol, etc. (Chain No.)
'flag' is set to 0 = OK, -1 = Not found or duplicate

The symbolic names are stored in a variable length format. The entries are chained according to type and entries on each chain are in collating sequence. SYSERR is called if SYMTAB maximum is exceeded.

GOAL COMPILER ROUTINE

NAME – MAIN

FUNCTION – Initial entry point of GOAL Compiler. Provides mainline sequencing of principal compiler functions. Controls re-initialization and compilation of embedded subroutines.

CALLED BY – Operating System for each GOAL compilation.

SUBROUTINES CALLED – SVSAVE, GINIT, INPUT, PREP, PARSER, FIXUP, EXLIST, RESET, SYMXRF, LBLXRF, FDXREE, DIAGSM, (FORTRAN I/O).

DESCRIPTION – Common data areas and direct access I/O files are defined. Data initialization is performed and the INPUT Buffer is primed. Stmt No. 10 is the start of the loop used to process each GOAL statement. If the statement is a comment it is listed on a separate line. The PARSER is called to process each GOAL statement at Stmt No. 40. If RC is set Non-Zero by Parser, FIXUP is called to find ';'. EXLIST is then called to list statement. If ENDFLG is set, the loop is finished and summary listings are generated at Stmt No. 60. The symbol table is written out to the symbol table file for use by the Translator. If ENDFLG is not set, RESET is called to update pointers in the INPUT Buffer and the loop is continued at Stmt No. 10. DIAGSM is always called before exit from MAIN.

MAIN also controls looping for embedded subroutine processing. If any subroutines were embedded in a GOAL program, the compiler is executed again to compile these separately.

GOAL COMPILER ROUTINE

NAME – NEXTCR

FUNCTION – This routine scans the input buffer to find the next significant character. Blanks and comments are ignored.

CALLED BY – Most of the 'ACTION' routines that process GOAL statements or elements.

SUBROUTINES CALLED – INPUT

DESCRIPTION – Each time this routine is called the input buffer is scanned for a significant character. The pointer, K, is advanced accordingly. Blanks and comments are ignored. If K is advanced past the currently loaded portion of STMTAB, INPUT is called to obtain additional data.

GOAL COMPILER ROUTINE

NAME – PARSER

FUNCTION – This routine controls the parsing of GOAL statements according to the SYNTAX table, STXTAB.

CALLED BY – MAIN

SUBROUTINES CALLED – INPUT, ACTION, NEXTCR, ERROR

DESCRIPTION – When this routine is entered, K points to the expected beginning of each GOAL statement. The statement is then analyzed according to the syntax rules contained in STXTAB. INPUT is called to provide additional data when the SCAN exceeds the currently loaded portion of STMTAB. ACTION is called as required according to STXTAB. If the parse fails, ERROR is called to log the diagnostic message. NEXTCR is called prior to this to position the error pointer. RC is set to: 0 = Good Stmt, -1 = error. Return is then made to the calling program.

GOAL COMPILER ROUTINE

NAME – PREP

FUNCTION – This routine initializes the input buffer before parsing each statement.

CALLED BY – MAIN

SUBROUTINES CALLED – NEXTCR, SYSERR

DESCRIPTION – The input buffer pointer, K, is set to 1. NEXTCR is then called to find the first significant position in the statement. This index is saved as STMTK. If a comment precedes this position in the record a new block is created for it so that it will be printed on a separate line in the expanded listing. If the block count maximum is thus exceeded, SYSERR is called to terminate the run. Otherwise return is made to the calling program.

GOAL COMPILER ROUTINE

NAME – RCRETN

FUNCTION – This routine is called to return control to the operating system.

CALLED BY – DIAGSM, SYSERR

SUBROUTINES CALLED – (None)

DESCRIPTION – This routine returns control to the operating system along with a condition code which is obtained as a parameter from the routine calling RCRETN. This parameter is used to cancel the execution of subsequent GOAL TRANSLATOR job steps.

GOAL COMPILER ROUTINE

NAME – RESET

FUNCTION – This routine is called after each GOAL statement has been listed to delete it from the input statement buffer

CALLED BY – MAIN

SUBROUTINES CALLED – INPUT

DESCRIPTION – The contents of the statement buffer following the terminal ';' of the current statement are examined. If these are blank the pointers are set to initial positions and INPUT is called to prime the buffer. Otherwise, the remaining contents are moved up in the statement buffer and the buffer pointers are adjusted for this data. Return is always made to the calling program.

GOAL COMPILER ROUTINE

NAME – SRLIST

FUNCTION – This routine is called to write a record in the GOAL source listing file.

CALLED BY – INPUT

SUBROUTINES CALLED – (FORTRAN I/O)

DESCRIPTION – This routine is called by INPUT each time a record is read from the compiler input stream. The contents of this record are stored in the common data area. This routine formats this data to generate the GOAL compiler source record listing. Page and line counts are maintained. The source record data is not changed in any way. The SRFLG control word is tested before each output record is written. If this compiler option is not selected no action is taken. Return is always made to the calling program.

GOAL COMPILER ROUTINE

NAME – SYMXRF

FUNCTION – This routine is called to generate the GOAL Internal name cross-reference listing

CALLED BY – MAIN

SUBROUTINES CALLED – BLDXRF, (FORTRAN I/O)

DESCRIPTION – BLDXRF is called, if required, to build XRFTAB. XRFTAB is then scanned to generate the GOAL internal name cross-reference listing. Undefined and unreferenced names are flagged. Return is made to the calling program.

GOAL COMPILER ROUTINE

NAME – SYSERR

FUNCTION – This routine is called to terminate the compilation in the event of a 'system' type error condition.

CALLED BY – Any GOAL routine

SUBROUTINES CALLED – RCRETN, (FORTRAN I/O)

DESCRIPTION – An error message is given to indicate the error type which is passed as a parameter from the routine calling SYSERR. Return is made via RCRETN to cancel subsequent TRANSLATOR job steps.

GOAL COMPILER ROUTINE

NAME – TXTOUT

FUNCTION – This routine is used to write a record in the GOAL compiler Intermediate Text output file.

CALLED BY – All action routines that generate intermediate text.

SUBROUTINES CALLED – SYSERR

DESCRIPTION – The Intermediate Text buffer is contained in a common data area. A word count is also provided. When this routine is called a variable length record is written in the output file. If the word count is not $0<count\leq406$ SYSERR is called to terminate the run.

Each record contains a standard 6 word header. This header contains a record count which is incremented each time a record is written.

3.2.2 SUBXX Action Routines

Table 3-2 is a listing of the Action Routines used in the compiler. Definitive information relating to these routines is provided by the ensuing pages.

Table 3-2

Subroutine Listing

SUB01	ACTIVATE TABLE
SUB02	APPLY ANALOG
SUB03	ASSIGN
SUB04	DECLARE TEXT TABLE
SUB05	BEGIN MACRO
SUB06	BEGIN PROGRAM
SUB07	BEGIN SUBROUTINE
SUB08	CONCURRENT
SUB09	DECLARE DATA
SUB10	DECLARE NUMERIC LIST
SUB11	DECLARE NUMERIC TABLE
SUB12	DECLARE QUANTITY LIST
SUB13	DECLARE QUANTITY TABLE
SUB14	DECLARE STATE LIST
SUB15	DECLARE STATE TABLE
SUB16	DECLARE TEXT LIST
SUB17	EXTERNAL DESIGNATOR
SUB18	DELAY
SUB19	DISABLE INTERRUPT
SUB20	END
SUB21	EXPAND/EXECUTE MACRO
SUB22	FREE DATA BANK
SUB23	GØ TØ
SUB24	INHIBIT TABLE
SUB25	ISSUE DIGITAL PATTERN
SUB26	LEAVE
SUB27	LET EQUAL
SUB28	NOT USED
SUB29	PERFORM PROGRAM/SUBROUTINE
SUB30	READ
SUB31	AVERAGE
SUB31	RECORD DATA
SUB33	RELEASE CONCURRENT
SUB34	REPEAT
SUB35	REPLACE

Table 3-2

(Continued)

SUB36	REQUEST KEYBOARD
SUB37	RESUME
SUB38	SET DISCRETE
SUB39	NOT USED
SUB40	STOP
SUB41	TERMINATE
SUB42	USE DATA BANK
SUB43	<FD> CHAIN GENERATOR
SUB44	COMPILER DIRECTIVES
SUB45	WHEN INTERRUPT
SUB46	NOT USED
SUB47	NOT USED
SUB48	PREFIX PROCESSOR
SUB49	INTERNAL NAME
SUB50	NOT USED
SUB51	SHORT FORM PROCESSOR
SUB52	DATA BANK PROCESSOR

GOAL COMPILER ROUTINE

NAME – SUB01

FUNCTION – This routine supports compilation of the 'ACTIVATE TABLE' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, TXTOUT, FDLKUP

DESCRIPTION – Options 1...7 supported.

#101 - No Action

#102 - Verify, save table name

#103 - No Action

#104 - Write type 30 TXT record

#105 - Verify, index name, write type 31 TXT record

#106 - Verify row No., write type 31 TXT record

#107 - Verify F. D.,write type 31 TXT

GOAL COMPILER ROUTINE

NAME – SUB02

FUNCTION – This routine supports compilation of the 'APPLY ANALOG' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Options 1...8 supported.

#201 - Set INNMCT = 0

#202 - Save 1st External Designator (Present Value)

#203 - Save 2nd External Designator (Present Value)

#204 - Write type 4 TXT record

#205 - Verify, save internal name

#206 - Verify, save External Designator

#207 - Write type 42, 43 TXT record

#208 - Write type 43 TXT record

GOAL COMPILER ROUTINE

NAME – SUB03

FUNCTION – This routine supports compilation of the 'ASSIGN' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Options 1...5 supported.

#301 - No Action

#302 - Verify, save internal name (1st)

#303 - Write type 38 TXT record

#304 - Verify, save internal name (2nd)

#305 - Save 'STATE'

GOAL COMPILER ROUTINE

NAME – SUB04

FUNCTION – This routine supports compilation of the 'DECLARE TEXT TABLE' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT, LOOKUP, FDLKUP, (FORTRAN I/O)

DESCRIPTION – Options 1...11 Supported.

#401 - Initialize flags, counters, and pointers

#402 - Write type 62, 63 TXT records

#403 - Verify/Save Table Name

#404 - Verify/Save number of rows integer

#405 - Verify/Save number of columns integer

#406 - Verify/Save column names

#407 - Verify/Save row Function Designators

#408 - Verify entries per row does not exceed the number of columns

#409 - Verify/Save text constants

#410 - Save maximum number of characters integer

#411 - Verify entries per row is not less than the number of columns

GOAL COMPILER ROUTINE

NAME – SUB05

FUNCTION – This routine supports compilation of the 'BEGIN MACRO' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, NEXTCR, LOOKUP, FORTRAN I/O, INPUT, ERROR

DESCRIPTION – Options 1...3 supported.

#501 - Set MACFLG =2, verify, save macro name, parameters

#502 - Process macro definition

#503 - Set MACFLG = 0, return to normal parsing procedure

GOAL COMPILER ROUTINE

NAME – SUB06

FUNCTION – This routine supports compilation of the 'BEGIN PROGRAM' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Options 1...4 supported

#601 - Reset compiler pointers

#602 - Save program name

#603 - Save program revision label

#604 - Write type 28 TXT record, then type 6 for ON, OFF constants and type 18 for DISPLAY, PRINT, RECORD Function Designators

GOAL COMPILER ROUTINE

NAME – SUB07

FUNCTION – This routine supports compilation of the 'BEGIN SUBROUTINE' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, TXTOUT, NEXTCR

DESCRIPTION – Options 1...14 supported

#701 - Reset compiler pointers

#702 - Save subroutine name

#703 - Verify, save 'NAME' parameter

#704 - Write type 61 TXT record then type 6 for ON, OFF constants then type 18 for PRINT, DISPLAY, RECORD Function Designators.

#705 - Count, save parameters

#706 - Verify, save F.D. parameter

#707 - Verify, save F.D.

#708...#711 - Save STYPE

#712 - Find ';'

#713 - Check for No procedural statements

#714 - Set STYPE = 5

GOAL COMPILER ROUTINE

NAME – SUB08

FUNCTION – This routine supports compilation of the 'CONCURRENT' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Option 1 supported

#801 - Write type 36 TXT records

GOAL COMPILER ROUTINE

NAME – SUB09

FUNCTION – This routine supports compilation of the 'DECLARE DATA' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT, LOOKUP

DESCRIPTION – Options 1...20 supported.

#901 - TD = 0

#902, #903 - No Action

#904 - Verify, save NAME (DECLARE NUMBER)

#905 - Write type 2 TXT record (DECLARE NUMBER)

#906 - Set flag for initial values list

#907 - No Action

#908 - Verify, save NAME (DECLARE QUALITY)

#909 - Write type 3 TXT record (DECLARE QUANTITY)

#910 - Set flag for initial values list

#911 - No Action

#912 - Verify, save NAME (DECLARE STATE)

GOAL COMPILER ROUTINE

NAME – SUB09 (continued)

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION –

#913 - Write type 6 TXT record (DECLARE STATE)

#914 - Set flag for initial values list

#915 - No Action

#916 - Verify, save NAME (DECLARE TEXT)

#917 - Write type 7 or 8 TXT record (DECLARE TEXT)

#918 - Move Number, set length

#919 - Check for subroutine parameter

#920 - Set flag for initial values list

GOAL COMPILER ROUTINE

NAME – SUB10

FUNCTION – This routine supports compilation of the 'DECLARE NUMERIC LIST' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, TXTOUT

DESCRIPTION – Options 1...8 supported.

#1001 - Initialize counters

#1002 - Verify, save list name

#1003 - Verify No. of entries

#1004 - Write type 9 TXT record

#1005 - Verify, save initialization list

#1006 - Verify, save initialization list

#1007 - No Action

#1008 - Count initialization entries, commas

GOAL COMPILER ROUTINE

NAME – SUB11

FUNCTION – This routine supports compilation of the 'DECLARE NUMERIC TABLE' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, FDLKUP, TXTOUT

DESCRIPTION – Options 1...10 supported.

#1101 - Initialize counters

#1102 - Verify, save table name

#1103 - Verify, save No. columns

#1104 - Verify, save No. rows

#1105 - INVALID (ACTERR)

#1106 - Verify F.D., write type 19 TXT record

#1107 - Write type 17, 18, or 19 TXT records

#1108 - Process column titles

#1109 - Check comma count

#1110 - Save initialization value

GOAL COMPILER ROUTINE

NAME – SUB12

FUNCTION – This routine supports compilation of the 'DECLARE QUANTITY LIST' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, TXTOUT

DESCRIPTION – Options 1...7 supported.

#1201 - Initialize flags/pointers

#1202 - Verify, save LIST name

#1203 - Check, save No. of entries

#1204 - Write type 11, 12 TXT record

#1205 - Save initialization data

#1206 - Count entries

#1207 - Initialize list entry

GOAL COMPILER ROUTINE

NAME – SUB13

FUNCTION – This routine supports compilation of the 'DECLARE QUANTITY TABLE' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, FDLKUP, TXTOUT

DESCRIPTION – Options 1...10 supported.

#1301 - Initialize flags and counters

#1302 - Verify, save table name

#1303 - Verify, save No. columns

#1304 - Verify, save No. rows

#1305 - INVALID (ACTERR)

#1306 - Verify row function designator

#1307 - Write type 17 TXT record

#1308 - Verify, save column titles

#1309 - Verify, save initialization values

#1310 - Verify No. of entries

GOAL COMPILER ROUTINE

NAME – SUB14

FUNCTION – This routine supports compilation of the 'DECLARE STATE LIST' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, TXTOUT

DESCRIPTION – Options 1...8 supported.

#1401 - Initialize flags, counters

#1402 - Verify, save list NAME

#1403 - Verify, save No. of entries

#1404 - No Action

#1405 - Save initialization values

#1406 - Write type 13, 14 record

#1407 - Count entries in list

#1408 - Verify list length

GOAL COMPILER ROUTINE

NAME – SUB15

FUNCTION – This routine supports compilation of the 'DECLARE STATE TABLE' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, FDLKUP, TXTOUT

DESCRIPTION – Options 1...10 supported.

#1501 - Initialize flags, counters

#1502 - Verify, save table name

#1503 - Verify, save No. columns in table

#1504 - Verify, save No. rows in table

#1505 - INVALID (ACTERR)

#1506 - Verify, save Row Function Designator

#1507 - Write type 22, 23, 18 TXT records

#1508 - Verify, save column NAMES

#1509 - Save initialization states

#1510 - Verify No. entries in table

GOAL COMPILER ROUTINE

NAME – SUB16

FUNCTION – This routine supports compilation of the 'DECLARE TEXT LIST' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, TXTOUT, (FORTRAN I/O)

DESCRIPTION – Options 1...8 supported.

#1601 - Initialize flags and counters

#1602 - Verify/save List name

#1603 - Verify/save No. entries in list

#1604 - Write type 15, 16 TXT records

#1605 - Save initialization data

#1606 - Verify size of Initialization data

#1607 - Count entries in list

#1608 - Verify No. entries in list

GOAL COMPILER ROUTINE

NAME – SUB17

FUNCTION – This routine supports compilation of 'EXTERNAL DESIGNATOR'.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, FDLKUP, TXTOUT, LOOKUP

DESCRIPTION – Options 1...5 supported.

#1701 - Initialize flags and counters

#1702 - Verify, save 1st function designator

#1703 - Verify, save remaining function designators

#1704 - Write type 18 TXT record

#1705 - Verify 'TABLENAME FUNCTIONS', save

GOAL COMPILER ROUTINE

NAME – SUB18

FUNCTION – This routine supports compilation of the 'DELAY' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Options 1...6 supported.

#1801 - Initialize flags and counters

#1802 - Write type 53 TXT record

#1803 - Save 'TIME' value

#1804 - Provide for 'COMPARISON TEST'

#1805 - No Action

#1806 - Initialize flags for comparison test

GOAL COMPILER ROUTINE

NAME – SUB19

FUNCTION – This routine supports compilation of the 'DISABLE INTERRUPT' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, TXTOUT

DESCRIPTION – Options 1...4 supported.

#1901 - Initialize flags

#1902 - No Action

#1903 - Save disable step number

#1904 - Write type 64 TXT records

GOAL COMPILER ROUTINE

NAME – SUB20

FUNCTION – This routine supports compilation of the 'END' statement. (Program and subroutine only.)

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT, (FORTRAN I/O)

DESCRIPTION – Options 1...6 supported.

#2001 - No Action

#2002 - No Action

#2003 - No Action

#2004 - Write type 29 TXT record

- Close file #17

- Set end flag

#2005 - Same as #2004 for subroutines

#2006 - No Action

GOAL COMPILER ROUTINE

NAME – SUB21

FUNCTION – This routine supports compilation of the 'EXPAND MACRO' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, NEXTCR, LOOKUP, INPUT, SYSERR, (FORTRAN I/O), RESET, ERROR

DESCRIPTION – Options 1...4 supported.

#2101 - Process 'BEGIN MACRO' statement

#2102 - Set 'EXPAND ONLY FLAG'

#2103 - Set 'EXECUTE ONLY FLAG'

#2104 - Perform MACRO parameter substitutions

GOAL COMPILER ROUTINE

NAME – SUB22

FUNCTION – This routine supports compilation of the 'FREE DATA BANK' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP

DESCRIPTION – Options 1...4 supported.

#2201 - Initialize flags and counters

#2202 - Save 'DATA BANK NAME'

#2203 - Verify 'DATA BANK NAME' and Revision Label, delete from use list

#2204 - Save Revision Label

GOAL COMPILER ROUTINE

NAME – SUB23

FUNCTION – This routine supports compilation of the 'GO TO' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, TXTOUT

DESCRIPTION – Options 2...3 supported.

#2301 - Set 'GO TO FLAG' for testing that next STMT is labeled.

#2302 - Verify stmt label, save in TXT

#2303 - Write type 27 TXT record

GOAL COMPILER ROUTINE

NAME – SUB24

FUNCTION – This routine supports compilation of the 'INHIBIT TABLE' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, TXTOUT, FDLKUP

DESCRIPTION – Options 1...7 supported.

#2401 - No Action

#2402 - Verify/save table NAME

#2403 - No Action

#2404 - Write type 32 TXT record

#2405 - Verify index, write type 33 TXT record

#2406 - Verify ROW No., write type 33 TXT record

#2407 - Verify ROW F.D., write type 33 TXT record

GOAL COMPILER ROUTINE

NAME – SUB25

FUNCTION – This routine supports compilation of the 'ISSUE DIGITAL PATTERN' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Options 1...8 supported.

#2501 - Initialize flags/counters

#2502 - Save 1st External/designator (present value)

#2503 - Save 2nd External/designator (present value)

#2504 - Number pattern constant - write type 2 TXT record

#2505 - Internal name - save

#2506 - External Designator for NON 'PRESENT VALUE' types

#2507 - Write TXT record

#2508 - Save External Designator

GOAL COMPILER ROUTINE

NAME – SUB26

FUNCTION – This routine supports compilation of the 'LEAVE' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT, INPUT, NEXTCR, RESET

DESCRIPTION – Options 1...10 supported.

#2601 - Initialize flags, check for subroutine compilation

#2602 - Write type 4 TXT record (Quantity value)

#2603 - Write type 2 TXT record (Number value)

#2604 - Write type 2 TXT record (Number Pattern)

#2605 - Save Self Defining State Parameter

#2606 - Write type 8 TXT record (Text Constant)

#2607 - Save internal name parameter

#2608 - Write Type 66 TXT record (with Parameters)

#2609 - Write Type 66 TXT record (without Parameters)

#2610 - Purge all data in the input stream until the word RESUME is encountered.

GOAL COMPILER ROUTINE

NAME – SUB27

FUNCTION – This routine supports compilation of the 'LET EQUAL' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Options 1...15 supported.

#2701 - Initialize flags/counters

#2702 - Verify, save internal name (on left of '=')

#2703 - Check parenthesis count, write type 60 TXT record

#2704 - Save '+'

#2705 - Save '-'

#2706 - Verify, save internal name in expression

#2707 - Save operator type

#2708 - Write type 4 TXT for self-defining quantity

#2709 - Write type 2 TXT for self-defining number

GOAL COMPILER ROUTINE

NAME – SUB27 (continued)

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION –

#2710 - Count, save '('

#2711 - Count, save ')'

#2712 - Operator type = 1

#2713 - Operator type increment by 1

#2714 - Namecount = 0

#2715 - Check Name count - must be 1

GOAL COMPILER ROUTINE

NAME – SUB29

FUNCTION – This routine supports compilation of the 'PERFORM SUBROUTINE' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT, FDLKUP

DESCRIPTION – Options 1...16 supported

#2901 - Initialize flags/counters

#2902 - Save 'PROGRAM NAME'

#2903 - Write type 34/35 TXT record

#2904 - No Action

#2905 - No Action

#2906 - Self-defining Number Pattern Parm

#2907 - Self-defining Number Parm

#2908 - Self-defining Quantity Parm

#2909 - Self-defining State Parm

GOAL COMPILER ROUTINE

NAME – SUB29

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION – #2910 - Self-defining Text Parm - Write Type 8 TXT record

#2911 - Function designator Parm - Write Type 8 TXT record

#2912 - Internal Name Parm

#2913 - Write type 34 TXT record

#2914 - Same as #2902

#2915 - Write type 59 TXT record

#2916 - Save Revision Label

GOAL COMPILER ROUTINE

NAME – SUB30

FUNCTION – This routine supports compilation of the 'READ' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Options 1...3 supported.

#3001 - Verify External designator, save

#3002 - Verify/save internal name

#3003 - Write type 47 TXT record

GOAL COMPILER ROUTINE

NAME – SUB31

FUNCTION – This routine supports compilation of the 'AVERAGE' statement

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Options 1...4 supported.

#3101 - Save Nbr readings

#3102 - Verify External designator

#3103 - Verify Internal name

#3104 - Write type 48 TXT record

GOAL COMPILER ROUTINE

NAME – SUB32

FUNCTION – This routine supports compilation of the 'RECORD DATA' statement

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Options 1...18 supported.

#3201 - Initialize flags/counters

#3202 - Write type 40 TXT record

#3203 - INVALID (ACTERR)

#3204 - Save External Designator for 'PRESENT VALUE' (sensor)

#3205 - Write type 39 TXT record

#3206 - INVALID (ACTERR)

#3207 - INVALID (ACTERR)

#3208 - Verify, save 'SYSTEM' type External Designator

#3209 - INVALID (ACTERR)

GOAL COMPILER ROUTINE

NAME – SUB32 (continued)

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION –

#3210 - INVALID (ACTERR)

#3211 - INVALID (ACTERR)

#3212 - Text constant - write type 8 TXT record

#3213 - 'New Line' entry

#3214 - 'Internal Name' entry

#3215 - INVALID (ACTERR)

#3216 - Verify 'SYSTEM' type External
Designator write type 18 TXT if required

#3217 - 'PRINT' request - set up External
Designator

#3218 - 'RECORD' request - set up External
Designator

GOAL COMPILER ROUTINE

NAME – SUB33

FUNCTION – This routine supports compilation of the 'RELEASE CONCURRENT' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT, LOOKUP

DESCRIPTION – Options 1...5 supported.

#3301 - Initialize flags/counters

#3302 - Write type 37 TXT record

#3303 - Verify/save STMT NO. reference

#3304 - No Action

#3305 - No Action

GOAL COMPILER ROUTINE

NAME – SUB34

FUNCTION – This routine supports compilation of the 'REPEAT' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, TXTOUT

DESCRIPTION – Options 1...7 supported.

#3401 - Initialize flags/counters

#3402 - Verify/save 1st STMT NO. reference

#3403 - Write type 24 TXT record

#3404 - Verify/save 2nd STMT NO. reference

#3405 - No Action

#3406 - Save repetition count

#3407 - No Action

GOAL COMPILER ROUTINE

NAME – SUB35

FUNCTION – This routine supports compilation of the 'REPLACE' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, NEXTCR, INPUT, LOOKUP, SYSERR

DESCRIPTION – Options 1...7 supported.

#3501 - Save 1st 'NAME'

#3502 - Save 2nd 'NAME'

#3503 - Save 1st 'TEXT'

#3504 - Save 2nd 'TEXT'

#3505 - Save 1st 'Funct designator'.

#3506 - Save 2nd 'Funct designator'.

#3507 - Update substitution table

GOAL COMPILER ROUTINE

NAME – SUB36

FUNCTION – This routine supports compilation of the 'REQUEST KEYBOARD' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, FDLKUP, TXTOUT

DESCRIPTION – Options 1...8 supported.

#3601 - Initialize flags/counters

#3602 - Verify, save 'SYSTEM' Function Designator
Write type 18 TXT record

#3603 - Verify/save Internal Name (for input)

#3604 - Write type 55 TXT record

#3605 - Save TEXT constant - write type 8 TXT record

#3606 - No Action

#3607 - 'New Line' entry - save

#3608 - Verify/Save 'Internal/Name' - Text type message

GOAL COMPILER ROUTINE

NAME – SUB37

FUNCTION – This routine supports compilation of the 'RESUME' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Option 1 is supported.

#3701 - Write type 67 TXT record

GOAL COMPILER ROUTINE

NAME – SUB38

FUNCTION – This routine supports compilation of the 'SET DISCRETE' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Options 1...12 supported.

#3801 - Initialize flags/counters

#3802 - 1st External Designator - save - 'PRESENT VALUE' option

#3803 - 2nd External Designator - save - 'PRESENT VALUE' option

#3804 - Save 1st External Designator - 'SET <FD> option

#3805 - Prep for type 46 TXT record

#3806 - Save 'STATE'

#3807 - Save 'INTERNAL NAME'

#3808 - Save 'TIME'

GOAL COMPILER ROUTINE

NAME – SUB38 (continued)

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION – #3809 - Set flag for 'OPEN/'TURN ON'

#3810 - Set flag for 'CLOSE'/'TURN OFF'

#3811 - Verify List counts

#3812 - Write TXT record

GOAL COMPILER ROUTINE

NAME – SUB40

FUNCTION – This routine supports compilation of the 'STOP' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT, LOOKUP

DESCRIPTION – Options 1...5 supported.

#4001 - Initialize flags/counters

#4002 - Write type 54 TXT record

#4003 - Note - No restart labels specified

#4004 - Save STMT LABELS

#4005 - Generate array for 'LABELS'

GOAL COMPILER ROUTINE

NAME – SUB41

FUNCTION – This routine supports compilation of the 'TERMINATE' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT

DESCRIPTION – Options 1...3 supported.

#4101 - Initialize flags/counters

#4102 - Write type 25 TXT record

#4103 - Set flag for 'TERMINATE SYSTEM'

GOAL COMPILER ROUTINE

NAME – SUB42

FUNCTION – This routine supports compilation of the 'USE DATA BANK' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, SEEKDB, LOOKUP

DESCRIPTION – Options 1...4 supported.

#4201 - Initialize flags/counters

#4202 - Save 'DATA BANK NAME'

#4203 - Verify 'Data Bank' add to use list

#4204 - Save Revision Label

GOAL COMPILER ROUTINE

NAME – SUB44

FUNCTION – This routine supports compilation of the 'DIRECTIVES' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, NEXTCR, INPUT

DESCRIPTION – Options 1...19 supported.

#4401 - Set EXLIST 'no print' FLAG

#4402 - Verify set 'sequencing field' length

#4403 - Set 'NO TXT' flag

#4404 - Clear all output listing enable flags

#4405 - Enable 'SOURCE LISTING'

#4406 - Enable 'EXPANDED LISTING'

#4407 - Enable 'LABEL XREF LISTING'

#4408 - Enable 'INTERNAL NAME XREF LISTING'

#4409 - Enable 'FUNCTION DESIGNATOR XREF LISTING'

GOAL COMPILER ROUTINE

NAME – SUB44 (continued)

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION – #4410 - Enable 'DIAGNOSTIC SUMMARY LISTING'

#4411 - Save 'TITLE'

#4412 - Save 'DATE'

#4413 - Save 'Page Size'

#4414 - Save 'Line Size'

#4415 - Set up for new page

#4416 - Set page count

#4417 - Set convert, reset punch

#4418 - Set line size to 80, set punch flag

#4419 - Reset punch and convert flags

GOAL COMPILER ROUTINE

NAME – SUB45

FUNCTION – The routine supports compilation of the 'WHEN INTERRUPT' statement.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, TXTOUT, LOOKUP, FDLKUP

DESCRIPTION – Options 1...16 supported.

#4501 - Initialize flags

#4502 - Save Subroutine Name

#4503 - Write type 34, 35, and 68 TXT records

#4504 - Save statement nbr, Write type 27 TXT record

#4505 - No Action

#4506 - Write type 2 TXT record (Number Pattern)

#4507 - Write type 2 TXT record (Number Value)

#4508 - Write type 4 TXT record (Quantity value)

#4509 - Save Self Defining State Parameter

#4510 - Write type 8 TXT record (Internal name)

GOAL COMPILER ROUTINE

NAME – SUB45 (continued)

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION – #4511 - Write type 18 TXT record (Function Designator)

#4512 - Save Internal Name

#4513 - Set "Critical" Subroutine Flag

#4514 - No Action

#4515 - Write type 65 TXT record

#4516 - Save 'RETURN TO' statement number

GOAL COMPILER ROUTINE

NAME – SUB48

FUNCTION – This routine supports compilation of the 'PREFIX'

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, FDLKUP, TXTOUT

DESCRIPTION – Options 1...45 supported.

#4801 - Note 'AFTER' option

#4802 - Note 'WHEN' option

#4803 - Verify Time <F.D.>, write type 18 TXT record

#4804 - Verify 'TIME VALUE'

#4805 - Verify 'INTERNAL NAME'

#4806 - Write type 52 TXT record for 'TIME PREFIX'

#4807 - No Action

#4808 - Initialize flags/counters for 'LIMITS TEST'

#4809 - Save 'INTERNAL NAMES' in 'LIMITS TEST'

#4810 - Save 'NUMBER' in 'LIMITS TEST', write type 2 TXT record

GOAL COMPILER ROUTINE

NAME – SUB48 (continued)

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION –

#4811 - Save 'QUANTITY' in 'LIMITS TEST', write type 4 TXT

#4812 - Note 'NOT BETWEEN' option

#4813 - Verify list counts

#4814 - No Action

#4815 - Note 'IF' option

#4816 - Negate 'GO TO' TEST

#4817 - Write type 56 TXT record

#4818 - Verify compatibility for 'RELATIONAL TEST'

#4819 - Save 'STATE'

#4820 - Determine Relational operator

GOAL COMPILER ROUTINE

NAME – SUB48 (continued)

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION –

#4821 - Process TEXT constant - write type 8 TXT record

#4822 - Initialize Relational operator test

#4823 - Write type 57/58 TXT records, process implied 'VERIFY/STOP' if required

#4824 - Note 'VERIFY THEN'

#4825 - Note 'VERIFY ELSE/AND'

#4826 - Note 'VERIFY ELSE'

#4827 - Note 'IMPLIED STOP'

#4828 - No Action

#4829 - Initialize for 'OUTPUT EXCEPTION'

#4830 - Note 'PRINT OPTION'

GOAL COMPILER ROUTINE

NAME – SUB48 (continued)

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION –

#4831 - Note 'DISPLAY OPTION'

#4832 - Note 'RECORD OPTION'

#4833 - Process 'TEXT'

#4834 - Process 'INTERNAL NAME' for 'VERIFY'

#4835 - Process 'EXTERNAL DESIGNATOR' for VERIFY

#4836 - Zero time buffer - Initialize

#4837 - Save 'DAYS'

#4838 - Save 'HOURS'

#4839 - Save 'MIN'

#4840 - Save 'SEC'

#4841 - Save 'MS'

#4842 - Update for signed 'TIME VALUE'

GOAL COMPILER ROUTINE

NAME – SUB48 (continued)

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION – #4843 - Save Time Value - Write Type 4 Text Record

#4844 - Save Pointers to Internal Name

#4845 - Save 'WITHIN' Time Value

GOAL COMPILER ROUTINE

NAME – SUB49

FUNCTION – This routine supports compilation of 'INTERNAL NAME'.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, LOOKUP, FDLKUP

DESCRIPTION – Options 1...10 supported.

#4901 - Verify 'NAME' is defined, save type, etc.

#4902 - Verify 'ROW DESIGNATOR'

#4903 - Verify 'COLUMN NAME'

#4904 - Only 'COLUMN NAME' given

#4905 - Set flag for column subscript

#4906 - Set flag for row subscript

#4907 - Process 'TABLE'

#4908 - Process 'LIST'

#4909 - Process 'LIST' subscript

#4910 - Process 'SCALAR' (single) NAME

GOAL COMPILER ROUTINE

NAME – SUB51

FUNCTION – This subroutine substitutes GOAL words and phrases for corresponding short form words and phrases.

CALLED BY – ACTION

SUBROUTINES CALLED – NEXTCR

DESCRIPTION – This subroutine is entered from the syntax action numbers #5101, #5102, #5103. This parser sets SUBTXT to an address in STXTAB which contains the number of characters in the substitute field. If CONVRT, which is set by the Compiler Directives Subroutine, is equal to one, the substitution is made.

#5101 - Save pointer to the first letter in a short form word

#5102 - If CONVRT equals one, make the substitution and apend a blank, the letter S, or both as appropriate

#5103 - Mark the short form word as singular or plural

GOAL COMPILER ROUTINE

NAME – SUB52

FUNCTION – This routine preprocesses free form data bank input records and outputs fixed form records for use by the data bank maintenance programs.

CALLED BY – ACTION

SUBROUTINES CALLED – ACTERR, (FORTRAN I/O)

DESCRIPTION – Options 1...26 supported

#5201 - Move name to output buffer

#5202 - Write fixed form 'DATABANK' record

#5203 - Write 'END DATABANK' statement

#5204 - Move function designator to output buffer

#5205 - Write fixed form 'SPECIFY' record

#5206 - Move 'LOAD' to output buffer

#5207 - Move 'SENSOR' to output buffer

#5208 - Move 'SYSTEM' to output buffer

#5209 - Move 'DISCRETE' to output buffer

#5210 - Move 'ANALOG to output buffer

GOAL COMPILER ROUTINE

NAME – SUB52 (Continued)

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION –

#5211 - Move 'CLOCK' to output buffer

#5212 - Move 'PRINTER' to output buffer

#5213 - Move 'CRT' to output buffer

#5214 - Move 'TAPE' to output buffer

#5215 - End of input data - set ENDFLG=1

#5216 - Support function designator alternate form - output first record

#5217 - Limit address to 4 digits and move it to output area

#5218 - Move subroutine name to output area

#5219 - Move revision label to output area

#5220 - Write fixed form 'DELETEDB' record

GOAL COMPILER ROUTINE

NAME – SUB52 (Continued)

FUNCTION –

CALLED BY –

SUBROUTINES CALLED –

DESCRIPTION –

#5221 - Write fixed form 'DELETE' record

#5222 - Move 'INTERRUPT' and value to output area

#5223 - Move 'FLAG' and value to output area

#5224 - Output 'NAME SUBROUTINE' record

#5225 - Set flag to indicate preprocessor mode

#5226 - Limit FORTRAN subroutine name to 6 characters

3.2.3 Intermediate Text Output Formats

The GOAL compiler generates an intermediate text output record for each of the syntactical elements of the GOAL language. The records are variable in length, with each having a standard 7 word header followed by up to 400 words of text data. Each word is a 16 bit integer which is referenced in IBM System/360 terminology as a halfword (HW). All records are of the following general format.

HW# 0 - Number of words following (6 through 400)

1 - Internal text record number (sequence 1 through n)

2 - Type Code (1 through 68)

3 - Continuation Code (0 = stand alone text record
(1 = additional text data to follow

4 - GOAL statement number

5 - GOAL statement label number

6 - Variable

7- n - Data

A standard format is used for representation of External Designators, Internal Names, and Comparison Tests. The individual format descriptions and the descriptions of the intermediate text records are given on the following pages.

STANDARD REPRESENTATION OF AN INTERNAL NAME

(Always 5 Half Words)

	Type	Name	Rows	Columns	
Numeric Scalar	1	Variable Sequence Number	0	0	1
Quantity Scaler	2	Variable Sequence Number	0	0	1
State Scalar	3	Variable Sequence Number	0	0	1
Text Scalar	4	Variable Sequence Number	0	0	Length of text string
Numeric List	5	Variable Sequence Number	0	0	Number of entries
Numeric List Indexed	5	Variable Sequence Number	I/-V#	0	1
Quantity List	6	Variable Sequence Number	0	0	Number of entries
Quantity List Indexed	6	Variable Sequence Number	I/-V#	0	1
State List	7	Variable Sequence Number	0	0	Number of entries
State List Indexed	7	Variable Sequence Number	I/-V#	0	1
Text List	8	Variable Sequence Number	0	Length of	Number of entries
Text List Indexed	8	Variable Sequence Number	I/-V#	text string	1
Numeric Table Column	9	Variable Sequence Number	0	I/-V#	Number of rows
Numeric Table Element	9	Variable Sequence Number	I/-V#	I/-V#	1
Quantity Table Column	10	Variable Sequence Number	0	I/-V#	Number of rows
Quantity Table Element	10	Variable Sequence Number		I/-V#	1
State Table Column	11	Variable Sequence Number	0	I/-V#	Number of rows
State Table Element	11	Variable Sequence Number	I/-V#	I/-V#	1
Text Table Column	12	Variable Sequence Number	0	I/-V#	Number of rows
Text Table Element	12	Variable Sequence Number	I/-V#	I/-V#	1

I = A positive Integer

-V# = The negative variable sequence number of the variable containing the index number

STANDARD REPRESENTATION OF AN EXTERNAL DESIGNATOR

(Always 4 Half Words)

Type	Variable Sequence Number	Number of Rows	Code

Type Data Bank	External Designator	Compiler
1	Load Discrete	4
2	Load Analog	2
3	Load Clock	2
4	Sensor Discrete	3
5	Sensor Analog	1
6	Sensor Clock	1
7	System Printer	5
8	System Display	5
9	System Tape	5
10	Subroutine	6
11	Interrupt	7
12	System Flag	8

Code

0 = Table Name Functions

1 = Number Inhibit Array

STANDARD REPRESENTATION OF COMPARISON TEST

(Always 17 Half Words)

Type				
1	1st Internal Name	2nd Internal Name	3rd Internal Name	OP Count
2	1st External Name	2nd Internal Name	3rd Internal Name	OP Count
3	1st Internal Name	2nd Internal Name	OP Count	
4	1st External Designator	2nd Internal Name	OP Count	
HW1	HW2-6	HW7-11	HW12-16	HW17

Type

1 - Limit Formula Internal Names

2 - Limit Formula External Designator

3 - Relational Formula Internal Names

4 - Relational Formula External Designator

OP Count

1 - GT

2 - LT

3 - GE

4 - LE

5 - EQ

6 - NE

7 - ON

8 - OFF

INTERMEDIATE TEXT TYPES

Type	Name
1	Declare Numeric Data (Uninitialized)
2	Declare Numeric Data (Initialized)
3	Declare Quantity Data (Uninitialized)
4	Declare Quantity Data (Initialized)
5	Declare State Data (Uninitialized)
6	Declare State (Initialized)
7	Declare Text (Uninitialized)
8	Declare Text (Initialized)
9	Declare Numeric List (Uninitialized)
10	Declare Numeric List (Initialized)
11	Declare Quantity List (Uninitialized)
12	Declare Quantity List (Initialized)
13	Declare State List (Uninitialized)
14	Declare State List (Initialized)
15	Declare Text List (Uninitialized/Initialized)
16	Declare Text List (Row Initialization)
17	Declare Numeric Table (Uninitialized/Initialized)
18	Function Designator Array
19	Declare Numeric Table (Row Initialization)
20	Declare Quantity Table (Uninitialized/Initialized)
21	Declare Quantity Table (Row Initialized)
22	Declare State Table (Uninitialized/Initialized)
23	Declare State Table (Row Initialization)

INTERMEDIATE TEXT TYPES
(Continued)

Type	Name
24	Repeat Statement
25	Terminate Statement
26	Statement Label
27	GO TO Statement
28	Begin Program
29	End Program
30	Activate Table (All)
31	Activate Table (Row)
32	Inhibit Table (All)
33	Inhibit Table (Row)
34	Enter/Leave Critical Mode
35	Perform Subroutine
36	Concurrently Perform
37	Release Concurrent Statement
38	Assign Statement
39	Record Present Value of
40	Record Statement
41	Apply Present Value of
42	Apply Analog (list or table column)
43	Apply Analog (scalars)
44	Set Present Value of
45	Set External Designator (list or table column)
46	Set Discrete (scalars)
47	Read Statement

INTERMEDIATE TEXT TYPES
(Continued)

Type	Name
48	Average
49	Issue Digital Pattern (present value of)
50	Issue Digital Pattern (list or table column)
51	Issue Digital Pattern (scalars)
52	Time Prefix
53	Delay Statement
54	Stop Statement
55	Request Keyboard
56	Condition Prefix (If/Then variation)
57	Condition Prefix (Verify)
58	Output Exceptions
59	Perform Program
60	Let Equal
61	Begin Subroutine
62	Declare Text Table (Uninitialized/Initialized)
63	Declare Text Table (Row Initialization)
64	Disable Interrupt
65	When Interrupt
66	Leave Statement
67	Resume Statement
68	Return To

DECLARE NUMERIC DATA (Uninitialized)

HW STANDARD HEADER

1 Intermediate Text Record Number

2 Record Type = 1

3 Continuation Code = 0

4 GOAL Statement Number

5 GOAL Statement Label

6 Not Used

* *

1 Variable Sequence Number

DECLARE NUMERIC DATA (Initialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 2
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used
* *	* *
1	Variable Sequence Number
2	0
3	0
4	0
5, 6	Initialization Value

DECLARE QUANTITY DATA (Uninitialized)

```
HW        STANDARD HEADER
 1  Intermediate Text Record Number
 2  Record Type = 3
 3  Continuation Code = 0
 4  GOAL Statement Number
 5  GOAL Statement Label
 6  Not Used
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 1  Variable Sequence Number
```

DECLARE QUANTITY DATA (Initialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 4
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	0
3	0
4	Quantity Number 1 thru 63 or - 1
	(The -1 denotes an initialized time entry.)
5, 6	Initialized Data

DECLARE STATE (Uninitialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 5
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number

DECLARE STATE (Initialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 6
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	0
3	0
4	0
5	State (0 or 1)

DECLARE TEXT (Uninitialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 7
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used
* *	* *
1	Variable Sequence Number
2	0
3	0
4	Length of text variable

DECLARE TEXT (Initialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 8
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	0
3	0
4	Length of text variable
5	1st text character
6	2nd text character
.	
.	
.	
n+4	nth text character ($1 \le n \le 80$)

DECLARE NUMERIC LIST (Uninitialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 9
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Number of entries in list

DECLARE NUMERIC LIST (Initialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 10
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used
* *	* *
1	Variable Sequence Number
2	Number of entries in list
3	0
4	0
5 6 }	1st Value
7 8 }	2nd Value
.	
.	
.	
2n+3 2n+4	nth value ($1 \le n \le 99$)

DECLARE QUANTITY LIST (Uninitialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 11
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Number of Rows

DECLARE QUANTITY LIST (Initialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 12
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Number of Rows
3	0
4	0
5, 6	1st Quantity Value
7, 8	2nd Quantity Value
.	
.	
.	
2n+3, 2n+4	nth Quantity Value
2n+5, 2n+6	1st Quantity Dimension
2n+7, 2n+8	2nd Quantity Dimension
.	
.	
.	
4n+3, 4n+4	nth Quantity Dimension ($1 \le n \le 99$)

DECLARE STATE LIST (Uninitialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 13
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Number of Rows

DECLARE STATE LIST (Initialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 14
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Number of Rows
3	0
4	0
5	1st State (0 or 1)
6	2nd State
.	
.	
.	
n+4	nth State ($1 \leq n \leq 99$)

DECLARE TEXT LIST (Uninitialized/Initialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 15
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Number of Rows
3	Maximum length of text data

*A type 15 Intermediate Text record must precede one or more type 16 Intermediate Text records.

DECLARE TEXT LIST (Row Initialization)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 16
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used
* *	* *
1	Variable Sequence Number
2	Row Number
3	Length of Text
4	1st Character
5	2nd Character
.	
.	
.	
n+4	nth Character ($1 \le n \le 80$)

*A type 15 Intermediate Text record must precede a group of type 16 Intermediate Text records.

DECLARE NUMERIC TABLE (Uninitialized/Initialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 17
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Number of Rows
3	Number of Columns

*A type 17 Intermediate Text record must precede a group of type 19 Intermediate Text records.

FUNCTION DESIGNATOR ARRAY

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 18
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

HW	
1	Variable Sequence Number
2	Number of Rows (Number of Function Designators)
3 4	Type Address } 1st Function Designator
5 6	Type Address } 2nd Function Designator
.	
.	
.	
n+2 n+3	Type Address } nth Function Designator ($1 \le n \le 45$)

DECLARE NUMERIC TABLE (Row Initialization)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 19
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used
* *	* *
1	Variable Sequence Number
2	Row Number
3	Number of Columns
4	0
5, 6 }	1st Entry in Row
7, 8 }	2nd Entry in Row
.	
.	
.	
n+4, n+5 }	nth Entry in Row ($1 \leq n \leq 10$)

*A type 17 Intermediate Text record must precede a group of type 19 Intermediate Text records.

DECLARE QUANTITY TABLE (Uninitialized/Initialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 20
3	Continuation Code = 0 or 1
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used
* *	* *
1	Variable Sequence Number
2	Number of Rows
3	Number of Columns

*A type 20 Intermediate Text record with HW3=1 must precede a group of type 21 Intermediate Text records.

DECLARE QUANTITY TABLE (Row Initialization)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 21
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Row Number
3	Number of Columns
4	0
5, 6	1st Entry in Row
7, 8	2nd Entry in Row
.	
.	
.	
2n+3, 2n+4	nth Entry in Row
2n+5	1st Dimension
2n+6	2nd Dimension
.	
.	
.	
3n+4	nth Dimension ($1 \le n \le 10$)

*There will be a type 21 Intermediate Text record for each row of the table. A type 20 Intermediate Text record with the continuation code Equal 1 must precede the type 21 Intermediate Text records.

DECLARE STATE TABLE (Uninitialized/Initialized)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 22
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Number of Rows
3	Number of Columns

*A type 22 Intermediate Text record will precede a group of type 23 Intermediate Text records.

DECLARE STATE TABLE (Row Initialization)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 23
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Row Number
3	Number of Columns
4	0
5	1st State (0 or 1)
6	2nd State
.	
.	
.	
n+4	nth State ($1 \leq n \leq 10$)

*There will be a type 23 Intermediate Text record for each row of the table. A type 22 Intermediate Text record must precede the type 23 Intermediate Text records.

.

REPEAT STATEMENT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 24
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used
* *	* *
1	Variable Sequence Number
2	Numer of times to repeat
3	Beginning Step Number
4	Ending Step Number

TERMINATE STATEMENT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 25
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	0 or 1
	0 = Terminate;
	1 = Terminate System;

STATEMENT LABEL

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 26
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

GO TO STATEMENT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 27
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

HW		
1	4	
2	1st Character	GOAL Statement Number to branch to
3	2nd Character	
4	3rd Character	
5	4th Character	

BEGIN PROGRAM

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 28
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Chain 12 Reference Number

* *

HW	
1-4	1st Four Characters of Program Name
5-8	1st Four Characters of Revision Label

END PROGRAM

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 29
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

ACTIVATE TABLE (ALL)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 30
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Number of Rows

ACTIVATE TABLE (ROW)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 31
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number		
2	0	or	1
3	Row Number		Variable Number of Index

INHIBIT TABLE (ALL)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 32
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Number of Rows

INHIBIT TABLE (ROW)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 33
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

HW			
1	Variable Sequence Number		
2	0	or	1
3	Row Number		Variable Number of Index

ENTER/LEAVE CRITICAL MODE

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 34
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used
* *	* *
1	0 or 1
	0 = Enter Critical
	1 = Leave Critical

PERFORM SUBROUTINE

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 35
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Chain 12 Reference Number

* *

1, 2, 3, 4	1st 4 characters of Subroutine name
5	Number of parameters
6, 7, 8, 9, 10	1st Parameter
11, 12, 13, 14, 15	2nd Parameter
. . .	
n+5, n+6, n+7, n+8, n+9	nth Parameter (0_n_10)

The parameters will be standard 5 word internal names or expanded Function Designator. The 1st word for Expanded Function Designators will be -1.

CONCURRENTLY PERFORM

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 36
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

HW	
1–5	Internal Time Value Variable (Zero in HW1 implies execute one time only)

RELEASE CONCURRENT STATEMENT(S)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 37
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Number of statements to be released (Zero implies release all)
2	1st statement number to be released
3	2nd statement number to be released
.	
.	
.	
n+1	last statement number to be released ($1 \leq n \leq 10$)

ASSIGN STATEMENT

HW	STANDARD HEADER	
1	Intermediate Text Record Number	
2	Record Type = 38	
3	Continuation Code = 0	
4	GOAL Statement Number	
5	GOAL Statement Label	
6	Not Used	

* *

HW	Field	
1	Internal Name Type (1-11)	1st Internal Name
2	Variable sequence Number	1st Internal Name
3	Row	1st Internal Name
4	Column	1st Internal Name
5	Length	
6	Internal Name Type (1-11)	2nd Internal Name
7	Variable Sequence Number	2nd Internal Name
8	Row	2nd Internal Name
9	Column	2nd Internal Name
10	Length	

RECORD PRESENT VALUE STATEMENT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 39
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1 2 3 4	Object External Designator. (Where the data is to be recorded)
5 6 7 8	Input External Designator (What data is to be recorded)

*Standard External Designator 4 half word format.

STOP STATEMENT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 54
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number (HW1=0 if HW2=0)
2	Number of Entries (restart labels, 0≤n≤10)
3	1st Restart Label
4	2nd Restart Label
.	
.	
.	
n+2	nth Restart Label

RECORD STATEMENT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 40
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

HW	
1–4	Object External Designator (Where the data is to be recorded.)
5	Number of internal names ($1 \le n \le 25$)
6–10	1st Internal Name
11–15	2nd Internal Name
. . .	
n+5 – n+9	nth Internal Name

* Zero in the first HW of an Internal Name indicates a new record or a carriage return

*Standard External Designator and Internal Name 4 and 5 half word formats.

APPLY PRESENT VALUE

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 41
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1, 2, 3, 4	External Designator (Load Analog)
5, 6, 7, 8	External Designator (Sensor Analog)

*Standard External Designator 4 half word format.

APPLY ANALOG STATEMENT
(List or Table Column)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 42
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1, 2, 3, 4	External Designator (Load Analog)
5, 6, 7, 8, 9	Internal Time Value Variable (Zero in HW5 implies one time only)
10, 11, 12, 13, 14	Internal Name (Must be numeric or quantity list or table column)

*Standard External Designator and Internal Name 4 and 5 half word formats.

APPLY ANALOG (Scalars)

HW	
	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 43
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1 2 3 4	External Designator (Load Analog)
5	Number of internal names ($0 \leq n \leq 15$). If n=0, then External Designator must be a subroutine.
6 7 8 9 10	1st Internal Name
11 12 13 14 15	2nd Internal Name
.	
.	
.	
n+5 n+6 n+7 n+8 n+9	nth Internal Name

*Standard External Designator and Internal Name 4 and 5 half word formats. Internal Names must be quantity or numeric scalar, list element or table element.

SET PRESENT VALUE STATEMENT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 44
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1, 2, 3, 4	Object External Designator (Load Discrete)
5, 6, 7, 8	Input External Designator (Sensor Discrete)

*Standard External Designator 4 half word formats.

SET EXTERNAL DESIGNATOR STATEMENT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 45
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1, 2, 3, 4	Object External Designator
5, 6	Time Value (0 if none specified)
7, 8, 9, 10, 11	Internal Name (must be a state list or table column)

*Standard External Designator and Internal Name 4 and 5 half word formats.

SET DISCRETE (Scalars)

HW STANDARD HEADER

1 Intermediate Text Record Number

2 Record Type = 46

3 Continuation Code = 0

4 GOAL Statement Number

5 GOAL Statement Label

6 Not Used

* *

1, 2, 3, 4 } Object External Designator

5, 6, 7, 8, 9 } Internal Time Value Variable
(HW5 is Zero if time not specified)

10 Number of Internal Names ($0 \leq n \leq 15$) If n=0, then External Designator must be a subroutine type.

11, 12, 13, 14, 15 } 1st Internal Name

16, 17, 18, 19, 20 } 2nd Internal Name

.
.
.

n+11, n+12, n+13, n+14, n+15 } nth Internal Name

*Standard External Designator and Internal Name 4 and 5 half word formats.

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 47
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1 2 3 4	External Designator
5 6 7 8 9	Internal Name

*Standard External Designator and Internal Name 4 and 5 half word formats.

AVERAGE STATEMENT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 48
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1, 2, 3, 4	External Designator
5	Number of readings to average
6, 7, 8, 9, 10	Internal Name

*Standard External Designator and Internal Name 4 and 5 half word formats.

ISSUE DIGITAL PATTERN STATEMENT
(Present Value of)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 49
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1 2 3 4	External Designator (Load)
5 6	External Designator (Sensor)
7 8	

*Standard External Designator 4 half word format.

ISSUE DIGITAL PATTERN STATEMENT

(List or Table Column)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 50
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1, 2, 3, 4	External Designator (Load)
5, 6, 7, 8, 9	Internal Name (must be list or table column

*Standard External Designator and Internal Name 4 and 5 half word format.

ISSUE DIGITAL PATTERN (Scalars)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 51
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1 2 3 4	External Designator (Load)
5	Number of Internal Names (0≤n≤15) If n=0, then External Designator must be subroutine type.
6 7 8 9 10	1st Internal Name (must be scalar, list element or table element)
11 12 13 14 15	2nd Internal Name
.	
.	
.	
n+5 n+6 n+7 n+8 n+9	nth Internal Name

*Standard External Designator and Internal Name 4 and 5 half word format.

TIME PREFIX

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 52
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	0 = After 1 = When
2	Function Designator array number
3 4 5 6 7	Internal Name for Time Value

*Standard Internal Name 5 half word format.

STOP STATEMENT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 54
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number (HW1=0 if HW2=0)
2	Number of Entries (Restart Labels, $0 \leq n \leq 10$)
3	1st Restart Label
4	2nd Restart Label
.	
.	
.	
.	
n+2	nth Restart Label

REQUEST KEYBOARD STATEMENT

HW	
	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 55
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

HW		
1	Function Designator Variable Number (Input Device)	
2 3 4 5 6	Internal Name	
7	Number of additional Internal Names (0≤n≤25)	
8 9 10 11 12	1st Internal Name	*Zero in the 1st HW of an Internal Name indicates a new record or a carriage return
13 14 15 16 17	2nd Internal Name	
. . .		
n+7 n+8 n+9 n+10 n+11	nth Internal Name	

*Standard Internal Name 5 half word format.

CONDITION PREFIX
(If/Then Variation)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 56
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Comparison Test
.	
.	
.	
17	

*Standard Comparison Test 17 half word format.

CONDITION PREFIX
(Verify)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 57
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1, 2	Time Variable (Zero in none specified)
3	Else/Then Code 0 = Then 1 = Else
4 . . . 20	Comparison Test
21 . . . 25	Internal Time Value Variable

*Standard Comparison Test 17 half word format.

OUTPUT EXCEPTIONS

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 58
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

HW	
1 2 3 4 5	Internal Name (Message List) Half words 1 thru 5 = 0 for Default Message
6 7 8 9	External Designator (Output Device)

*Standard External Designator and Internal Name 4 and 5 half word format.

PERFORM PROGRAM

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 59
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Chain 12 Reference Number

* *

1 2 3 4	1st 4 characters of Program Name
5 6 7 8	1st 4 characters of Revision Label (blank if none specified)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 60
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1, 2, 3, 4, 5	Internal Name
6	Number of Operators and Variables ($1 \leq n \leq 55$)
7	1st Operator
8	2nd Operator or 2nd Quantity/Number/Internal Name
.	
.	
.	
.	
.	
n+6, n+7, n+8, n+9, n+10	Last Quantity/Number/Internal Name

*Standard Internal Name 5 half word format will be used

*Operators require 1 HW and are defined as follows:

(= -1
) = -2
+ = -3
- = -4
* = -5
/ = -6
** = -7

BEGIN SUBROUTINE

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 61
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Chain 12 Reference Number

* *

1, 2, 3, 4	1st 4 characters of subroutine name
5	Number of parameters ($0 \leq n \leq 10$)
6	1st Parameter type {0 = Internal Name; 1 = Function Designator}
7	1st parameter Variable Sequence Number
8	2nd parameter type
9	2nd parameter Variable Sequence Number
. . .	
n+5	nth parameter type
n+6	nth parameter Variable Sequence Number

DECLARE TEXT TABLE

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 62
3	Continuation Code = 0 or 1
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Number of Rows
3	Number of Columns
4	Number of Characters Per Entry

*A type 62 Intermediate Text Record must precede a group of type 63 Intermediate Text Records.

DECLARE TEXT TABLE

(Row Initialization)

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 63
3	Continuation Code = 0 or 1 (1 if continuation of row)
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Row Number
3	1st Column Number This Entry
4	Last Column Number This Entry
5 . . . n	Text Data for 1st Column this row
y y+1 y+2 y+3 y+4	Text Data for Last Column This row

A type 62 Intermediate Text Record must precede a group of type 63 Intermediate Text records. Each row requires a separate type 63 Intermediate Text Record.

DISABLE INTERRUPT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 64
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Number of statements to be disabled 0 implies "disable all"
2	1st statement number to be disabled
3	2nd statement number to be disabled
.	
.	
.	
n+1	Last statement number to be disabled

$0 \le n \le 10$

WHEN INTERRUPT STATEMENT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 65
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Variable Sequence Number
2	Variable Type
3	Variable Address

GOAL STATEMENT

HW	
	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 66
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1	Number of parameters (0≤n≤10)
2 3 4 5 6	1st Parameter
7 8 9 10 11	2nd Parameter
. . .	
n+1 n+2 n+3 n+4 n+5	nth Parameter

*Standard External Designator or Internal Name 4 and 5 half word format. Function Designator 1st half word will be -1.

RESUME STATEMENT

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 67
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

RETURN TO

HW	STANDARD HEADER
1	Intermediate Text Record Number
2	Record Type = 68
3	Continuation Code = 0
4	GOAL Statement Number
5	GOAL Statement Label
6	Not Used

* *

1 { 0 = Perform Subroutine and return
{ + integer = Perform subroutine and return to this step number

*This Intermediate Text record type is found in conjunction with a type 65 Intermediate Text Record.

3.2.4 Chain Definitions

The Symbol Table used by the GOAL Compiler consists of ten separate tables or chains. Each chain contains a unique type of symbol, i.e., internal names, step numbers, etc. The number given to a chain is the location in the symbol table of the pointer to the first entry of the chain. Each entry of the chain has a pointer to the next entry. The last entry of each chain points to a location in the table which contains a -1.

The chains are numbered 1 through 7 and 10 through 12. The locations that would normally be used for chains 8 and 9 are reserved for special use. Location 8 contains the -1 which is used by the chains to signify the last entry. Location 9 contains the pointer to the next available location in the table.

The following pages contain a graphical description of the SYMTAB table, the standard chain headers, and the format for each chain type.

SYMTAB TABLE FORMAT

The following is a representation of the SYMTAB table before any entries have been added to the chains.

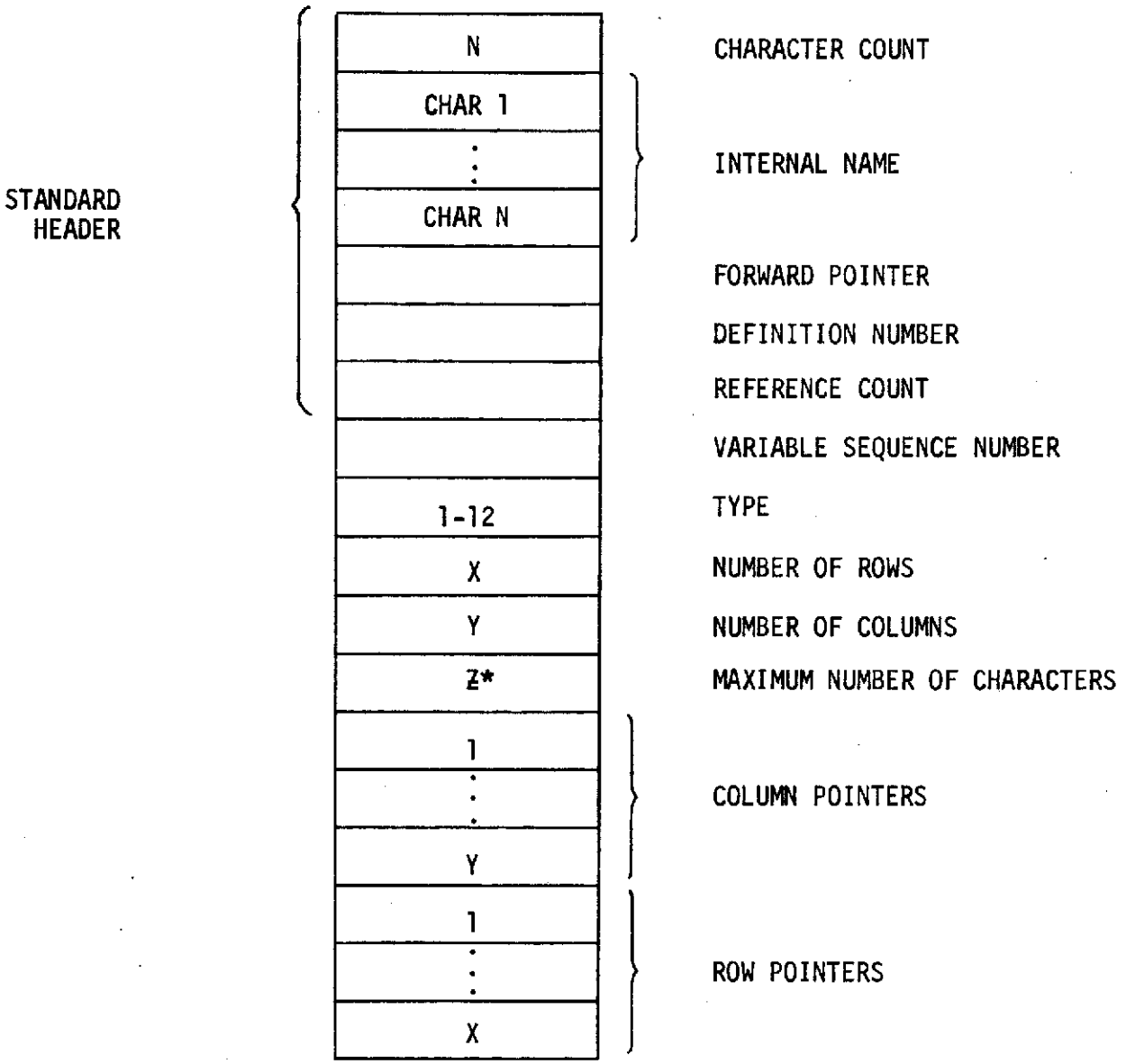
SYMTAB LOCATION	CONTENTS	CHAIN TYPE
1	8	Internal Names
2	8	Statement Labels
3	8	Function Designators
4	8	Abbreviations (replace Statement)
5	8	Data Bank Names
6	8	MACRO Names
7	8	MACRO Parameters
8	-1	-1
9	13	Next Available Location
10	8	Subroutine Names
11	8	Statement Labels (special usage)
12	8	Subroutine/Program Names
13		Chain Entries
.		
.		
.		
n		

CHAIN HEADER FORMAT

Each chain has a standard header which precedes the variable data associated with each individual chain. The standard header is as follows:

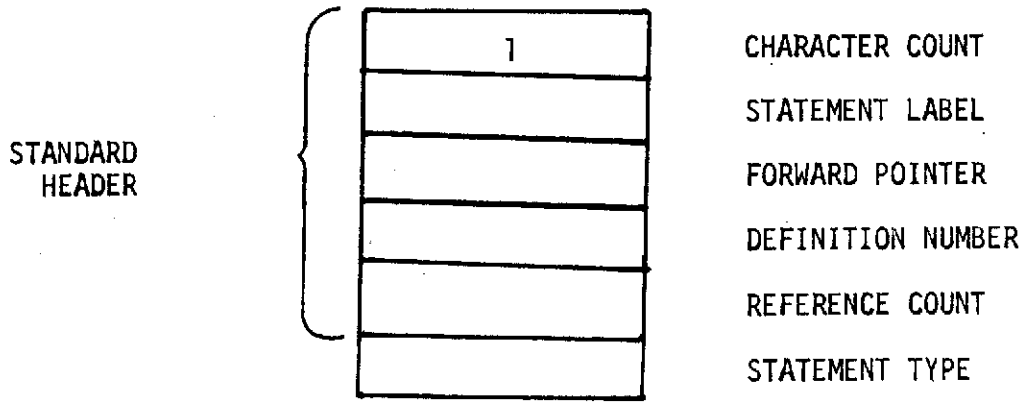
HW#	CONTENTS	DEFINITION
1	n	Number of characters in locations
2	1st Character	2 through m
.	.	Symbol variable for each type of chain
.	.	
.	.	
m	nth Character	
m+1	Forward Pointer	Pointer to next entry in the chain
m+2	Definition Number	Expanded source statement listing number
m+3	Reference Count	Number of times the variable has been referenced

CHAIN 1. INTERNAL NAMES

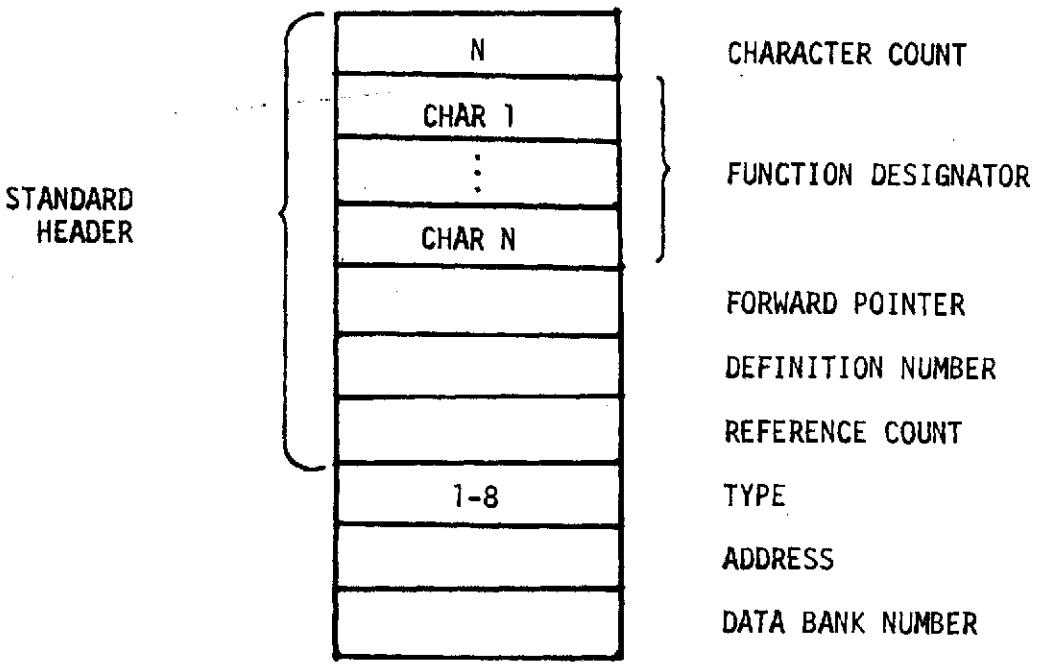


*This entry is included only with Type 12 data.

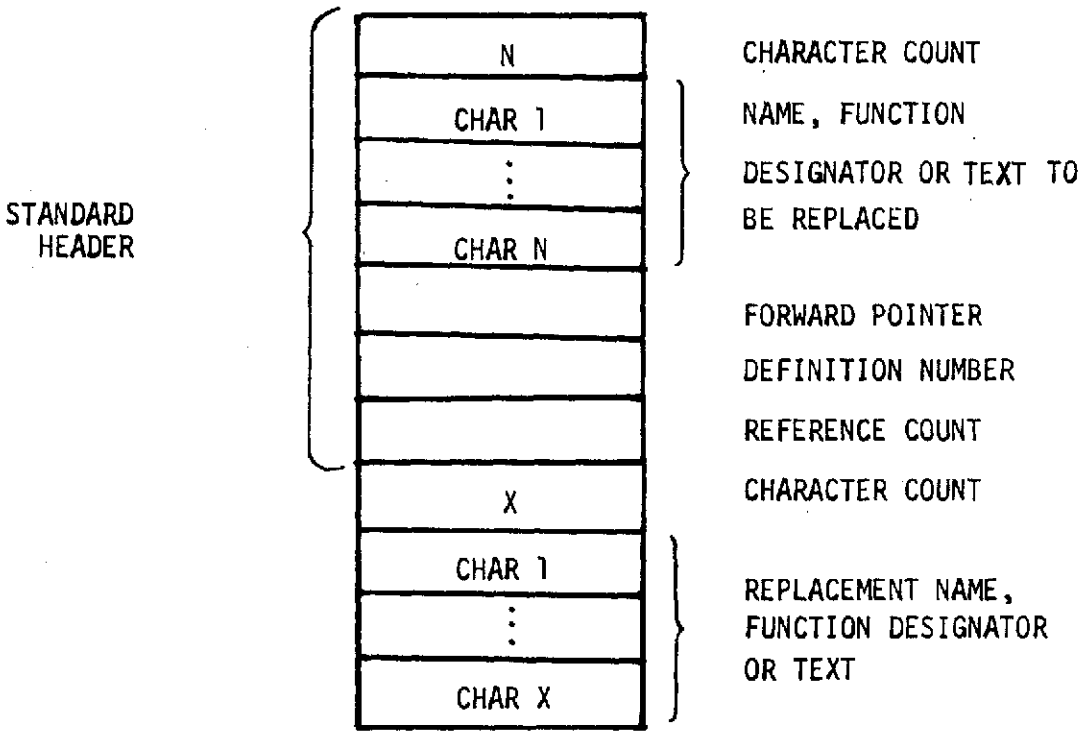
CHAIN 2. STATEMENT LABELS



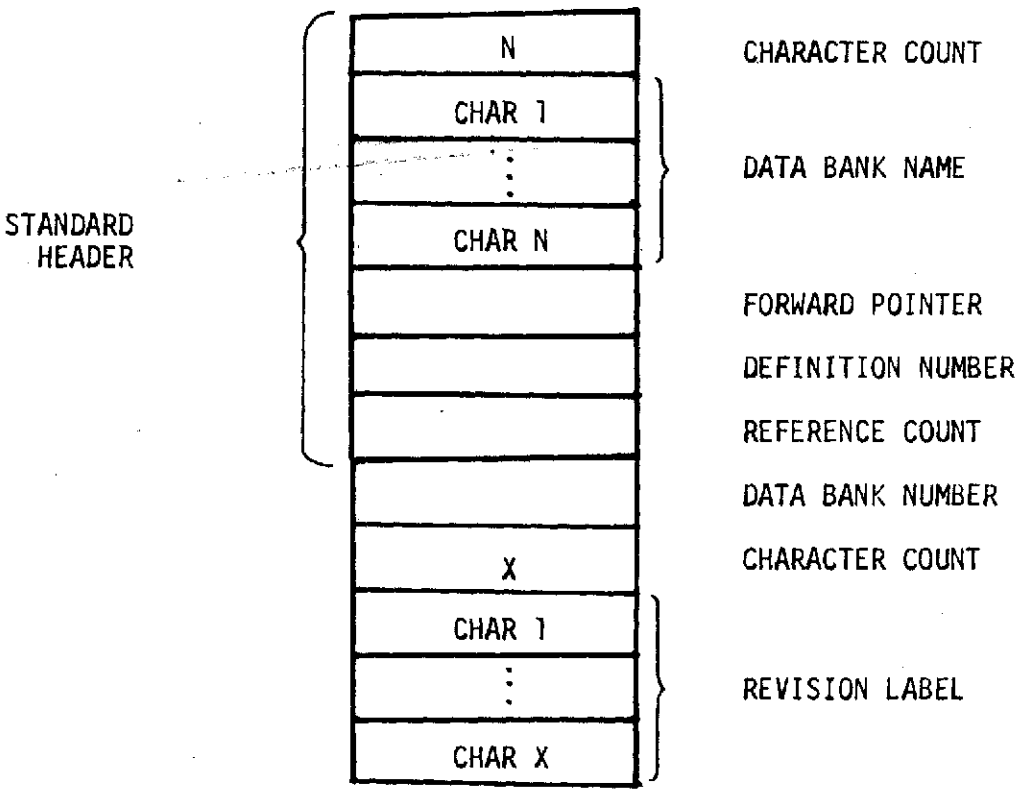
CHAIN 3. FUNCTION DESIGNATOR



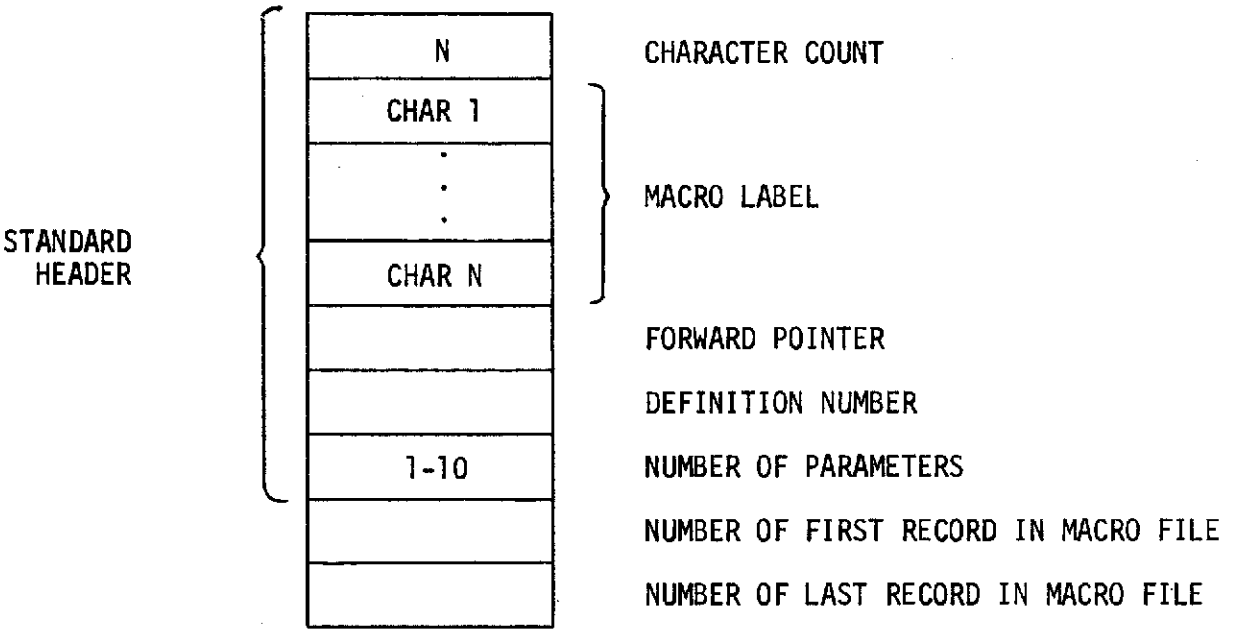
CHAIN 4. ABBREVIATIONS (REPLACEMENTS)



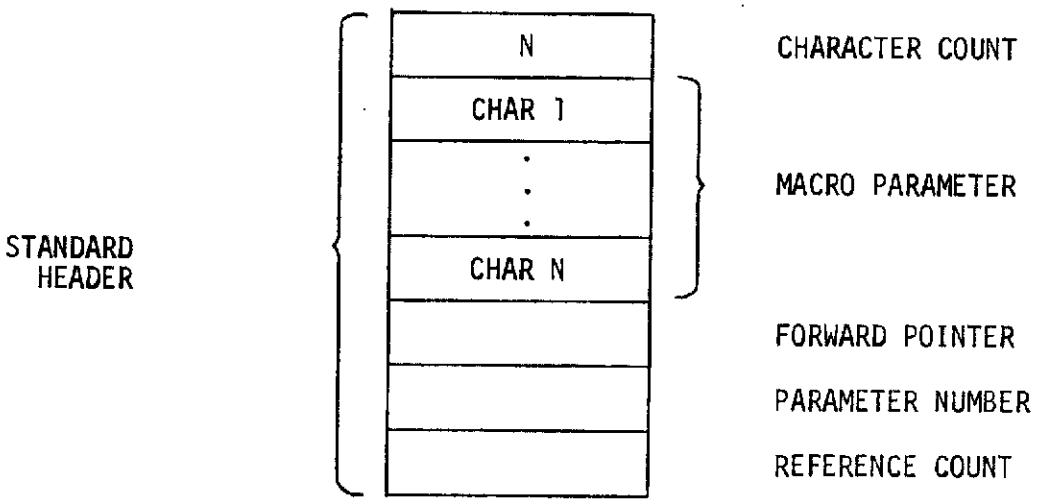
CHAIN 5. DATA BANK NAMES



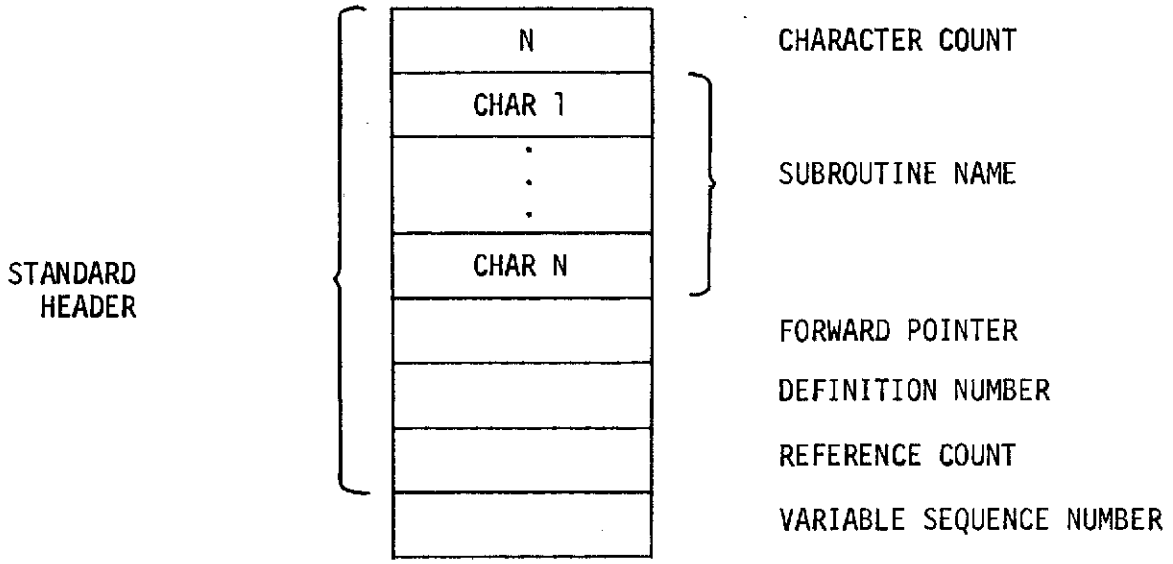
CHAIN 6. MACRO LABELS



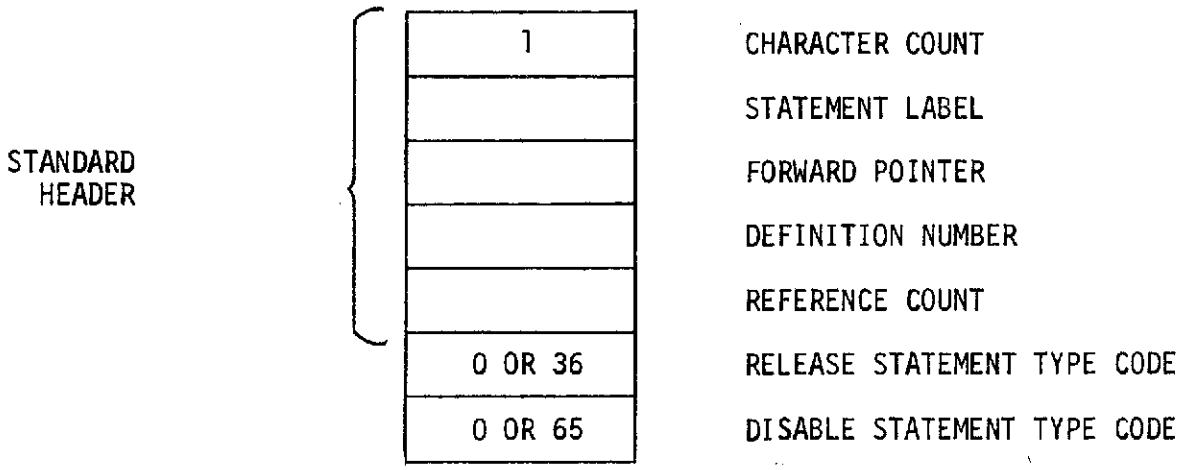
CHAIN 7. MACRO PARAMETERS



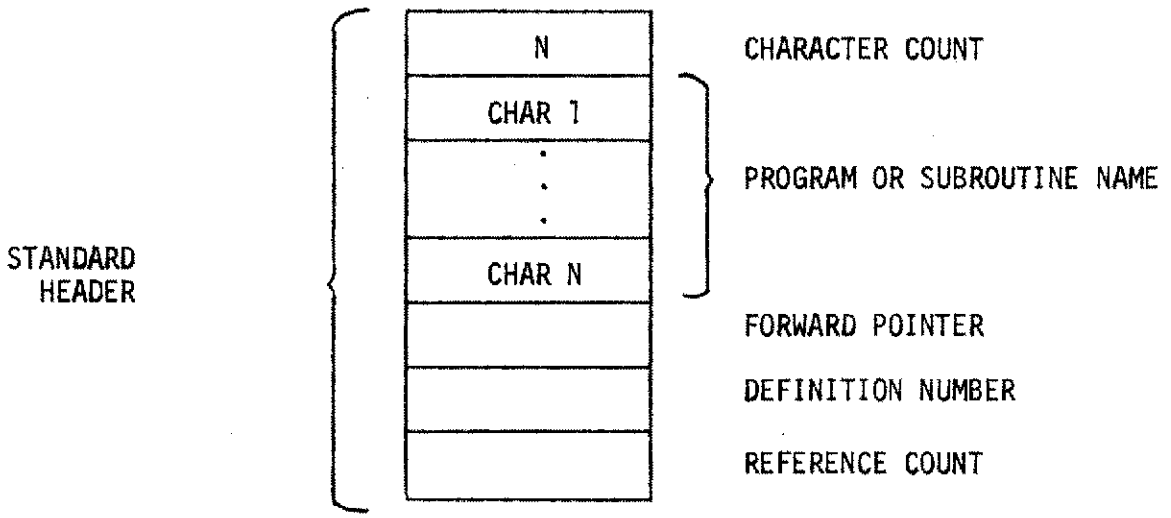
CHAIN 10. SUBROUTINES NAMES



CHAIN 11. STATEMENT LABELS TO BE VALIDATED



CHAIN 12. PROGRAM/SUBROUTINE NAME



3.2.5 Common Definitions

GOAL routines use 'common' storage locations for passing data between GOAL mainline routines, action routines, and data bank maintenance routines. This section gives the relative order of 'common' locations within each 'common block' and describes briefly each 'common' location.

COMMON BLOCK		COMMON LOCATIONS
BLANK COMMON	-	COMBLK, STXMAX, STMMAX, SYMMAX, STPMAX, K, RC, Z, ROOT, STMTK, LASTK, ENDK, STMTNO, STPNO, INREC, SEQFLD, ENDFLG, FDFLG, SYMFLG, LBLFLG, DGFLG, SRFLG, EXFLG, RECCNT, REC1ST, SRLNCT, SRPGCT, ERRCTR, EXLNSZ, EXPGCT, EXPGSZ, EXLNCT, ERRTAB, DATE, PRTLNE, TITLE, CHRTAB, ACTCOM, TLNOER, XRFFLG, TXTFLG, CTLFLG.
COMSØ7	-	PARMCT, PARTYP, TEXPT, SAVCC, STYPE, NAMSV4.
COMS17	-	FDCTR, TYPPT, FDCHK, FDTPPT, FDPKU, EXSUBR.
COMS22	-	CMCNT, IHDBN.
COMS44	-	ELSSVE, PNCHFG
COMS48	-	AFTRWN, FDAFN, RTHR, INCNT, OPCNT, LBPNT, VART, ILNG, ELSTHN, DPR, OUTEX
COMS49	-	TYPPT, SUB, FDCHK, FDROWS, COLSUB
COMS51	-	CONVRT
DBCM	-	MBLOCK, DBLOCK, DBREC, PREC, MLO, MMID, MHI
DBFWIN	-	DBFWIN
DBHWIN	-	DBHWIN
DBNKNM	-	DBNKNM
DSCOM	-	TLNOWR, UNRFNM, UNRFSN, UNDFNM, UNDFSN, UNDFFD, RFDSWI, RFRLCC
INPCOM	-	LTFLG, ABBRFG
INTTXT	-	TXTRCD
LMBUF	-	LMBUF

COMMON BLOCK		COMMON LOCATIONS
LVECOM	-	LVEFLG
MACCOM	-	MACFLG, FLSHFG, MEXPFG, EXPDFG, EXECFG, LSTREC, NXTMAX, RLOTFG, FSTREC, NXTMAC, PLSTAB
PRECOM	-	PREFLG
REPEAT	-	REPEAT
STMTAB	-	STMTAB
STPTAB	-	STPTAB
STXTAB	-	STXTAB
SUBCOM	-	SUBTXT, J, SUBFLG, SUBCNT, SVPRCC, STPSVB, STPFLG, Q, ZAP
SYMTAB	-	SYMTAB

Brief description of 'common' locations:

ABBRFG - Halfword integer in labeled common /INPCOM/. Used by INPUT to indicate that an abbreviation is being processed.

ACTCOM - Array of 200 halfwords in blank common. Symbolic names used for communication between 'action' routines are equivalenced to positions in ACTCOM.

AFTRWN - Halfword integer in labeled common /COMS48/. Used as a flag to indicate whether SUB48 is processing the 'AFTER' or 'WHEN' option of the time prefix. AFTRWN = 0 indicates the 'AFTER' option; AFTRWN = 1 indicates 'WHEN'.

ALPHA - Array of 26 halfwords beginning at CHRTAB (11) in blank common. Used as a table to contain the 26 alphabetic characters.

BEGINP - Halfword integer equivalenced to ACTCOM (95). It is initialized to zero in GINIT. It is set to 1 in SUB06 to indicate that a 'Program' is being compiled.

BEGINS - Halfword integer equivalenced to ACTCOM (94). It is initialized to zero in GINIT. It is set to 1 in SUB07 to indicate that a 'Subroutine' is being compiled.

BVAL - Fullword integer equivalenced to ACTCOM (3). It is used to contain the integer value, (binary value), of numeric type fields used in GOAL statements. It is computed and set in ACTION.

CHRTAB - Array of 80 halfword integers in blank common. List of 80 characters used to contain the GOAL character set. It is loaded from the syntax file by GINIT. CHRTAB is used by all 'ACTION' routines that test characters in the GOAL source statements.

CLABEL - Halfword integer equivalenced to ACTCOM (61). It is used to contain the integer value of the numeric field of a statement label.

CLFW - Fullword integer equivalenced to DBFWIN (11) in labeled common /DBFWIN/. CLFW is equal to zero, and is used to insure that there will be a zero location following the data bank sequence numbers.

CLHW - Halfword integer equivalenced to DBHWIN (11) in labeled common /DBHWIN/. CLHW is equal to zero, and is used to insure that there will be a zero location following the pointers into the data bank sequence table.

CMCNT - 1. Halfword integer in labeled common /COMS22/. Used by SUB22 to count the number of commas in a GOAL statement.

2. Halfword integer equivalenced to ACTCOM (184). It is used as a counter for initial values in DECLARE LIST/TABLE statements.

CNT - Halfword integer equivalenced to ACTCOM (9). It is used as a utility counter in action routines.

CNVRT - Fullword integer equivalenced to ACTCOM (53). It is used as a control word in converting numeric fields to actual values.

COLSUB - Halfword integer in labeled common /COMS49/. Used by SUB49 to contain the number of columns in a table.

COMBLK - Halfword integer used to insure proper boundary alignment in blank common.

CONDIF - Halfword integer equivalenced to LMBUF (18). Used by SUB48 to indicate whether the 'IF' or 'VERIFY' option of the 'VERIFY' prefix is being processed. CONDIF = 1 indicates the 'IF' option; CONDIF = 0 indicates 'VERIFY'.

CONTC - Halfword integer equivalenced to HEADER (3) in labeled common /INTTXT/. Used as a continuation flag in intermediate text records.

CONVRT - Halfword integer in labeled common /COMS51/. Used as a flag by the Compiler Directives subroutine SUB44. When CONVRT is set to one, SUB51 will substitute GOAL words and phrases for a short form dialect.

CTLFLG - Halfword integer in blank common. Used as a flag to indicate to EXLIST that the statement in the buffer is a compiler directive, and it is not to be printed. CTLFLG is set by SUB44.

DATAPT - Halfword integer equivalenced to ACTCOM (200). It is used to save a pointer to the 'table name functions' entry in SYMTAB. It is set by SUB17.

DATE - Array of 8 halfword integers in blank common. Used to contain the 'date' field printed in the EXPANDED SOURCE listing. This array is set to 'blanks' in GINIT. It may be modified by compiler directive in SUB44.

DBFWIN - Array of 11 fullwords in labeled common /DBFWIN/. Used as a table to contain the data bank numbers which are in use.

DBHWIN - Array of 12 halfwords in labeled common /DBHWIN/. Used as a table to contain pointers into table DBFWIN. This table determines the sequence which will be used to look up entries in data banks when more than one data bank is in use.

DBINT - Halfword integer equivalenced to DBNKNM (1) in labeled common /DBNKNM/. DBINT is the integer length of the data bank name contained in DBNKNM. It is also the first character of the data bank name.

DBLOCK - Array of 383 fullwords in labeled common /DBCM/. Used by data bank maintenance routines for upper and lower level directory blocks.

DBNKNM - Array of 34 halfwords in labeled common /DBNKNM/. Used by SUB22, SUB42, and FDLKUP to contain the data bank name.

DBNUM - Halfword integer equivalenced to DBHWIN (12). Used to count the total number of data banks that have been requested by 'USE data bank' statements.

DBREC - Array of 43 fullword integers in labeled common /DBCM/. Used to contain a record from the data bank.

DBUSE - Halfword integer equivalenced to ACTCOM (199). It is used to indicate the number of DATA BANKS in use. It is initialized to zero in GINIT. It is updated in SUB42 and SUB22.

DGFLG - Halfword integer in blank common. Used as a control word to enable generation of the Diagnostic Summary listing. It is set to 1 (enabled) in GINIT. It may be modified via compiler directive in SUB44. It is set to zero to inhibit generation of the listing.

DIGIT - Array of 10 halfwords beginning at CHRTAB (1) in blank common. Used as a table to contain the numeric digits 1 through 9 and zero.

DIMTYP - Halfword integer equivalenced to ACTCOM (98). It is used as a counter to determine the engineering units code. It is set to 1 by ACTION routine #18. It is incremented by 1 in ACTION routine #52. When the units field is recognized by the PARSER, DIMTYP contains the correct code.

DPLY - Halfword integer equivalenced to ACTCOM (44). Contains the FORTV variable number representing the internal name for the DISPLAY option when processing 'OUTPUT EXCEPTION'.

DPNT - Halfword integer equivalenced to ACTCOM (187). It is used to contain the (number of columns) x 2 + 3 for DECLARE tables.

DPR - Halfword integer in labeled common /COMS48/. It is used by SUB48 when processing the 'OUTPUT EXCEPTION' to indicate whether the 'DISPLAY', 'PRINT', or 'RECORD' option was specified. DPR = 0 indicates 'DISPLAY', DPR = 1 indicates 'PRINT', and DPR = 2 indicates 'RECORD'.

ELSSVE - Halfword integer in labeled common /COMS44/. It is used to save the expanded listing line size when entering the 'PUNCH' mode. When leaving the 'PUNCH' mode, the line size is restored to the value in ELSSVE.

ELSTHN - Halfword integer in labelled common /COMS48/. It is used to indicate which option of the 'VERIFY' prefix is being processed. ELSTHN = 0 indicates 'THEN' (or comma) option; ELSTHN = 1 indicates 'ELSE OUTPUT EXCEPTION'; ELSTHN = 2 indicates 'ELSE'; and ELSTHN = 3 indicates semicolon (;).

ENDFLG - Halfword integer in blank common. It is the flag used to indicate that the parsing phase of the GOAL compilation is complete. It is set to 1 when the program 'END' statement is parsed. This cues the compiler to generate compilation summary listings.

ENDK - Halfword integer in blank common. It is a pointer containing the subscript of the last significant character of the GOAL statement being parsed. It is set when the semicolon (;) is found. If the parse fails, it is set by FIXUP.

ENTCNT - Halfword integer equivalenced to ACTCOM (185). It is used to save the specified number of entries in DECLARE LIST/TABLE.

ERRCTR - Halfword integer in blank common. It is a counter containing the number of errors detected in the current GOAL statement. It is incremented by one each time ERROR is called and is reset by EXLIST for each GOAL statement.

ERRTAB - Array of 15 halfword integers in blank common. It is a list of 15 pointers containing the positions of errors identified in STMTAB. ERRCTR contains the number of entries in ERRTAB for the current GOAL statement. ERRTAB is updated by the routine, ERROR.

EXECFG - Halfword integer in labeled common /MACCOM/. Used as a flag to signify 'EXECUTE Only' statement. It is set to 1 in SUBØ4 if an 'EXECUTE Only' statement is encountered.

EXFLG - Halfword integer in blank common. Used as a control word to enable generation of the Expanded Source Record listing. It is set to 1, (enabled), in GINIT. It may be modified via compiler directive in SUB44.

EXLNCT - Halfword integer in blank common. Used as a counter to contain the number of lines used on the current page of the Expanded Source Listing. It is set to 50 in GINIT. It is incremented by 1 before each line is printed in EXLIST. When EXLNCT exceeds EXPGSZ, a new page is started and EXLNCT is reset to zero.

EXLNSZ - Halfword integer in blank common. Used as a control word to contain the number of GOAL statement characters per line in the Expanded Source Listing. It is set to 100 in GINIT. It may be modified via compiler directive in SUB44.

EXPDFG - Halfword integer in labeled common /MACCOM/. Used as a flag to signify 'EXPAND' only statement was found. EXPDFG is set to one in SUB21 if an 'EXPAND' only statement is found.

EXPGCT - Halfword integer in blank common. Used as a counter to contain the number of pages generated in the Expanded Source listing. It is incremented by EXLIST each time a new page is started. It may be modified via compiler directive in SUB44.

EXPGSZ - Halfword integer in blank common. Used as a control word to contain the number of lines per page on the Expanded Source listing. It is set to 50 in GINIT and may be modified via compiler directive in SUB44.

EXSUBR - Halfword integer in labeled common /COMS17/. Used as a flag to indicate whether 'function designator' or 'table name functions' of an external designator is being processed.

EXTDES - Array of 4 halfword integers equivalenced to ACTCOM (195). This array contains the 'standard' representation for 'external designator'.

FDAFN - Halfword integer in labeled common /COMS48/. Contains the FORTV variable name assigned to a function designator.

FDCHK - Halfword integer in labeled common /COMS17/ and /COMS49/. Used as a variable subscript for accessing in SYMTAB function designators which are row names of a table.

FDCNT - Halfword integer equivalenced to ACTCOM (62). It is used to contain the number of characters stored in NAMEBF for function designators. It is updated in ACTION.

FDCTR - Half integer in labeled common /COMS17/. Used by SUB17 to contain the number of function designators contained within an external designator.

FDFLG - Halfword integer in blank common. Control word used to enable generation of the function designator cross-reference listing. It is initialized to 1 in GINIT (enabled). It may be modified via compiler directive in SUB44.

FDFND - Halfword integer equivalenced to ACTCOM (189). It is used to indicate if a 'function designator' was found in action #20. It is set in ACTION.

FDPKU - Halfword integer in labeled common /COMS17/. Used as a variable subscript to access the FORTV name of a function designator.

FDROWS - Halfword integer in labeled common /COMS49/. Used by SUB49 to indicate the number of rows in a table.

FDTPPT - Halfword integer in labeled common /COMS17/. Used by SUB17 to indicate type of function designator which was accessed from SYMTAB by the variable subscript TYPPT.

FLSHFG - Halfword integer in labeled common /MACCOM/. Used by SUBØ5 to cause the macro body to be flushed following an error in the 'BEGIN MACRO' statement. A search for the 'END MACRO' statement is made disregarding all intermediate data. When found, all flags are cleared and processing continues normally.

FORTV - Halfword integer equivalenced to ACTCOM (60). It is used as a counter to determine the internal identification sequence number assigned to internal names which are explicitly or implicitly defined in the GOAL program. FORTV is initialized to zero in GINIT.

FRSTP - Halfword integer equivalenced to REPEAT (204) in labeled common /REPEAT/. Contains first step number to be repeated in a 'REPEAT' statement.

FSTREC - Fullword integer in labelled common /MACCOM/. Used as a counter for the next record to be read during macro expansion. It is set to the first record number of a macro body in SUB21 when an 'EXPAND AND/OR EXECUTE' statement is encountered. It is used in INPUT as a relative record number in a direct access 'WRITE' statement which requires a fullword integer. It is incremented after each 'READ' in INPUT. It is checked against LSTREC before each 'READ' in INPUT to determine the end of the macro body.

FVAL - Fullword floating point variable equivalenced to ACTCOM (55). It is used to contain the floating point value for numeric fields processed in GOAL statements.

FVALT - Fullword floating point variable equivalenced to ACTCOM (57). It is used for temporary storage of FVAL.

FWTEXT - Array of 200 fullword integers in labeled common /INTTXT/ used to assemble the 'data' portion of intermediate text records. FWTEXT is equivalenced to TXTRCD (7).

GTOFLG - Halfword integer equivalenced to ACTCOM (97). It is initialized to zero in GINIT. It is incremented by 1 in SUB23 to indicate that a GO TO statement had been found. It is subsequently tested to insure that the following statement has a label.

HEADER - Array of 6 halfword integers in labeled common /INTTXT/. HEADER is equivalenced to TXTRCD (1). It is used to assemble the standard 6 word header of the intermediate text records.

HWTEXT - Array of 400 halfword integers equivalenced to TXTRCD (7). Used to assemble the 'data' portion of intermediate text records.

IHDBN - Halfword integer in labeled common /COMS22/ used to store the sequence number of data bank numbers assigned in table DBHWIN.

ILNG - Halfword integer in labeled common /COMS48/ used by SUB48 to contain the length of intermediate text.

INCNT - Halfword integer in labeled common /COMS48/ used to indicate whether an external designator and associated names of a limit test have been placed into the output buffer. It points to the next five word array within that buffer into which the names will be moved. It also insures that no more than three names are to be moved.

INREC - Array of 80 halfword integers in blank common. INREC is the input buffer for GOAL source statements. It is loaded in routine INPUT from input stream or macro file. It is used to generate the source record listing in SRLIST. Characters are moved from INREC to STMTAB for parsing.

INTCOL - Halfword integer equivalenced to ACTCOM (193). It is used to contain the 'column' code for the current 'internal name'.

INTLNG - Halfword integer equivalenced to ACTCOM (194). It is used to contain the 'length' code for the current 'internal name'.

INTNME - Halfword integer equivalenced to ACTCOM (191). It is used to contain the internal identification number for the current 'internal name.'

INTRNM - ARRAY of 5 halfword integers equivalenced at ACTCOM (190). This array contains the following variables:

INTTYP	-	ACTCOM (190)
INTNME	-	ACTCOM (191)
INTROW	-	ACTCOM (192)
INTCOL	-	ACTCOM (193)
INTLNG	-	ACTCOM (194)

It is used to assemble the 'standard' representation of the 'internal name.'

INTROW - Halfword integer equivalenced to ACTCOM (192). It is used to contain the 'row' code for the current 'internal name'.

INTTYP - Halfword integer equivalenced to ACTCOM (190). It is used to contain the 'type' code for the current 'internal name'.

J - Halfword integer in labeled common /SUBCOM/. PARSER uses this as a pointer in STXTAB to the header of the syntax equation being parsed. In SUB26 this same common location is called MARKER.

K - Halfword integer in blank common used as a pointer to contain the subscript of the current character position in STMTAB for the statement being parsed. It is initialized in PREP prior to parsing and it is updated by the PARSER and any 'action' routines that process the statement. K must be saved and restored by action routines that fail (RC $\neq$ 0).

KSAVE - Halfword integer equivalenced to ACTCOM (8). It is used to save current value of K at entry to ACTION.

LASTK - Halfword integer in blank common used as a pointer to contain the subscript of the word in STMTAB immediately following the last character loaded with current data. LASTK is updated by INPUT. It is tested, as required, to insure that data in STMTAB is current.

LASTP - Halfword integer equivalenced to REPEAT (205) in labeled common /REPEAT/. Contains last step number to be repeated in a 'REPEAT' statement.

LBLFLG - Halfword integer in blank common used as a control word to enable generation of the statement label cross-reference listing. It is set to one (enabled) in GINIT. It may be modified via compiler directive in SUB44. It is set to zero to inhibit generation of the listing.

LBPNT - Halfword integer in labeled common /COMS48/ used by SUB48 as a pointer to the next entry in LMBUF.

LMBUF - Array of 18 halfwords in labeled common /LMBUF/ used by SUB48 to contain FORTY names assigned to quantities used for limit testing.

LNG - Halfword integer equivalenced to ACTCOM (59). It is used to contain field lengths during processing in ACTION.

LOPCTR - Halfword integer equivalent to REPEAT (206) in labeled common /REPEAT/. Used as a counter to contain the number of times steps are to be repeated in a 'REPEAT' statement.

LSTREC - Halfword integer in labeled common /MACCOM/ used to contain the last record number of a macro body. It is set in SUB21 after an 'EXPAND AND/OR EXECUTE' macro statement has been found. It is tested in INPUT to determine if the entire macro body has been used as input to the compiler.

LTFLG - Halfword integer in labeled common /INPCOM/. Used by INPUT to indicate that a character string began with a 'less than' symbol (<) and therefore should close with a 'greater than' (>) symbol rather than 'close parenthesis'.

LVEFLG - Halfword integer in labeled common /LVECOM/ which is set to one when a 'LEAVE' statement is encountered. This causes all data following the 'LEAVE' statement to be ignored until a 'RESUME' statement is parsed. When the 'RESUME' statement is parsed LVEFLG is set to zero.

MACFLG - Halfword integer in labeled common /MACCOM/ used as a flag for 'BEGIN MACRO' statement in EXLIST. Can have value of 0, 1, or 2.

0 - implies no macro being processed
continue normally

1 - implies record being processed is in macro body and should be written in macro file

2 - implies 'BEGIN MACRO' statement has been found and macro body follows.

This flag is initialized to zero in GINIT. It is set to two in SUBØ5 and reset to 1 in EXLIST. It is cleared to zero in SUBØ5 when an 'END MACRO' statement is found.

MARKER - Halfword integer in labeled common /SUBCOM/ used by SUB26 as a pointer to the header in STXTAB of the syntax equation being parsed. PARSER calls this same common location J.

MBLOCK - Array of 383 fullword integers in labeled common /DBCM/ used by the data bank maintenance routines for the master data bank directory block.

MEXPFG - Halfword integer in labeled common /MACCOM/ used as a flag to signify 'macro expansion mode' is in effect. This means that an 'EXPAND AND/OR EXECUTE' statement was encountered and input records to the compiler should come from the body of the previously defined macro with the same name. This flag is set to 1 in SUB21 when statement is encountered. It is checked in EXLIST to determine if EXECFG and EXPDFG should be checked for printing purposes. It is also checked in INPUT to see if records should come from the macro file or the input stream. It is cleared in INPUT when the entire macro has been used as input to the compiler.

MHI - Halfword integer in labeled common /DBCM/ used by data bank maintenance routines during search of the directory.

MLO - Halfword integer in labeled common /DBCM/ used by data bank maintenance routines during search of the directory.

MMID - Halfword integer in labeled common /DCBM/ used by data bank maintenance routines during search of the directory.

NAMCNT - Halfword integer equivalenced to ACTCOM (63). It is used to contain the number of characters stored in NAMEBF for 'names'. It is updated in ACTION.

NAMEBF - Array of 32 halfword integers used to assemble the characters of a 'name' used in GOAL statements. The number of characters used is contained in NAMCNT. NAMEBF is equivalenced to (starts at) ACTCOM (10) and uses consecutive locations.

NAMSV4 - Array of 4 halfwords in labeled common /COMSØ7/ used by SUBØ7 to save the first 4 digits of a name in NAMEBF.

NCR - Halfword integer equivalenced to ACTCOM (5). It is used to contain the most recent significant character found by NEXTCR. It is set in NEXTCR, (NCR = STMTAB (K-1)) on return.

NMFLD - Halfword integer equivalenced to ACTCOM (183). It is used to contain the position of the beginning of a 'name' field in STMTAB. It is set in ACTION.

NMFND - Halfword integer equivalenced to ACTCOM (188). It is used to indicate if a 'name' was found in action #29. It is set in ACTION.

NUMBUF - Array of 32 halfword integers equivalenced to ACTCOM (66) and occupying sequential locations. It is used to contain the characters of numeric fields.

NUMCNT - Halfword integer equivalenced to ACTCOM (64). It is used to contain the number of characters stored in NUMBUF for numeric fields. It is updated in ACTION.

NXTMAC - Fullword integer in labeled common /MACCOM/ used as a counter for the next available record in the macro file when creating a macro. It is initialized to 11 in GINIT. Each time a macro body record is written into the file, it is incremented by 1. It is used in EXLIST as a counter and as a relative record number in a direct access 'WRITE' statement which requires a fullword integer.

NXTMAX - Halfword integer in labeled common /MACCOM/ which contains the maximum record number allowable in the macro file. It is initialized to 1000 in GINIT. It is tested in EXLIST when macro body records are written in the macro file. If an attempt is made to exceed this maximum, a call to SYSERR with a parameter of 6 is made. This terminates the compiler.

OPCNT - Halfword integer in labeled common /COMS48/ which contains a number that represents the type of relational test in a condition prefix.

OUTEX - Array of 10 halfwords in labeled common /COMS48/. Contains variables which control displaying, printing, or recording of the 'OUTPUT EXCEPTIONS' option of the condition prefix.

PARMCNT - Halfword integer in labeled common /COMSØ7/ used by SUBØ7 to indicate the number of parameters passed to a subroutine in a 'BEGIN SUBROUTINE' statement.

PARTYP - Halfword integer in labeled common /COMSØ7/ used by SUBØ7 to indicate type of parameter passed in a 'BEGIN SUBROUTINE' statement. PARTYP = 1 indicates a function designator; PARTYP = 0 indicates an internal name.

PLSTAB - Array of 51 halfwords in labeled common /MACCOM/. PLSTAB parallels STPTAB. It is set by INPUT to mark for EXLIST which statements should have a plus sign (+) signifying macro expansion.

PNCHFG - Halfword integer in labeled common /COMS44/ which signifies that 'PUNCH' has been specified in a compiler directive. The line size is reduced to 80, and a deck is output. This flag is tested by EXLIST.

PREC - Fullword integer in labeled common /DBCM/ which is used by the data bank maintenance routines in a search of the data bank directory.

PREFLG - Halfword integer in labeled common /PRECOM/. It is set by SUB52 when data bank input is being preprocessed. It is tested by SUBØ5. If PREFLG = 1, SUBØ5 outputs processed macros with data bank input rather than writing them to the macro file.

PRNT - Halfword integer equivalanced to ACTCOM (45). Contains the FORTV variable number representing the internal name for the PRINT option when processing 'OUTPUT EXCEPTION'.

PROCFG - Halfword integer equivalenced to ACTCOM (96). PROCFG = 1 indicates the first procedural statement has been encountered. EXLIST prints the message

**********BEGIN OPERATING STEPS**********

PRTLNE - Array of 130 halfword integers in blank common. Buffer of 130 characters used to assemble the print line in EXLIST. The first 100 characters are set to blanks in GINIT. It is subsequently cleared in EXLIST after printing each line.

Q - Halfword integer in labeled common /SUBCOM/ which contains the syntax table number from the input source deck control card. It is used by GINIT to load the correct syntax table.

QTYPE - Halfword integer in blank common equivalenced to ACTCOM (99). Control word used to indicate if a 'quantity' type name has been assigned engineering units. It is updated in ACTION.

RC - Halfword integer in blank common used as a flag to indicate success or failure in parsing elements of GOAL statements. It is set by PARSER and/or 'action' routines. It is used by PARSER in finding successful paths through the syntax tables.

```
RC =  0  implies 'successful'
RC = -1  implies 'try alternate' or 'error'
RC >  0  implies 'error', RC = error number
```

RECCNT - Halfword integer in blank common used as a counter to contain the number of the current source record. It is set to zero in GINIT. It is incremented by 1 in SRLIST for each record processed from the GOAL source input stream.

RECD - Halfword integer equivalenced to ACTCOM (46). Contains the FORTV variable number representing the internal name for the RECORD option when processing 'OUTPUT EXCEPTION'.

RECLST - Halfword integer in blank common used as a control word to contain the source record number for the first record of the current GOAL statement. It is set to RECCNT in PARSER.

REPEAT - Array of 208 halfword integers in labeled common /REPEAT/. Consists of 8 working locations followed by 100 two-halfword entries containing the last step number in the loop and the FORTV name of the loop for all 'REPEAT' statements referenced in a program.

REPTEN - Halfword integer equivalenced to REPEAT (203) which contains the number of table entries in REPEAT.

RFDWSI - Halfword integer in labeled common /DSCOM/ used by DIAGSM to indicate the total number of step numbers referenced on a DISABLE statement but not defined on a WHEN INTERRUPT statement.

RFLCC - Halfword integer in labeled common /DSCOM/ used by DIAGSM to indicate the total number of step numbers referenced on a RELEASE statement but not defined on a CONCURRENT statement.

RLOTFG - Halfword integer in labeled common /MACCOM/ used as a flag for statement table roll-out. It is set to 1 in SUB21 when an 'EXPAND AND/OR EXECUTE' macro statement is encountered. It is checked in RESET to determine if the statement table should be saved. If it is on, the statement table is temporarily written to disk so a macro body can be processed and normal processing can continue following the macro expansion. It is set to zero following roll-out in RESET.

RLTXT - Array of 200 fullword floating point words equivalenced to TXTRCD (7) in labeled common /INTTXT/. Used to assemble the 'data' portion of intermediate text records.

ROOT - Halfword integer in blank common used as a pointer to contain the subscript of the 'root' syntactical element control block in STXTAB. ROOT is initialized when the syntax table is loaded in GINIT. It is used by PARSER to locate the starting point in the syntax table for parsing each GOAL statement.

RPNAM - Halfword integer equivalenced to REPEAT (207). Contains the FORTV variable name assigned to the repeat loop.

RTHR - Halfword integer in labeled common /COMS48/ used as a return parameter to allow common coding to be used in a subroutine. It is used as the parameter in a 'computed GO TO' following the shared coding.

SAVCC - Halfword integer in labeled common /COMSØ7/ used by SUBØ7 to save the completion code from LOOKUP.

SEQFLD - Halfword integer in blank common used as a control word to contain the number of columns reserved for sequencing data on the input source records. The sequencing field is the right-most portion of the input record. SEQFLD is initialized to zero in GINIT and may be modified via compiler directive in SUB44.

SPCHAR - Array of 21 halfwords starting at CHRTAB (37) used as a table to contain the GOAL special characters.

SRFLG - Halfword integer in blank common used as a control word to enable generation of the source record listing. It is set to 1 (enabled) in GINIT. It may be modified via compiler directive in SUB44.

SRLNCT - Halfword integer in blank common used as a counter to contain the number of lines contained in the current page of the source record listing. It is initialized to 50 in GINIT. It is incremented by 1 before each line is printed. If greater than 50, a new page is generated.

SRPGCT - Halfword integer in blank common used as a counter to contain the number of pages generated in the source record listing. It is set to zero in GINIT and is incremented by 1 in SRLIST each time a new page is started.

STATE - Halfword integer equivalenced to ACTCOM (6). It is used to contain the status of logical type fields used in GOAL statements. It is computed and set in ACTION.

STATOF - Same as STOFF.

STATON - Same as STONN.

STMLBL - Array of 5 halfword integers used to assemble the digit characters of a GOAL statement label. It is equivalenced to ACTCOM (48).

STMMAX - Halfword integer in blank common which contains the system limit of the size of SYMTAB. It is set to 3501 in GINIT.

STMTAB - Array of 3500 halfword integers in labeled common /STMTAB/ used as a table to contain GOAL source statements during parsing. STMTAB is loaded by calling routine INPUT. STMMAX is used to contain the size of STMTAB. STPTAB contains pointers to individual records contained in STMTAB. STMTAB is used to generate the expanded source listing.

STMTK - Halfword integer in blank common used as a pointer to contain the subscript of the first significant character of the current GOAL statement. It is set by routine PREP. It is used by PARSER to locate the beginning of the statement in STMTAB.

STMTNO - Halfword integer in blank common used as a counter to contain the 'internal statement number' for the GOAL statement being parsed. It is set to zero in GINIT and is incremented by 1 for each statement parsed.

STOFF - Halfword integer equivalenced to ACTCOM (48). Contains the FORTV variable number representing the condition 'state off'.

STONN - Halfword integer equivalenced to ACTCOM (47). Contains the FORTV variable number representing the condition 'state on'.

STPFLG - Halfword integer in labeled common /SUBCOM/ used as a flag to indicate 'END SUBROUTINE' statement has been parsed. The subroutine is written to the subroutine file, STPSUB is cleared and SUBCNT is incremented by 1.

STPMAX - Halfword integer in blank common used to contain the system limit of the size of STPTAB minus 1. It is initialized to 50 in GINIT.

STPNO - Halfword integer in blank common used as a counter to contain the number of blocks of expanded source data contained in STMTAB. STPNO +1 is the number of associated entries in STPTAB.

STPSUB - Halfword integer in labeled common /SUBCOM/ used as a flag to indicate whether or not the compiler is in the 'strip subroutine' mode. STPSUB = 1 signals the compiler to 'strip' until an 'END SUBROUTINE' statement is parsed.

STPTAB - Array of 51 halfword integers in labeled common /STPTAB/ used as a table of pointers to locate individual records in STMTAB. This table is updated when the routine INPUT is called. STPMAX contains the size of STPTAB. STPNO is equal to the number of records in STMTAB. STPNO +1 is equal to the number of entries in STPTAB.

STXMAX - Halfword integer in blank common used to contain the system limit of the size of STXTAB. It is set to 12,000 in GINIT.

STXTAB - Array of 12,000 halfword integers in labeled common /STXTAB/ used as a table to contain syntax data used by the PARSER. STXTAB is loaded from the syntax file by GINIT. STXMAX contains the size of STXTAB.

STYPE - Halfword integer in labeled common /COMSØ7/. Not used.

SUB - Halfword integer in labeled common /COMS49/ used by SUB49 as a variable subscript to allow common coding to be used for processing 'row index name' and 'column index name' data into INTRNM.

SUBCNT - Halfword integer in labeled common /SUBCOM/ used as a counter to indicate the number of successfully 'stripped' subroutines in the subroutine file. SUBCNT is incremented by ACTION.

SUBFLG - Halfword integer in labeled common /SUBCOM/ used to determine whether or not the subroutine file is to be rewound. The subroutine file is to be rewound (SUBFLG = Ø) only at the end of a GOAL program compilation and at the beginning of the first GOAL subroutine compilation.

SUBTXT - Halfword integer in labeled common /SUBCOM/. PARSER sets SUBTXT to an address in STXTAB which contains the number of characters in any field which fails to parse. This is used by the converter subroutine SUB51 to accomplish substitution of GOAL words and phrases for a short form dialect. The characters to be substituted begin at the address SUBTXT+1.

SVPRCC - Halfword integer in labeled common /SUBCOM/ used to contain the highest condition code in a GOAL compilation. This is the condition code that will be passed at the end of the run.

SYMFLG - Halfword integer in blank common used as a control word to enable generation of the internal name cross-reference listing. It is set to one (enabled) in GINIT and may be modified via compiler directive in SUB44.

SYMMAX - Halfword integer in blank common used to contain the system limit of the size of SYMTAB. It is set to 3000 in GINIT.

SYMTAB - Array of 3000 halfword integers in labeled common /SYMTAB/ used to contain symbolic names defined in GOAL source statements. Names are entered and/or verified by LOOKUP. Other data related to the name may be placed in the table immediately following it, provided that the 'free area pointer' TBF is updated. SYMMAX contains the size of SYMTAB.

TBF - Halfword integer equivalenced to SYMTAB (9) used as a pointer to contain the subscript of the first entry in the unused portion of SYMTAB. TBF is initialized to 13 in GINIT and is updated by LOOKUP when a symbol is entered in SYMTAB. Action routines may place additional data in SYMTAB if they update TBF accordingly. TBF may not exceed SYMMAX.

TBFSAV - Halfword integer equivalenced to ACTCOM (93) used by PREP to save the value of TBF. This allows for deletion of names from SYMTAB if the statements containing them are in error.

TCCNT - Halfword integer equivalenced to ACTCOM (65). It is used to contain the number of characters stored in TSAVE for text constants. It is updated in ACTION.

TD - Halfword integer equivalenced to ACTCOM (186). It is used as a flag to indicate if initial values are provided in DECLARE statement.

TEXPT - Halfword integer in labeled common /COMSØ7/ used by SUBØ7 as a pointer into HWTEXT to the type of parameter being parsed (PARTYP).

TIME - Fullword folating point integer equivalenced to ACTCOM (181). It is used to contain the time value for 'time constants'. It is updated in SUB48.

TITLE - Array of 100 halfword integers in blank common. It is a list of characters used to contain the header printed on each page of the expanded source listing. This array is initialized to blanks in GINIT. It may be modified via compiler directive in SUB44.

TLNOER - Halfword integer in blank common used as a counter for the total number of errors in a GOAL compile. It is used by DIAGSM to set the condition code. The counter is incremented by ERROR.

TLNOWR - Halfword integer in labeled common /DSCOM/ used by DIAGSM to indicate the total number of warnings in a GOAL compile.

TRCD - Halfword integer equivalenced to HEADER (1) in labeled common /INTTXT/. It is used to contain the record number in intermediate text records.

TSAVE - Array of 80 halfword integers equivalenced to ACTCOM (100). It is used to contain the characters of a 'text constant'. It is updated in ACTION.

TXTFLG - Halfword integer in blank common used as a flag to indicate to TXTOUT whether or not intermediate text is to be generated. If TXTFLG = 1 generation of intermediate text is inhibited. SUB44 sets TXTFLG. DIAGSM tests TXTFLG when setting the condition code. TXTFLG = 1 gives a condition code of 12.

TXTRCD - Array of 406 halfwords in labeled common /INTTXT/ used as the output buffer for intermediate text records.

TYPE - Halfword integer in labeled common /INTTXT/ used to contain the record type in intermediate text records. TYPE is equivalenced to HEADER (2).

TYPPT - Halfword integer in labeled common /COMS17/ and /COMS49/. It is set to the type of name in SYMTAB and is used for processing internal names within SYMTAB for setting up internal name intermediate text.

UNDFFD - Halfword integer in labeled common /DSCOM/ used by FDXREF and DIAGSM to indicate the total number of undefined function designators in a GOAL compile.

UNDFNM - Halfword integer in labeled common /DSCOM/ used by SYMXRF and DIAGSM to indicate the total number of undefined names in a GOAL compile.

UNDFSN - Halfword integer in labeled common /DSCOM/ used by LBLXRF and DIAGSM to indicate the total number of undefined step numbers in a GOAL compile.

UNRFNM - Halfword integer in labeled common /DSCOM/ used by DIAGSM and SYMXRF to indicate the total number of unreferenced names in a GOAL compile.

UNRFSN - Halfword integer in labeled common /DSCOM/ used by LBLXRF and DIAGSM to indicate the total number of unreferenced step numbers in a GOAL compile.

VART - Halfword integer in labeled common /COMS48/. VART divided by 2 is a value representing the type of internal name being generated as intermediate text.

XRFFLG - Halfword integer in blank common used as a flag to indicate whether or not the cross-reference table has been built. If XRFFLG = 0, BLDXRF is called to allocate and initialize the cross-reference table. This flag is tested by FDXREF, SYMXRF and LBLXRF.

XX - Halfword integer equivalenced to ACTCOM (1). It is used to contain the 'overlay' action routine number, (i.e. SUB xx). It is computed and set in ACTION. If the action is 'resident', XX is set to zero.

YY - Halfword integer equivalenced to ACTCOM (2). It is used to contain the 'resident' action number or the option number for 'overlay' action routines. It is computed and set in ACTION.

Z - Halfword integer in blank common used as a control word to contain the value of the current 'action code'. It is set by PARSER according to the syntax tables. It is used by ACTION to select the appropriate 'action' routines to process the GOAL statements.

ZAP - Array of 10 halfwords in labeled common /SUBCOM/ used to contain hard-coded patches to the syntax table. ZAP can contain up to 5 two-entry (i.e. 'location, patch') patches.

APPENDIX A

GOAL CATALOGED PROCEDURES

A INTRODUCTION

Cataloged procedures have been provided for use with the GOAL Compiler/ Translator, Data Bank, and utility programs in order to minimize the requirement for user preparation of Job Control Language statements. This Appendix describes those procedures currently implemented in the GOAL system and gives examples of their use. Additional information regarding cataloged procedures may be found in the IBM Systems Reference Library publication, Job Control Language Users Guide, reference number GC28-6703.

A description of each procedure is given followed by an example of its use. Listings of the cataloged procedures are also included.

A.1 COMPILER/TRANSLATOR

This section describes the cataloged procedures used for the GOAL Compile and Translate steps. GOAL compiler revision REVØ2 is used in the examples.

GOAL Compiler Step - This step processes the user's GOAL program source deck. Syntax checks are performed, and the standard GOAL compilation listings are generated. In addition the 'Intermediate GOAL' data file is generated. This file is retained for subsequent use in the 'Translator' step.

A.1.1 GOALC - (GOAL Compiler)

This procedure is used when only the 'GOAL Compiler' step is desired. It is useful in syntax debugging and/or listing GOAL source programs. The standard GOAL compilation listings are generated. The 'Intermediate GOAL' data file and symbol table are generated and are retained at completion of this step. GOAL source program decks are used as input. The GOAL compiler, action routines, and syntax table are required for this procedure.

```
//SAMPLE JOB
//       EXEC  GOALC,COMP=REV02
//GOALC.SYSIN  DD  *

....... GOAL PROGRAM DECK .......

/*
```

GOALC Example

A.1.2 GOALCT - (GOAL Compile and Translate)

This procedure is equivalent to GOALC followed by the 'Translation' step. It is used to produce GOAL interpretive code for interpretive execution. The interpretive code is contained in a tabular file which is retained at completion of this procedure. The standard interpretive translator listings may be generated as well as a binary program tape. Interpreter control cards follow the GOAL source deck.

```
//SAMPLE JOB
//      EXEC  GOALCT,COMP=REV02
//GOALC.SYSIN  DD  *

....... GOAL PROGRAM DECK .......

/*
//GOALT.SYSIN  DD  *

....... TRANSLATOR CONTROL CARDS .......

/*
```

GOALCT Example

A.1.3 GOALT - (GOAL Translate)

This procedure may be used when only the 'Translation' step is desired. It is useful when the 'Intermediate GOAL' data file and symbol table have been previously generated. The interpretive code binary program tape is generated and retained at completion of this procedure. This procedure uses the GOAL translator.

```
//SAMPLE JOB
//      EXEC  GOALT,COMP=REV02
//GOALT.SYSIN  DD  *

....... TRANSLATOR CONTROL CARDS .......

/*
```

GOALT Example

A.2 DATA BANK

This section describes the cataloged procedures used for data bank maintenance. Procedures are provided for initializing, updating, and listing data banks.

A.2.1 GOALDBI - (GOAL Data Bank Initialization)

Before the data bank files can be loaded with specific data, they must be created and/or initialized. The process of initializing an existing data bank can be used to delete the contents of the data bank files and to restore their status to initial conditions. This procedure uses the GOAL data bank program DBI.

```
//SAMPLE JOB
//       EXEC  GOALDBI,DISP=NEW    **CREATE AND INITIALIZE DB**
//STEP1.CHARSET DD *
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ=:;><()'+-*/?#$¬&|,.
/*
//STEP1.CONTROL DD *
      8000     TYPE-1 CARD:    TOTAL # RECORDS AVAILABLE IN DATABANK.
        30     TYPE-2 CARD:    TOTAL # DATABANKS TO BE ALLOWED.
/*
```

Example: Create and Initialize Data Bank

```
//SAMPLE JOB
//       EXEC  GOALDBI              **INITIALIZE EXISTING DB**
//STEP1.CHARSET DD *
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ=:;><()'+-*/?#$¬&|,.
/*
//STEP1.CONTROL DD *
      8000     TYPE-1 CARD:    TOTAL # RECORDS AVAILABLE IN DATABANK.
        30     TYPE-2 CARD:    TOTAL # DATABANKS TO BE ALLOWED.
/*
```

Example: Initialize Existing Data Bank

A.2.2 GOALDBUP - (GOAL Data Bank Update)

This procedure enables the user to load specific information into the data bank files. The GOAL compiler, the data bank syntax table, data bank programs MAINT and DCON, and OS/360 sort-merge program IERRCOØØ are used by this procedure. A data bank source deck is used as input.

```
//SAMPLE JOB
//       EXEC  GOALDBUP,COMP=REV02
//GOALC.SYSIN  DD  *

....... DATABANK UPDATE INPUT DECK .......

/*
//STEP1.CHARSET DD *
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ=:;><()'+-*/?#$¬&|,.
/*
//STEP2.SORTFLD DD *
 SORT FIELDS=(1,4,BI,A,13,4,BI,A,17,64,CH,A),SIZE=E1000
/*
```

GOALDBUP Example

A.2.3 GOALDBL - (GOAL Data Bank List)

This procedure enables the user to obtain a summary listing of the entire contents of the data bank files. Data bank program SUPERD is called by GOALDBL.

```
//SAMPLE JOB
//       EXEC  GOALDBL
/*
```

GOALDBL Example

A.3 UTILITIES

This section describes the cataloged procedures used with the GOAL utility programs. Procedures are provided for syntax table initialization and generation, error message file generation, GOAL module updates, and link editing.

A.3.1 Syntax Table Maintenance

A.3.1.1 GOALINTL - (GOAL Syntax Table Initialization)

This procedure initializes the syntax table file. It is run prior to generating any syntax tables (GOALXGEN), and it is not run again unless it is desired to return the file to initial conditions. GOALINTL uses OS/360 utility program IEHPROGM. The input is shown in the example. Output is syntax Table 1 which is used by the program GOALSNTX (cataloged procedure GOALXGEN) to build the GOAL syntax tables.

```
//SAMPLE JOB
// EXEC GOALINTL
//INIT.SYSIN DD *
1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ=:;><()'+-*/?#$¬&|,.
1    1 9                    0     1     0     0     0     0     3    14     5
1   10 9                   90     9     8    -1     0     1     0     0     0
1   19 9                    0     9     1     5    94     1    31     5    96
1   28 9                    9     7    -1     0     2     0     0     0     0
1   37 9                    1    51     1    42    -1     0     1     0     0
1   46 9                    0     0     3    75    -1     0     1     0     0
1   55 9                    0     0     1    75     3    64     9     6    -1
1   64 9                    0     1     0     0     0     0     5    98     1
1   73 9                   75    -1     0     2     0     0     0     0     9
1   82 8                    2     9     3     9     4     9     5    -1
1   90 1                    3
2   91 3  END
1   94 1                    1
2   95 1  =
1   96 1                    1
2   97 1  ;
1   98 1                    1
2   99 1  |
3    1
/*
```

GOALINTL Example

A.3.1.2 GOALXGEN _ (GOAL Syntax Generator)

This procedure is used to generate syntax tables for use by the GOAL compiler. The program used is GOAL utility program GOALSNTX. Input is a syntax table card deck, and output is a GOAL syntax table in the syntax table file.

```
//SAMPLE JOB
//      EXEC  GOALXGEN
//XGEN.SYSIN  DD  *

....... CONTROL CARD .......

....... SYNTAX TABLE DECK .......
/*
```

GOALXGEN Example

A.3.2 GOALDIAG - (GOAL Diagnostic Messages)

This procedure generates the GOAL error message file. The entire error message input deck must be included each time GOALDIAG is run. For a listing of current GOAL error messages, refer to Appendix B of Volume II. GOAL utility program EMSGINIT and OS/360 utility IEHPROGM are called by this procedure.

```
//SAMPLE JOB
//       EXEC  GOALDIAG
//INIT.SYSIN  DD  *
.....GOAL ERROR MESSAGE DECK.....
/*
```

GOALDIAG Example

A.3.3 GOAL Module Updates

A.3.3.1 GOALUPDT - (GOAL Update)

GOAL module updates are accomplished using GOALUPDT except as noted in A.3.3.2 and A.3.3.3 below. This procedure uses OS/360 utility program IEBUPDTE, the FORTRAN compiler IEYFORT, and the Linkage-Editor program IEWL. Both GOAL.SOURCLIB and GOAL.LINKLIB are updated. Procedure GOALLINK must be run before modules in GOAL.LINKLIB can be executed.

```
//SAMPLE JOB
         EXEC  GOALUPDT,NAME=SUB52
//UP.SYSIN  DD  *
.......IEBUPDTE (OS UTILITY) CONTROL CARDS & UPDATES.......
/*
```

GOALUPDT Example

A.3.3.2 GOALFTCL - (GOAL FORTRAN Compile and Link)

This procedure uses OS/360 utility IEBUPDTE, the FORTRAN compiler IEYFORT and the Linkage-Editor program IEWL. Modules updated with this procedure are:

CRL	DBI	GOALINIT	SUPERD
CRR	DCON	GOALSNTX	
DBD	EMSGINIT	MAINT	

External references are resolved and modules in GOAL.LINKLIB are in executable form.

```
//SAMPLE JOB
//       EXEC  GOALFTCL,NAME=SUPERD
//UP.SYSIN  DD  *

.......IEBUPDTE (OS UTILITY) CONTROL CARDS & UPDATES.......

/*
```

GOALFTCL Example

A.3.3.3 GOALASM - (GOAL Assembler Language Module Update)

This procedure is used to update the assembler language modules RCRETN and COMPAR. GOALASM uses the OS/360 assembler program IEUASM and the linkage editor program IEWL.

```
//SAMPLE JOB
//       EXEC  GOALASM,NAME=RCRETN
//UP.SYSIN  DD  *

.......IEBUPDTE (OS UTILITY) CONTROL CARDS & UPDATES.......

/*
```

GOALASM Example

A.2.3.4 GOALLINK - (GOAL Linkage Editor)

This procedure resolves external references in GOAL.LINKLIB. After running GOALLINK modules in GOAL.LINKLIB are in executable form.

```
//SAMPLE JOB
//       EXEC  GOALLINK,COMP=REV02
/*
```

GOALLINK Example

A.4 LISTINGS OF CATALOGED PROCEDURES

```
//GOALC EXEC PGM=&COMP
//STEPLIB  DD  DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT08F001 DD DSN=GOAL.SYNTAX,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT09F001 DD DSN=GOAL.EMESSG,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT10F001 DD UNIT=2314,SPACE=(TRK,10),DCB=BLKSIZE=200
//FT11F001 DD SYSOUT=A,DCB=(BLKSIZE=133,RECFM=UA)
//FT12F001 DD SYSOUT=A,DCB=(BLKSIZE=133,RECFM=UA)
//FT13F001 DD UNIT=2314,SPACE=(TRK,(1,1)),DCB=BLKSIZE=200
//FT14F001 DD DSN=GOAL.INTERTXT,UNIT=2314,VOL=SER=GOAL01,DISP=OLD,
//  DCB=BLKSIZE=200
//FT15F001 DD UNIT=2314,SPACE=(TRK,(10,1)),DCB=BLKSIZE=200
//FT16F001 DD DSN=GOAL.MACROS,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT17F001 DD DSN=GOAL.RPTTBL,UNIT=2314,VOL=SER=GOAL01,DISP=OLD,
//  DCB=BLKSIZE=200
//FT18F001     DD     DSN=GOAL.DATAB1,       ** DATABANK DATASET **
//              UNIT=2314,VOL=SER=GOAL01,
//              DISP=(OLD,KEEP)
//FT19F001     DD     DSN=GOAL.DATAD1,       ** DATABANK DIRECTORY **
//              UNIT=2314,VOL=SER=GOAL01,
//              DISP=(OLD,KEEP)
//FT21F001 DD DUMMY
//FT22F001 DD DSN=GOAL.SYMTAB,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT23F001 DD UNIT=2314,SPACE=(80,(200,200)),
//              DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
```

GOALC

A.4 LISTINGS OF CATALOGED PROCEDURES (Continued)

```
//GOALC EXEC PGM=&COMP
//STEPLIB  DD  DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT08F001 DD DSN=GOAL.SYNTAX,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT09F001 DD DSN=GOAL.EMESSG,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT10F001 DD UNIT=2314,SPACE=(TRK,10),DCB=BLKSIZE=200
//FT11F001 DD SYSOUT=A,DCB=(BLKSIZE=133,RECFM=UA)
//FT12F001 DD SYSOUT=A,DCB=(BLKSIZE=133,RECFM=UA)
//FT13F001 DD UNIT=2314,SPACE=(TRK,(1,1)),DCB=BLKSIZE=200
//FT14F001 DD DSN=GOAL.INTERTXT,UNIT=2314,VOL=SER=GOAL01,DISP=OLD,
//  DCB=BLKSIZE=200
//FT15F001 DD UNIT=2314,SPACE=(TRK,(10,1)),DCB=BLKSIZE=200
//FT16F001 DD DSN=GOAL.MACROS,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT17F001 DD DSN=GOAL.RPTTBL,UNIT=2314,VOL=SER=GOAL01,DISP=OLD,
//  DCB=BLKSIZE=200
//FT18F001      DD     DSN=GOAL.DATAB1,       ** DATABANK DATASET **
//               UNIT=2314,VOL=SER=GOAL01,
//               DISP=(OLD,KEEP)
//FT19F001      DD     DSN=GOAL.DATAD1,       ** DATABANK DIRECTORY **
//               UNIT=2314,VOL=SER=GOAL01,
//               DISP=(OLD,KEEP)
//FT21F001 DD DUMMY
//FT22F001 DD DSN=GOAL.SYMTAB,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT23F001 DD UNIT=2314,SPACE=(80,(200,200)),
//               DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
//GOALT  EXEC  PGM=XLATOR,COND=(8,LT)
//STEPLIB  DD  DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//FT06F001 DD  SYSOUT=A
//FT10F001 DD  DSN=GOAL.INTERTXT,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT11F001 DD  DSN=GOAL.SYMTAB,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT12F001 DD  UNIT=2314,SPACE=(CYL,10)
//FT13F001 DD  UNIT=2314,SPACE=(CYL,1)
//TAPE7    DD  UNIT=(180,,DEFER),LABEL=(,NL),DISP=(NEW,KEEP)
//TAPE9    DD  UNIT=(181,,DEFER),LABEL=(,NL),DISP=(NEW,KEEP)
//FT05F001 DD  DDNAME=SYSIN
```

GOALCT

```
//GOALT  EXEC  PGM=XLATOR
//STEPLIB  DD  DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//FT10F001 DD  DSN=GOAL.INTERTXT,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT11F001 DD  DSN=GOAL.SYMTAB,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT12F001 DD  UNIT=2314,SPACE=(CYL,10)
//FT13F001 DD  UNIT=2314,SPACE=(CYL,1)
//FT06F001 DD  SYSOUT=A
//TAPE7    DD  UNIT=(180,,DEFER),LABEL=(,NL),DISP=(NEW,KEEP)
//TAPE9    DD  UNIT=(181,,DEFER),LABEL=(,NL),DISP=(NEW,KEEP)
//FT05F001 DD  DDNAME=SYSIN
```

GOALT

A.4 LISTINGS OF CATALOGED PROCEDURES (Continued)

```
//              PROC DISP=OLD
//STEP1         EXEC PGM=DBI       **DATABANK INITIAL FORMATTER**
//STEPLIB       DD DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//FT06F001      DD      SYSOUT=A
//*
//FT10F001      DD      DSN=GOAL.DATAB1,      **DATABANK DATASET**
//              UNIT=2314,VOL=SER=GOAL01,
//              SPACE=(172,(8000),,CONTIG),
//              DCB=(RECFM=F,LRECL=172,BLKSIZE=172),
//              DISP=(&DISP,KEEP,DELETE)
//*
//FT11F001      DD DSN=GOAL.DATAD1,           **DATABANK DIRECTORY**
//              UNIT=2314,VOL=SER=GOAL01,
//              SPACE=(1532,(421),,CONTIG),
//              DCB=(RECFM=F,LRECL=1532,BLKSIZE=1532),
//              DISP=(&DISP,KEEP,DELETE)
//*
//FT13F001      DD      DSN=GOAL.UTILD,  **TEMPORARY DIRECTORY OUTPUT
//              UNIT=2314,VOL=SER=GOAL01,
//              SPACE=(1532,(421),,CONTIG),
//              DCB=(RECFM=F,LRECL=1532,BLKSIZE=1532),
//              DISP=(&DISP,KEEP,DELETE)
//*
//FT09F001      DD      DDNAME=CHARSET   **ALPHA CHARACTER REFERENCE**
//FT05F001      DD      DDNAME=CONTROL
```

GOALDBI

A.4 LISTINGS OF CATALOGED PROCEDURES (Continued)

```
//GOALC EXEC PGM=&COMP
//STEPLIB  DD  DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT08F001 DD DSN=GOAL.SYNTAX,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT09F001 DD DSN=GOAL.EMESSG,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT10F001 DD UNIT=2314,SPACE=(TRK,10),DCB=BLKSIZE=200
//FT11F001 DD SYSOUT=A,DCB=(BLKSIZE=133,RECFM=UA)
//FT12F001 DD SYSOUT=A,DCB=(BLKSIZE=133,RECFM=UA)
//FT13F001 DD UNIT=2314,SPACE=(TRK,(1,1)),DCB=BLKSIZE=200
//FT14F001 DD DSN=GOAL.INTERTXT,UNIT=2314,VOL=SER=GOAL01,DISP=OLD,
//  DCB=BLKSIZE=200
//FT15F001 DD UNIT=2314,SPACE=(TRK,(10,1)),DCB=BLKSIZE=200
//FT16F001 DD DSN=GOAL.MACROS,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//FT17F001 DD DSN=GOAL.RPTTBL,UNIT=2314,VOL=SER=GOAL01,DISP=OLD,
//  DCB=BLKSIZE=200
//FT18F001 DD DSN=GOAL.DATAB1,
//         UNIT=2314,VOL=SER=GOAL01,
//         DISP=(OLD,KEEP)
//FT19F001 DD DSN=GOAL.DATAD1,
//         UNIT=2314,VOL=SER=GOAL01,
//         DISP=(OLD,KEEP)
//FT20F001 DD DSN=&&INPUT,UNIT=2314,DISP=(NEW,PASS),SPACE=(TRK,(5,5)),
//         DCB=(BLKSIZE=80,LRECL=80,RECFM=F)
//FT21F001 DD DUMMY
//FT22F001 DD DSN=GOAL.SYMTAB,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
//STEP1      EXEC PGM=MAINT,COND=(4,LT)
//STEPLIB    DD DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//*
//FT06F001   DD     SYSOUT=A
//*
//FT10F001   DD     DSN=GOAL.DATAB1,     **DATABANK DATASET**
//           UNIT=2314,VOL=SER=GOAL01,
//           DISP=(OLD,KEEP)
//*
//FT11F001   DD     DSN=GOAL.DATAD1,      **DATABANK DIRECTORY**
//           UNIT=2314,VOL=SER=GOAL01,
//           DISP=(OLD,KEEP)
//*
//FT12F001   DD     DSN=&&UNSORT,
//           UNIT=2314,
//           SPACE=(TRK,(5,5)),
//           DCB=(RECFM=F,LRECL=80,BLKSIZE=80),
//           DISP=(NEW,PASS)
//*
//FT09F001   DD DDNAME=CHARSET
//*
//FT05F001   DD     DSN=&&INPUT,DISP=(OLD,DELETE)
```

GOALDBUP

(Continued next page)

A.4 LISTINGS OF CATALOGED PROCEDURES (Continued)

```
//STEP2        EXEC PGM=IERRCOOO,   **SORT OF DIRECTORY ENTRIES**
//             COND=(4,LT),
//             PARM='MSG=AP'
//SYSOUT       DD     SYSOUT=A
//SORTLIB      DD     DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01     DD     UNIT=SYSDA,SPACE=(TRK,(100),,CONTIG)
//SORTWK02     DD     UNIT=SYSDA,SPACE=(TRK,(100),,CONTIG)
//SORTWK03     DD     UNIT=SYSDA,SPACE=(TRK,(100),,CONTIG)
//SORTIN       DD     DSN=&&UNSORT,
//             DCB=(RECFM=F,LRECL=80,BLKSIZE=80),
//             DISP=(OLD,DELETE)
//SORTOUT      DD     DSN=&&SORTED,
//             UNIT=2314,
//             SPACE=(TRK,(5,5)),
//             DCB=(RECFM=F,LRECL=80,BLKSIZE=80),
//             DISP=(NEW,PASS)
//SYSIN        DD     DDNAME=SORTFLD
//STEP3        EXEC PGM=DCON,    **DIRECTORY BUILD MODULE**
//             COND=(4,LT)
//STEPLIB DD DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//FT06F001     DD     SYSOUT=A
//FT10F001     DD     DSN=GOAL.DATAB1,     **DATABANK DATASET**
//             UNIT=2314,VOL=SER=GOAL01,
//             DISP=(OLD,KEEP)
//FT11F001     DD     DSN=GOAL.DATAD1,     **DATABANK DIRECTORY**
//             UNIT=2314,VOL=SER=GOAL01,
//             DISP=(OLD,KEEP)
//FT12F001     DD     DSN=&&SORTED,   **SORTED DIRECTORY ENTRIES**
//             DISP=(OLD,DELETE)
//FT13F001     DD     DSN=GOAL.UTILD,  **TEMPORARY DIRECTORY OUTPUT**
//             UNIT=2314,VOL=SER=GOAL01,
//             DISP=(OLD,KEEP)
```

GOALDBUP

A.4 LISTINGS OF CATALOGED PROCEDURES (Continued)

```
//              EXEC PGM=SUPERD
//STEPLIB       DD DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//FT06F001      DD      SYSOUT=A
//FT10F001      DD      DSN=GOAL.DATAB1,
//              UNIT=2314,VOL=SER=GOAL01,
//              DISP=(OLD,KEEP)
//FT11F001      DD      DSN=GOAL.DATAD1,
//              UNIT=2314,VOL=SER=GOAL01,
//              DISP=(OLD,KEEP)
```

GOALDBL

```
//IEHPROGM EXEC PGM=IEHPROGM
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=A
//DD1 DD DSN=GOAL.SYNTAX,UNIT=2314,VOL=SER=GOAL01,DISP=(OLD,DELETE)
//INIT EXEC PGM=GOALINIT,COND=EVEN
//STEPLIB DD DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT08F001 DD DSN=GOAL.SYNTAX,UNIT=2314,VOL=SER=GOAL01,
//              DISP=(NEW,KEEP),SPACE=(CYL,5)
```

GOALINTL

```
//XGEN EXEC PGM=GOALSNTX
//STEPLIB DD DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT08F001 DD DSN=GOAL.SYNTAX,UNIT=2314,VOL=SER=GOAL01,DISP=OLD
```

GOALXGEN

A.4 LISTINGS OF CATALOGED PROCEDURES (Continued)

```
//IEHPROGM EXEC PGM=IEHPROGM
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=A
//DD1 DD DSN=GOAL.EMESSG,UNIT=2314,VOL=SER=GOAL01,DISP=(OLD,DELETE)
//INIT EXEC PGM=EMSGINIT,COND=EVEN
//STEPLIB DD DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT09F001 DD DSN=GOAL.EMESSG,UNIT=2314,VOL=SER=GOAL01,
//            DISP=(NEW,KEEP),SPACE=(80,1000)
```

GOALDIAG

```
//UP EXEC PGM=IEBUPDTE,PARM=MOD
//SYSUT1 DD DSN=GOAL.SOURCLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSUT2 DD DSN=GOAL.SOURCLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSPRINT DD SYSOUT=A
//FORT EXEC PGM=IEYFORT,PARM=(SOURCE,NODECK,LOAD,MAP),COND=(1,LT,UP)
//SYSIN DD DSN=GOAL.SOURCLIB(&NAME),UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSLIN DD DSN=&LOADSET,DISP=(NEW,PASS),UNIT=2314,
//            SPACE=(80,(200,100)),DCB=BLKSIZE=80
//LINK EXEC PGM=IEWL,PARM=(MAP,LET,NCAL),COND=((1,LT,FORT),(1,LT,UP))
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)
//SYSLMOD DD DSN=GOAL.LINKLIB(&NAME),UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=2314,SPACE=(1024,(100,100)),DCB=BLKSIZE=1024
```

GOALUPDT

```
//UP EXEC PGM=IEBUPDTE,PARM=MOD
//SYSUT1 DD DSN=GOAL.SOURCLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSUT2 DD DSN=GOAL.SOURCLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSPRINT DD SYSOUT=A
//FORT EXEC PGM=IEYFORT,PARM=(SOURCE,NODECK,LOAD,MAP),COND=(1,LT,UP)
//SYSIN DD DSN=GOAL.SOURCLIB(&NAME),UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSLIN DD DSN=&LOADSET,DISP=(NEW,PASS),UNIT=2314,
//            SPACE=(80,(200,100)),DCB=BLKSIZE=80
//LINK EXEC PGM=IEWL,PARM=(MAP,LET,LIST),COND=((1,LT,FORT),(1,LT,UP))
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)
//SYSLMOD DD DSN=GOAL.LINKLIB(&NAME),UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
//       DD DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=2314,SPACE=(1024,(100,100)),DCB=BLKSIZE=1024
```

GOALFTCL

A.4 LISTINGS OF CATALOGED PROCEDURES (Continued)

```
//UP EXEC PGM=IEBUPDTE,PARM=MOD
//SYSUT1 DD DSN=GOAL.SOURCLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSUT2 DD DSN=GOAL.SOURCLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSPRINT DD SYSOUT=A
//ASM EXEC PGM=IEUASM,PARM=(LOAD,NODECK,LIST),COND=(1,LE,UP)
//SYSIN DD DSN=GOAL.SOURCLIB(&NAME),UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSGO DD DSN=&LOADSET,DISP=(NEW,PASS),UNIT=SYSSQ,SPACE=(80,(500,200))
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1 DD UNIT=(SYSSQ,SEP=SYSLIB),SPACE=(1700,(500,50))
//SYSUT2 DD UNIT=SYSSQ,SPACE=(1700,(500,50))
//SYSUT3 DD UNIT=(SYSSQ,SEP=(SYSLIB,SYSUT1,SYSUT2)),
//             SPACE=(1700,(500,50))
//LKED EXEC PGM=IEWL,PARM=(MAP,LET,NCAL),COND=((1,LE,UP),(1,LE,ASM))
//SYSLIB DD DSN=SYS1.TELCMLIB,DISP=SHR
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)
//SYSLMOD DD DSN=GOAL.LINKLIB(&NAME),UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=2314,SPACE=(1024,(100,100)),DCB=BLKSIZE=1024
```

GOALASM

```
//LINK EXEC PGM=IEWL,PARM='MAP,LET,OVLY,XCAL,SIZE=(120K,50K)'
//SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
//  DD DSN=GOAL.LINKLIB,UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSLIN DD DSN=GOAL.DATA(LINKDATA),UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSLMOD DD DSN=GOAL.LINKLIB(&COMP),UNIT=2314,VOL=SER=GOAL01,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=2314,SPACE=(1024,(100,100)),DCB=BLKSIZE=1024
```

GOALLINK

APPENDIX B

GOAL DIAGNOSTIC MESSAGES

B.1 INTRODUCTION

Two types of diagnostic messages are output by the GOAL compiler - GOAL system error messages and GOAL compilation error messages. Compilation error messages indicate errors in the GOAL input source data. The error conditions are flagged, error messages are printed, and compilation continues. System errors indicate that the GOAL compiler cannot continue to process input source data due to internal GOAL system conditions. The condition code is set to 16 and the JOB step is terminated.

B.2 GOAL SYSTEM ERROR MESSAGES

GOAL system error messages are printed in the following format:

```
*** TERMINAL ERROR -1 ***
```

The GOAL system error conditions are:

System Error Number	Description
1	The maximum number of statement pointers in STPTAB has been exceeded
2	Not Used
3	Overflow of STMTAB has occurred
4	'Computed GO TO' range error in 'Action' routines
5	Overflow of cross-reference table XRFTAB has occurred
6	Overflow of macro file has occurred
7	Invalid text length (less than zero or greater than 406) passed to routine TXTOUT
8	Not Used
9	Not Used
10	Overflow of SYMTAB has occurred.

In addition, data bank maintenance routines YEFIND and SEEKDB use system error numbers 500 and 501 to indicate logical error conditions internal to the data bank routines. No user error is indicated in these cases.

B.3 GOAL COMPILATION ERROR MESSAGES

The following error messages indicate error conditions detected in GOAL source input data. These errors are flagged in the expanded source listing and are defined in the GOAL COMPILER DIAGNOSTIC SUMMARY.

100 INVALID ROW DESIGNATOR OR KEYWORD 'ROW' IS MISSING .
101 INVALID COLUMN INDEX NAME OR COLUMN INTEGER NUMBER.
102 INVALID ROW INDEX NAME OR ROW INTEGER NUMBER.
103 INVALID LIST INDEX NAME OR LIST INTEGER NUMBER.
104 INVALID REFERENCE OR KEYWORD FOLLOWING KEYWORD 'SEND' OR 'APPLY'.
106 INVALID OR MISSING EXTERNAL DESIGNATOR -FROM-
108 INVALID OR MISSING EXTERNAL DESIGNATOR - TO -
110 INVALID INTERNAL NAME WHICH MUST BE DECLARED AS A STATE VALUE.
112 INVALID INTERNAL NAME OR STATE WHICH MUST BE DECLARED AS STATE VALUES.
114 INVALID INTERNAL NAME WHICH MUST NOT BE DECLARED AS STATE OR TEXT .
122 INVALID INTEGER NUMBER OF ENTRIES.
124 INVALID INTERNAL NAME OR STATE .
128 INVALID NUMBER NAME.
129 INVALID NUMBER NAME. THIS NAME IS PREVIOUSLY DEFINED.
130 INVALID NUMBER PATTERN OR NUMBER.
131 INVALID NUMERIC VALUE - MUST BE 1-4 DIGITS.
132 INVALID QUANTITY NAME.
133 INVALID QUANTITY NAME. THIS NAME IS PREVIOUSLY DEFINED .
134 INVALID QUANTITY VALUE.
136 INVALID STATE NAME.
137 INVALID STATE NAME. THIS NAME IS PREVIOUSLY DEFINED.
138 INVALID STATE VALUE.
140 INVALID TEXT NAME.
141 INVALID TEXT NAME. THIS NAME IS PREVIOUSLY DEFINED.
142 INVALID NUMERIC LIST NAME.
143 INVALID NUMERIC LIST NAME. THIS NAME IS PREVIOUSLY DEFINED.
144 INVALID NUMERIC TABLE NAME
145 INVALID NUMERIC TABLE NAME . THIS NAME IS PREVIOUSLY DEFINED.
146 INVALID INTEGER NUMBER OF COLUMNS.
147 INVALID INTEGER NUMBER OF COLUMNS. THE LIMITS ARE 0 THROUGH 45.
148 INVALID INTEGER NUMBER OF ROWS.
149 INVALID INTEGER NUMBER OF ROWS. THE LIMITS ARE 1 THROUGH 45.
150 INVALID COLUMN NAME.
151 INVALID COLUMN NAME OR KEYWORD 'COLUMN' IS MISSING.
152 INVALID QUANTITY LIST NAME.
153 INVALID QUANTITY LIST NAME. THIS NAME IS PREVIOUSLY DEFINED.
154 INVALID QUANTITY TABLE NAME.
155 INVALID QUANTITY TABLE NAME. THIS NAME IS PREVIOUSLY DEFINED.
156 INVALID STATE LIST NAME.
157 INVALID STATE LIST NAME. THIS NAME IS PREVIOUSLY DEFINED.
158 INVALID STATE TABLE NAME.
159 INVALID STATE TABLE NAME. THIS NAME IS PREVIOUSLY DEFINED.
160 INVALID INTERNAL NAME OR NUMBER PATTERN .

```
162 INVALID TEXT LIST NAME.
163 INVALID TEXT LIST NAME. THIS NAME IS PREVIOUSLY DEFINED.
164 INVALID INTEGER NUMBER OF CHARACTERS.
165 INVALID INTEGER NUMBER OF CHARACTERS. THE LIMITS ARE 1 THROUGH 132.
166 INVALID TEXT TABLE NAME.
167 INVALID TEXT TABLE NAME.  THIS NAME IS PREVIOUSLY DEFINED.
168 INVALID DELAY STATEMENT FOLLOWING THE VERB DELAY OR WAIT.
172 INVALID REFERENCE OR KEYWORD FOLLOWING THE VERB ISSUE .
173 INVALID LEAVE STATEMENT - LEAVE CAN ONLY BE USED WITHIN A SUBROUTINE
174 INVALID RESUME STATEMENT.
175 INVALID LEAVE STATEMENT.
176 INVALID PERFORM SUBROUTINE STATEMENT FOLLOWING THE SUBROUTINE NAME.
180 INVALID RECORD DATA STATEMENT FOLLOWING THE KEYWORD DISPLAY,PRINT OR RECORD.
182 INVALID STEP NUMBER OR KEYWORD 'ALL' IS MISSING.
184 INVALID TEXT, NAME OR FUNCTION DESIGNATOR FOLLOWING THE VERB REPLACE.
186 INVALID TEXT OR KEYWORD 'ENTRY' IS MISSING FOLLOWING THE VERB REQUEST.
190 INVALID REFERENCE OR KEYWORD 'PRESENT VALUE OF' FOLLOWING THE VERB SET.
195 INVALID WHEN INTERRUPT STATEMENT FOLLOWING THE KEYWORD 'OCCURS'.
200 THE NUMBER OF ENTRIES INITIALIZED EXCEEDS THE NUMBER SPECIFIED.
201 THE NUMBER OF COLUMN TITLES EXCEEDS THE SPECIFIED NUMBER OF COLUMNS.
202 THE NUMBER OF ENTRIES INITIALIZED IS LESS THAN THE NUMBER SPECIFIED.
203 THE NUMBER OF COLUMN TITLES IS LESS THAN THE SPECIFIED NUMBER OF COLUMNS.
204 THE FUNCTION DESIGNATOR SPECIFIED IS NOT DEFINED IN THE DATA BANK.
206 INVALID ROW FUNCTION DESIGNATOR. IT IS PREVIOUSLY DEFINED IN THIS TABLE.
210 INVALID COLUMN TITLE NAME. THIS NAME IS PREVIOUSLY DEFINED IN THIS TABLE.
212 EXECUTION RATE AS SPECIFIED IS GREATER THAN TEN MINUTES.
214 CONCURRENT STATEMENT DOES NOT HAVE A STEP NUMBER.
216 CORRESPONDENCE IS INVALID (SHOULD BE 1 TO 1, 1 TO MANY OR MANY = MANY)
218 INVALID NUMERIC FORMULA (UNBALANCED PARENTHESES)
220 INVALID INTERNAL NAME (NOT DECLARED AS NUMERIC OR QUANTITY)
222 INVALID INTERNAL NAME (NOT A SINGLE ELEMENT)
224 INVALID NUMERIC FORMULA (SIZE EXCEEDS COMPILER CAPACITY)
228 FUNCTION DESIGNATOR SPECIFIED IS NOT A SUBROUTINE PARAMETER.
300 INVALID MACRO LABEL- DOES NOT START WITH A LETTER.
301 INVALID MACRO LABEL- LONGER THAN 32 CHARACTERS.
302 INVALID MACRO LABEL- CONTAINS AN ILLEGAL CHARACTER.
303 INVALID MACRO LABEL- MACRO LABEL IS MULTI-DEFINED.
304 INVALID MACRO PARAMETER - DOES NOT START WITH A LETTER.
305 INVALID MACRO PARAMETER - LONGER THAN 32 CHARACTERS.
306 INVALID MACRO PARAMETER - CONTAINS AN ILLEGAL CHARACTER.
307 INVALID MACRO PARAMETER - MACRO PARAMETER IS MULTI-DEFINED.
308 EXPECTED SEMICOLON ';' NOT FOUND AFTER PROCESSING THE 10 MAXIMUM PARAMETERS.
309 EITHER COMMA ',' OR SEMICOLON ';' WAS OMITTED.
```

310 LEFT PARENTHESIS '(' MISSING ON PARAMETER FOLLOWING COMMA.
311 MACRO TO BE EXPANDED AND/OR EXECUTED IS NOT DEFINED.
312 MACRO TO BE EXPANDED AND/OR EXECUTED NEEDS PARAMETERS - NONE WERE SUPPLIED.
313 INVALID SUBSTITUTION PARAMETER - CONTAINS AN ILLEGAL CHARACTER.
314 INVALID SUBSTITUTION PARAMETER - CONTAINS NO CHARACTERS.
315 NUMBER OF PARAMETERS IN STATEMENT AND MACRO ARE NOT THE SAME.
316 NUMBER OF PARAMETERS IN STATEMENT EXCEEDS NUMBER OF PARAMETERS IN MACRO.
317 INVALID SUBSTUTUTION PARAMETER - LONGER THAN 79 CHARACTERS.
318 INVALID MACRO BODY - CONTAINS NO CHARACTERS.
350 INVALID CHARACTER STRING - CONTAINS AN ILLEGAL CHARACTER.
351 INVALID CHARACTER STRING - CONTAINS MORE THAN 32 CHARACTERS.
352 INVALID REPLACEMENT CHARACTER STRING. CONTAINS MORE THAN 80 CHARACTERS.
353 INVALID REPLACEMENT CHARACTER STRING. CONTAINS AN ILLEGAL CHARACTER.
354 REPLACEMENT NAME, CHARACTER STRING OR FUNCTION DESIGNATOR IS MULTI-DEFINED
400 NUMBER OF DATA BANKS IN USE HAS EXCEEDED THE MAXIMUM OF 10.
402 DATA BANK SPECIFIED IS ALREADY IN USE.
406 INVALID DATA BANK NAME. THE DATA BANK NAME IS MULTI-DEFINED.
408 UNABLE TO FREE DATA BANK AS NONE IS BEING USED AT THIS TIME.
410 SPECIFIED DATA BANK NAME DOES NOT EXIST.
412 UNABLE TO FREE DATA BANK AS IT IS NOT IN USE AT THIS TIME.
413 LABEL ERROR - THE STATEMENT FOLLOWING AN UNCONDITIONAL GO TO IS NOT NUMBERED
414 STRUCTURAL ERROR ** PREAMBLE STATEMENT FOUND IN PROCEDURAL BODY.
415 SYMBOL TABLE OVERFLOW HAS OCCURRED. A MAXIMUM OF 9999 ENTRIES IS ALLOWED.
800 INVALID ADDRESS - MUST BE 1-4 DIGITS.
802 INVALID COMPARISON TEST.
804 INVALID DATA BANK NAME.
805 INITIALIZATION OF REFERENCED **SUBROUTINE PARAMETER** NAME IS NOT ALLOWED.
806 INVALID OR MISSING EXTERNAL DESIGNATOR.
807 END PROGRAM STATEMENT IS INVALID DURING A SUBROUTINE COMPILATION.
808 INVALID FUNCTION DESIGNATOR.
809 END SUBROUTINE STATEMENT IS INVALID DURING A PROGRAM COMPILATION.
810 INVALID NUMBER,NUMBER PATTERN,QUANTITY,STATE,TEXT OR INTERNAL NAME.
812 INVALID INTEGER NUMBER.
814 INVALID INTERNAL NAME.
816 INVALID OR MISSING REFERENCE FOLLOWING THE COMMA.
826 INVALID NUMERIC FORMULA.
828 INVALID OUTPUT EXCEPTION.
829 INVALID NAME OR FUNCTION DESIGNATOR.
830 INVALID SUBROUTINE PARAMETER (NAME OR FUNCTION DESIGNATOR).
832 INVALID OR MISSING PROGRAM NAME.
834 INVALID QUANTITY OR INTERNAL NAME .
836 INVALID REVISION LABEL.
838 INVALID ROW DESIGNATOR.

841 INVALID STEP NUMBER. THIS STEP NUMBER IS PREVIOUSLY DEFINED.
842 INVALID STEP NUMBER.
843 INVALID PERFORM PROGRAM OR PERFORM SUBROUTINE STATEMENT.
844 INVALID SUBROUTINE NAME.
845 BEGIN PROGRAM FOUND DURING A PROGRAM COMPILATION.
846 INVALID TABLE NAME.
847 INVALID FORTRAN SUBROUTINE NAME.
848 INVALID TEXT CONSTANT.
849 A TEXT CONSTANT ENTRY EXCEEDED THE MAXIMUM NUMBER OF CHARACTERS SPECIFIED.
850 INVALID TIME VALUE.
852 INVALID FUNCTION DESIGNATOR TYPE IN THE SPECIFY STATEMENT.
853 INVALID ROW FUNCTION DESIGNATOR TYPE. MUST BE A LOAD OR SENSOR ANALOG.
854 INVALID ROW FUNCTION DESIGNATOR TYPE. MUST BE A LOAD OR SENSOR DISCRETE.
855 INVALID ROW FUNCTION DESIGNATOR TYPE. MUST BE A SYSTEM FUNCTION DESIGNATOR.
856 THE NUMBER OF ROW FUNCTION DESIGNATORS EXCEEDS THE NUMBER OF ROWS.
857 THE NUMBER OF ROW FUNCTION DESIGNATORS IS LESS THEN THE NUMBER OF ROWS.
900 KEYWORD NOT FOUND - AND.
901 KEYWORD NOT FOUND - RETURN.
902 KEYWORD NOT FOUND - AND SAVE AS.
903 KEYWORD NOT FOUND - ADDRESS.
904 KEYWORD NOT FOUND - AS.
907 KEYWORD NOT FOUND - READINGS OF.
908 KEYWORD NOT FOUND - CHARACTERS.
909 KEYWORD NOT FOUND - CRT, PRINTER, TAPE, INTERRUPT, OR FLAG.
910 KEYWORD NOT FOUND - DATABANK OR MACRO.
911 KEYWORD NOT FOUND - ANALOG, CLOCK, OR DISCRETE.
912 KEYWORD NOT FOUND - ENTRIES.
913 KEYWORD NOT FOUND - EXCEPTIONS.
914 KEYWORD NOT FOUND - EQUAL TO OR =.
916 KEYWORD NOT FOUND - FROM
918 KEYWORD NOT FOUND - LOAD OR SENSOR OR SYSTEM.
920 KEYWORD NOT FOUND - OCCURS.
922 KEYWORD NOT FOUND - UNTIL.
924 KEYWORD NOT FOUND - PRESENT VALUE OF.
925 KEYWORD NOT FOUND - COLUMNS.
926 KEYWORD NOT FOUND - ROWS AND.
927 KEYWORD NOT FOUND - REVISION.
930 KEYWORD NOT FOUND - SUBROUTINE.
934 KEYWORD NOT FOUND - TIMES.
938 KEYWORD NOT FOUND - TO
939 KEYWORD NOT FOUND - TYPE.
940 KEYWORD NOT FOUND - WITH.
941 KEYWORD NOT FOUND - WITH ENTRIES.

```
944 KEYWORD NOT FOUND - WITH A MAXIMUM OF, EQUAL TO OR =.
945 BEGIN PROGRAM OR BEGIN SUBROUTINE FOUND DURING A SUBROUTINE COMPILATION.
946 KEYWORD NOT FOUND - PERFORM PROGRAM, VERIFY, DISPLAY, PRINT, OR RECORD.
948 KEYWORD NOT FOUND - NUMBER, QUANTITY, STATE OR TEXT.
952 KEYWORD NOT FOUND - PROGRAM OR SUBROUTINE.
954 KEYWORD NOT FOUND - AND INDICATE RESTART LABELS OR SEMICOLON ';'.
986 KEYWORD NOT FOUND - THEN OR COMMA ',' .
987 INVALID PAGE NUMBER FOLLOWING THE WORD PAGE. LIMITS ARE 1-999.
988 INVALID LINE SIZE FOLLOWING PAGE SIZE. LIMITS ARE 80-110.
989 INVALID PAGE SIZE FOLLOWING THE WORD LINE. LIMITS ARE 1-32767.
990 INVALID DATE TEXT CONSTANT FOLLOWING THE WORD DATE. LIMITS ARE 1-8.
991 INVALID TITLE TEXT CONSTANT FOLLOWING THE WORD TITLE. LIMITS ARE 1-100.
992 INVALID SEQUENCE FIELD NUMBER FOLLOWING THE WORD SEQ. LIMITS ARE 0-10.
993 INVALID COMPOUND COMPILER CONTROL CARD.
994 INVALID COMPILER CONTROL CARD.
995 THIS STATEMENT IS NOT RECOGNIZED AS A GOAL STATEMENT .
996 EXPECTED DOUBLE DOLLAR SIGN '$$' NOT FOUND
997 END STATEMENT NOT FOUND - SOURCE DECK DEPLETED - COMPILATION TERMINATED.
998 EXPECTED COMMA ',' NOT FOUND.
999 EXPECTED SEMICOLON ';' NOT FOUND.
```