

Report No. 75-0045
Contract No. NAS8-26990

(NASA-CR-144099) HAL/SM SYSTEM SOFTWARE
REQUIREMENTS SPECIFICATION (M&S Computing,
Inc.) 146 p HC [REDACTED] CSCI 09B

N76-14844

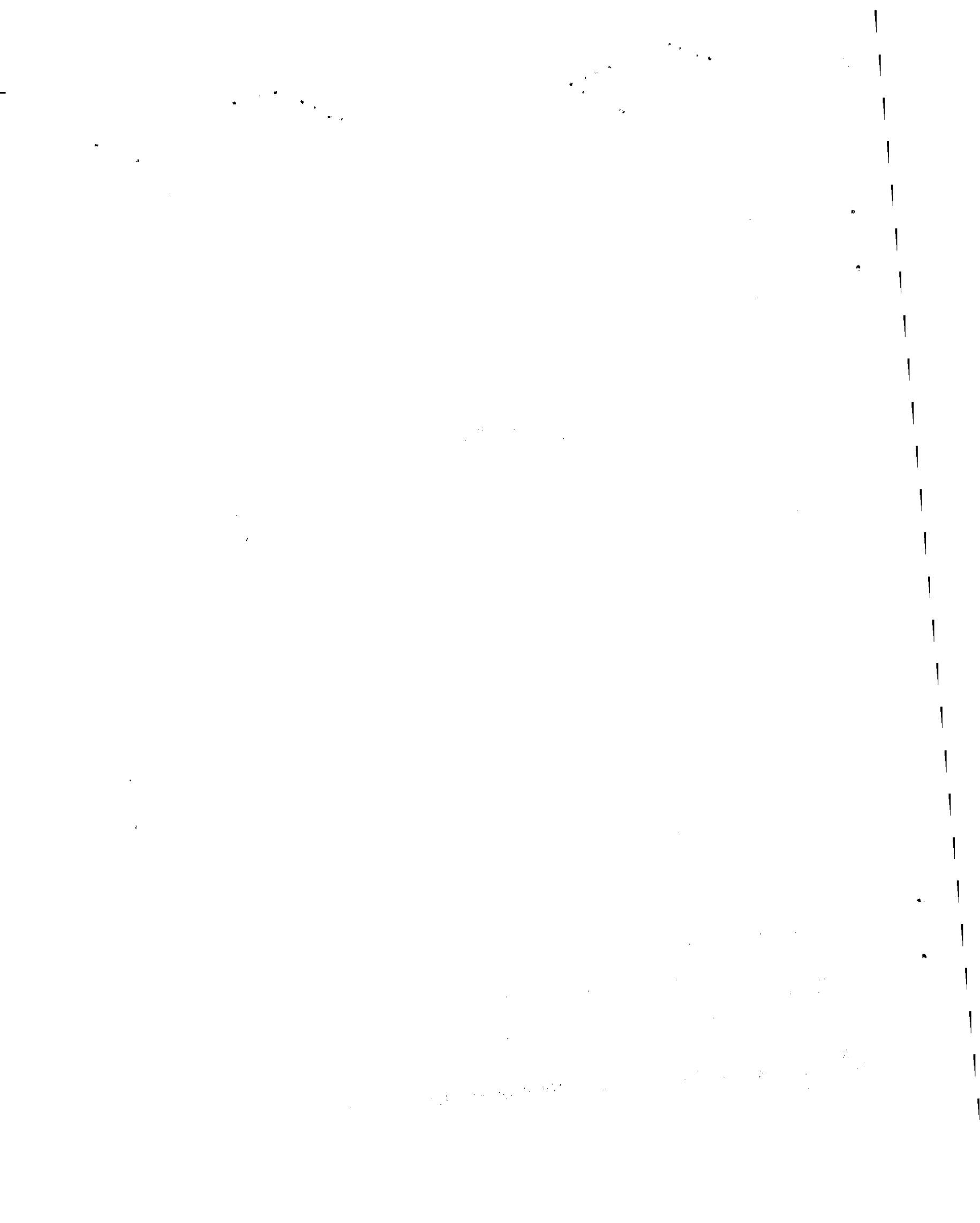
g3/61 Unclass
 06780

HAL/SM SYSTEM
SOFTWARE REQUIREMENTS SPECIFICATION

December 1, 1975

Prepared for:

George C. Marshall Space Flight Center
NASA
Marshall Space Flight Center, Alabama



PREFACE

This is a reference manual document describing the requirements for the HAL/SM programming system to be developed to provide the ability to use the HAL programming language in the SUMC/MOSS environment.

This document is partially based on, supplements, and in part, supercedes the Higher Order Language (HOL) Preprocessor Requirements Specification Document (Reference 8). As such, this document and the HAL/SM Language Specification (Reference 1) shall be the final controlling specifications for HAL/SM software.

Section 1 of this document discusses the basic structure and major objectives of the system. Section 2 describes the major subsystems of the implementation and their functional requirements. Section 3 details the interfaces between the major subsystems and implicitly defines their processing requirements. Section 4 discusses restrictions and limitations inherent in the implementation.

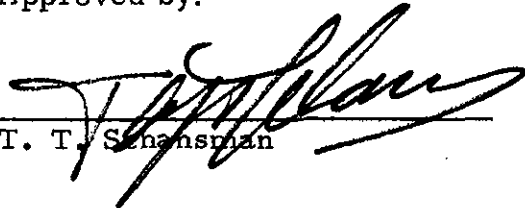
Prepared by:

C. Ross
G. P. W. Williams, Jr.

Project Manager:

J. L. Pruitt

Approved by:


T. T. Schansman

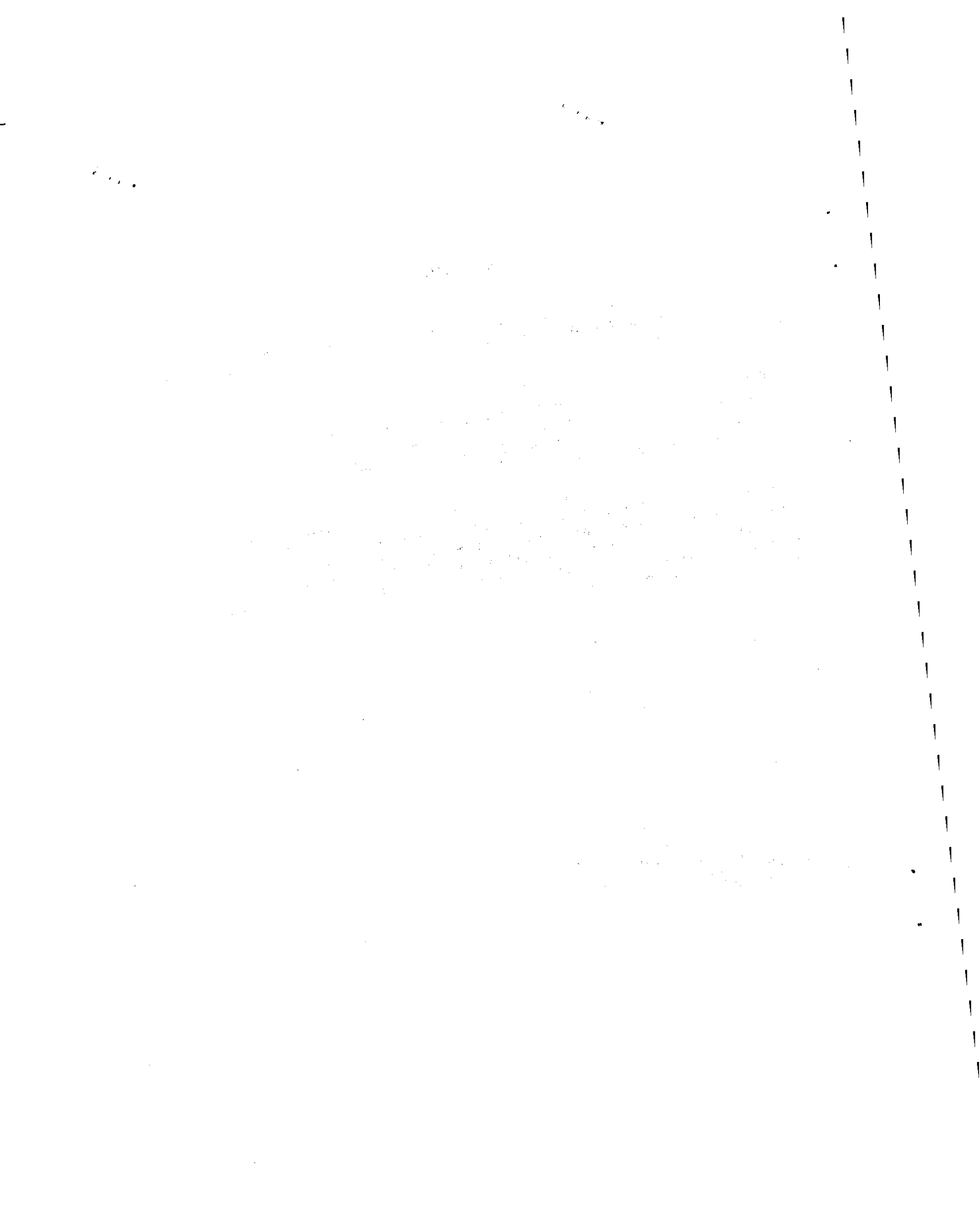


TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
LIST OF ACRONYMS	v
LIST OF FIGURES	vii
1. INTRODUCTION	1
2. SUBSYSTEMS	3
2.1 Preprocessor Subsystem	3
2.1.1 Inputs	5
2.1.2 Symbolic Library Interface	7
2.1.3 M&CD Interface	7
2.1.4 Display Data Interface Utility (DDIU) Interface	7
2.1.5 Listings	7
2.1.6 Diagnostic Information	8
2.1.7 HAL/S-360 Compiler Interface	8
2.1.8 User Interface	8
2.2 HALLINK Subsystem	9
2.3 RTL Subsystem	9
2.3.1 Efficiency Criteria	9
2.3.2 Modifications to Support MOSS External Interface	12
2.3.3 Modifications to Support HAL/SM Calling Sequences	12
2.3.4 Unsupported Functions	12
2.3.5 Unsupported Execution-Time JCL Options	13
3. INTERFACES	15
3.1 Preprocessor	15
3.1.1 TASK Header	16
3.1.2 COMPOOL Header	17
3.1.3 FUNCTION Header	18
3.1.4 PROCEDURE Header	19
3.1.5 REPLACE Statement	20

TABLE OF CONTENTS
(continued)

<u>Section</u>		<u>Page</u>
	3.1.6 JOB Attribute	21
	3.1.7 Type Specification	23
	3.1.8 Initialization of DCW-Type Data	25
	3.1.9 ON ERROR Statement (Form 1)	27
	3.1.10 ON ERROR Statement (Form 2)	29
	3.1.11 OFF ERROR Statement	31
	3.1.12 SIGNAL Statement	33
	3.1.13 RESET Statement	34
	3.1.14 TERMINATE Statement	35
	3.1.15 ABORT Statement	37
	3.1.16 LOG Statement	39
	3.1.17 UNLOCK Statement	40
	3.1.18 LOAD Statement	41
	3.1.19 INITIATE Statement	42
	3.1.20 DELETE Statement	43
	3.1.21 SCHEDULE Statement	44
	3.1.22 WAIT Statement	49
	3.1.23 ALERT Statement	53
	3.1.24 AVERAGE AI Statement	55
	3.1.25 READ ERP Statement	57
	3.1.26 ISSUE Statement	59
	3.1.27 SET Discrete Statement	61
	3.1.28 APPLY Analog Statement	63
	3.1.29 DISPLAY TO OPERATOR Statement	65
	3.1.30 DISPLAY CONTROL Statement	67
	3.1.31 Display Data Statement	69
	3.1.32 Modify VCW Statement	73
	3.1.33 REQUEST Keyboard Statement	74
	3.1.34 SELECT Statement	76
	3.1.35 RELEASE Statement	78
	3.1.36 CHANNEL Control Statement	80
	3.1.37 CRITICAL SECTION	82
	3.1.38 Time Literals	84
	3.1.39 EVENT Variable	85
	3.1.40 OPEN and CLOSE Literals	86
3.2	HALLINK	87
3.3	RTL	88
	3.3.1 ON ERROR Interface	89
	3.3.2 OFF ERROR Interface	91

TABLE OF CONTENTS
(continued)

<u>Section</u>	<u>Page</u>
3.3.3 SIGNAL Interface	92
3.3.4 RESET Interface	93
3.3.5 CANCEL Interface	94
3.3.6 TERMINATE Interface	95
3.3.7 ABORT Interface	96
3.3.8 LOG Interface	97
3.3.9 UNLOCK Interface	98
3.3.10 LOAD Interface	99
3.3.11 INITIATE Interface	100
3.3.12 DELETE Interface	101
3.3.13 SCHEDULE Interface	102
3.3.14 WAIT Interface	104
3.3.15 ALERT Interface	106
3.3.16 AVERAGE Interface	108
3.3.17 READ ERP Interface	109
3.3.18 ISSUE Interface	110
3.3.19 SET DISCRETE Interface	111
3.3.20 APPLY ANALOG Interface	112
3.3.21 WRITE TO OPERATOR Interface	114
3.3.22 DISPLAY CONTROL Interface	115
3.3.23 DISPLAY DATA Interface	117
3.3.24 REQUEST KEYBOARD Interface	122
3.3.25 SELECT Interface	123
3.3.26 RELEASE Interface	124
3.3.27 CHANNEL CONTROL Interface	125
3.3.28 CRITICAL SECTION Interface	126
3.3.29 EVENT VARIABLE Interface	127
3.3.30 TIME Interface	128
3.3.31 DATE Interface	129
3.3.32 SPIOs Interface	130
3.3.33 Interrupt and Error Handling Modules	131
3.3.34 Miscellaneous Modules	132
3.3.35 LOCK Interface	133
3.3.36 RTL Modules to be Deleted	134
 3.4 User	 136
3.4.1 OS JCL Procedures	136
3.4.2 MOSS JCL	136

TABLE OF CONTENTS
(continued)

<u>Section</u>	<u>Page</u>
4. RESTRICTIONS AND LIMITATIONS	137
4.1 HAL/S Dependencies	137
4.2 Separation of Host and Target Machines	137
4.3 Diagnostic Capabilities	137
4.4 RTL Reentrancy	138
REFERENCES	139

LIST OF ACRONYMS

C&D	Control and Display
CVT	Concept Verification
DDIU	Display Data Interface Utility
I/O	Input/Output
JCL	Job Control Language
MOSS	Modular Operating System for the SUMC
PAF	Program Access File
RTL	Run Time Library
SUMC	Space Ultrareliable Modular Computer
SUMC-S	SUMC-Simplex
SVC	Supervisor Call

(BLANK)

LIST OF FIGURES

<u>Figure No.</u>	<u>Title</u>	<u>Page</u>
2-1	HAL/SM Subsystems and System Flow	4
2-2	HAL/SM Preprocessor Subsystem	6
2-3	HALLINK Subsystem	10
2-4	HAL/SM Run-Time Library (RTL)	11

PRECEDING PAGE BLANK NOT FILMED

(BLANK)

1. INTRODUCTION

The HAL/SM programming system shall implement a version of the HAL programming language (see Reference 9) which has been specifically adapted for running in the Concept Verification Test (CVT) environment on the Space Ultrareliable Modular Computer - Simplex (SUMC-S) under the Modular Operating System for the SUMC (MOSS). The HAL/SM programming language is defined in the HAL/SM Language Specification (Reference 1, referred to herein as the Language Spec); familiarity with the Language Spec shall be assumed throughout this document.

The HAL/SM programming system shall be implemented as an adaptation of the HAL/S-360 Compiler System, Release 11.0 (see References 3 through 5), and shall consist of a HAL/SM Preprocessor, the HAL/S-360 Compiler, and modified versions of the HAL/S-360 HALLINK program and Run Time Library (RTL). The preprocessor, compiler, and HALLINK (collectively referred to herein as the language processor system) shall all operate on the IBM S/360-370 family of computers under suitable operating systems as specified in Reference 4. The HAL/SM object programs produced by the language processor system when properly combined with the RTL and linked in the MOSS environment (see References 6 and 7) shall be capable of executing under and utilizing the full capabilities of the MOSS Operating System on the SUMC-S computer system.

(BLANK)

2. SUBSYSTEMS

The major subsystems of the HAL/SM system shall be as follows (see Figure 2-1):

- o HAL/SM Preprocessor - converts HAL/SM source language modules into HAL/S source language modules, performs minimal syntax verification, provides various support features which cannot be conveniently or adequately performed by the HAL/S-360 compiler (e.g., M&CD reference verification, C&D display message data set generation).
- o HAL/S-360 Compiler - converts HAL/S source language modules into IBM S/360 compatible object modules, performs complete syntax verification.
- o HALLINK - combines HAL/S object modules into load modules, calculates run-time stack size requirements and adds the stack to the load module, adds and/or deletes certain other CSECTS from the generated load module depending on options specified.
- o Run Time Library - when properly linked with each HAL/SM task, provides computational routines and MOSS interface routines to support various features of the HAL/SM language.

The functions and interactions are described in more detail in the remainder of this section on an individual subsystem basis, except for the compiler. For more information on the compiler, see References 2 through 5.

2.1 Preprocessor Subsystem

This subsection describes the functional requirements for the HAL/SM preprocessor. In this implementation of HAL/SM, the combination of the preprocessor and the HAL/S-360 compiler takes the place of a HAL/SM-360 compiler. This approach to language translation has the advantage of being considerably less costly than a compiler development effort. To take full advantage of this fact, the requirements for the preprocessor have been specified with the following ground rules in mind:

- o Minimize complexity by making no modifications to the HAL/S-360 compiler.

HAL/SM SUBSYSTEMS AND SYSTEM FLOW

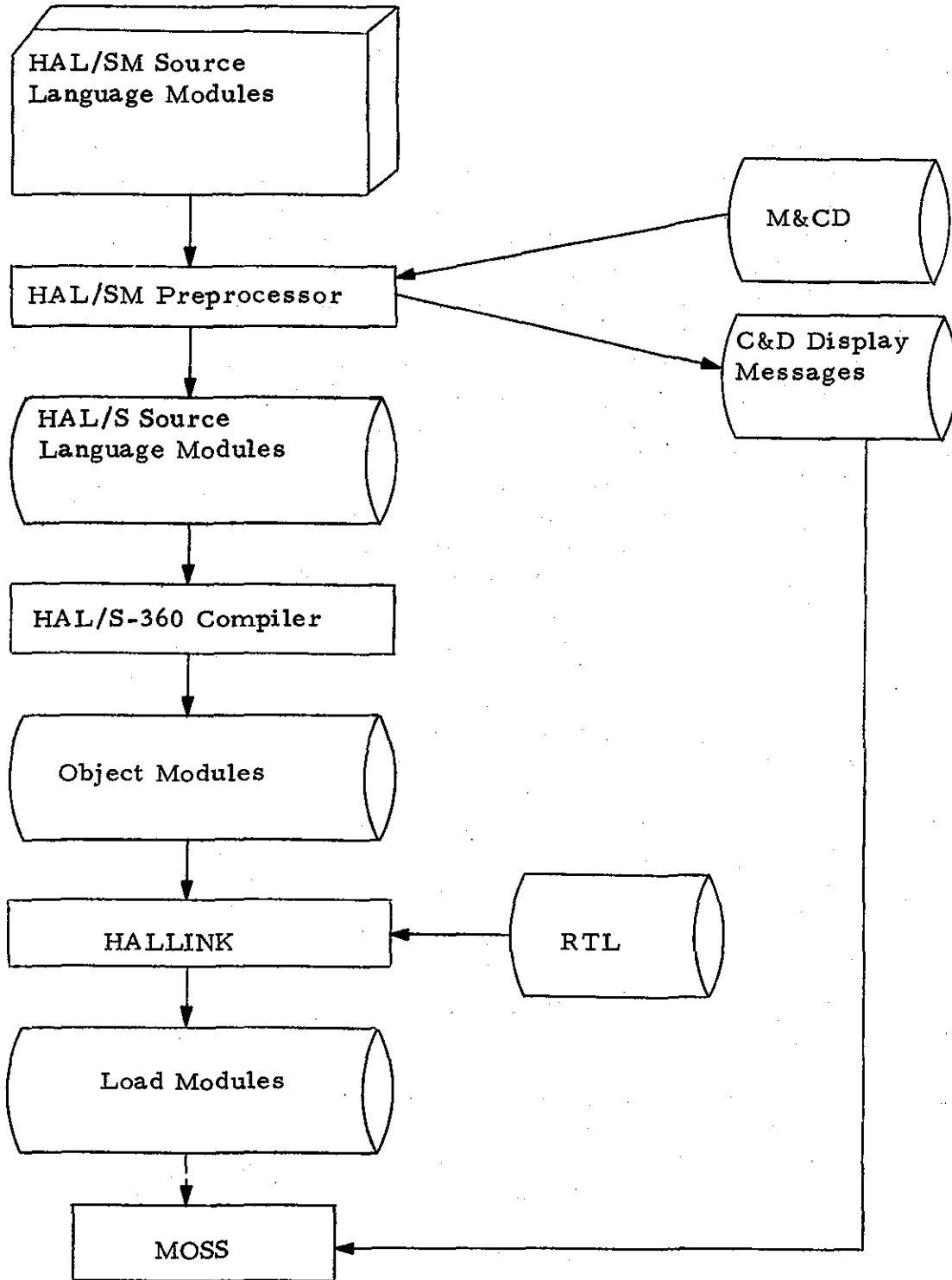


Figure 2-1

- o Perform no processing in the preprocessor which can be performed by the HAL/S-360 compiler.

Within these constraints the requirements listed in the following paragraphs have been defined such that the preprocessor will provide the same facilities as would otherwise be provided by a HAL/SM compiler. The single major area where this is not possible is in syntax error detection and diagnostic information (see Section 2.1.6).

The remainder of this subsection is divided into detailed requirements specifications by major functional area of the processing performed by the preprocessor (see Figure 2-2).

2.1.1 Inputs

The preprocessor shall accept two classes of primary inputs: HAL/SM symbolic source code and directives. An additional input shall be the Program Access File.

Symbolic Source Code: The preprocessor shall accept HAL/SM source code in the one-dimensional form defined in the Language Specification. Symbolic source code may be presented to the preprocessor from either a primary input data set (default; normally from the card reader) or a secondary input data set, as specified by preprocessor directives. The secondary input data set is called the Symbolic Library and contains block templates generated by the preprocessor and other standard symbolic modules which may be placed in the library using standard IBM supplied utility programs.

Directives: Directives shall provide control information required by the preprocessor and the compiler. The preprocessor shall accept two types of directives:

- o HAL/SM preprocessor directives (denoted by a "P" in column 1 of the input record).
- o HAL/S-360 compiler directives (denoted by a "D" in column 1 of the input record).

Certain of the directives defined for the HAL/S-360 compiler (see Reference 5) shall be restricted from use by the preprocessor in the HAL/SM system (TBD). These directives shall be replaced by preprocessor directives of the same form and intent. All other compiler directives shall be passed by the preprocessor to the compiler in the compiler's primary input data set. Additional preprocessor directives shall be defined to control optional processing performed by the preprocessor.

HAL/SM PREPROCESSOR SUBSYSTEM

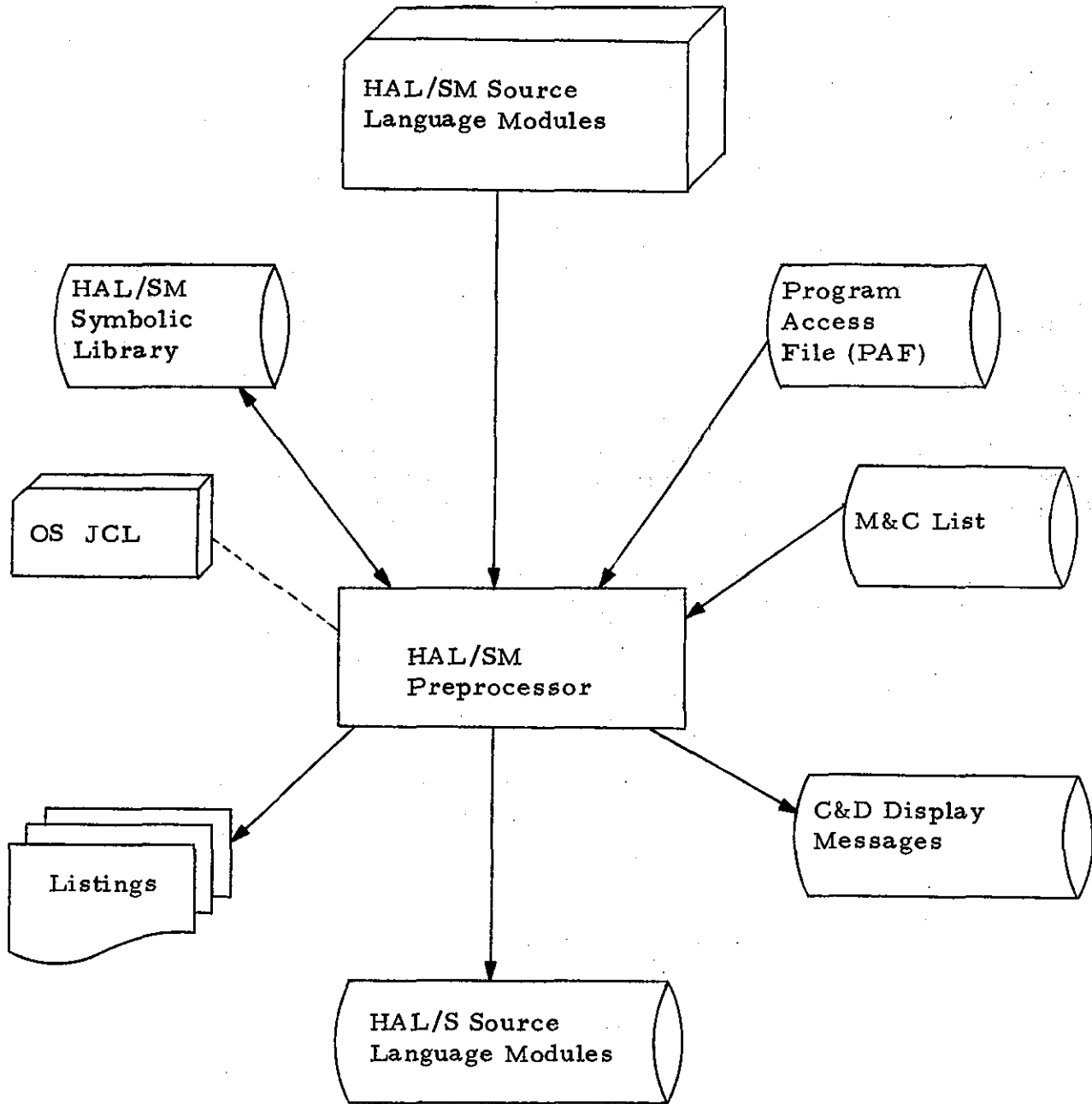


Figure 2-2

Program Access File (PAF): Implementation of the HAL/SM ACCESS attribute shall be via a PAF mechanism of the same form and intent as in the HAL/S-360 implementation (see Reference 5, paragraph 6.2.2.6). Although this capability is contained in the compiler, it must be duplicated in the preprocessor because of certain transformations made by the preprocessor to the text of the preprocessed symbolic modules.

2.1.2 Symbolic Library Interface

The Symbolic Library facility of the HAL/SM system shall be implemented via a mechanism of the same form and intent as that of the HAL/S-360 Include Library (see Reference 5, paragraph 6.2.2.5).

All block templates generated by the preprocessor shall be placed in the Symbolic Library. Template generation shall be performed in a manner similar to that performed by the compiler (see Reference 5, paragraph 6.2.2.7) differing only as dictated by the definition of the HAL/SM language.

2.1.3 M&CD Interface

The preprocessor shall verify all references to ERP's made by the HAL/SM programmer by looking up the M&CD ID in the M&CD. All references to ERP's shall be checked for correct usage and the reference will be translated to the standard "O-name" defined for that particular ERP to permit processing by the MOSS Linker.

2.1.4 Display Data Interface Utility (DDIU) Interface

The preprocessor shall remove all C&D display message text from the HAL/SM source code and create a data set which contains this information. The data set shall be constructed by the DDIU routines. Coded references to the display data set shall be placed in the output HAL/S code where each message is referenced.

2.1.5 Listings

Several types of listings may be generated by the preprocessor as specified by preprocessor directives. Some or all of these listings may be optional (TBD).

Unformatted Source Listing: This shall be an "80-80" listing of the primary input data set and all INCLUDED text (from the Symbolic Library) before annotation or reformatting by the preprocessor.

Reformatted Source Listing: This shall be a reformatted and annotated representation of the symbolic HAL/SM source text. Indentation and annotation shall be in essentially the same form as the primary listing produced by the compiler (see Reference 5, paragraphs 3.1-3.3).

Cross Reference Listing: This shall be an alphabetized listing of all programmer-defined symbols together with a list of the line numbers referencing each occurrence of each symbol in the reformatted source listing.

Output Source Listing: This shall be a listing of the HAL/SM symbolic source code produced by the preprocessor.

2.1.6 Diagnostic Information

The preprocessor shall produce diagnostic error messages for all errors which it detects. These error messages shall consist of descriptive text indicating the type of error detected and they shall be placed in the reformatted source listing to indicate the source of the error.

The preprocessor shall not perform a complete syntactic or semantic analysis of all statements. However, sufficient analysis shall be performed such that errors not detected by the preprocessor shall cause incorrect HAL/S symbolic code to be produced and the compiler will detect the error; i. e., syntactically incorrect HAL/SM source text will cause an error message to be generated by either the preprocessor or the compiler or both.

2.1.7 HAL/S-360 Compiler Interface

The implementation shall provide for automatic invocation of the compiler to process the HAL/S code produced by the preprocessor when no errors are detected. All information required by the compiler shall be provided to it by the preprocessor when the HAL/SM code is error free. This includes certain control information in the form of compiler directives (see Reference 5).

The version of the compiler system to be used in the HAL/SM system shall be fixed at Release 11.0 for the entire duration of the software development phase.

2.1.8 User Interface

OS/360-370 JCL procedures shall be implemented to permit convenient use of the HAL/SM preprocessor. Suitable options and defaults shall be defined and implemented into the procedures to allow for the normal expected range of applications of the system.

2.2 HALLINK Subsystem

The HAL/S-360 HALLINK program (see Figure 2-3) is defined to produce program complexes which operate in a different environment from that of MOSS and it, therefore, performs certain processing which is unnecessary for HAL/SM programs. HALLINK also allows certain configurations of programs which are not valid in the HAL/SM System. This subsection describes the functional requirements for modifications to be made to the HAL/S-360 HALLINK program to make it usable and reliable within the HAL/SM System. The modified version of HALLINK shall be known as the HAL/SM HALLINK program.

The modifications to be made to HALLINK are primarily to enforce restrictions which must be imposed either as a result of the HAL/SM language definition and the MOSS environment or as a result of the implementation of the HAL/SM System. Building these restrictions into HALLINK will permit a certain degree of integrity to be guaranteed when including non-HAL/SM modules in a HAL/SM task.

The HAL/SM HALLINK program shall operate in the same manner as that for HAL/S-360 with the following changes:

- o Only one process shall be permitted in each address space.
- o Certain control sections shall be deleted from the output load modules which are not needed in the MOSS environment.
- o The Execution Monitoring System (see Reference 5, Section 4.3) shall not be supported.

2.3 RTL Subsystem

This subsection describes the functional requirements for modifications to be made to the HAL/S-360 RTL (see Figure 2-4) to make it usable under MOSS in the HAL/SM System.

2.3.1 Efficiency Criteria

The configuration of the particular SUMC-S for which this implementation of HAL/SM is to be designed is currently envisioned as a system on which main memory is more of a scarce resource than processor time. For this reason, the primary efficiency criteria to be

HALLINK SUBSYSTEM

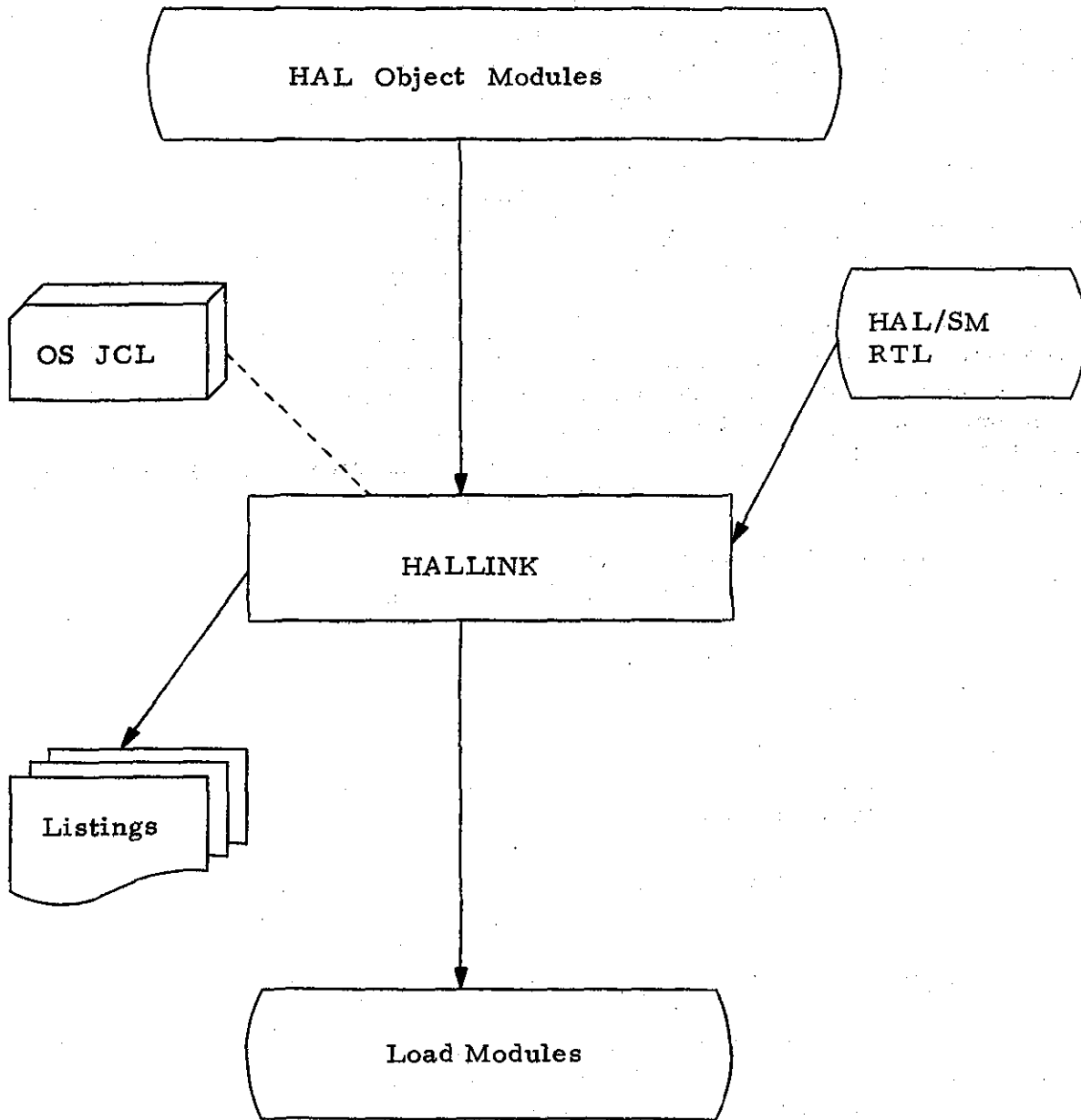


Figure 2-3

HAL/SM RUN-TIME LIBRARY (RTL)

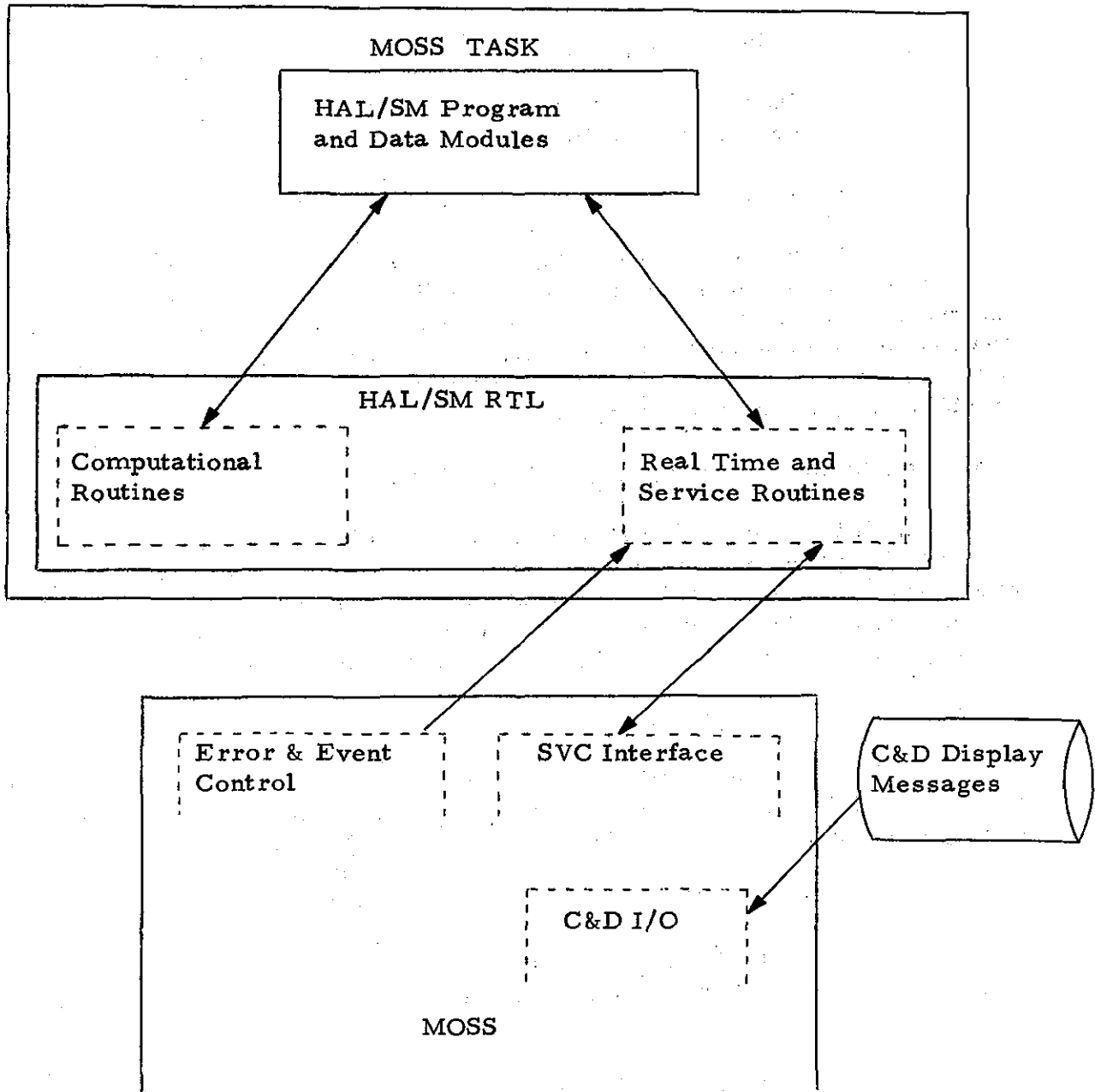


Figure 2-4

used in implementing the required modifications to the RTL shall be as follows:

- o Main Memory requirements shall be kept to a minimum.
- o Execution speed shall be considered secondary to memory utilization.

2.3.2 Modifications to Support MOSS External Interfaces

Modifications shall be made at all operating system interfaces due to different implementation of similar functions. Some RTL modules and interfaces shall be deleted where their functions are not supported; others shall be added to support new capabilities. Some functions which were previously performed (or simulated) within the RTL shall be performed by MOSS (e.g., task scheduling).

2.3.3 Modifications to Support HAL/SM Calling Sequences

Modifications shall be made to the RTL to provide interfaces to the HAL/SM program for all new calling sequences generated by the pre-processor (see Section 3.1). Some of the new interfaces shall be modified versions of existing modules, while others shall require implementation of completely new modules to support language mechanisms.

2.3.4 Unsupported Functions

Major deletions shall be made from the RTL of modules which handle functions which are unsupported by this implementation. The major deletions pertain primarily to the following:

- o "Pseudo-RTE" - This refers to the RTL routines which simulate process scheduling, event management, etc., for the HAL/S-360 implementation. These functions shall be performed by MOSS, frequently with modified semantics.
- o Execution Monitoring System - This facility (including the Diagnostic Command Language, see Reference 5, Section 4.3) shall not be supported for HAL/SM.
- o Multiple Processes - Under MOSS, exactly one process (or "task") is permitted per address space. The HAL/SM RTL shall support only one process per copy.

- o Flight Computer Timing Simulation - This facility shall not be supported for HAL/SM.
- o Miscellaneous HAL/S Features not in HAL/SM - Certain HAL/S language features are not provided in HAL/SM because either the capability cannot be provided under MOSS (e.g., the HAL/S UPDATE PRIORITY statement), or because the capability is considered either dangerous or difficult to support in this implementation.

2.3.5 Unsupported Execution-Time JCL Options

Because of the deletions mentioned in Paragraph 2.3.4, the following execution-time options (see Reference 5, Appendix B) are not supported:

- o MSGLEVEL = 2
- o SIMTIME = n
- o SPEED = n
- o PCBS = n
- o FIRSTPGM = name
- o PROGRAM = name
- o TRACE = n
- o NOTIME
- o FAST
- o DUMPALL = n

(BLANK)

3. INTERFACES

This section defines the preliminary processing requirements of each of the major subsystems in terms of their interfaces. As in Section 2, the HAL/S-360 compiler is not discussed directly; see References 2 through 5 for more information. Since the relationships of the HAL/SM programs to each of the subsystems are quite different, it is useful to interpret the term "interface" differently for each subsystem discussed. Specifically, for each subsystem the interfaces are discussed as follows:

- o Preprocessor - For each HAL/SM statement type or construct requiring a modified implementation in HAL/S the emitted HAL/S source code is described.
- o HALLINK - The allowable inputs and the types of information in the output load modules are discussed.
- o RTL - For each new entry point into the RTL defined in support of the preprocessor, the calling sequence and function performed by the entry point is described. For each RTL module whose implementation is affected by the difference between the OS and MOSS environment, the required modifications are described on a functional level.

In addition to the inter-subsystem interfaces discussed above, the user interface is discussed in terms of OS and MOSS JCL requirements.

3.1 Preprocessor

This subsection enumerates the HAL/SM constructs which require a modified implementation in HAL/S. Each numbered paragraph discusses a single HAL/SM construct in the following form:

3.1. _ Construct Name

Syntax Diagram

Emitted HAL/S Code Template(s)

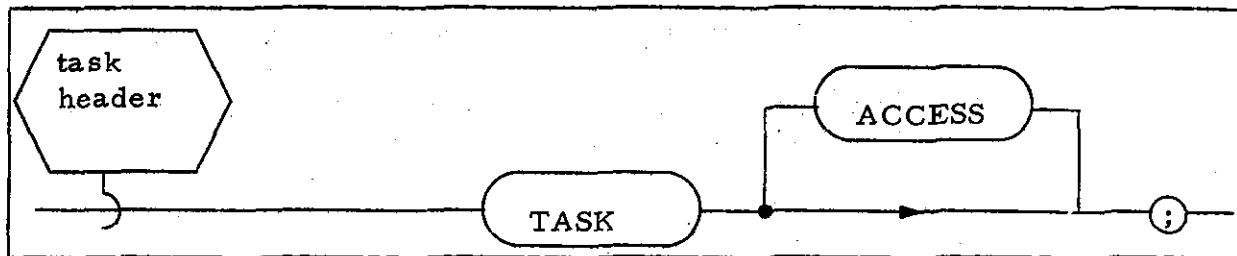
Notes

Examples

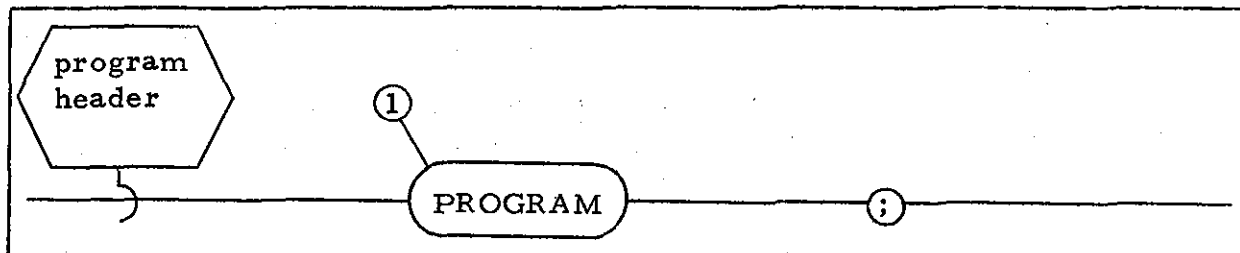
The syntax diagrams are of the same form as those in the language specification, though they are not necessarily identical. Non-terminal symbols which are not defined in the language specification are defined elsewhere in this subsection. The notes are used to correlate various elements of the HAL/SM construct with the emitted HAL/S code and to explain differences in alternate versions of the emitted HAL/S code template required by options or alternatives in the HAL/SM syntax.

3.1.1 TASK Header

HAL/SM Syntax



HAL/S Code Template



Notes

1. The TASK in the HAL/SM construct is equivalent to PROGRAM in HAL/S.
2. Access control for HAL/SM constructs is performed by the preprocessor.

Example

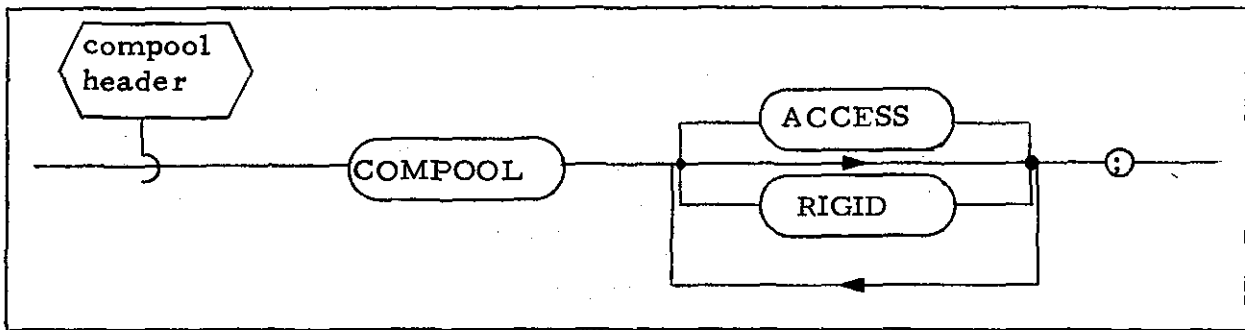
T1: TASK ACCESS;

becomes

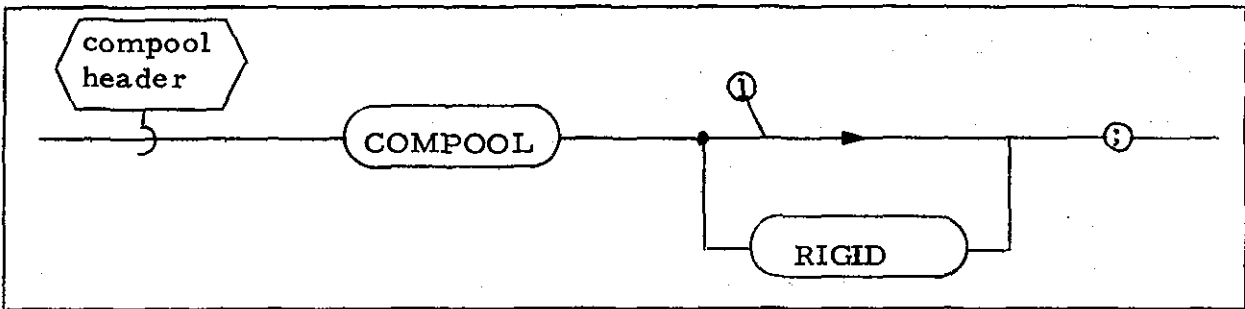
T1: PROGRAM;

3.1.2 COMPOOL Header

HAL/SM Syntax



HAL/S Code Template



Note

1. Access control for HAL/SM constructs is performed by the preprocessor.

Example

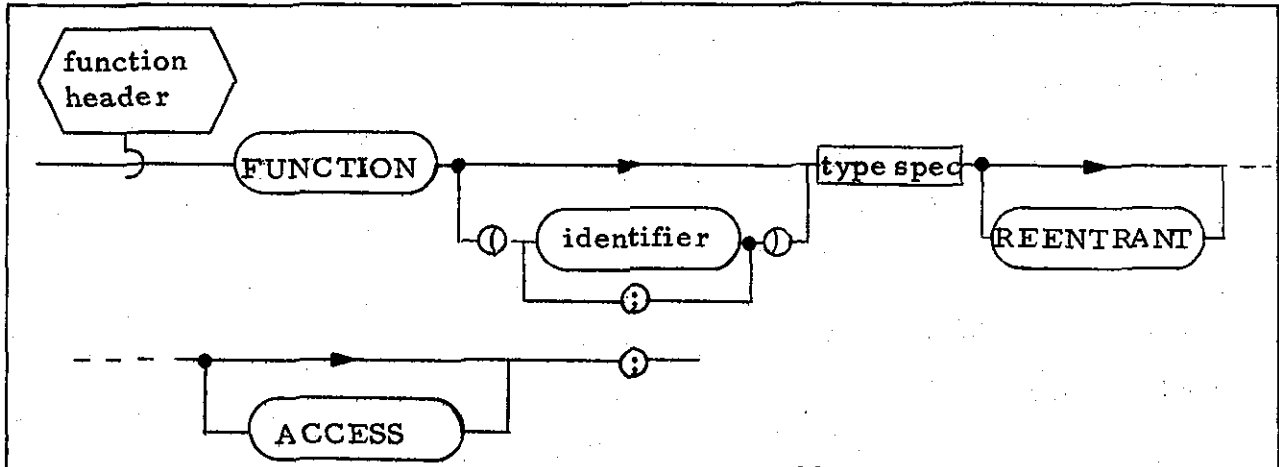
C1: COMPOOL ACCESS;

becomes

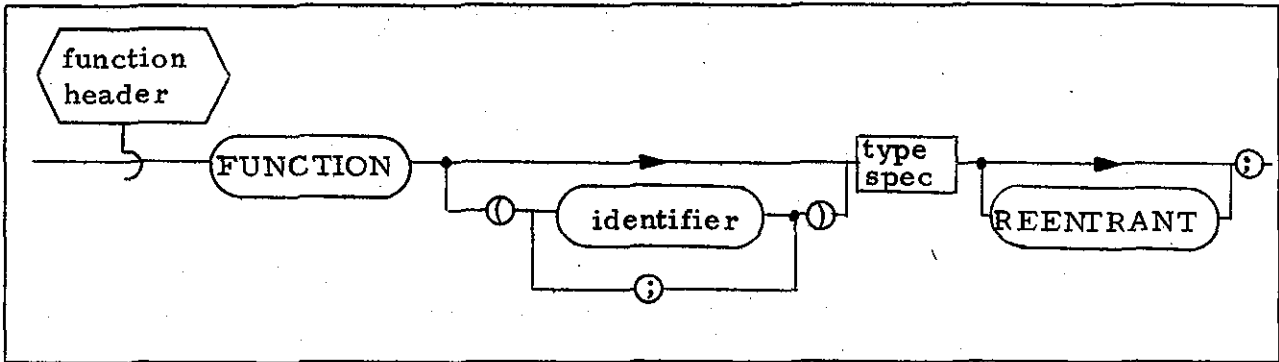
C1: COMPOOL;

3.1.3 FUNCTION Header

HAL/SM Syntax



HAL/S Code Template



Note

1. Access control for HAL/SM constructs is performed by the preprocessor.

Example

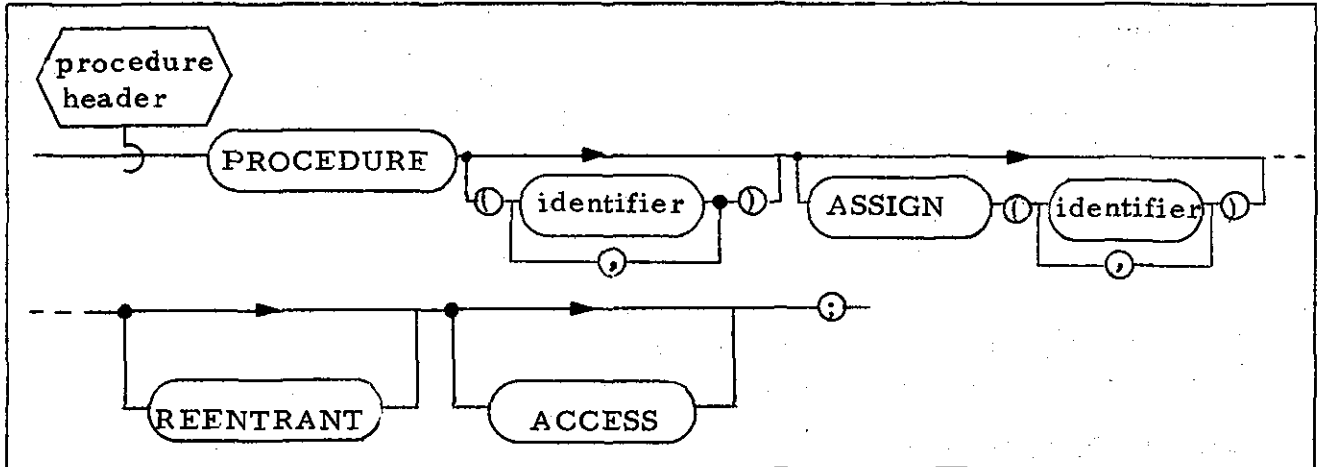
F1: FUNCTION (TESTX) SCALAR REENTRANT ACCESS;

becomes

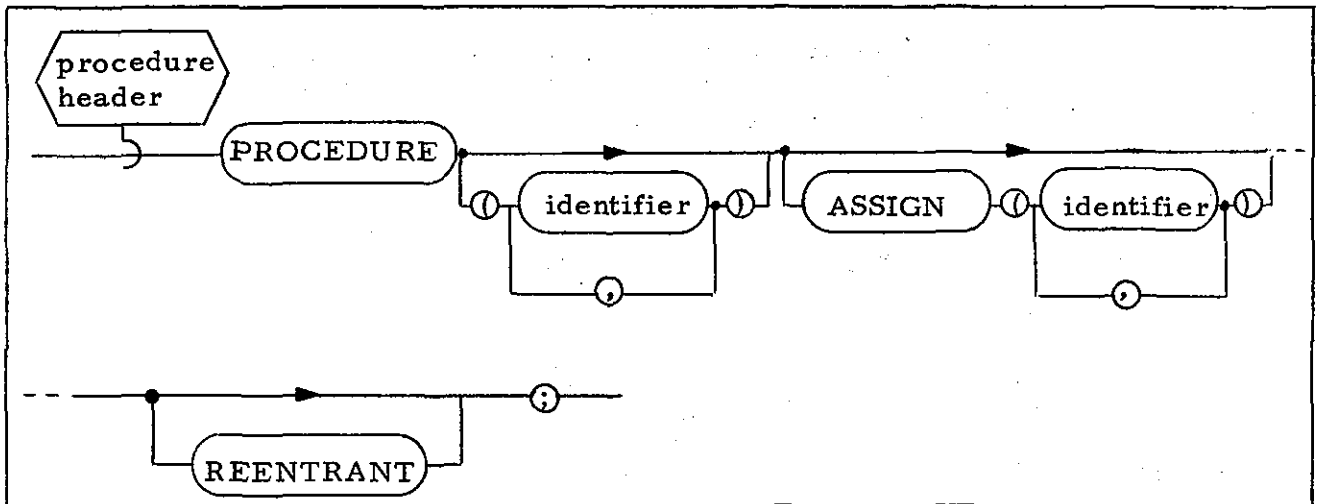
F1: FUNCTION (TESTX) SCALAR REENTRANT;

3.1.4 PROCEDURE Header

HAL/SM Syntax



HAL/S Code Template



Note

1. Access control for HAL/SM constructs is performed by the preprocessor.

Example

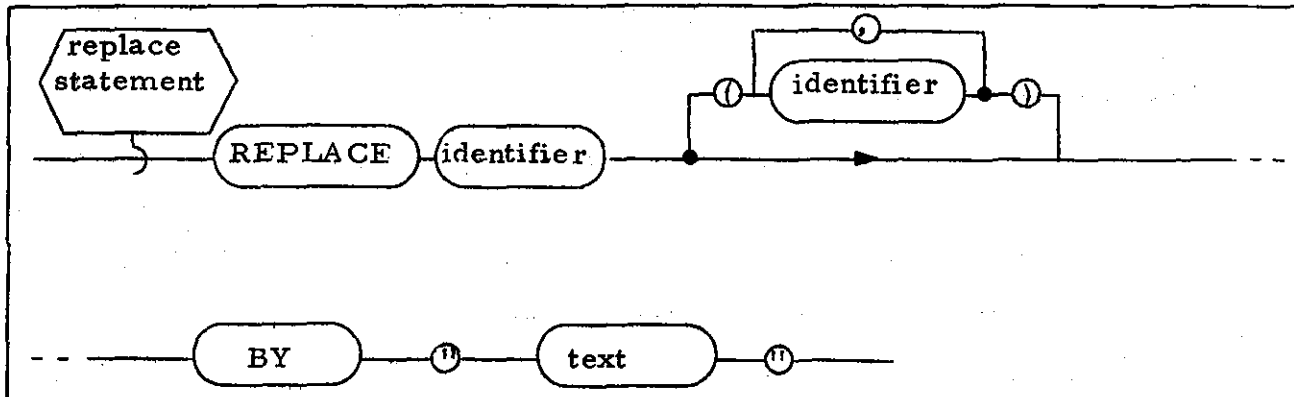
P1: PROCEDURE (PROCl) ASSIGN (ARGU1) REENTRANT ACCESS;

becomes

P1: PROCEDURE (PROCl) ASSIGN (ARGU1) REENTRANT;

3.1.5 REPLACE Statement

HAL/SM Syntax



HAL/S Code Template

Not Applicable.

Note

1. The REPLACE statement is processed by the preprocessor (i.e., the preprocessor shall replace each < identifier > with the appropriate < text >).

Example

```
REPLACE ALPHAZ BY "J + 2 SIN(ALPHAB)";
```

•
•
•

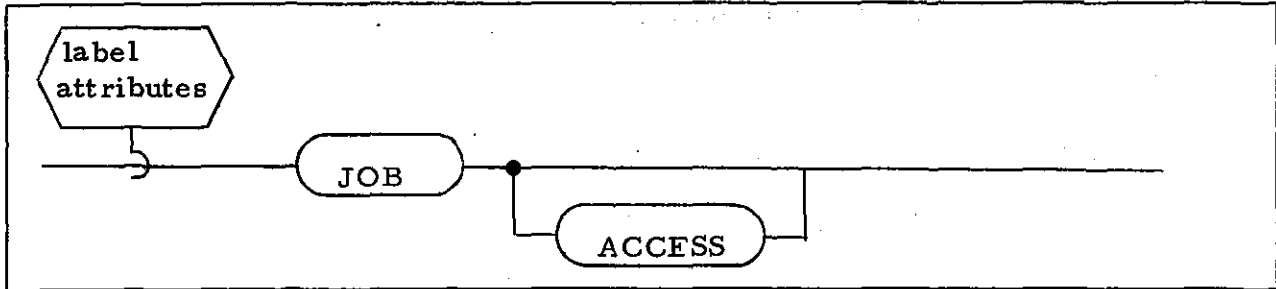
```
BETAG = ALPHAZ - C05(Y);
```

becomes

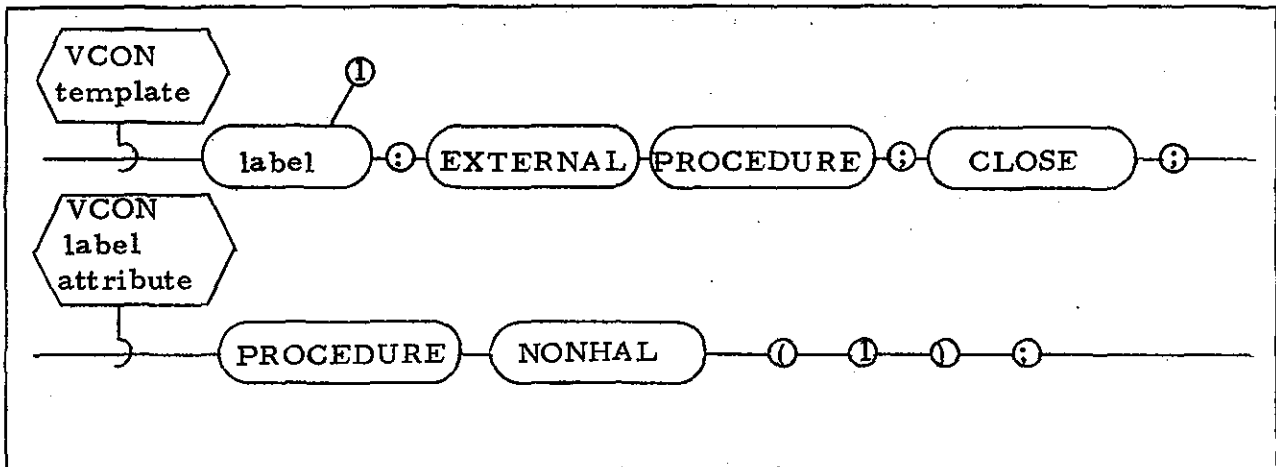
```
BETAG = J + 2 SIN(ALPHAB) - C05(Y);
```


3.1.6 JOB Attribute

HAL/SM Syntax



HAL/S Code Template



Notes

1. All <identifier> s declared with the JOB <label attribute> shall cause the preprocessor to produce a <VCON template> (at the beginning of the compilation) with the declared identifier as its <label> and the JOB <label attribute> shall be translated into a <VCON label attribute>.
2. All other <label attribute> s shall be handled the same in HAL/S as in HAL/SM (i.e., no transformation).

Example

DECLARE

LOGANAL JOB ACCESS;

becomes

LOGANAL : EXTERNAL PROCEDURE;

CLOSE;

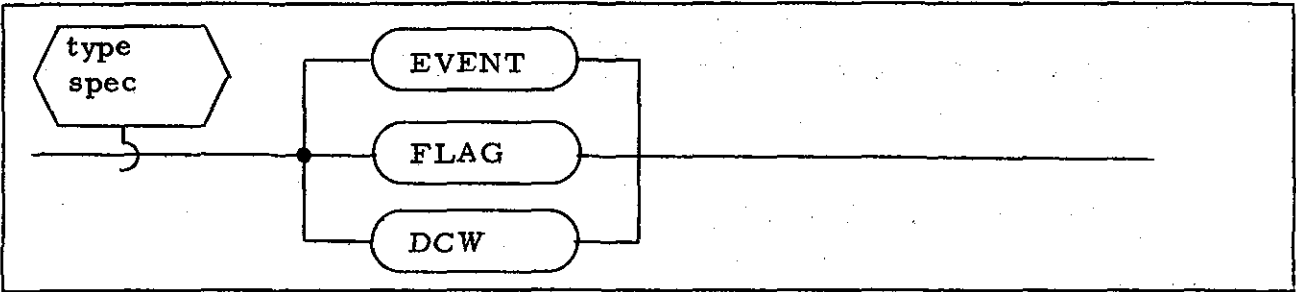
•
•
•

DECLARE

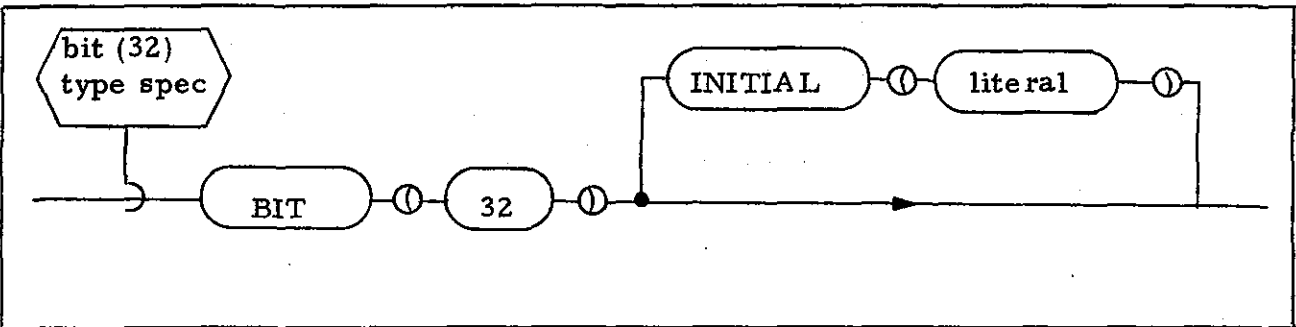
LOGANAL PROCEDURE NONHAL (1);

3.1.7 Type Specification

HAL/SM Syntax



HAL/S Code Template



Notes

1. DCW and EVENT keywords appearing in a <type spec> shall be translated into a <bit (32) type spec>.
2. Identifiers declared with a FLAG <type spec> shall cause the preprocessor to produce a <VCON template> (see Section 3.1.6) with the declared identifiers as its <label> and the FLAG <type spec> itself shall be translated into a <VCON label attribute> (see Section 3.1.6).
3. All other <type spec> s shall be handled the same in HAL/S as in HAL/SM (i. e., no transformations).
4. When no initialization is present for DCW type data (see Section 3.1.8), a default value shall be inserted by the preprocessor.

Examples

```
DECLARE VARCW1 DCW;
```

becomes

```
DECLARE VARCW1 BIT(32) INITIAL (HEX'40086000');
```

and

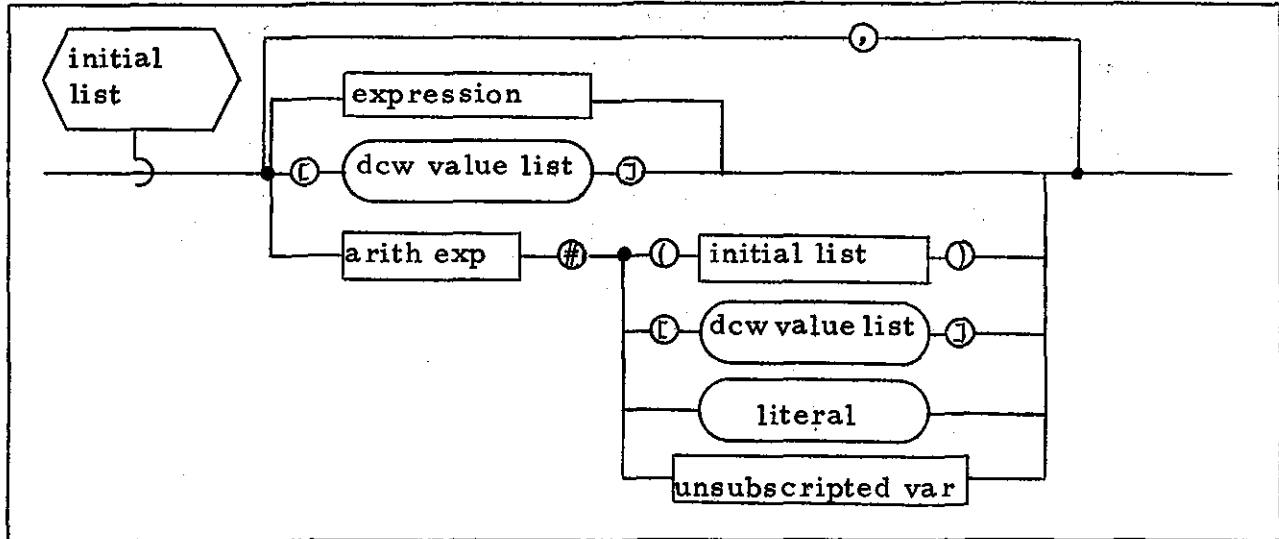
```
DECLARE EV_1 EVENT;
```

becomes

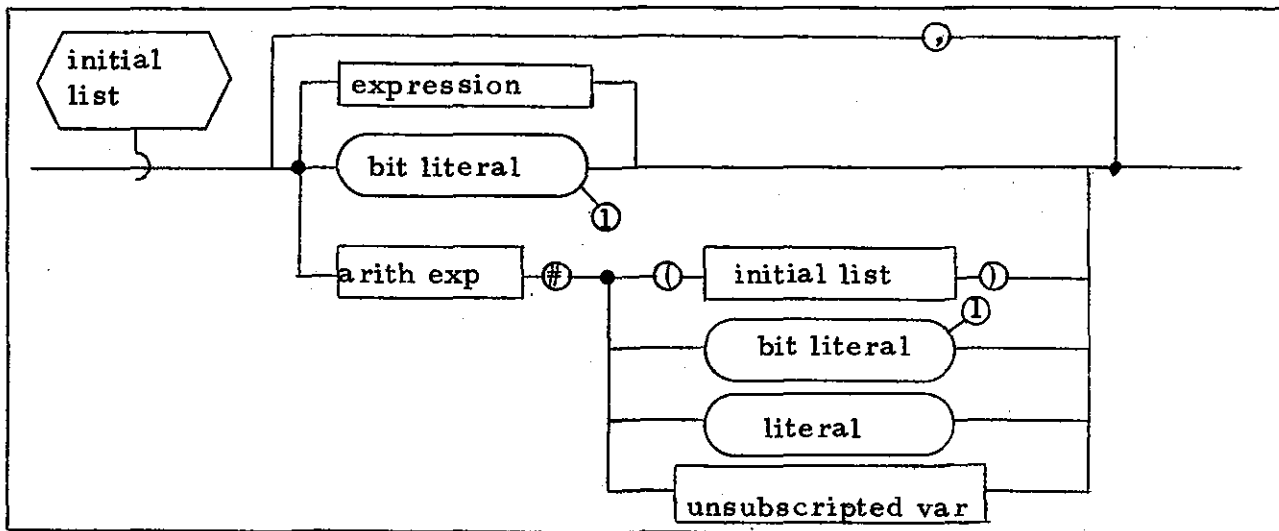
```
DECLARE EV_1 BIT(32) INITIAL(0);
```

3.1.8 Initialization of DCW-Type Data

HAL/SM Syntax



HAL/S Code Template



Note

1. This <bit literal> shall be an encoded value specifying the display control word options specified in the <dcw value list> in the HAL/SM construct. Any omitted options in the <dcw value list> shall be assigned default values.

Example

```
DECLARE
```

```
    DCW_1 DCW INITIAL([ YELLOW,10MM,BLINK OFF,6 ]);
```

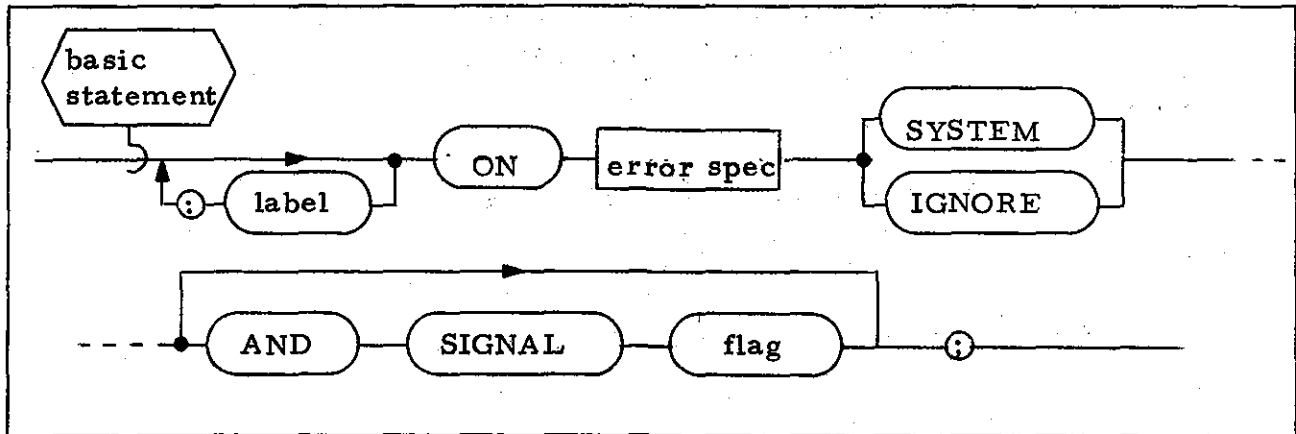
becomes

```
DECLARE
```

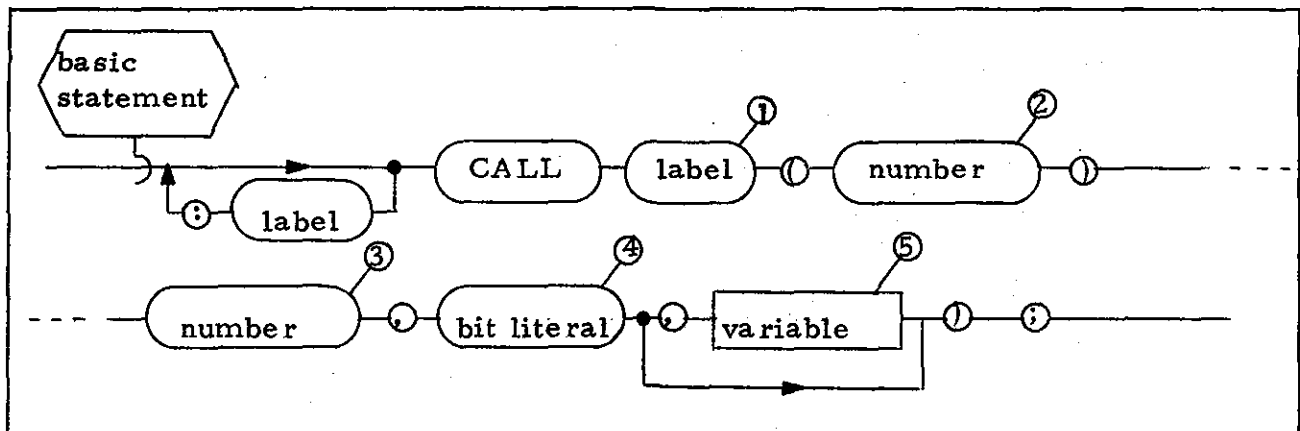
```
    DCW_1 BIT(32) INITIAL(HEX'202C4000');
```

3.1.9 ON ERROR Statement (Form 1)

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry into the RTL to communicate the required information to the HAL/SM error monitor.
2. This <number> shall indicate the error group specified in the <error spec> of the HAL/SM syntax.
3. This <number> shall indicate the error number within the selected error group as specified in the <error spec> of the HAL/SM syntax.

4. This <bit literal> shall indicate whether the SYSTEM or IGNORE option was selected in the HAL/SM syntax, and whether a flag is to be signaled.
5. This optional <variable> shall specify the HAL/S variable corresponding to the HAL/SM <flag> , when present.

Example

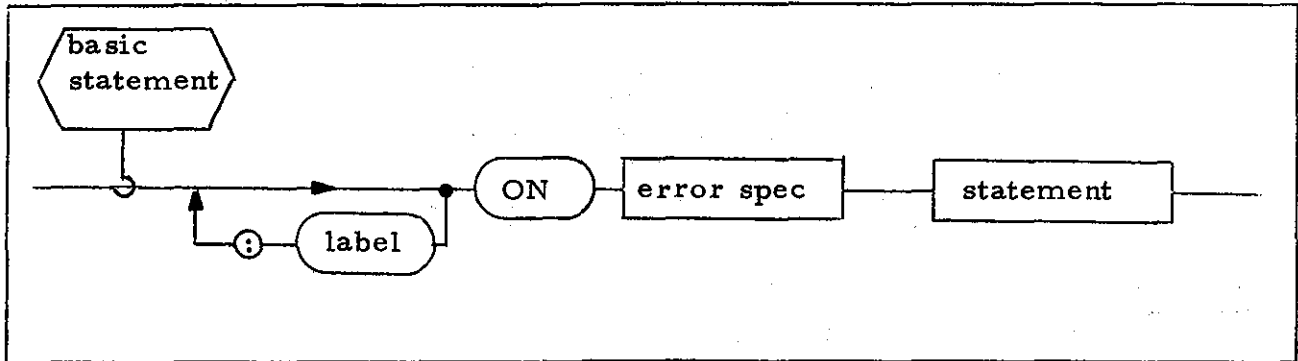
```
LAB1:ON ERROR3:1 IGNORE AND SIGNAL FLAG_A;
```

becomes

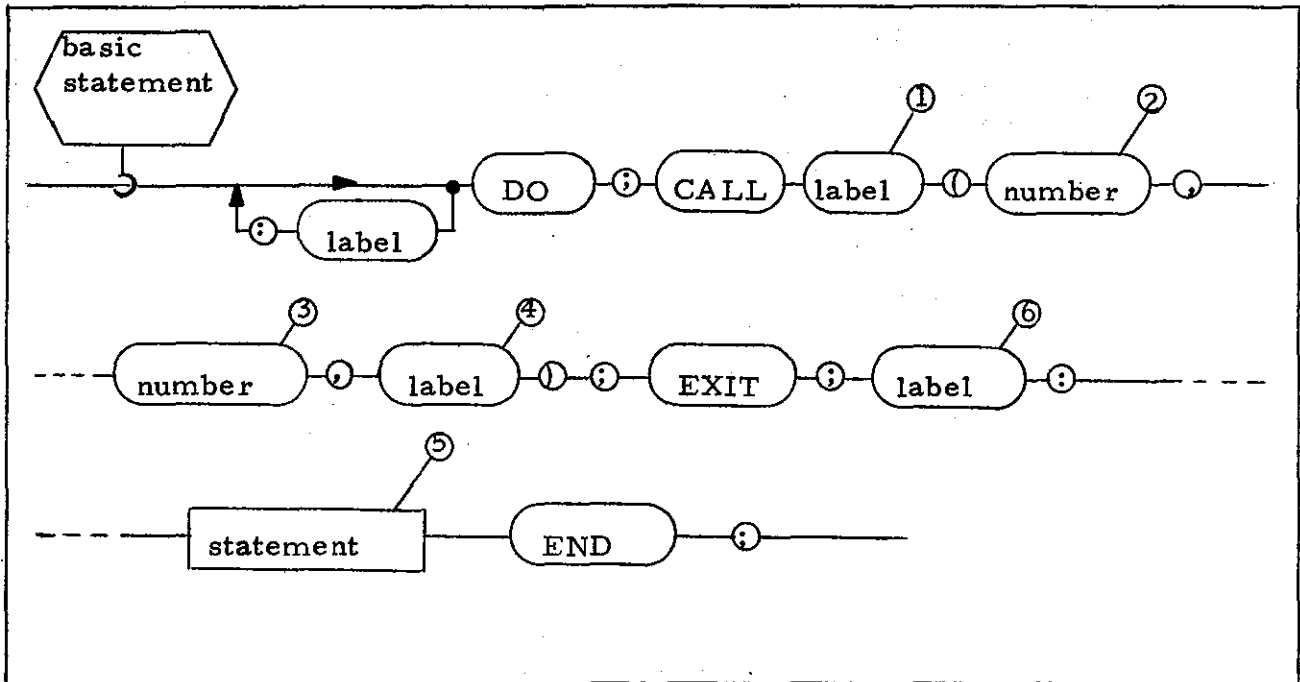
```
LAB1:CALL RTL_ON_ERROR1(3, 1, BIN'11', FLAG_A);
```


3.1.10 ON ERROR Statement (Form 2)

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry into the RTL to communicate the required information to the HAL/SM error monitor.

2. This <number> shall indicate the error group specified in the <error spec> of the HAL/SM syntax.
3. This <number> shall indicate the error number within the selected error group as specified in the <error spec> of the HAL/SM syntax.
4. This <label> shall be generated by the preprocessor and shall be the same identifier as indicated by 6.
5. This <statement> shall correspond to the <statement> in the HAL/SM syntax.

Example

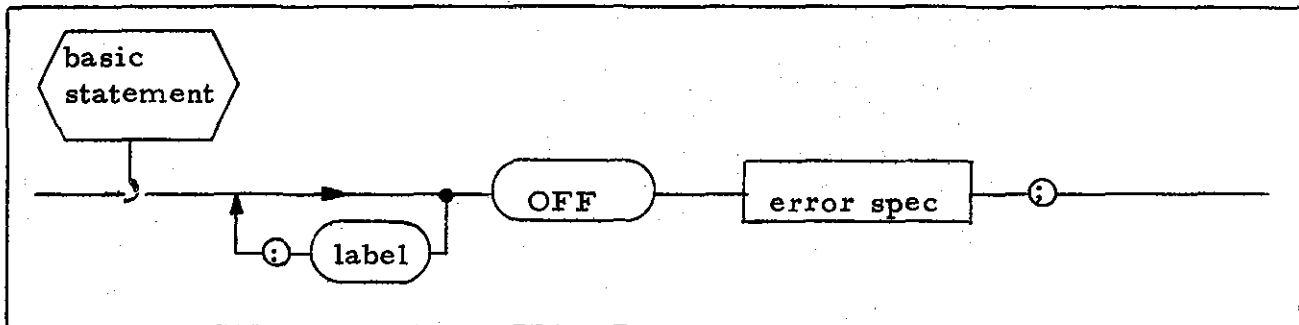
```
ON ERROR3;1 GO TO ERREXIT;
```

becomes

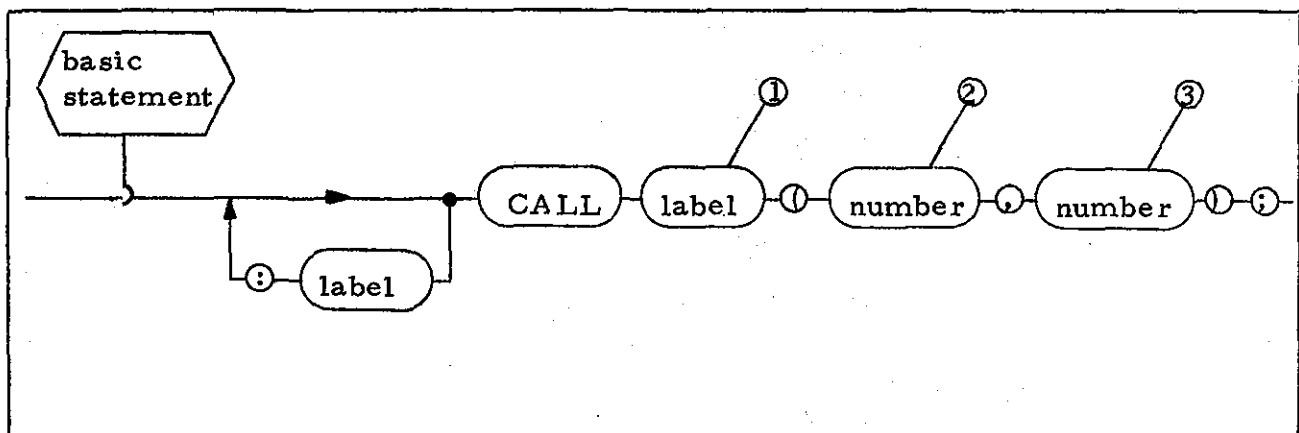
```
DO;  
  
    CALL RTL_ON_ERROR2 (3, 1, G000005);  
    EXIT;  
G000005; GO TO ERREXIT;  
END;
```

3.1.11 OFF ERROR Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry into the RTL to communicate the required information to the HAL/SM error monitor.
2. This <number> shall indicate the error group specified in the <error spec> of the HAL/SM syntax.
3. This <number> shall indicate the error number within the selected error group as specified in the <error spec> of the HAL/SM syntax.

Example

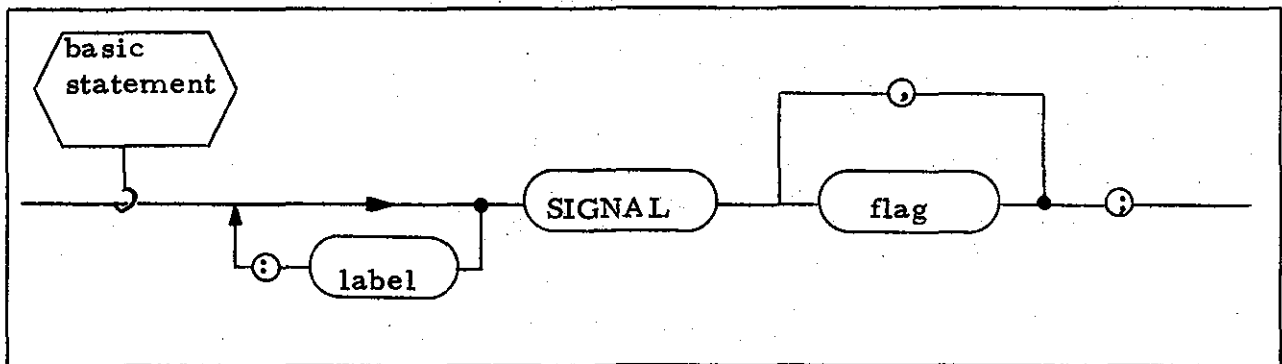
OFF ERROR_{1,2};

becomes

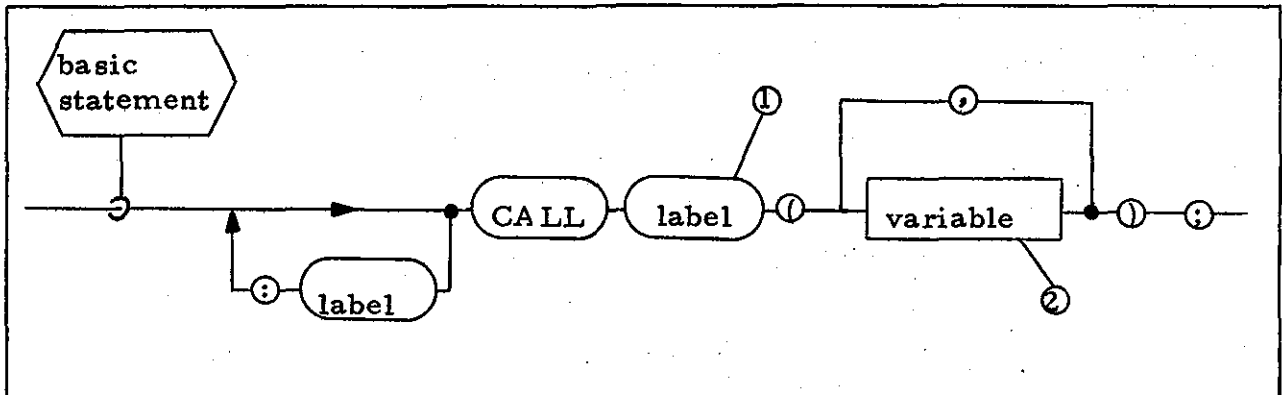
CALL RTL_OFF_ERROR(1,2);

3.1.12 SIGNAL Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry into the RTL to perform the required functions.
2. Each <variable> shall specify the HAL/S variable corresponding to each <flag> appearing in the HAL/SM construct.

Example

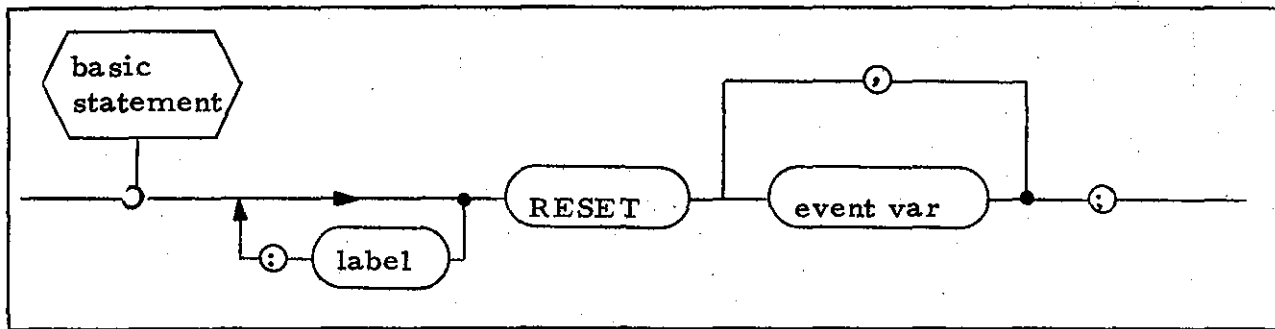
```
SIGNAL FLAG_A, FLAG_B;
```

becomes

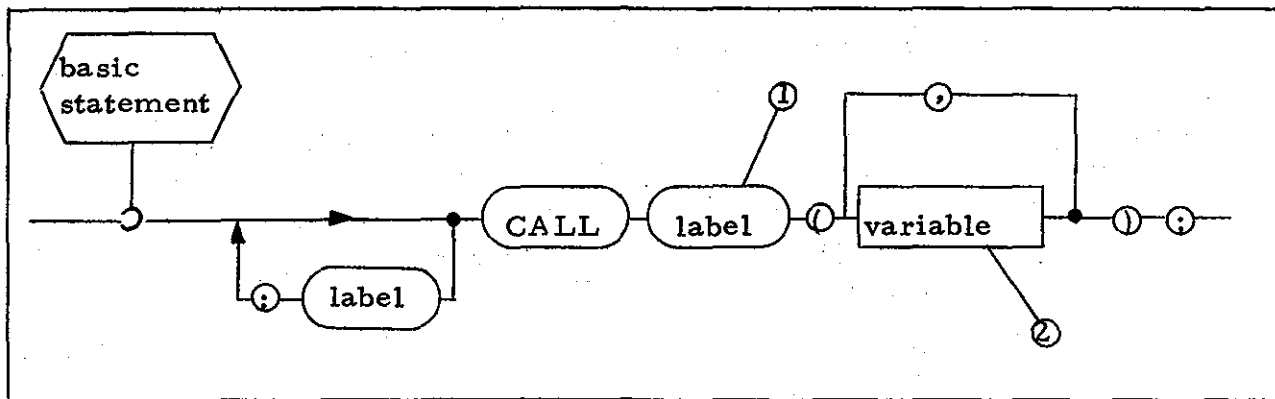
```
CALL RTL_SIGNAL (FLAG_A, FLAG_B);
```

3.1.13 RESET Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry into the RTL to perform the required functions.
2. Each <variable> shall specify the HAL/S variable corresponding to each <event var> appearing in the HAL/SM construct.

Example

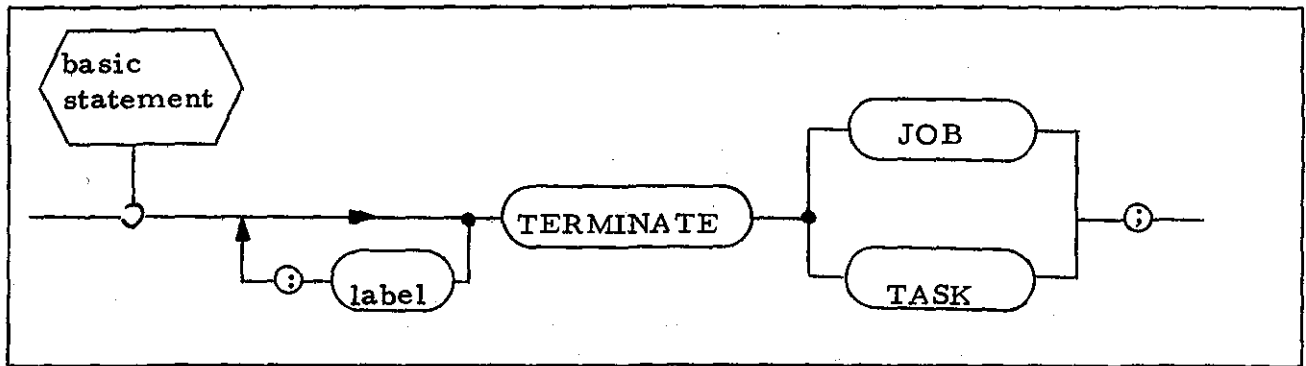
```
RESET EV_A;
```

becomes

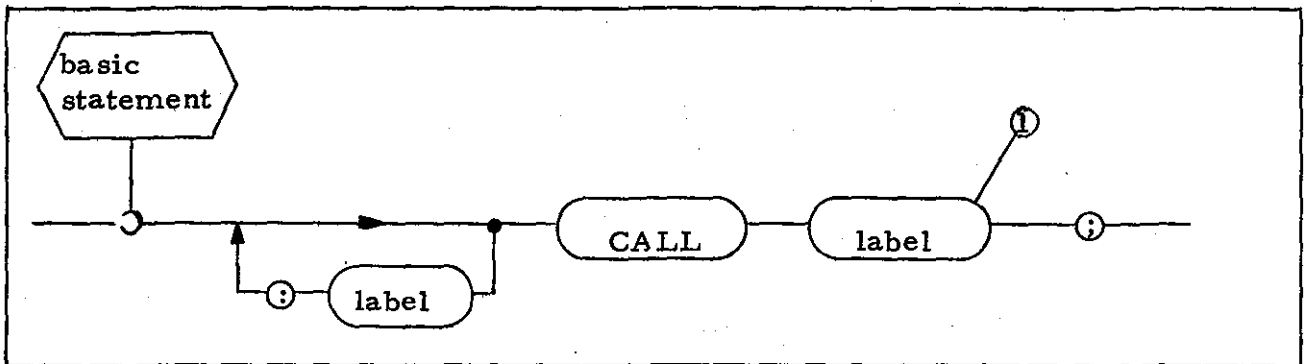
```
CALL RTL_RESET (EV_A);
```

3.1.14 TERMINATE Statement

HAL/SM Syntax



HAL/S Code Template



Note

1. This <label> shall be one of two entry points into the RTL to perform the required functions.

Example

TERMINATE JOB;

becomes

CALL RTL_JTERM;

and

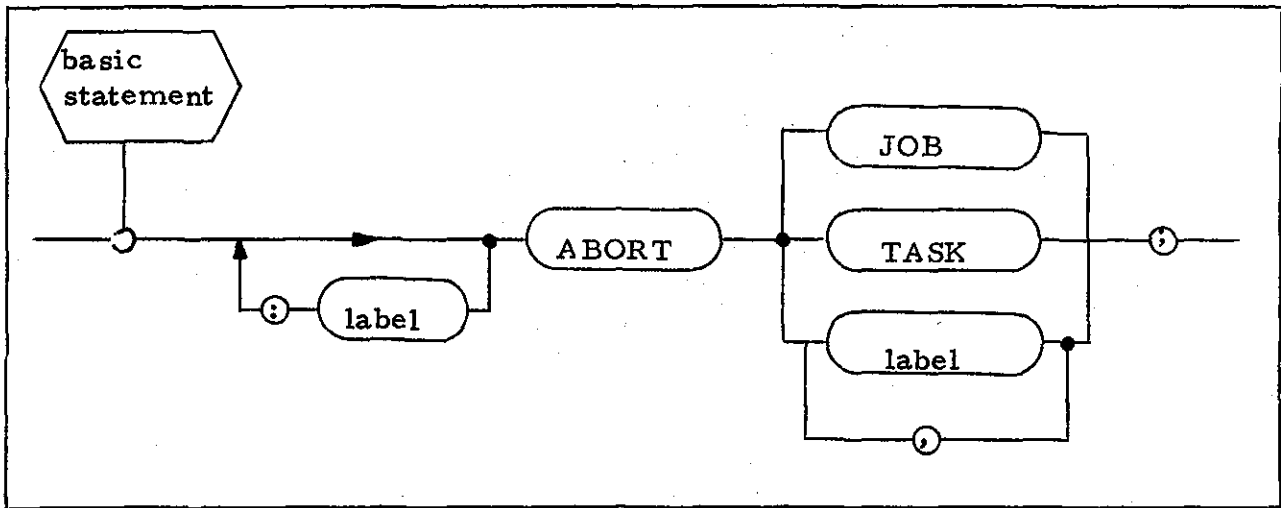
TERMINATE TASK;

becomes

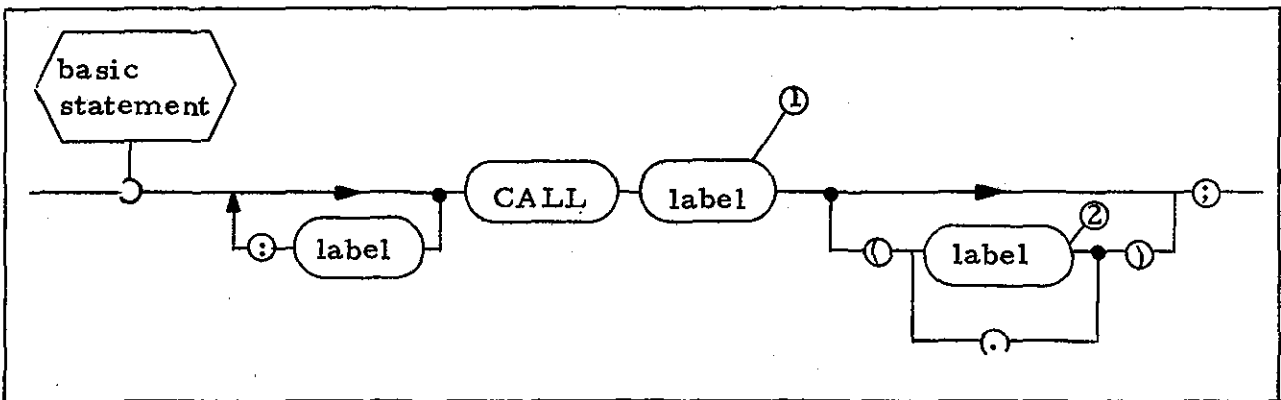
CALL RTL_TTERM;

3.1.15 A BORT Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be one of three entry points into the RTL to perform the required functions.
2. Each <label> shall specify the HAL/S label corresponding to each <label> in the HAL/SM construct.

Examples

ABORT JOB;

becomes

CALL RTL_JABORT;

and

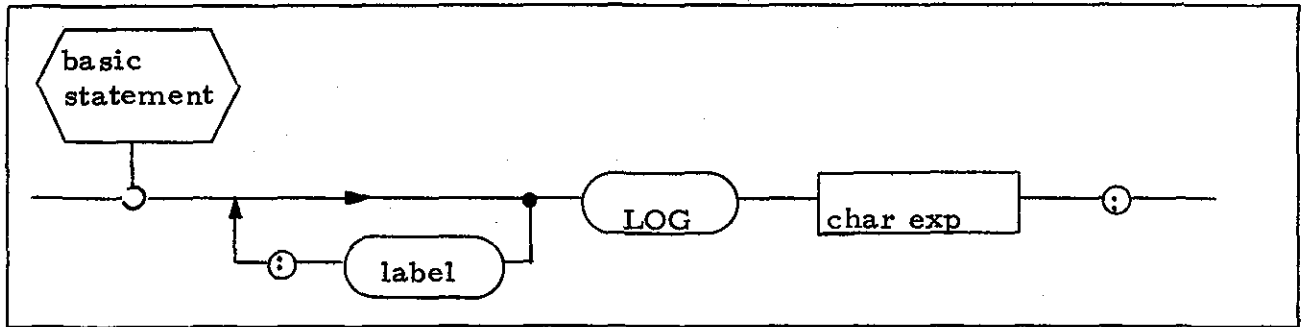
L:ABORT TASK_1, TASK_2;

becomes

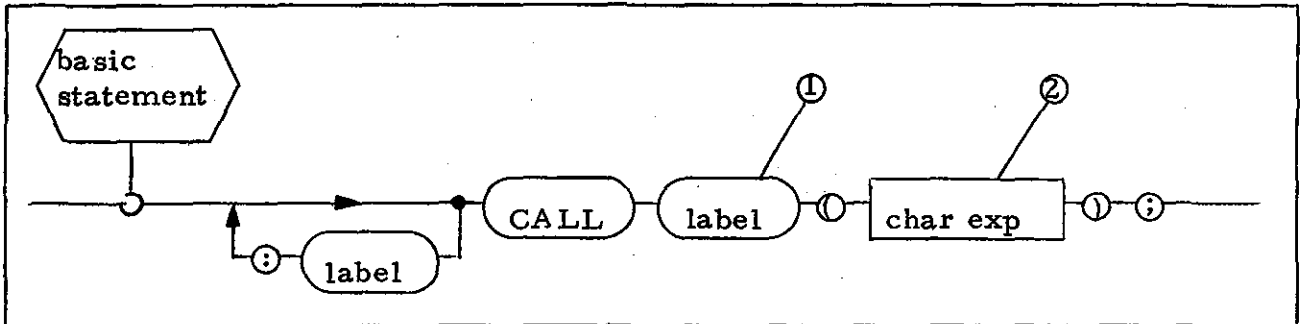
L:CALL RTL_ABORTL (TASK_1, TASK_2);

3.1.16 LOG Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required functions.
2. The <char exp> shall be the character expression specified in the HAL/SM construct.

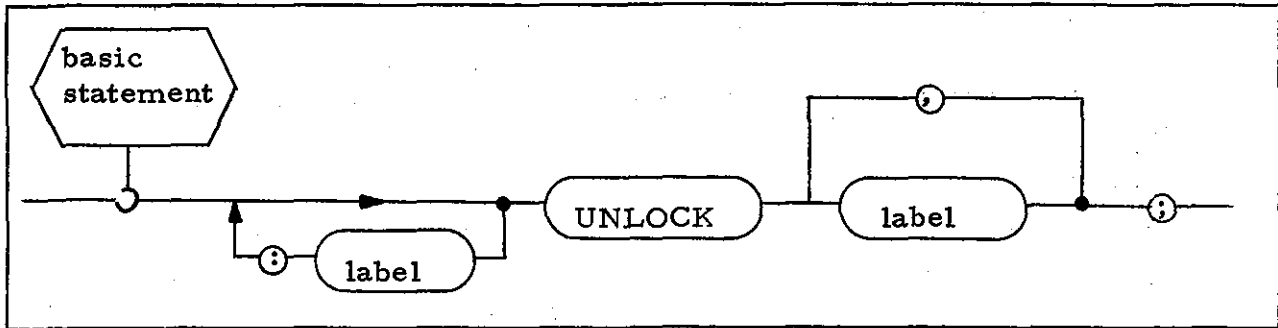
Example

```
LOG 'SIMULATION ERROR' //ERROR_NUMBER;
```

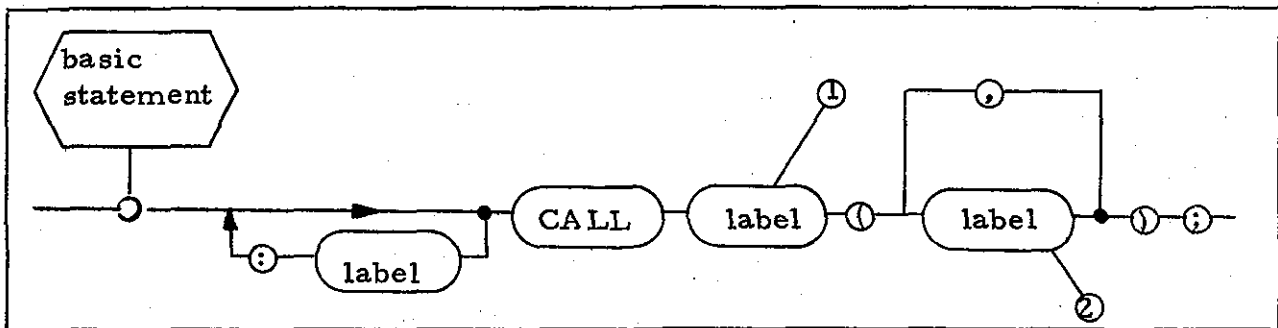
becomes

```
CALL RTL_LOG ('SIMULATION ERROR' //ERROR_NUMBER);
```

3.1.17 UNLOCK Statement



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required functions.
2. Each <label> shall specify the HAL/S label corresponding to each <label> in the HAL/SM construct.

Example

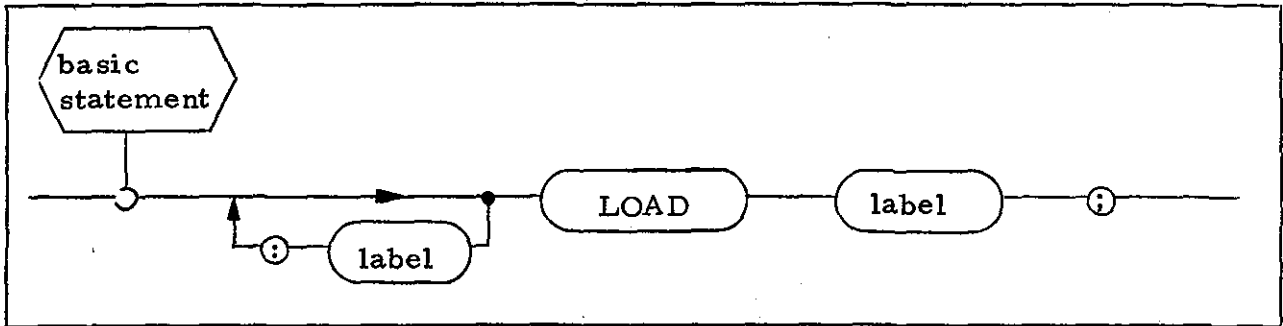
```
UNLOCK TAKE_OFF;
```

becomes

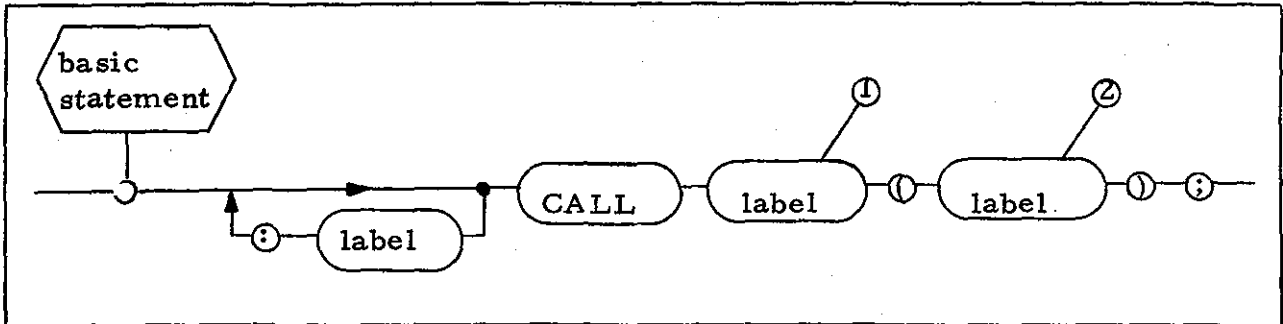
```
CALL RTL_UNLOCK (TAKE_OFF);
```

3.1.18 LOAD Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required functions.
2. This <label> shall specify the HAL/S label corresponding to the <label> in the HAL/SM construct.

Example

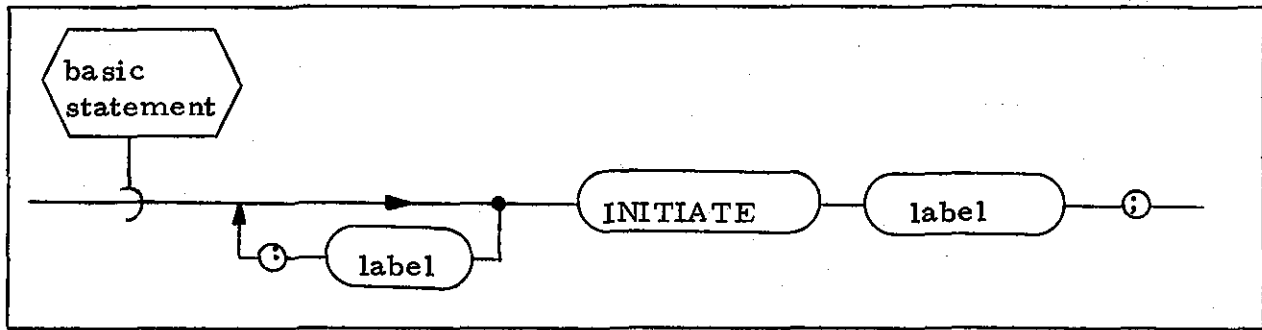
```
LOAD LOG_ANALYZER;
```

becomes

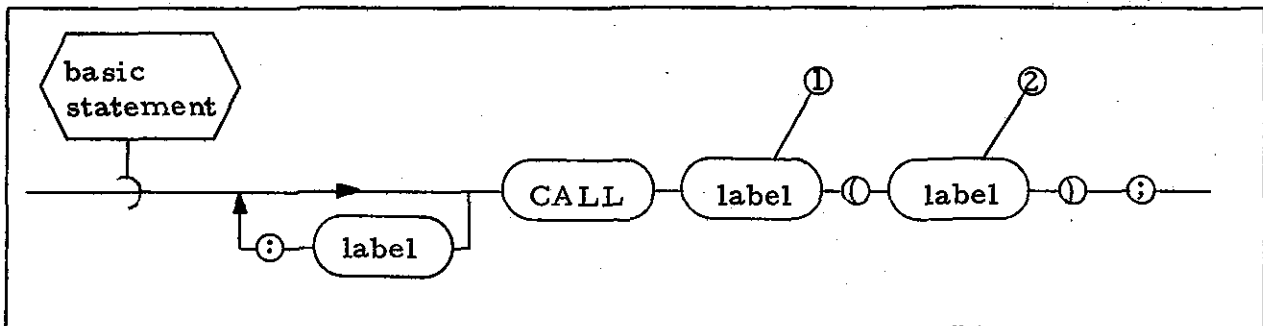
```
CALL RTL_LOAD (LOG_ANALYZER);
```

3.1.19 INITIATE Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required functions.
2. This <label> shall specify the HAL/S label corresponding to the <label> in the HAL/SM construct.

Example

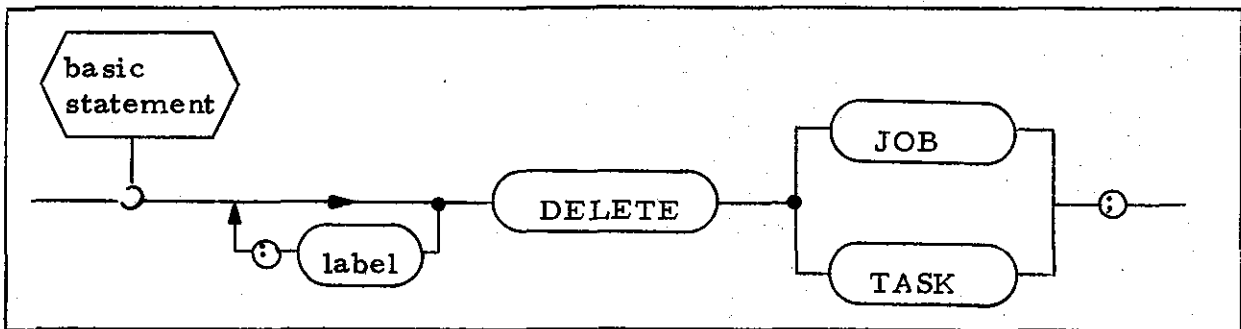
```
INITIATE LOG_ANALYZER;
```

becomes

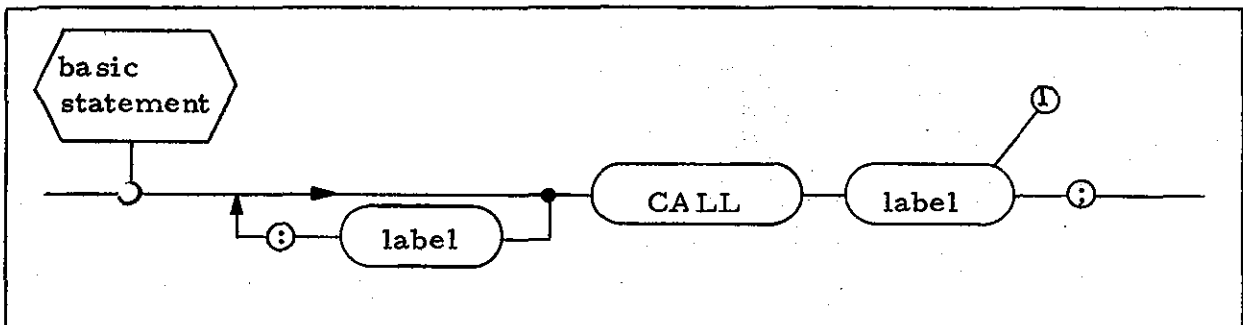
```
CALL RTL_INIT (LOG_ANALYZER);
```

3.1.20 DELETE Statement

HAL/SM Syntax



HAL/S Code Template



Note

1. This <label> shall be one of two entry points into the RTL to perform the required functions.

Examples

DELETE JOB;

becomes

CALL RTL_JDEL;

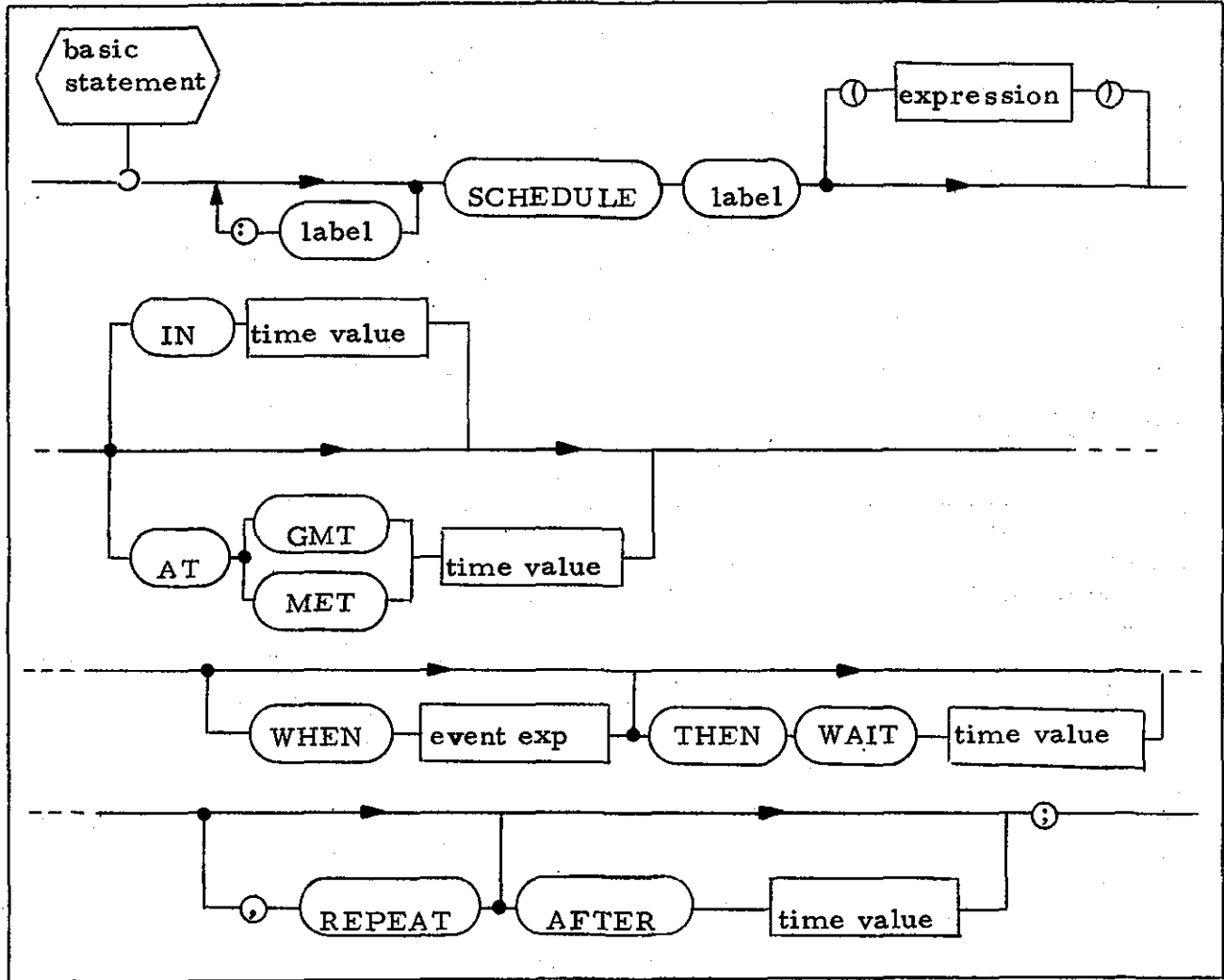
and

DELETE TASK;

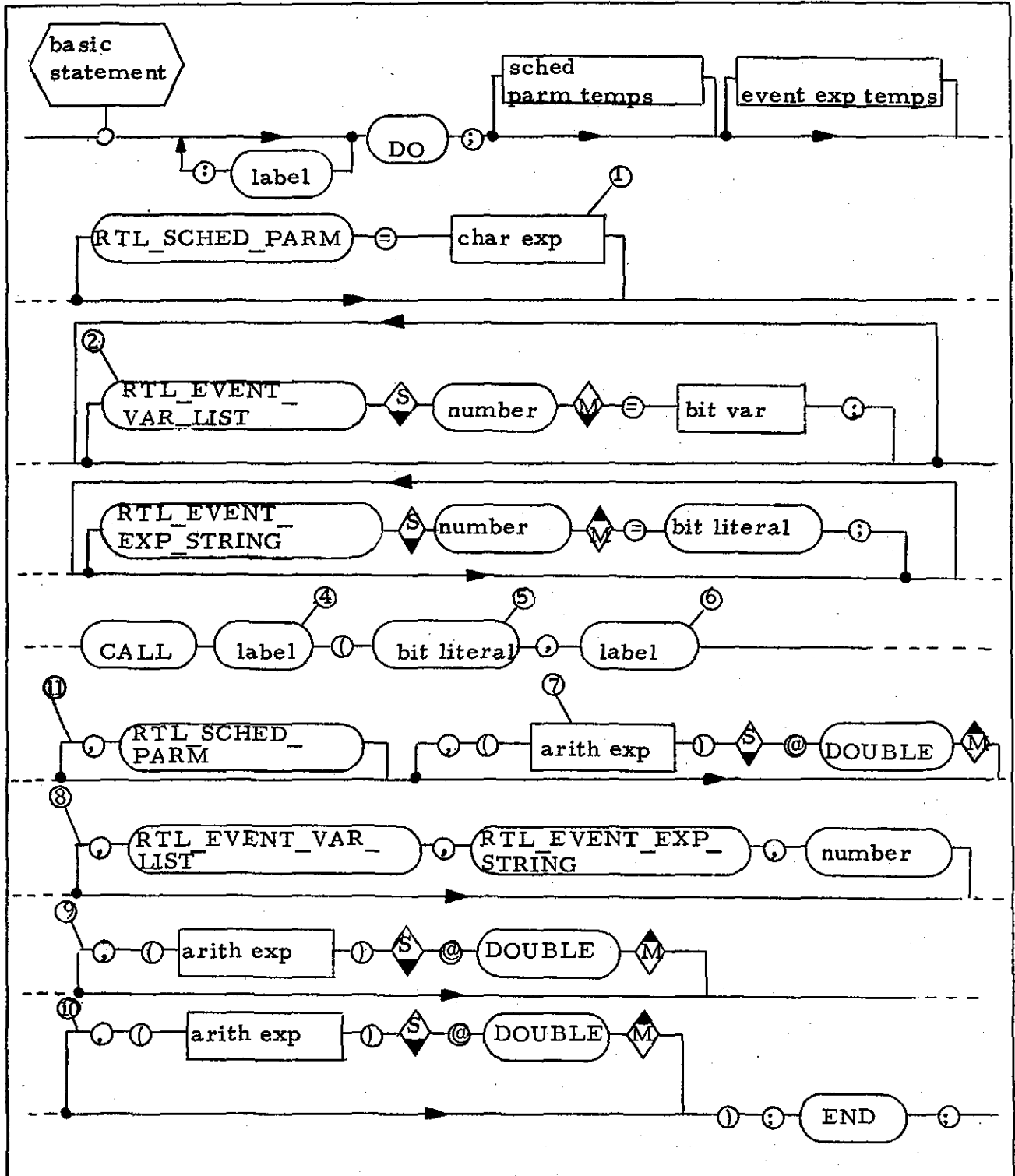
becomes

CALL RTL_TDEL;

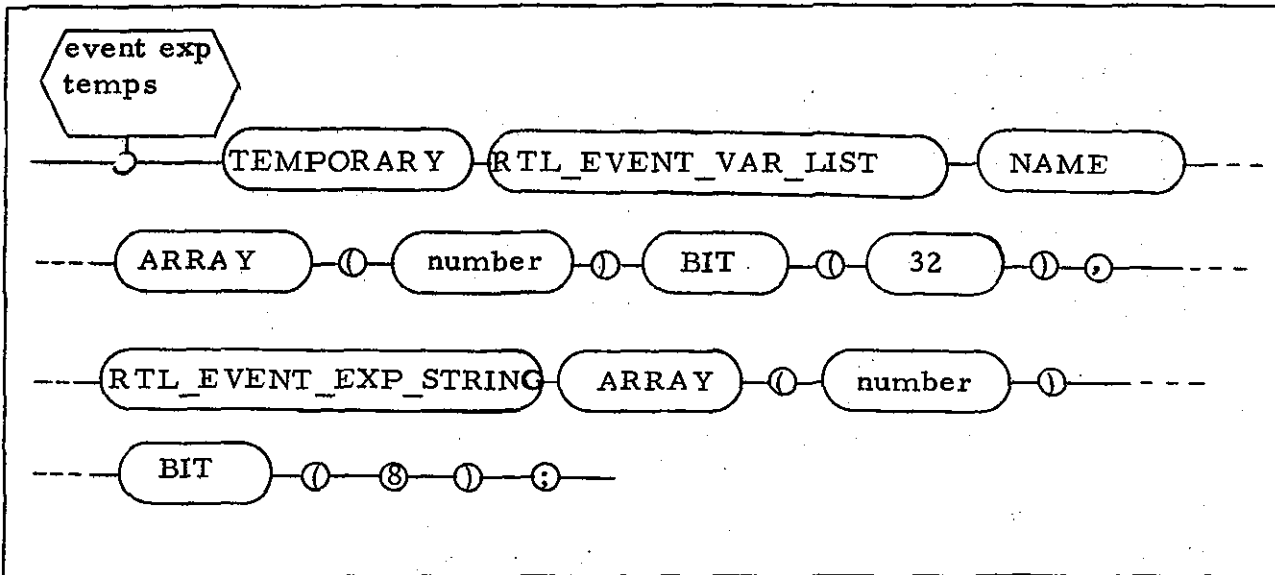
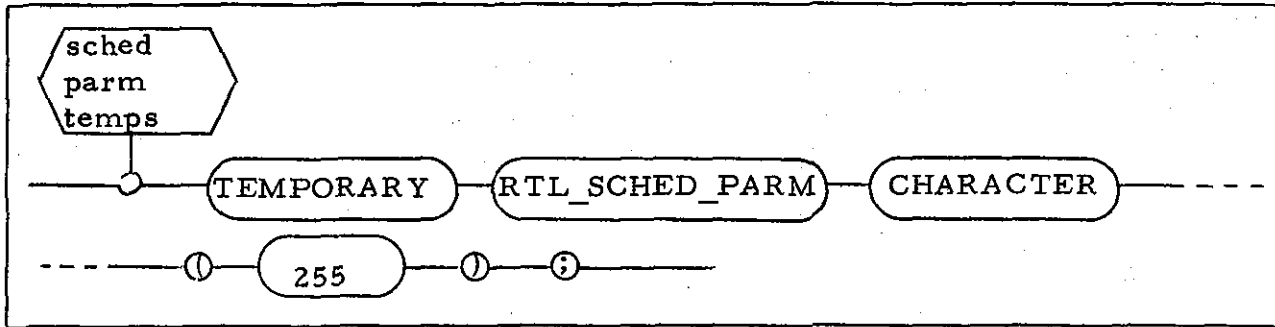
3.1.21 SCHEDULE Statement



HAL/S Code Template



where:



Notes

1. This <char exp> shall correspond to the parameter expression to be passed to the scheduled task.
2. This optional path shall be taken when an <event exp> appears in the HAL/SM construct. The path shall be repeated via a feedback loop as many times as there are event variables in the event expression. The <number> on this path shall be a one (1) the first time through, and shall be incremented each time the path is taken for a given HAL/SM construct. Each <bit var> on the path shall correspond to each of the <event var> s appearing in the HAL/SM construct.

3. This optional path shall be taken when an <event exp> appears in the HAL/SM construct. The path shall be repeated via a feedback loop as many times as necessary to initialize the array with the Reverse Polish representation of the <event exp> (see Reference 7, Section 5.2.1). The <number> on this path shall be one the first time through, and shall be incremented each time the path is taken for a given HAL/SM construct. Each <bit var> on the path shall correspond to the code for either an event operand in the HAL/SM construct or operator according to the conventions specified for MOSS event expressions.
4. This <label> shall be an entry point into the RTL to perform the required functions.
5. This <bit literal> shall be a bit vector indicating the presence of certain optional parameters in the RTL invocation.
6. This <label> shall be the HAL/S label corresponding to the task label (of the task being scheduled) in the HAL/SM construct.
7. This <arith exp> shall correspond to the <time value> in the IN or AT phrase of the HAL/SM construct, when present.
8. This optional path shall be taken when an <event exp> appears in the HAL/SM construct. The <number> indicates the number of <event var> s present in the <event exp>.
9. This optional path shall be taken when a THEN WAIT phrase appears in the HAL/SM construct. The <arith exp> shall correspond to the <time value> in the HAL/SM construct.
10. This optional path shall be taken when a REPEAT phrase appears in the HAL/SM construct. The <arith exp> shall correspond to the <time value> in the HAL/SM construct when the AFTER clause is present and shall be zero when it is absent.
11. This optional path shall be taken when a parameter to be passed to the scheduled task is specified in the HAL/SM construct.

Examples

```
L: SCHEDULE IOTA;
```

becomes

```
L: DO;
```

```
    CALL RTL_SCHED (BIN '0', IOTA);
```

```
END;
```

and

```
SCHEDULE DELTA IN 30 SECS WHEN EV1 & (EV2/EV3);
```

becomes

```
DO;
```

```
    TEMPORARY
```

```
        RTL_EVENT_VAR_LIST NAME ARRAY (3) BIT (32),  
        RTL_EVENT_EXP_STRING ARRAY (6) BIT (8);
```

```
    RTL_EVENT_VAR_LIST1 = EV1;
```

```
    RTL_EVENT_VAR_LIST2 = EV2;
```

```
    RTL_EVENT_VAR_LIST3 = EV3;
```

```
    RTL_EVENT_EXP_STRING1 = HEX '01';
```

```
    RTL_EVENT_EXP_STRING2 = HEX '02';
```

```
    RTL_EVENT_EXP_STRING3 = HEX '03';
```

```
    RTL_EVENT_EXP_STRING4 = HEX '4E';
```

```
    RTL_EVENT_EXP_STRING5 = HEX '5C';
```

```
    RTL_EVENT_EXP_STRING6 = HEX '7E';
```

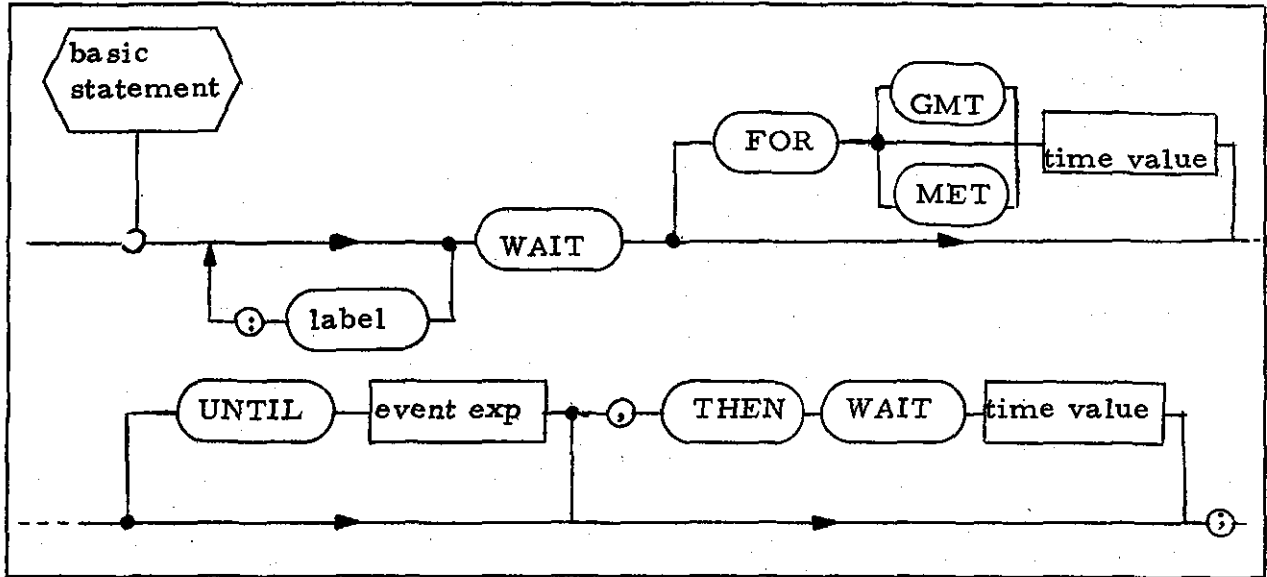
```
    CALL RTL_SCHED (BIN '001100', DELTA, (30000)@ DOUBLE,
```

```
        RTL_EVENT_VAR_LIST, RTL_EVENT_EXP_STRING, 3);
```

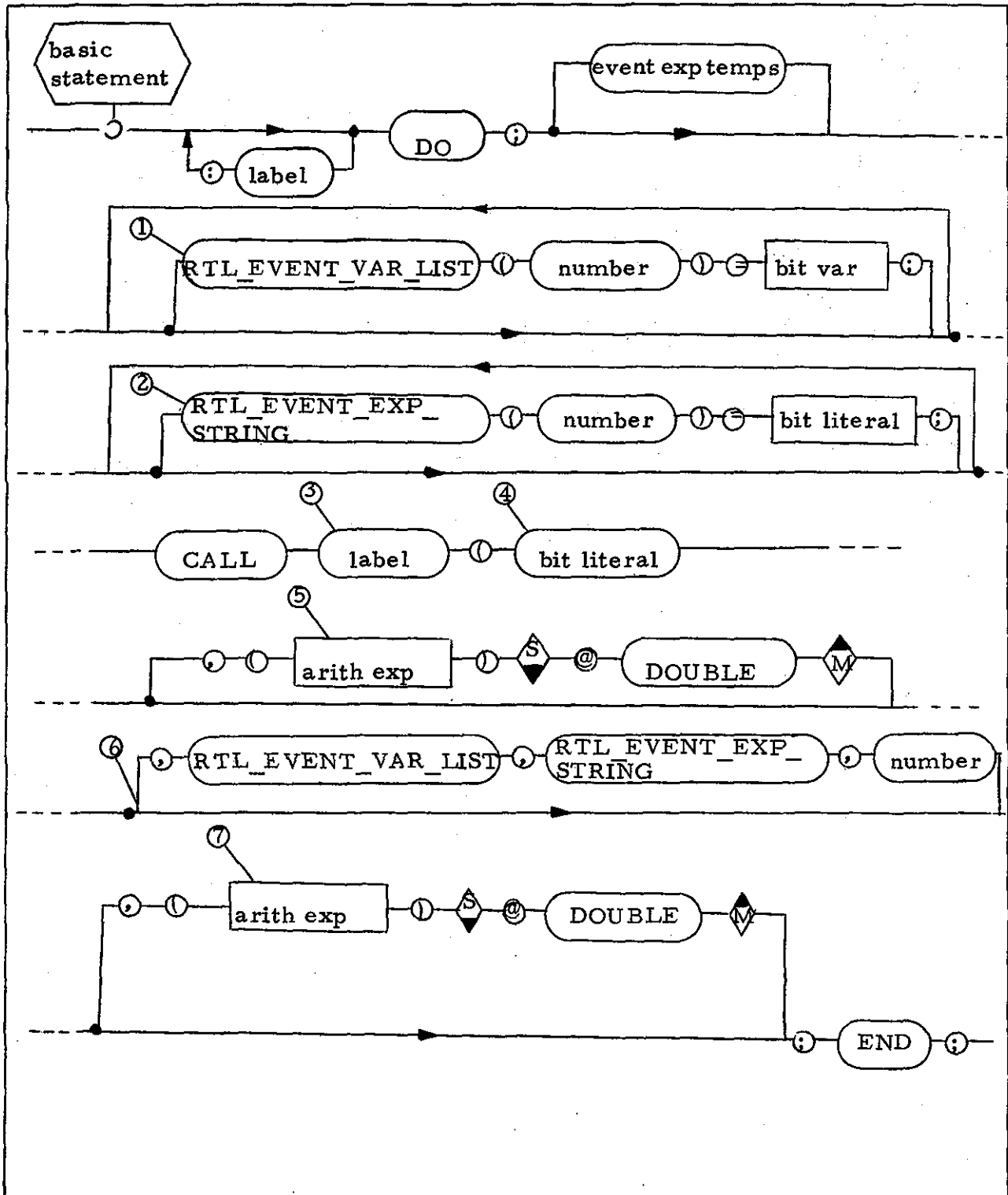
```
END;
```

3.1.22 WAIT Statement

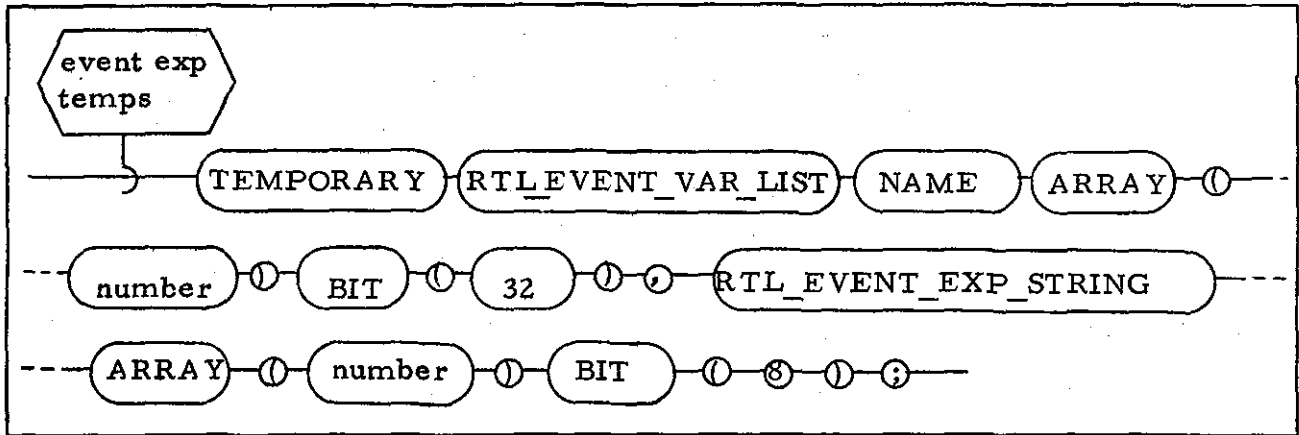
HAL/SM Syntax



HAL/S Code Template



where:



Notes

1. This optional path shall be taken when an <event exp> appears in the HAL/SM construct. The path shall be repeated via a feedback loop as many times as there are event variables in the event expression. The <number> on this path shall be a one the first time through, and shall be incremented each time the path is taken for a given HAL/SM construct. Each <bit var> on the path shall correspond to each of the <event var> s appearing in the HAL/SM construct.
2. This optional path shall be taken when an <event exp> appears in the HAL/SM construct. The path shall be repeated via a feedback loop as many times as necessary to initiate the array with the Reverse Polish representation of the <event exp> (see Reference 3, Section 5.2.1). The <number> on this path shall be one the first time through, and shall be incremented each time the path is taken for a given HAL/SM construct. Each <bit literal> on the path shall correspond to the code for either an event operand or operator in the HAL/SM construct according to the conventions specified for MOSS event expressions.
3. This <label> shall be an entry point into the RTL to perform the required function.
4. This <bit literal> shall be a bit vector indicating the presence of certain optional parameters in the RTL invocation.

5. This <arith exp> shall correspond to the <time value> in the FOR phrase of the HAL/SM construct, when present.
6. This optional path shall be taken when an <event exp> appears in the HAL/SM construct. The <number> indicates the number of <event var> s present in the <event exp> .
7. This optional path shall be taken when a THEN WAIT phrase appears in the HAL/SM construct. The <arith exp> corresponds to the <time value> in the HAL/SM construct.

Examples

```
STATE3:      WAIT FOR MET 50 SECS 25 MSECS
              UNTIL EVAR1/EVAR10;
```

becomes

```
STATE3:      DO;
              TEMPORARY
              RTL_EVENT_VAR_LIST NAME ARRAY (2)
              BIT (32),
              RTL_EVENT_EXP_STRING ARRAY (4)
              BIT (8);
              RTL_EVENT_VAR_LIST1 = EVAR1;
              RTL_EVENT_VAR_LIST2 = EVAR10;
              RTL_EVENT_EXP_STRING1 = HEX '01';
              RTL_EVENT_EXP_STRING2 = HEX '02';
              RTL_EVENT_EXP_STRING3 = HEX '4E';
              RTL_EVENT_EXP_STRING4 = HEX '7E';
              CALL RTL_WAIT(BIN '01110', (50000)@DOUBLE,
              RTL_EVENT_VAR_LIST, RTL_EVENT_EXP_
              STRING);
              END;
```

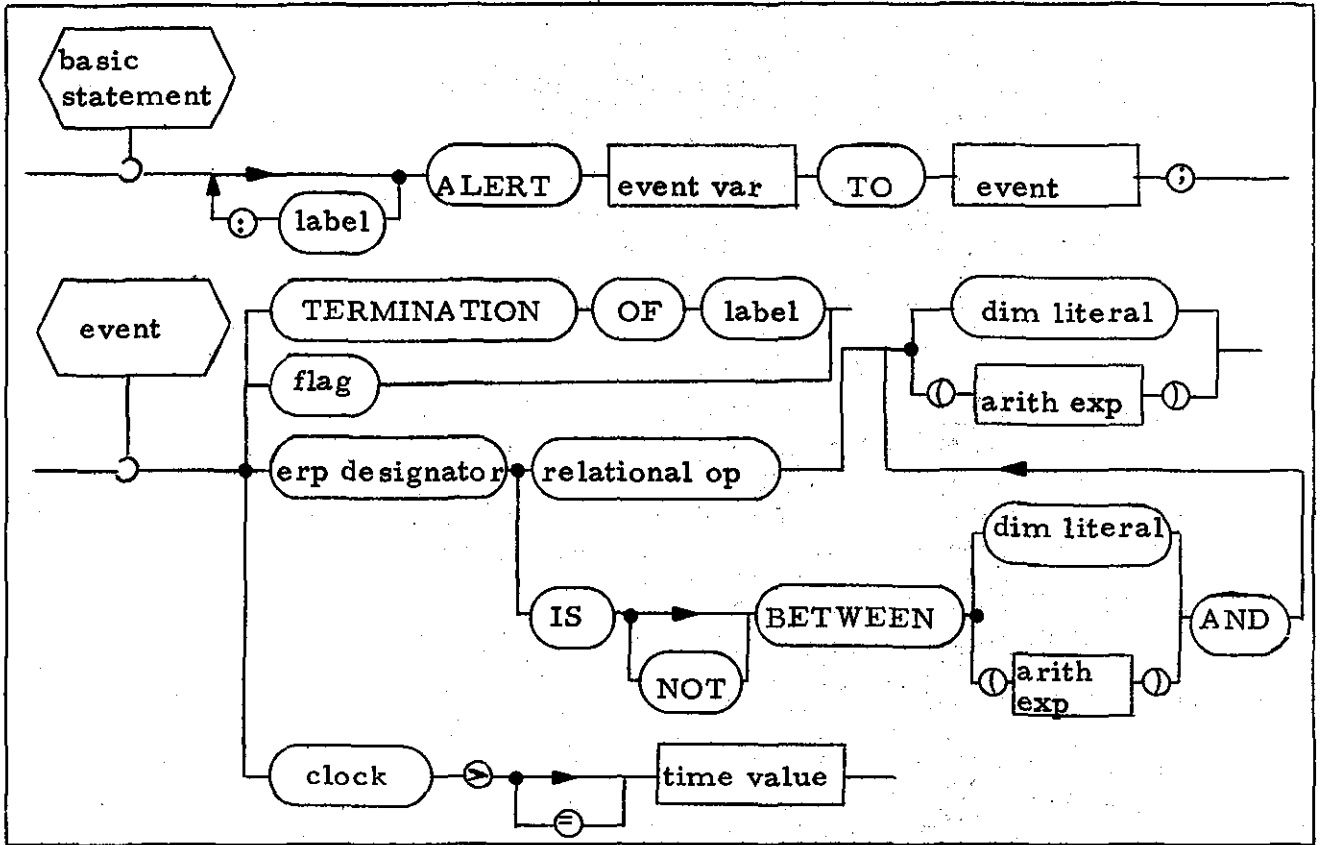
```
STATE4:      WAIT FOR GMT 2 TSTART;
```

becomes

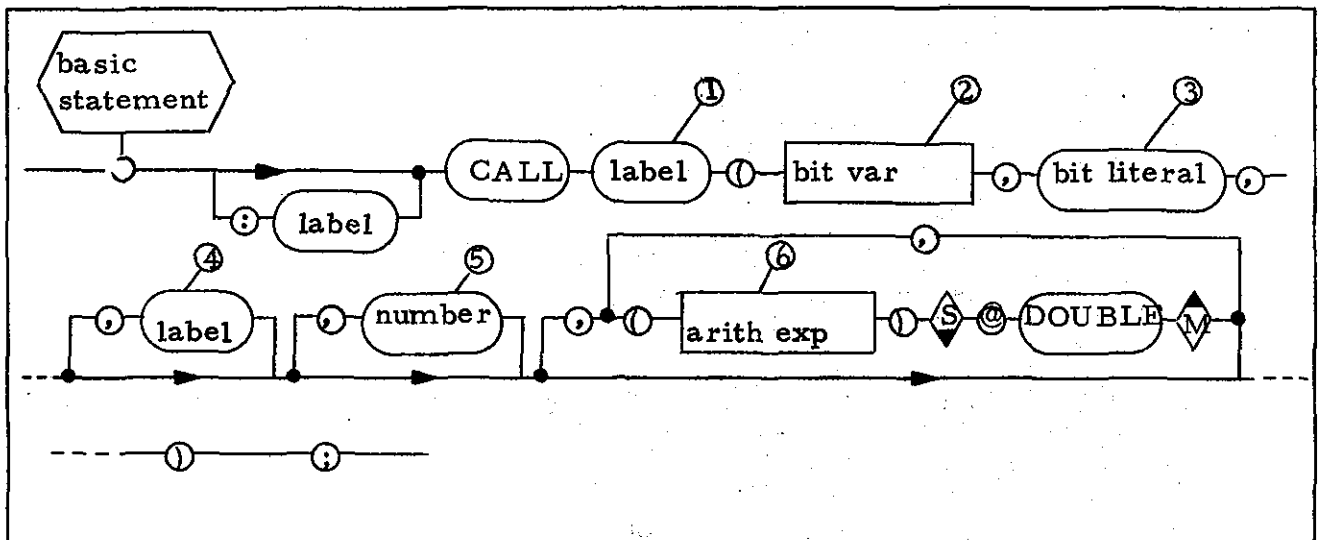
```
STATE4:      DO;
              CALL RTL_WAIT(BIN '0', 2 TSTART);
              END;
```


3.1.23 ALERT Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required functions.
2. This <bit var> shall correspond to the <event var> in the HAL/SM construct.
3. This <bit literal> shall indicate the presence of certain optional parameters in the RTL invocation.
4. This <label> optionally specifies the HAL/S <label> corresponding to the task label, the program flag, or the ERP designator in the HAL/SM construct.
5. This optional path shall be taken when the <ERP designator> or <clock> is specified in the HAL/SM construct. The <number> shall correspond to the <relational op>, the IS BETWEEN, IS NOT BETWEEN, or the >, = specification in the HAL/SM construct.
6. This optional path shall be taken when the <erp designator> or <clock> is specified in the HAL/SM construct. The <arith exp> shall correspond to the <dim literal> or <arith exp> following the <erp designator> or the <time value> following the <clock> in the HAL/SM construct.

Examples

```
STATE5:    ALERT EVENT_VAR1 TO TERMINATION OF TASKA;
```

becomes

```
STATE5:    CALL RTL_ALERT (EVENT VAR1, BIN '0', TASKA);
```

and

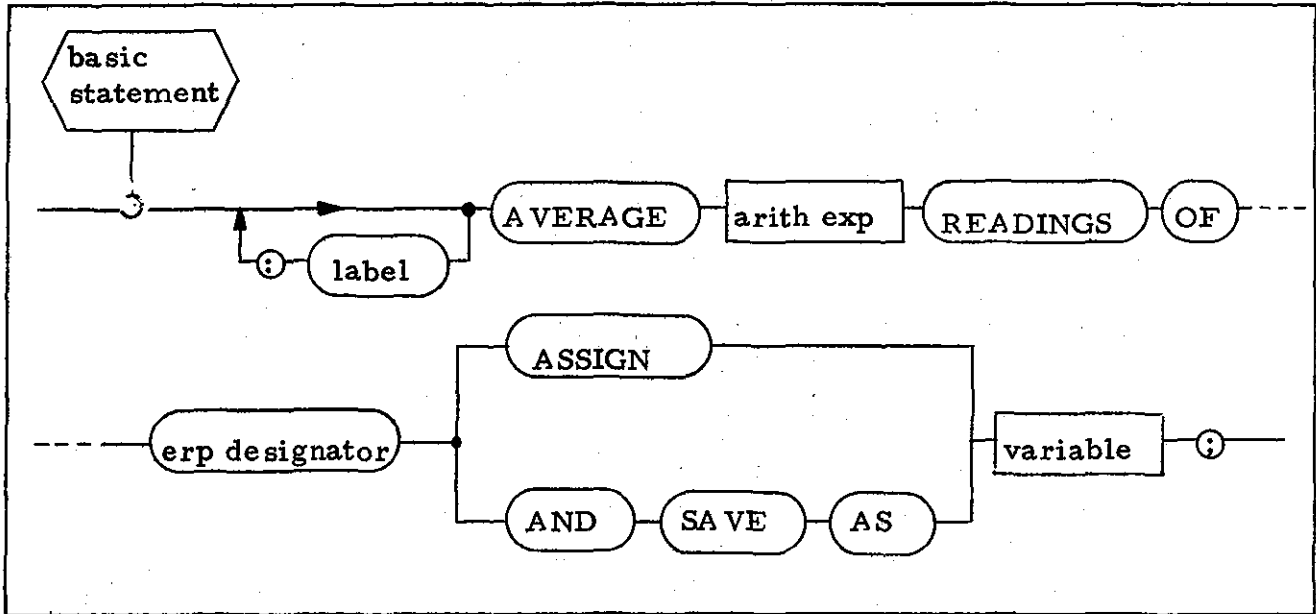
```
STATE7:    ALERT EVENT_VAR5 TO A115  2 DELTAV;
```

becomes

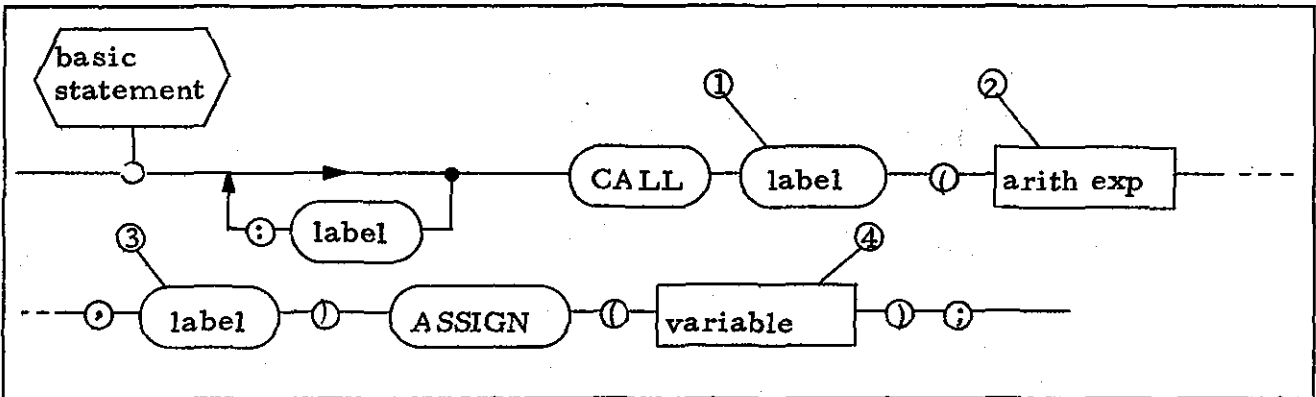
```
STATE7:    CALL RTL_ALERT (EVENT_VAR5, BIN '11', 0A115, 0,  
                          (2 DELTV)@DOUBLE);
```

3.1.24 AVERAGE AI Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required function.
2. This <arith exp> corresponds to the <arith exp> which specifies the number of readings to be averaged in the HAL/SM construct.

3. This <label> shall specify the HAL/S <label> which corresponds to the HAL/SM <erp designator>.
4. This <variable> shall correspond to the <variable> which specifies the data storage area in the HAL/SM construct.

Example

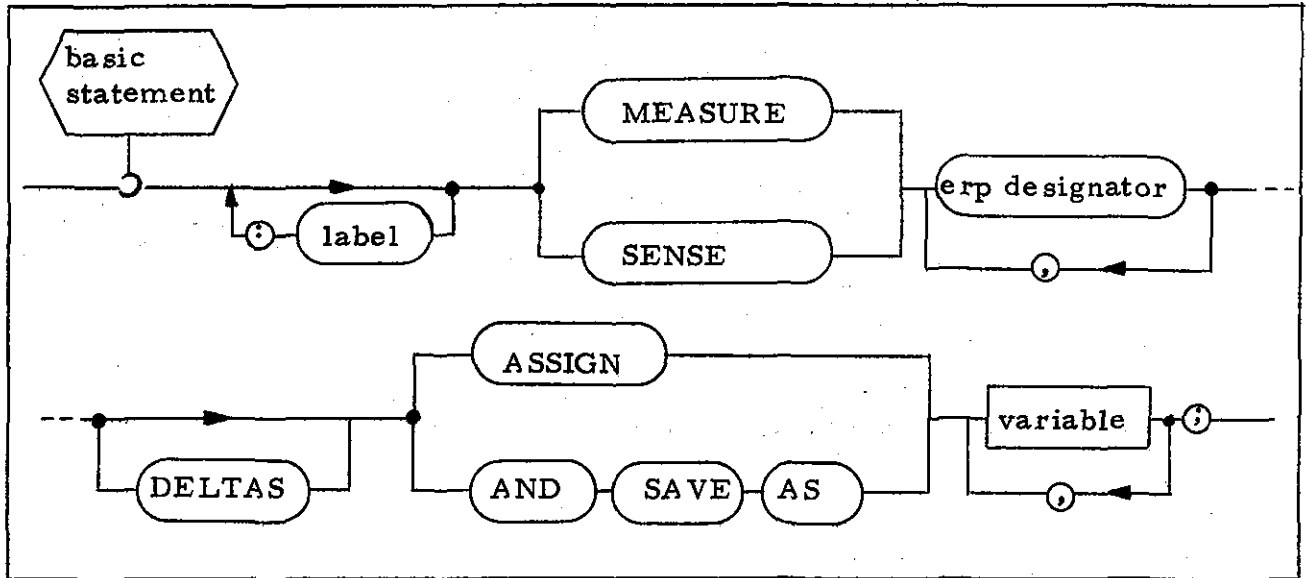
```
STATE10:  AVERAGE 2*N READINGS OF AINPUT_25  
          ASSIGN AVERAGE_VALUE;
```

becomes

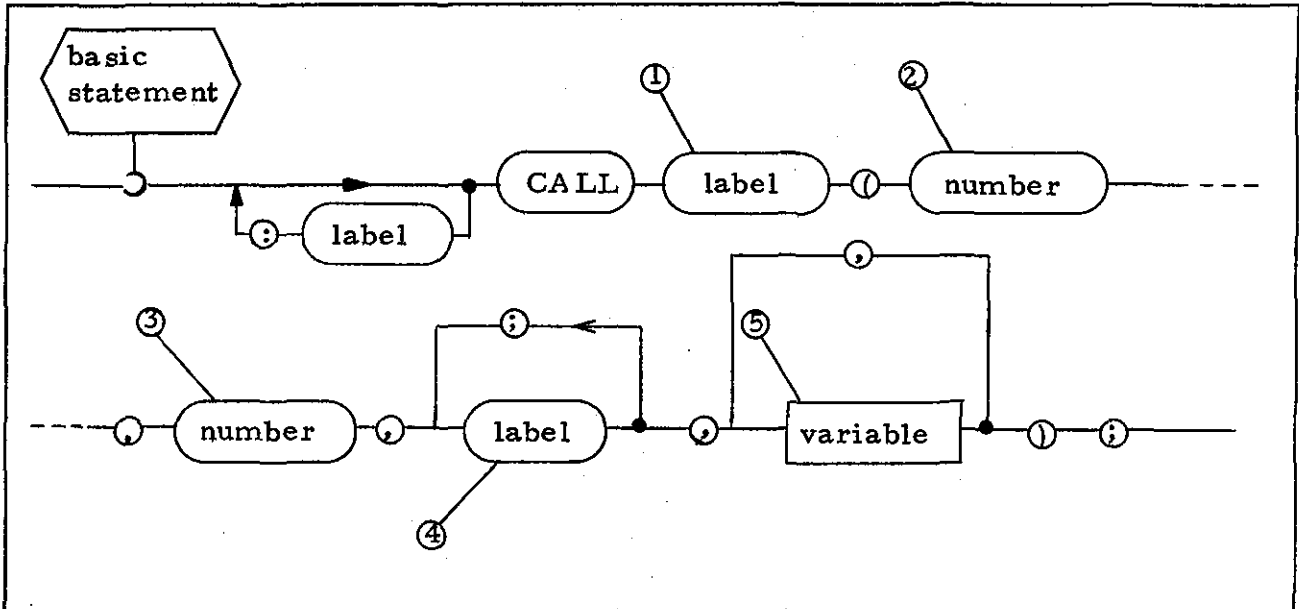
```
STATE10:  CALL RTL_AVERAGE(2*N, 0AI25) ASSIGN  
          (AVERAGE_VALUE);
```

3.1.25 READ ERP Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required function.
2. This <number> shall indicate the presence of the DELTAS option in the HAL/SM construct.
3. This <number> shall specify the number of <label> s and < variable> s which follow.
4. This <label> shall specify the HAL/S <label> which corresponds to the HAL/SM <erp designator>.
5. This <variable> shall correspond to the <variable> which specifies the data storage area in the HAL/SM construct.

Example

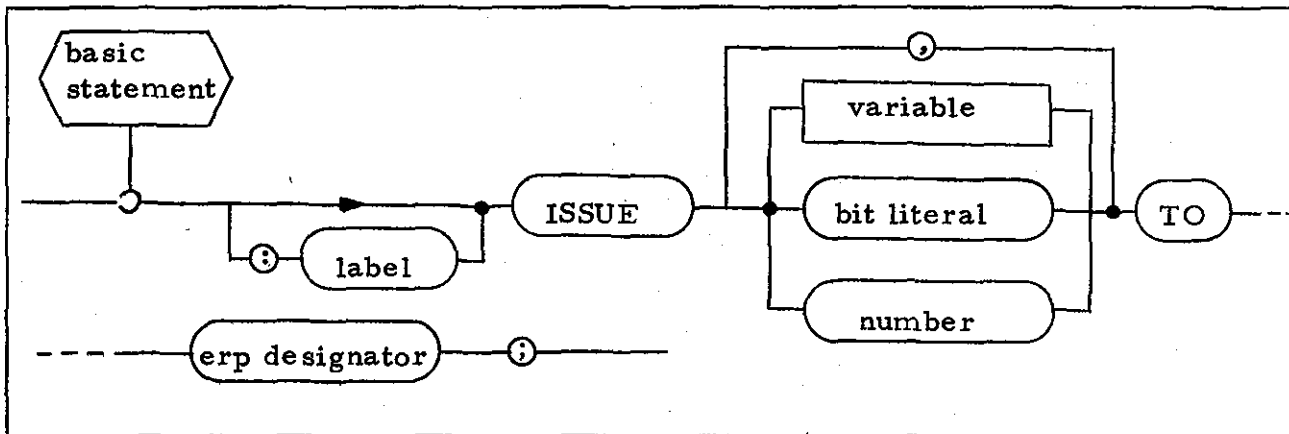
SENSE AINPUT-10 AND SAVE AS ALPHA;

becomes

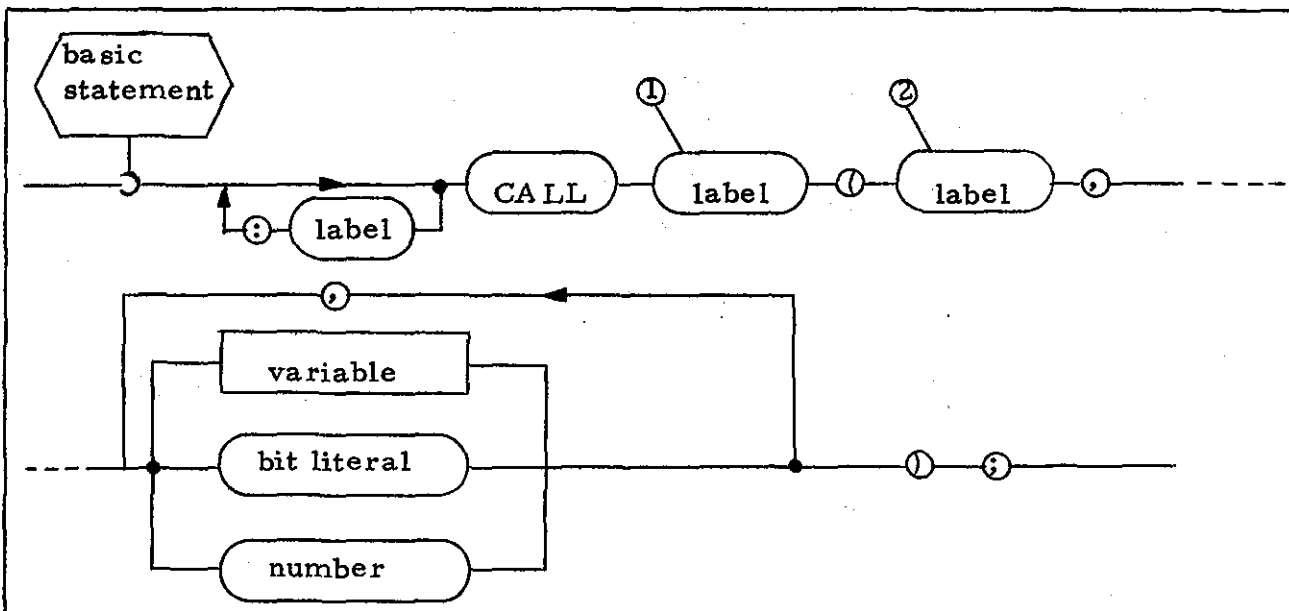
CALL RTL_SENSE (0, 1, 0A110, ALPHA);

3.1.26 ISSUE Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required functions.
2. This <label> shall specify the HAL/S <label> which corresponds to the HAL/SM <erp designator>.

Example

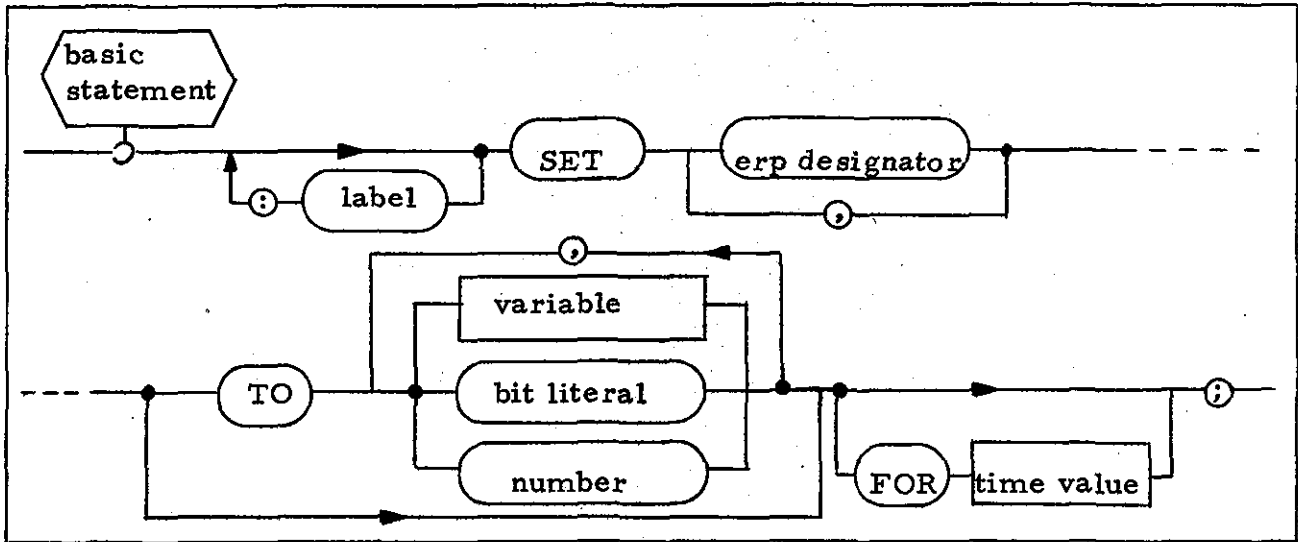
STATE1: ISSUE B21, B32 TO RECORD_23;

becomes

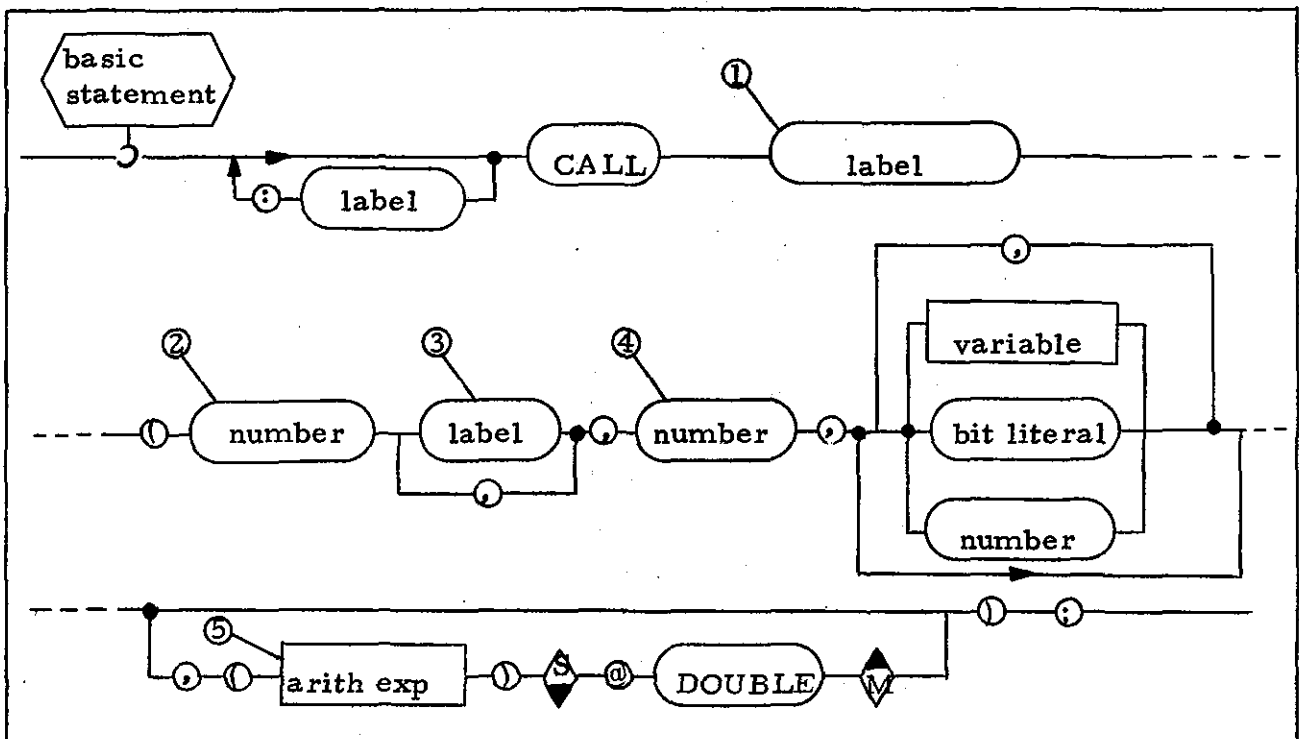
STATE1: CALL RTL_ISSUE (0R23, B21, B23);

3.1.27 SET Discrete Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required function.
2. This <number> shall specify the number of <label> s to follow.
3. This <label> shall specify the HAL/S <label> which corresponds to the HAL/SM <erp designator>.
4. This <number> shall specify the number of values to follow.
5. This <arith exp> shall correspond to the <time value> in the FOR phrase of the HAL/SM construct, when present.

Example

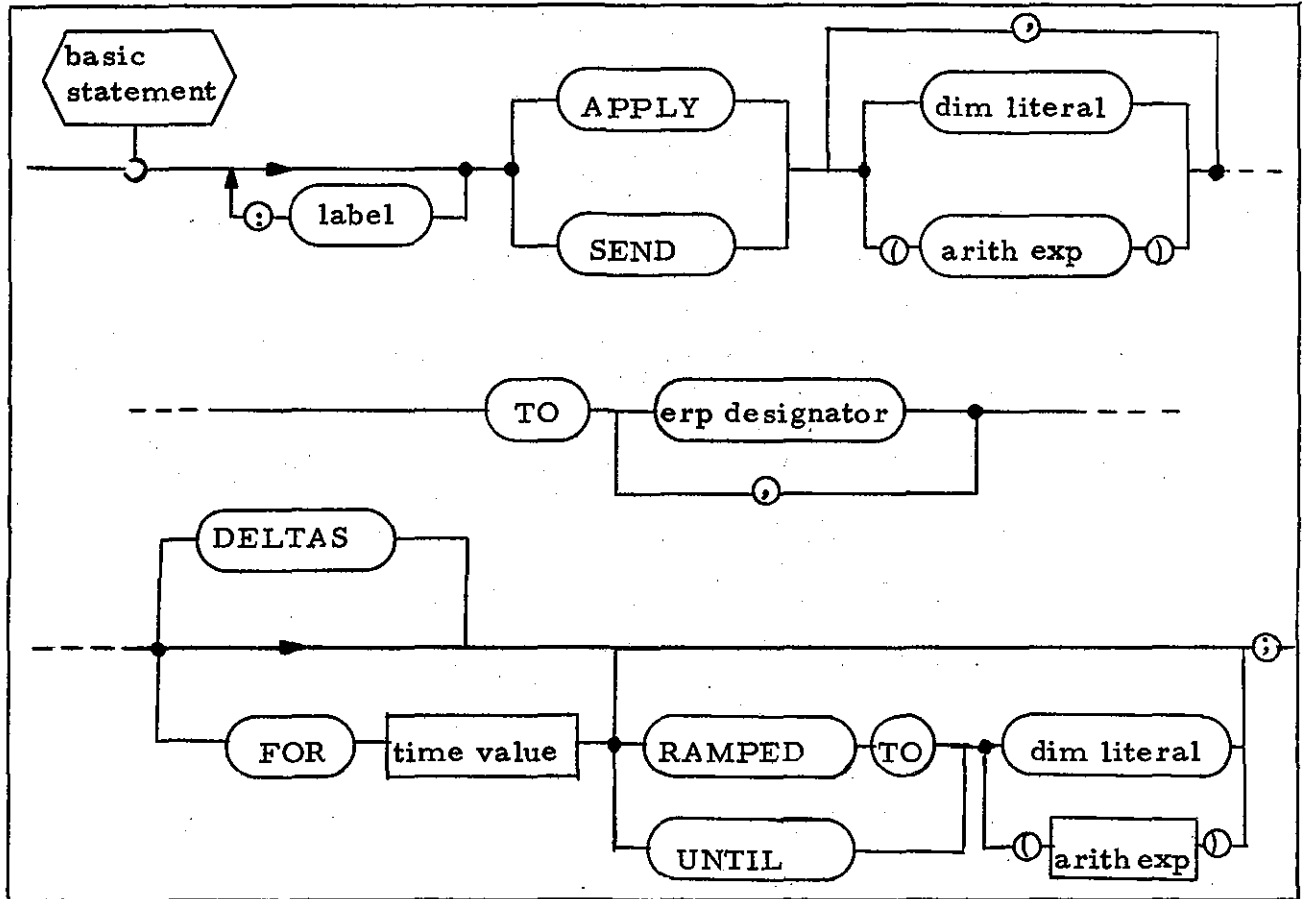
```
STATE5:      SET DO11, DO12 TO 10, 12 FOR DELTAT;
```

becomes

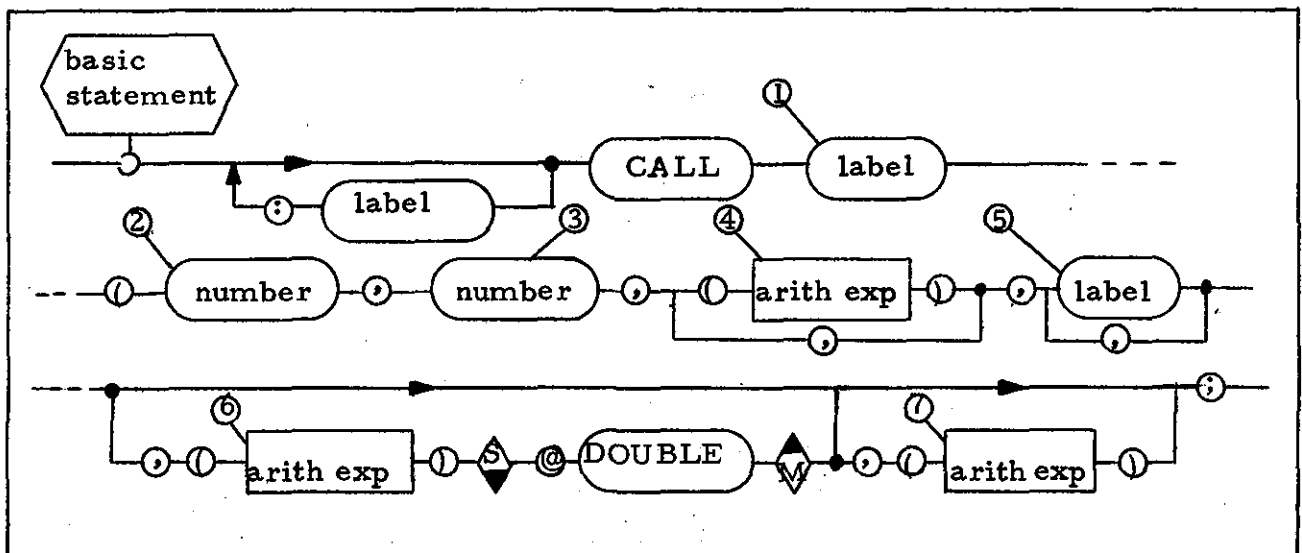
```
STATE5:      CALL RTL_SETD(2, 0DO11, 0DO12, 2, 10, 12,  
                      (DELTAT)@DOUBLE);
```

3.1.28 APPLY Analog Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required function.
2. This <number> shall indicate the presence of the DELTAS, RAMPED TO, or pulsed options.
3. This <number> shall specify the number of <arith exp> s and <label> pairs to follow.
4. This <arith exp> shall specify the value or values to be APPLIED to the <erp designator>.
5. This <label> shall specify the HAL/S <label> which corresponds to the HAL/SM <erp designator>.
6. This optional path shall be taken when the value(s) is to be pulsed or ramped. The <arith exp> shall correspond to the <time value> in the FOR phrase of the HAL/SM construct.
7. This <arith exp> shall correspond to the <dim literal> or <arith exp> following RAMPED TO or UNTIL in the HAL/SM construct.

Example

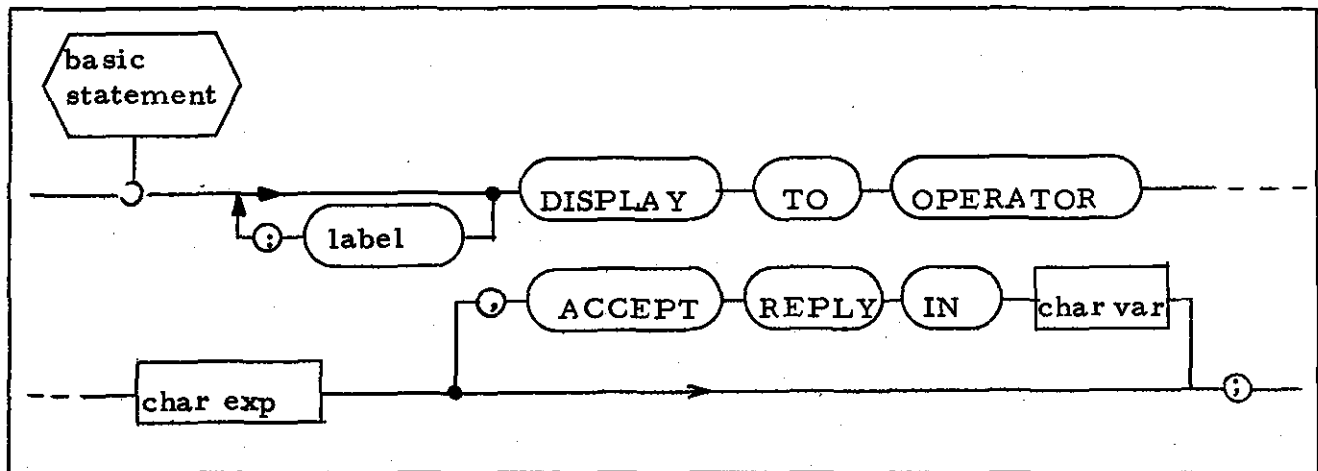
```
STATE25:    APPLY DELTAV TO AO24 FOR 2 TSTART UNTIL MAXV;
```

becomes

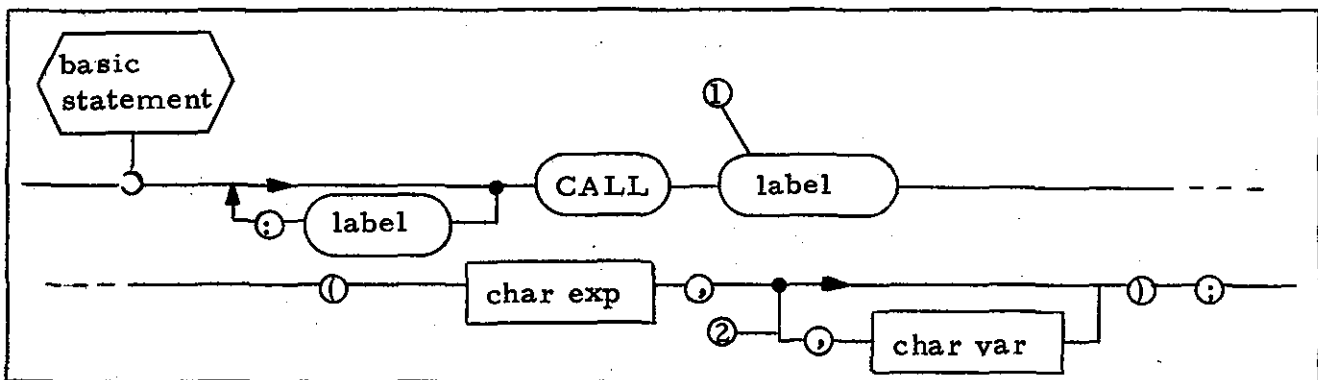
```
STATE25:    CALL RTL_APPLY (1, 1, (DELTAV), OA024, (2 TSTART)
                @DOUBLE, (MAXV));
```

3.1.29 DISPLAY TO OPERATOR Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required function.
2. This optional path shall be taken when ACCEPT REPLY IN appears in HAL/SM construct.

Example

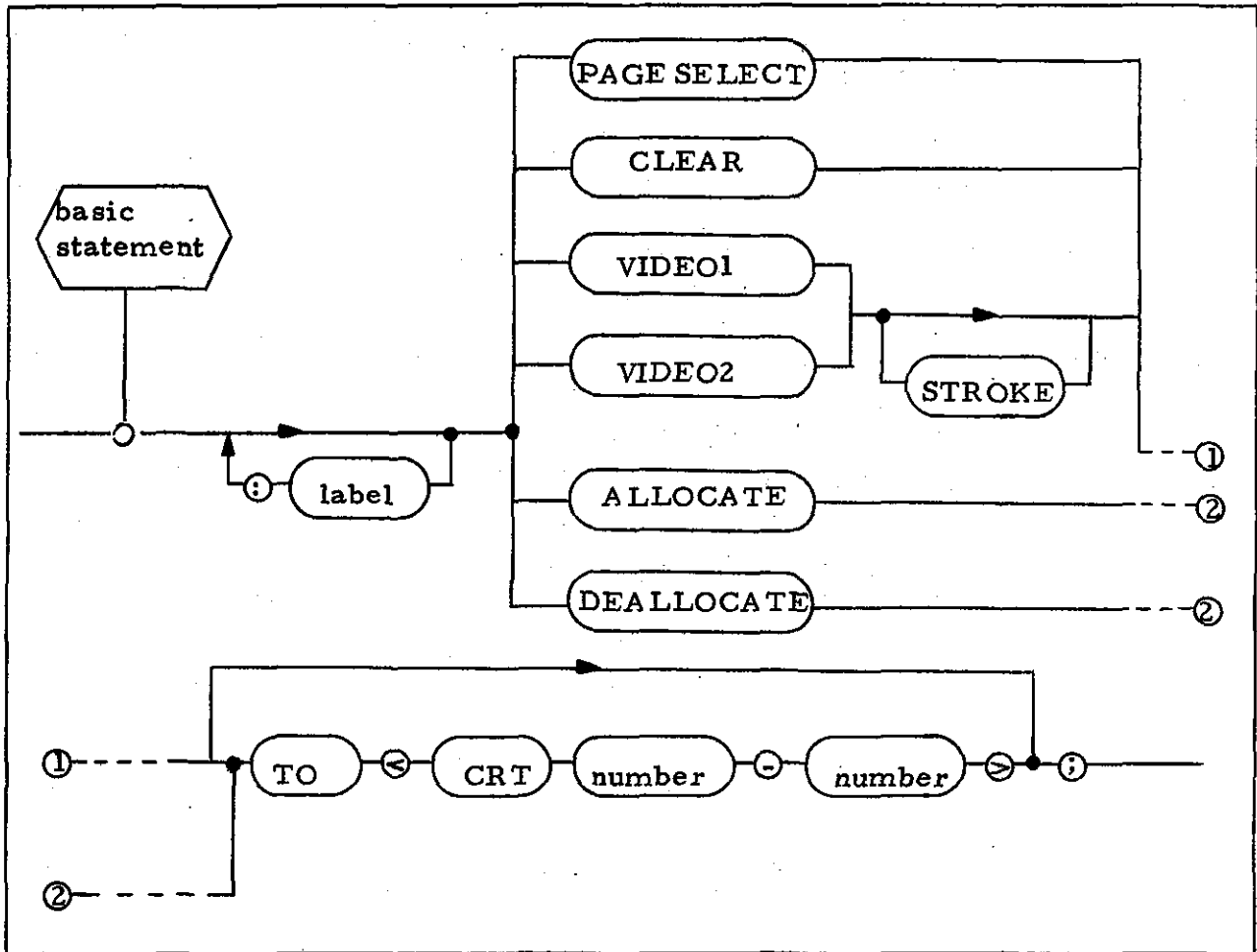
```
MESS1:      DISPLAY TO OPERATOR 'HAS JOBA BEEN INITIATED',  
            ACCEPT REPLY IN REPLY_BUFFER;
```

becomes

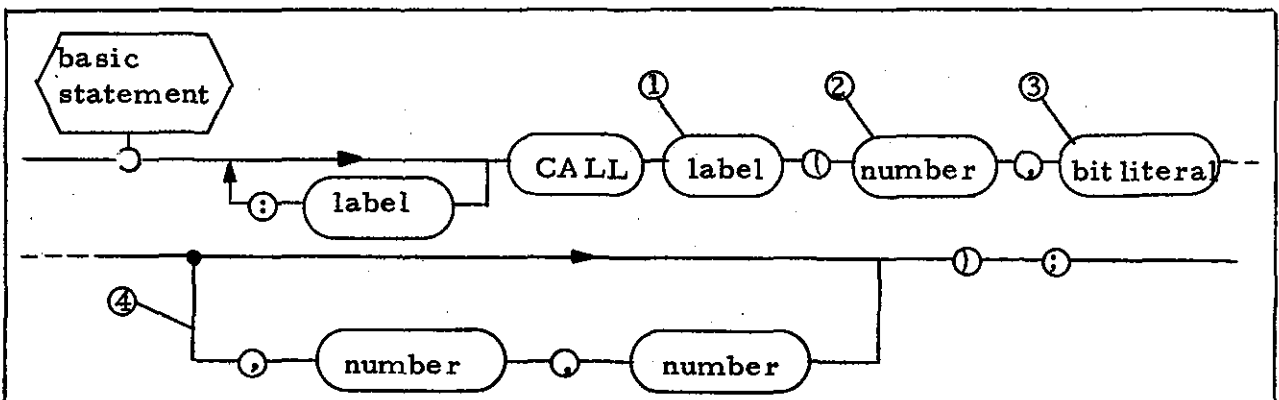
```
MESS1:      CALL RTL_WTOR ('HAS JOBA BEEN INITIATED',  
                          REPLY_BUFFER);
```

3.1.30 DISPLAY CONTROL Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point in the RTL to perform the required function.
2. This <number> shall indicate which control function is to be performed.
3. This <bit literal> shall indicate the presence of the optional RTL CRT parameters.
4. This optional path shall be taken when the TO <CRT <number> - <number>> is specified in the HAL/SM construct.

Example

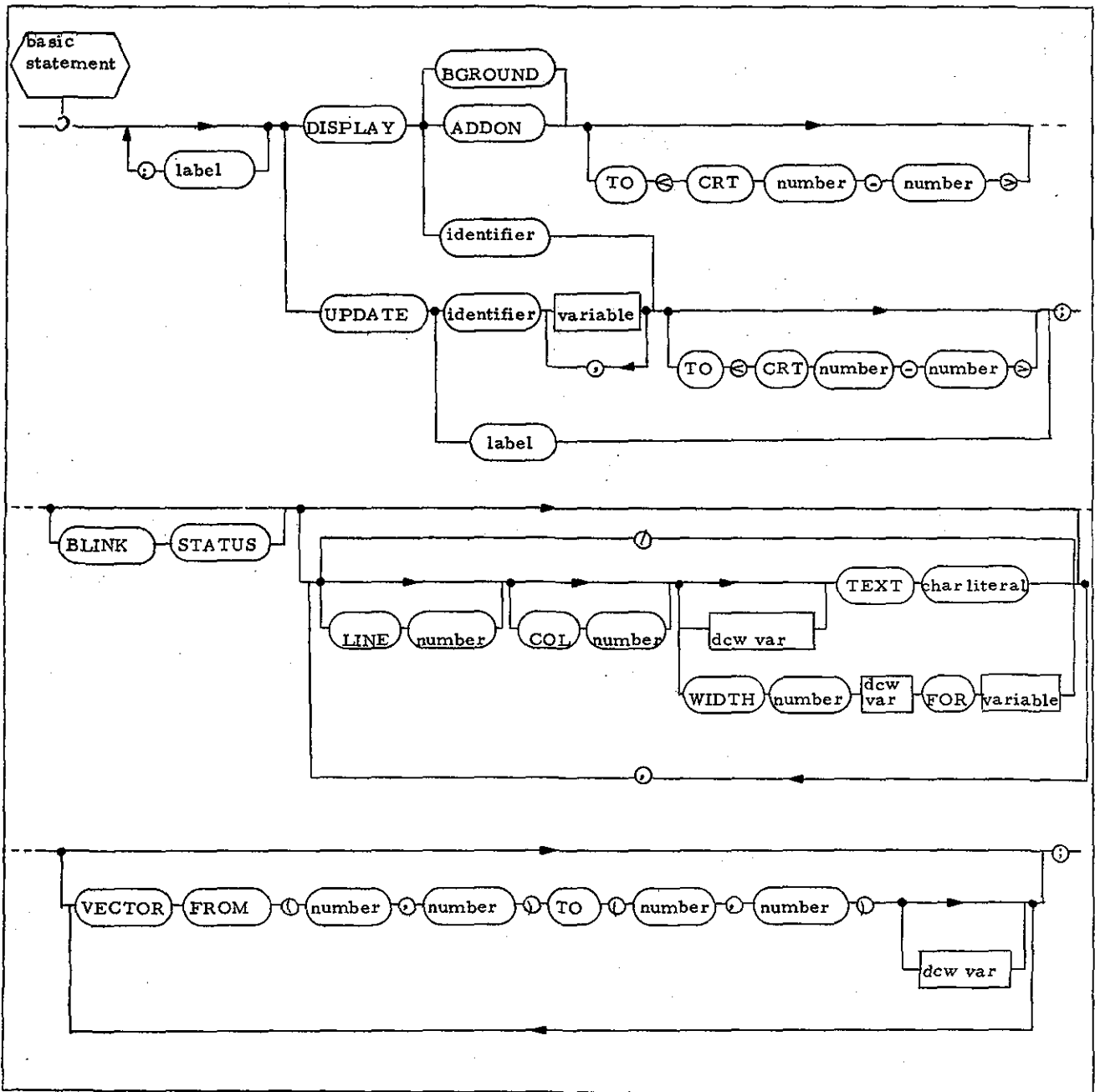
CONTROL1: VIDEO2 STROKE TO <CRT 1-4>;

becomes

CONTROL1: CALL RTL_DISCON(6, BIN '1', 1, 4);

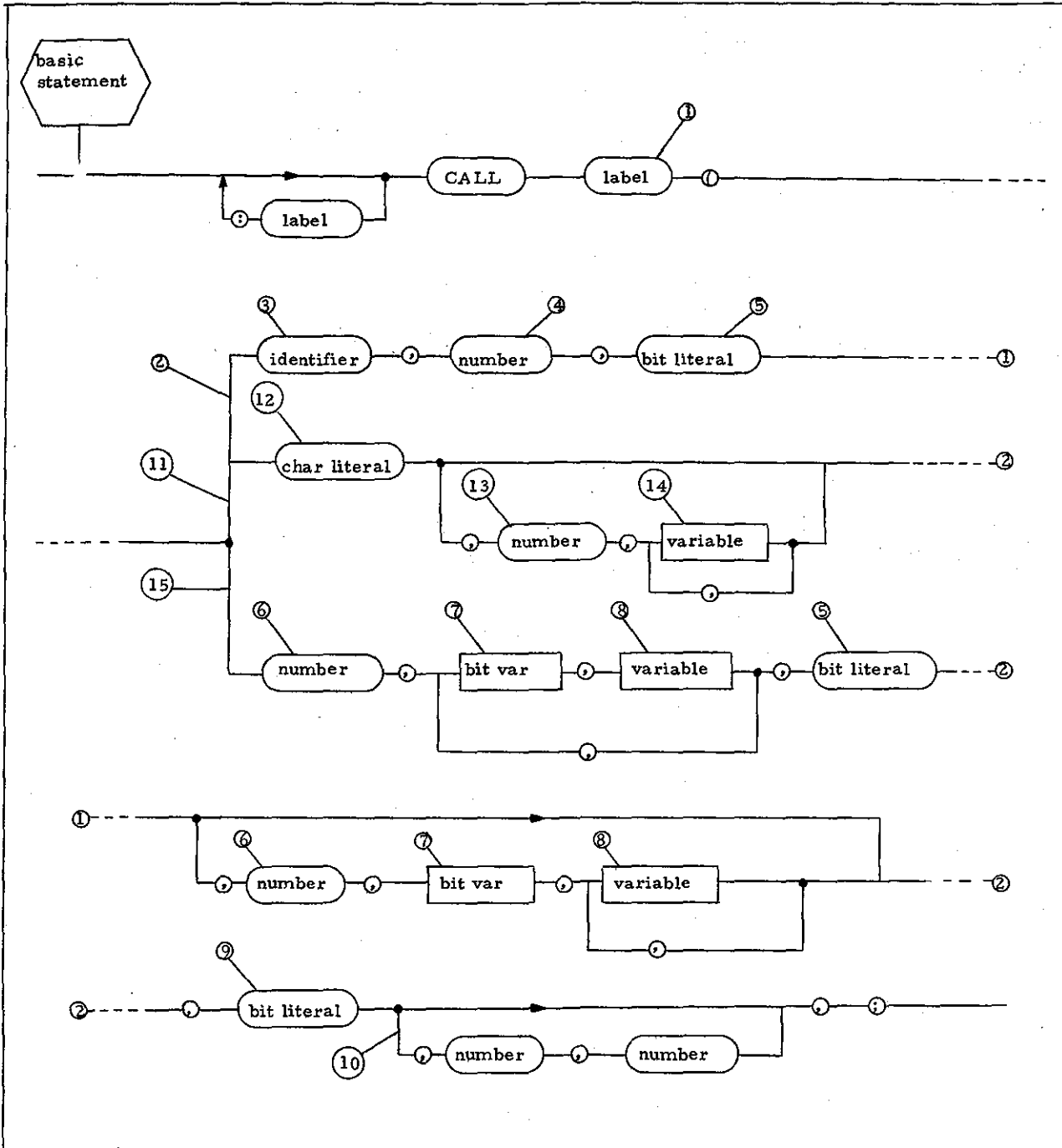
3.1.31 Display Data Statement

HAL/SM Syntax



ORIGINAL PAGE IS
OF POOR QUALITY

HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required functions.
2. This optional path shall be taken when DISPLAY BGROUND or DISPLAY ADDON are specified in the HAL/SM construct.
3. This <identifier> shall specify the program label by which the BACKGROUND and ADDON disk messages are identified.
4. This <number> shall identify the record within the data set specified by note 2 that contains the BACKGROUND or ADDON message.
5. This <bit literal> shall specify the presence of the BLINK STATUS option in the HAL/SM construct.
6. This <number> shall specify the number of <variable> and associated <dcw var> pairs in the DISPLAY BGROUND and DISPLAY ADDON options of the HAL/SM construct.
7. This <bit var> shall correspond to the <dcw var> following the WIDTH <number> specification in the HAL/SM construct.
8. This <variable> corresponds to the <variable> following the FOR in the HAL/SM construct.
9. This <bit literal> shall specify the presence of the <CRT <number <- <number << specification in the HAL/SM construct.
10. This optional path is taken when <CRT <number> - <number>> is specified in the HAL/SM construct.
11. This optional path is taken when the DISPLAY <identifier> or UPDATE <identifier> is specified in the HAL/SM construct.
12. This <char literal> corresponds to the <identifier> s mentioned in note 11.
13. This <number> shall specify the number of <variable> s in the UPDATE <identifier> option of the HAL/SM construct.

14. This optional path shall correspond to the UPDATE <identifier> option in the HAL/SM construct. The <variable> shall correspond to the <variable> after <identifier> in the HAL/SM construct.
15. This optional path shall correspond to the UPDATE <label> in the HAL/SM construct.

Examples

```
STATE1:    DISPLAY ADDON TO <CRT 1 - 4> BLINK STATUS
           TEXTCW1 TEXT 'THIS IS SAMPLE DATA',
           WIDTH 10 VARCW1 FOR DATA_ARRAY;
```

becomes

```
STATE1:    CALL RTL_DISADD(PROG1, 10, BIN '1', 1, VARCW1,
           DATA_ARRAY, BIN '1', 1, 4);
```

and

```
STATE5:    DISPLAY BGROUND LINE 3 COL 10 TEXTCW1 TEXT
           'THE ALTITUDE VARIATIONS ARE', LINE 5 COL
           15 WIDTH 3 VARCW2 FOR ALTITUDE;
```

becomes

```
STATE5:    CALL RTL_DISBACK(PROG1, 15, BIN '0', BIN '0');
```

and

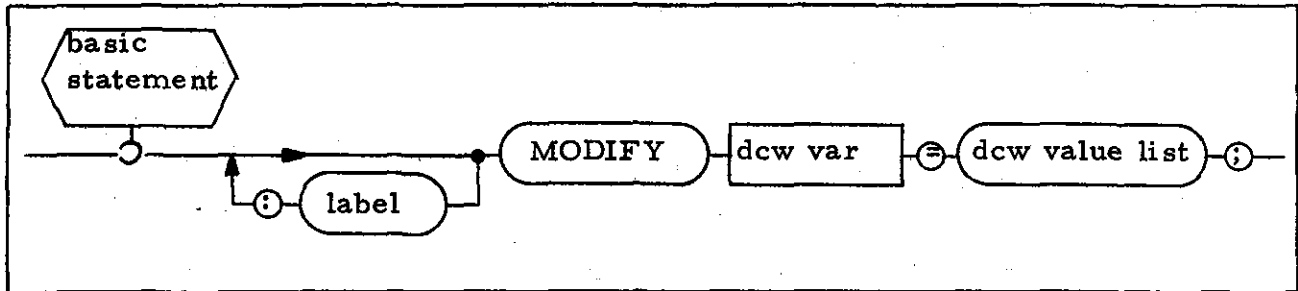
```
STATE10:   UPDATE STATE5;
```

becomes

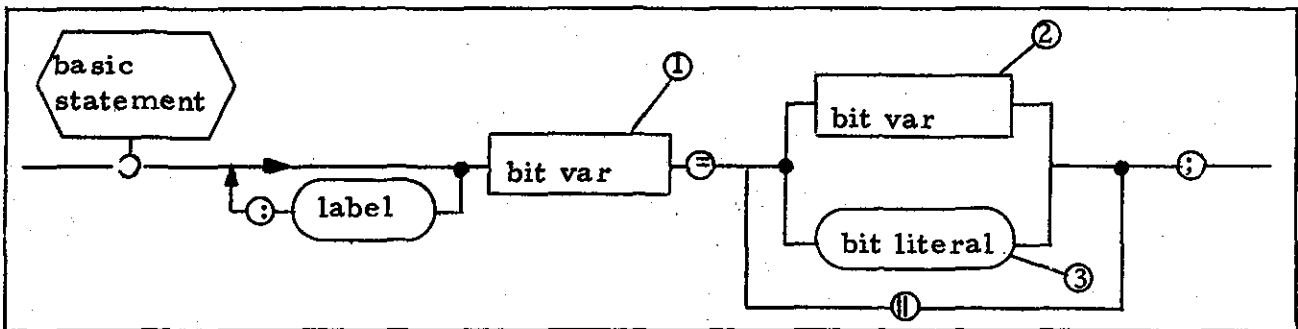
```
STATE10:   CALL RTL_DISUPD(1, VARCW2, ALTITUDE, BIN '0',
           BIN '0');
```

3.1.32 Modify VCW Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <bit var> shall be the HAL/S variable corresponding to the <dcw var> in the HAL/SM construct.
2. This <bit var> shall be the HAL/S variable corresponding to the <dcw var> in the HAL/SM construct with added component subscripting to select a specific bit field within the variable.
3. This <bit literal> shall be an encoded value specifying one or more of the dcw options specified to be modified by the <dcw value list> of the HAL/SM construct.

Example

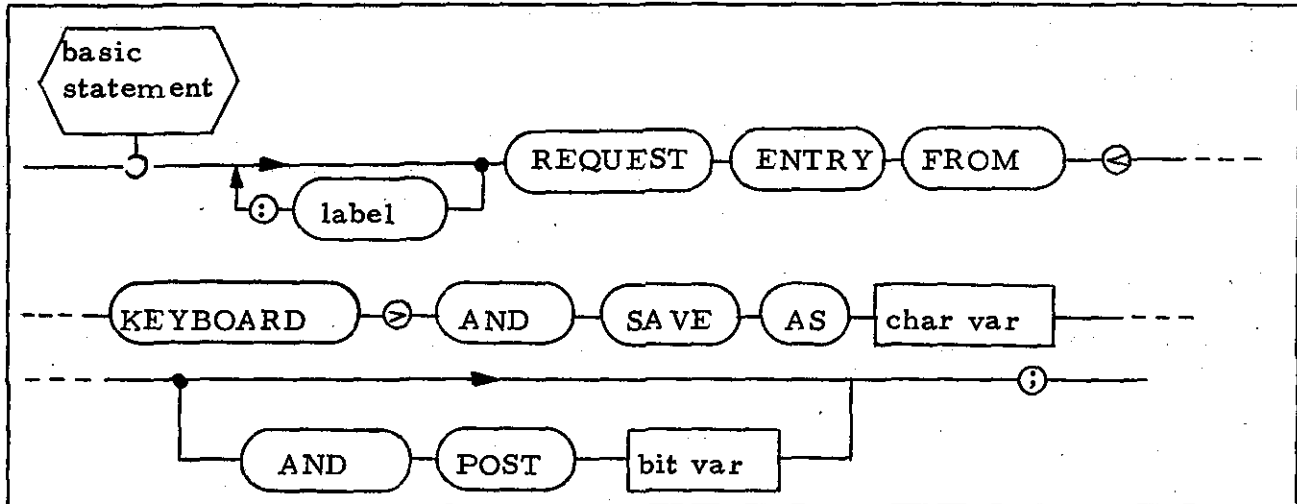
```
MODIFY DCW_1 = BLINK ON;
```

becomes

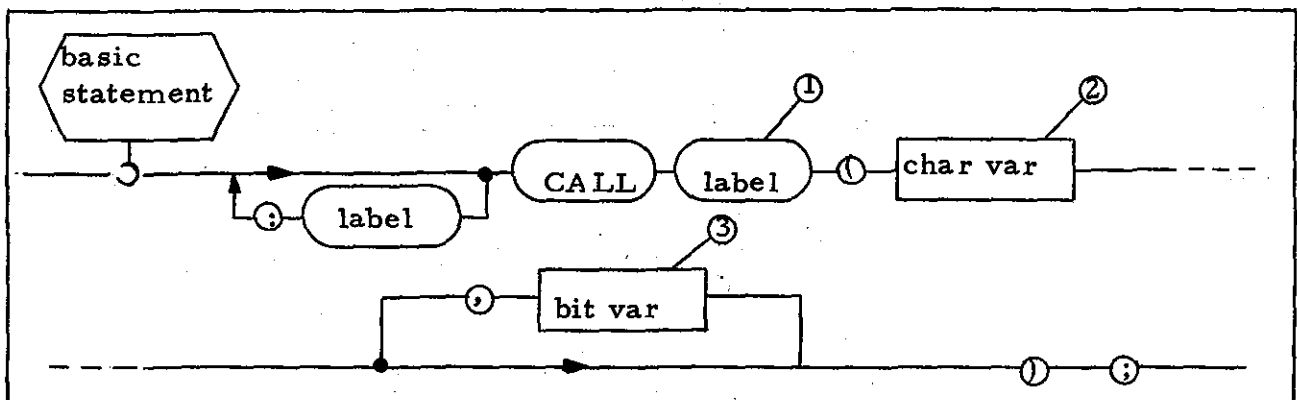
```
DCW_1 = DCW_1_0 TO 14 ||BIN '1' ||DCW_1_16 TO 31;
```

3.1.33 REQUEST Keyboard Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required function.
2. This <char var> shall be the same as the corresponding character variable in the HAL/SM construct.
3. This <bit var> shall be the same as the corresponding bit variable in the HAL/SM construct, when present.

Example

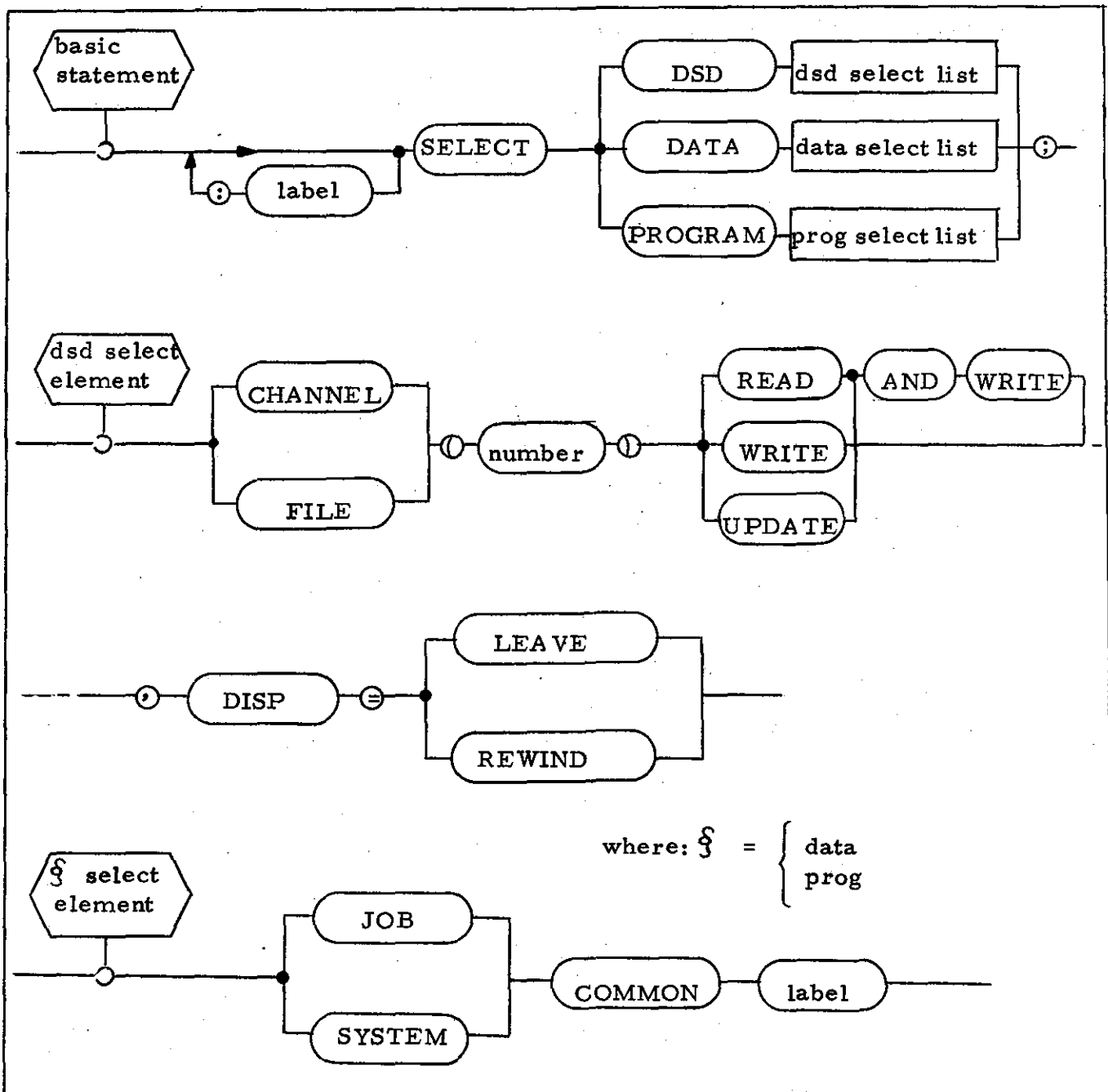
REQUEST ENTRY FROM<KEYBOARD>AND SAVE AS RESPONSE;

becomes

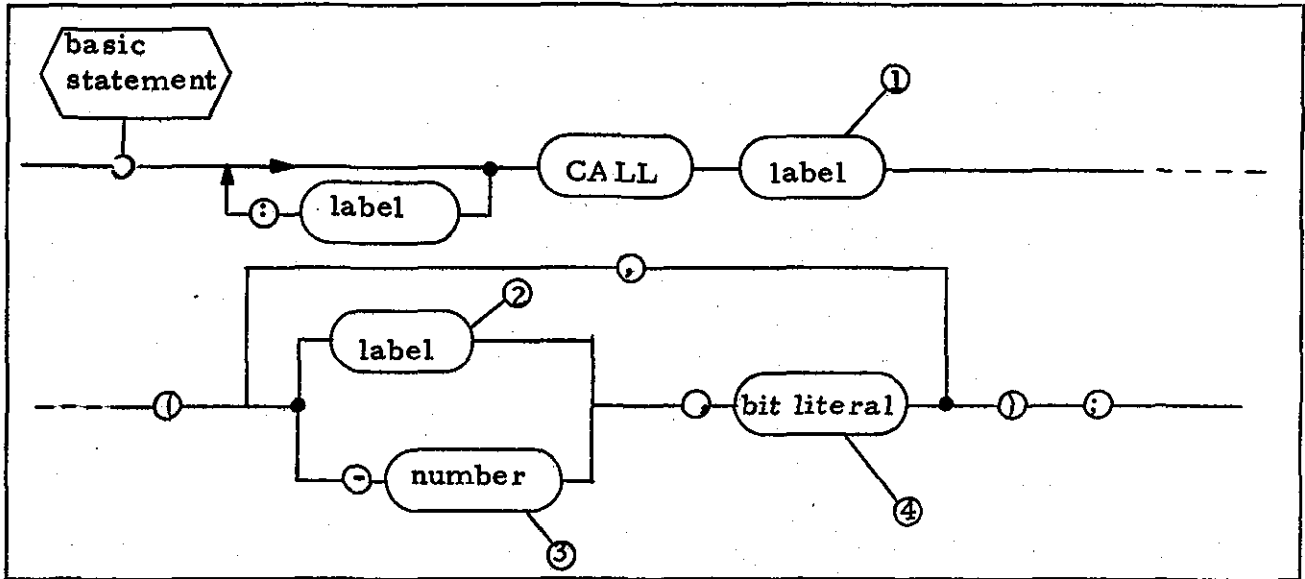
CALL RTL_RQST_KYBD(RESPONSE);

3.1.34 SELECT Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required function.
2. This <label> shall correspond to the DATA or PROGRAM module label specified in the HAL/SM construct, when present.
3. This <number> shall correspond to the SPIOS I/O channel or file number specified in the HAL/SM construct, when present. The range of values shall distinguish between channels and files.
4. This <bit literal> shall indicate the access rights specified in the HAL/SM construct for each resource. The format is the same as the parameter list entry 2 for the SELECT SVC interface to MOSS (see Reference 7, Section 5.5.1).

Example

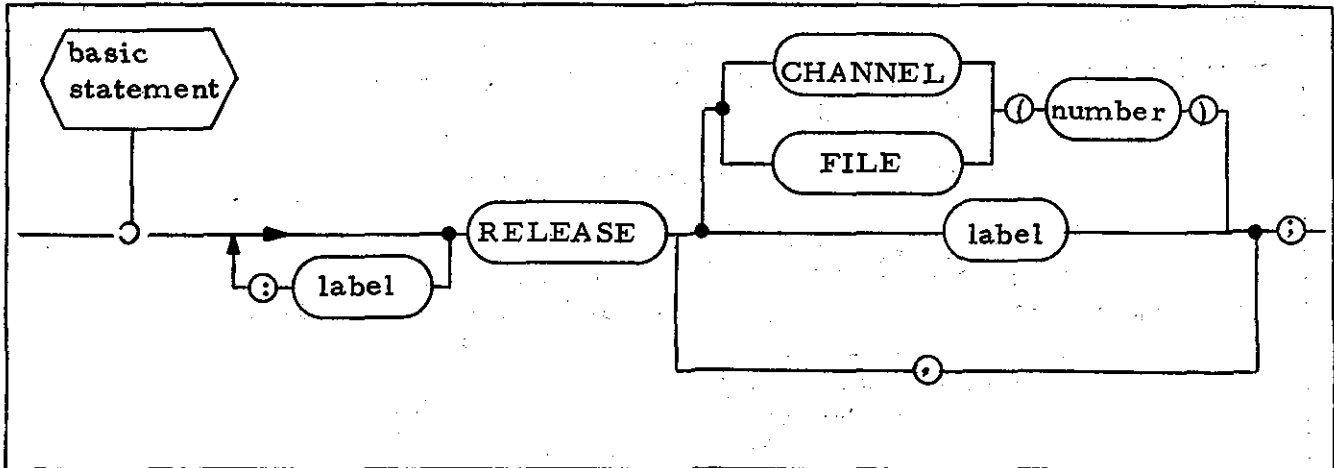
```
SELECT DSD CHANNEL (6) WRITE, DISP = LEAVE;
```

becomes

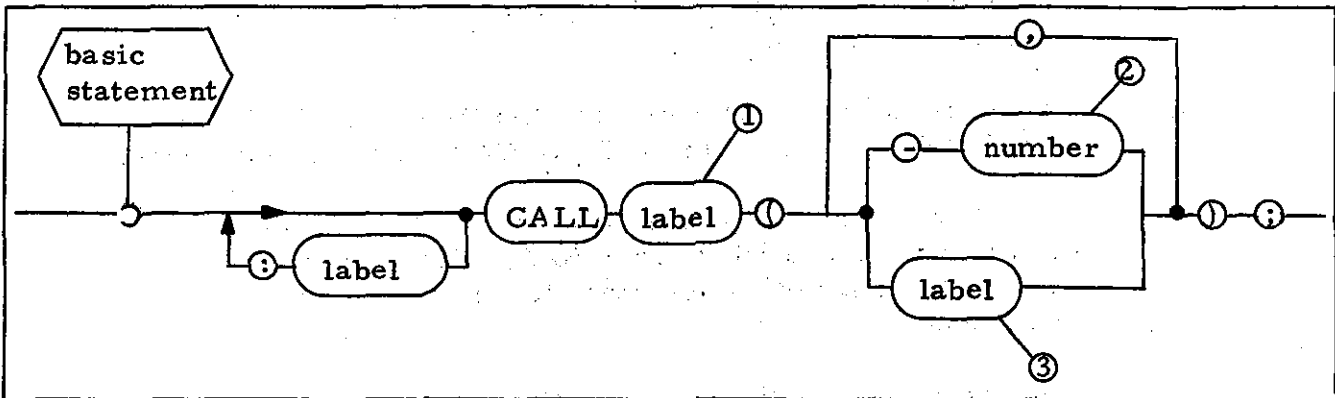
```
CALL RTL_SELECT (-6, HEX'02010102');
```

3.1.35 RELEASE Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required function.
2. This <number> shall correspond to the SPIOS I/O channel or file number specified in the HAL/SM construct.
3. This <label> shall correspond to the resource label specified in the HAL/SM construct.

Example

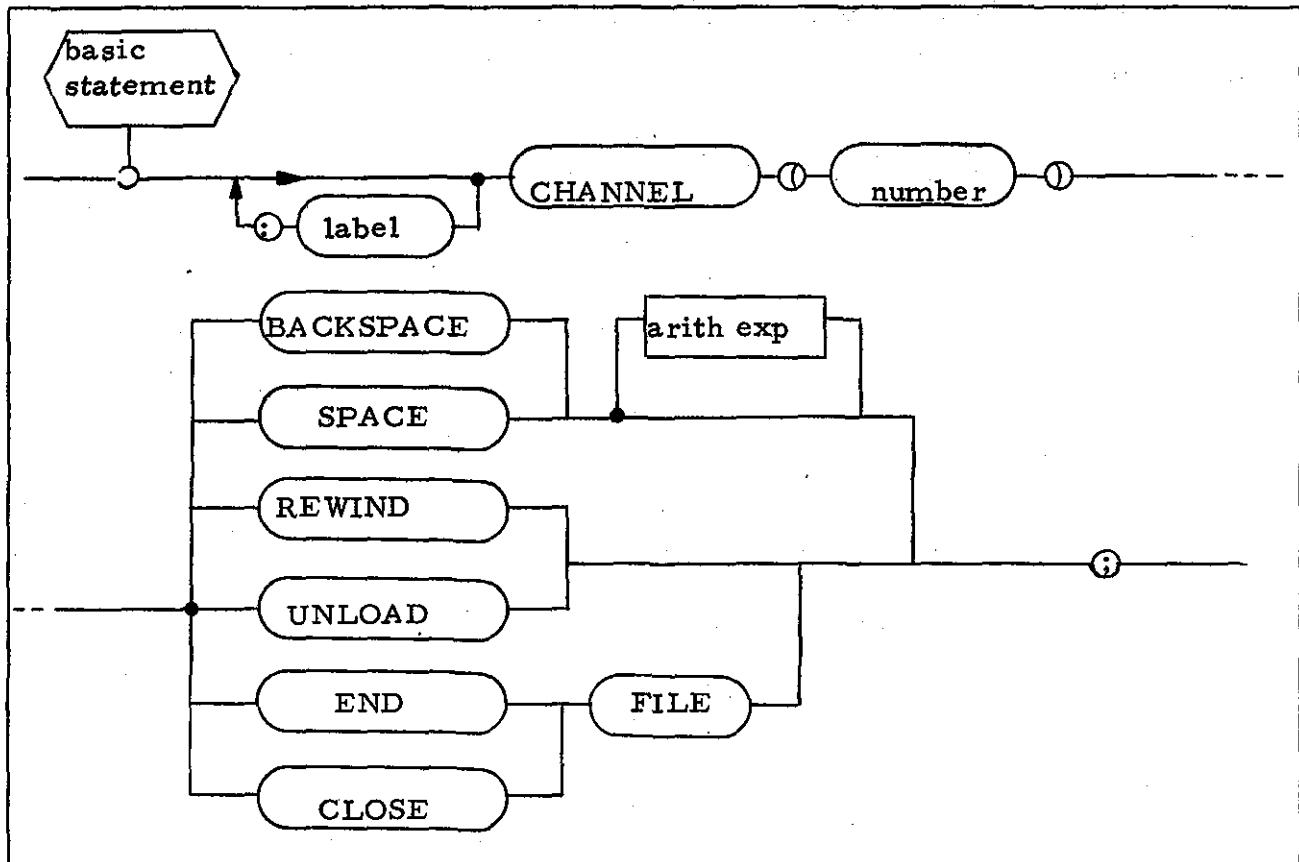
RELEASE DATAMOD, CHANNEL (3);

becomes

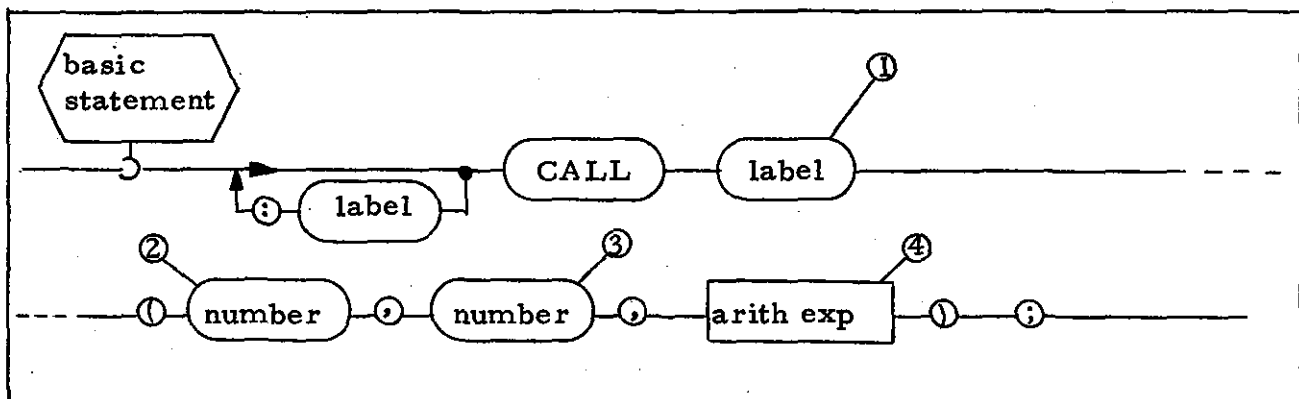
CALL RTL_RLSE (DATAMOD, -3);

3.1.36 CHANNEL Control Statement

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be an entry point into the RTL to perform the required function.
2. This <number> shall be the channel number of the SPIOS I/O channel being manipulated.
3. This <number> shall indicate the function to be performed.
4. This <arith exp> shall correspond to the <arith exp> in the HAL/SM construct, when present, and shall have a value of one, otherwise.

Example

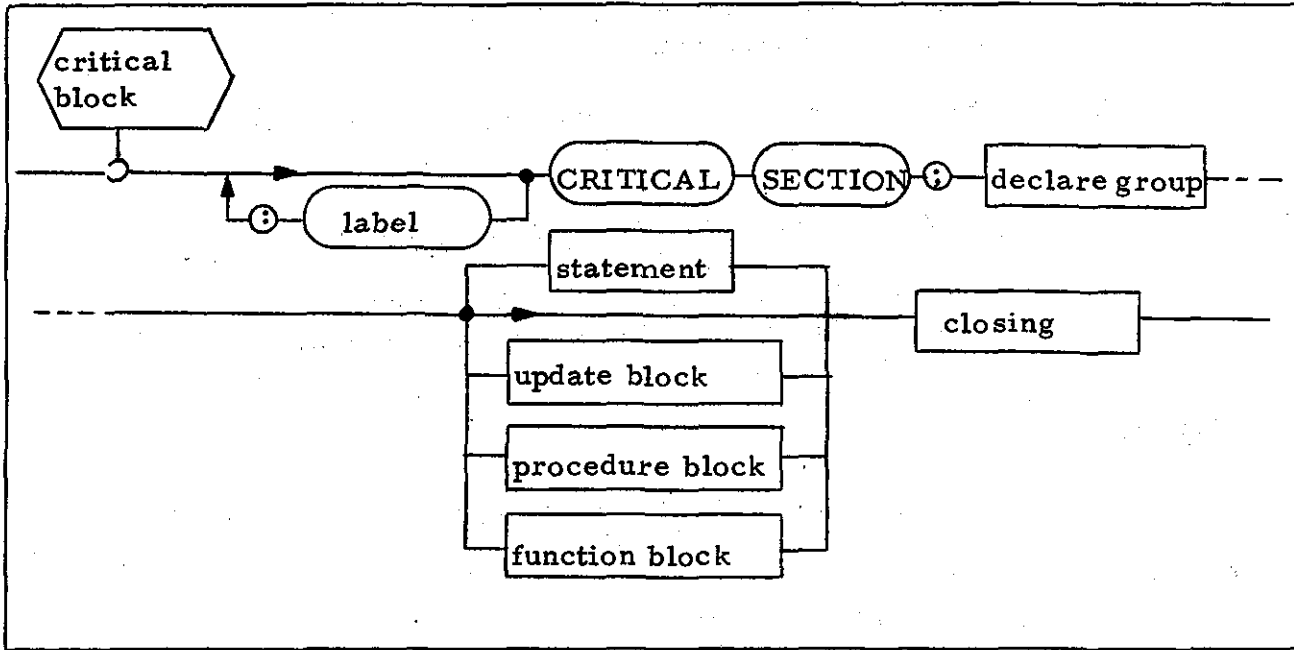
L: CHANNEL (3) SPACE N_FILES + 1;

becomes

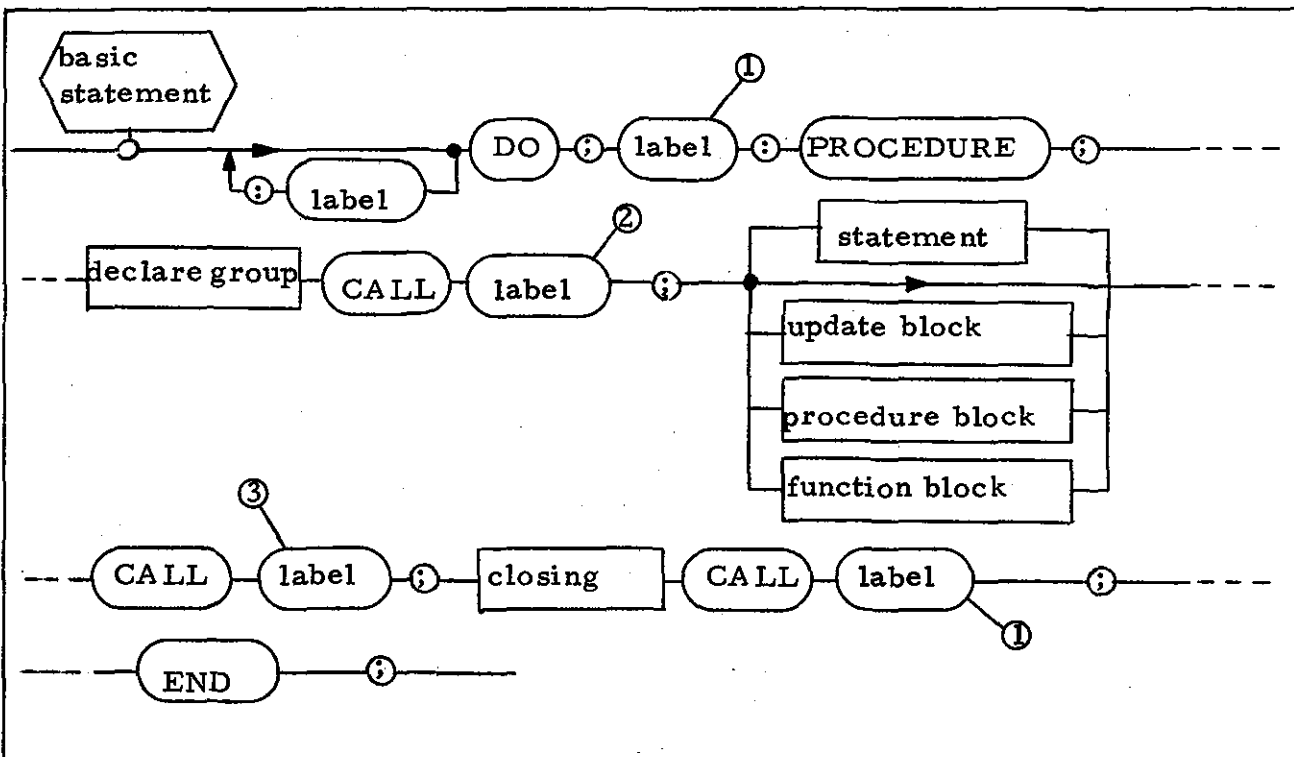
L: CALL RTL_CHAN (3,2,N_FILES+1);

3.1.37 CRITICAL SECTION

HAL/SM Syntax



HAL/S Code Template



Notes

1. This <label> shall be a symbol generated by the preprocessor to identify the nested procedure. This procedure is an artifice to force the compiler to enter a new name scope.
2. This <label> shall be an entry point into the RTL to cause the task to enter critical mode.
3. This <label> shall be an entry point into the RTL to cause the task to exit from critical mode.

Example

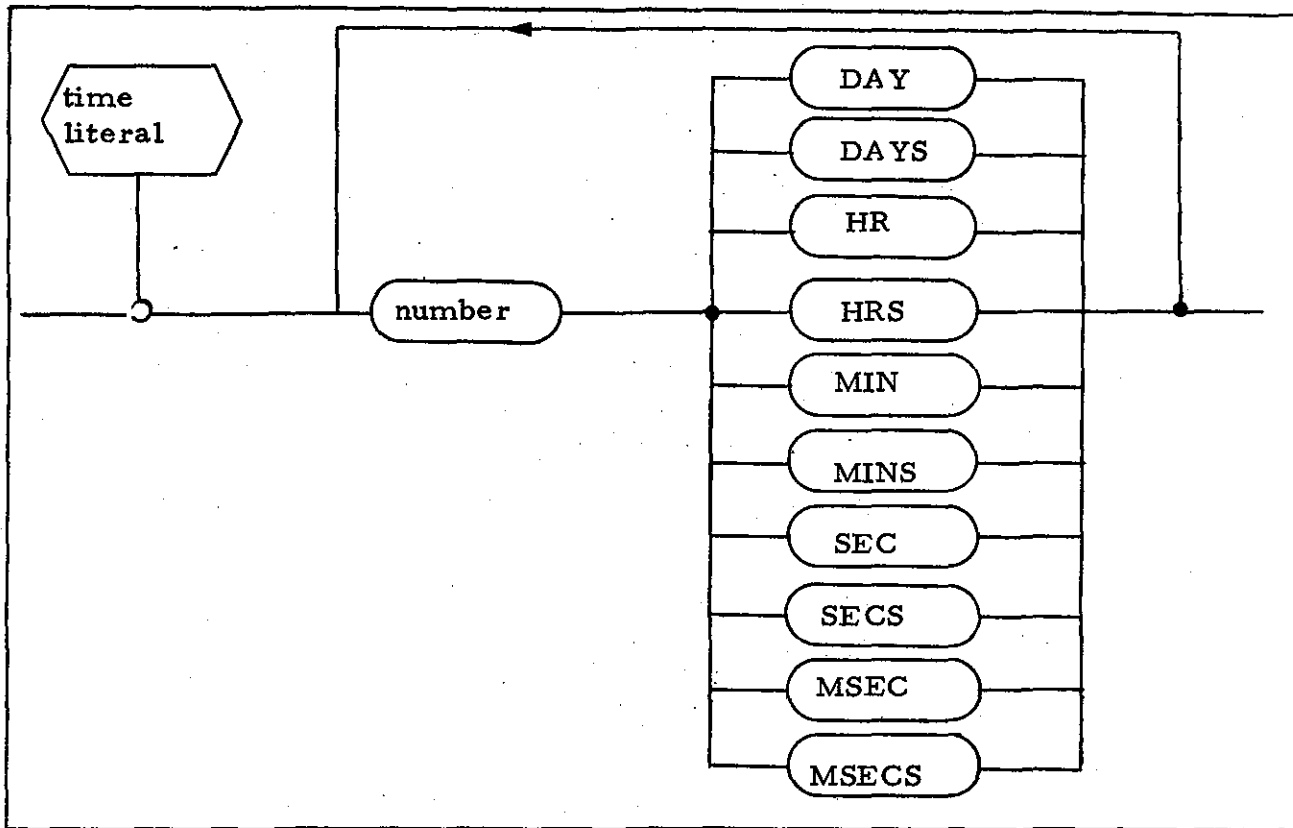
```
DO_IT:    CRITICAL SECTION;  
  
          CALL DO_IT_FAST;  
  
          CLOSE;
```

becomes

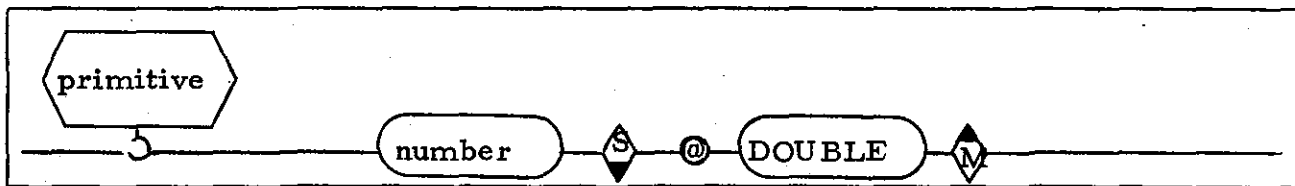
```
DO_IT:    DO;  
  
          G000001: PROCEDURE;  
  
          CALL RTL_GO_CRITICAL;  
  
          CALL DO_IT_FAST;  
  
          CALL RTL_STOP_CRITICAL;  
  
          CLOSE;  
  
          CALL G000001;  
  
END;
```

3.1.38 Time Literals

HAL/SM Syntax



HAL/S Code Template



Note

1. The preprocessor shall convert <time literal> s to a <number> indicating the required time in milliseconds.

Example

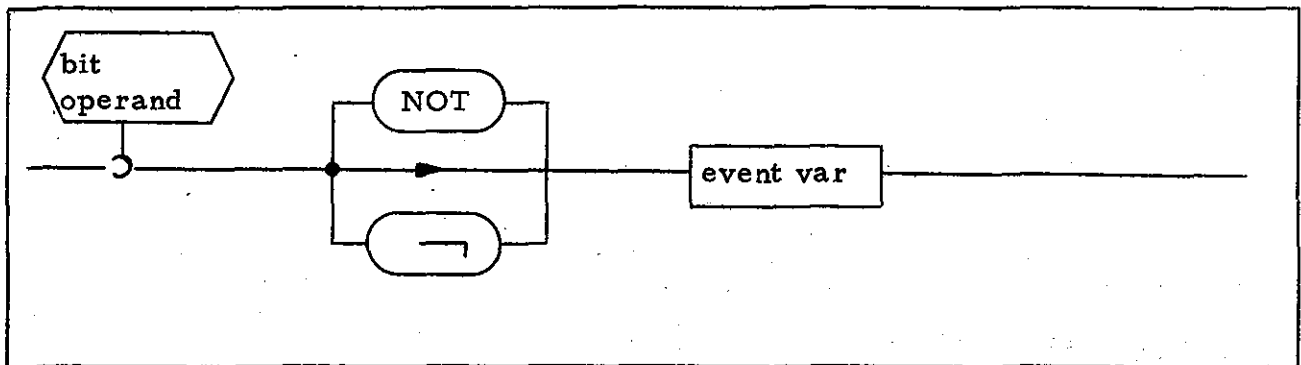
30 SECS

becomes

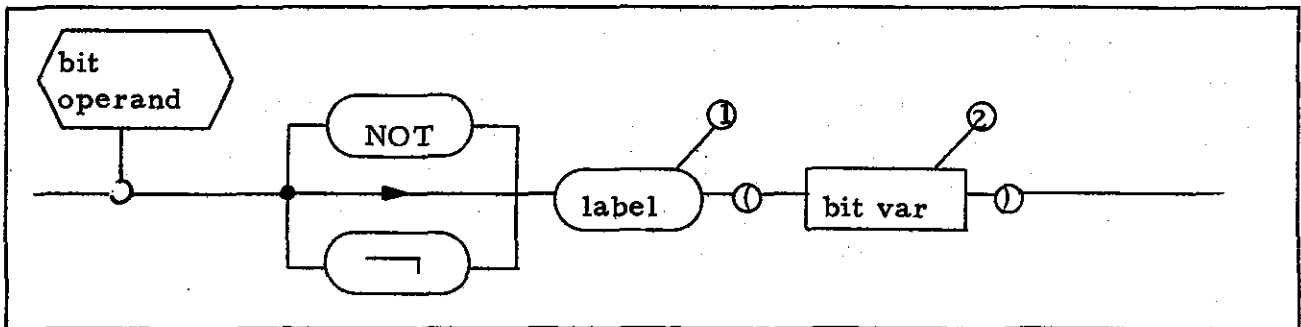
30000@DOUBLE

3.1.39 EVENT Variable

HAL/SM Syntax



HAL/S Code Template

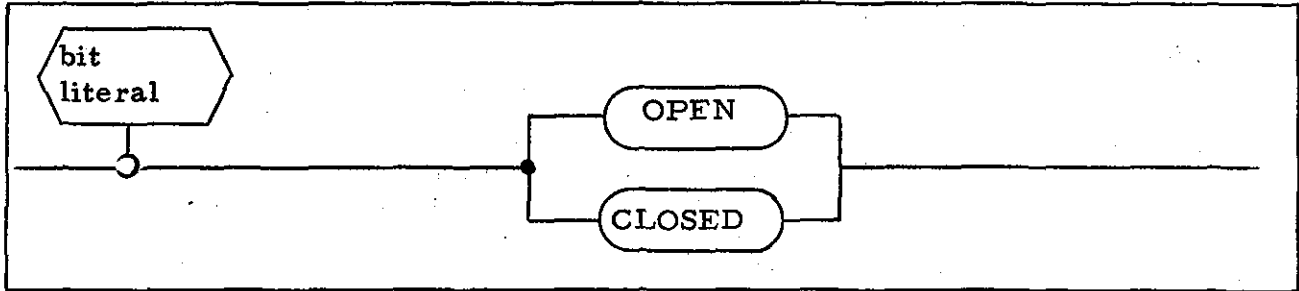


Notes

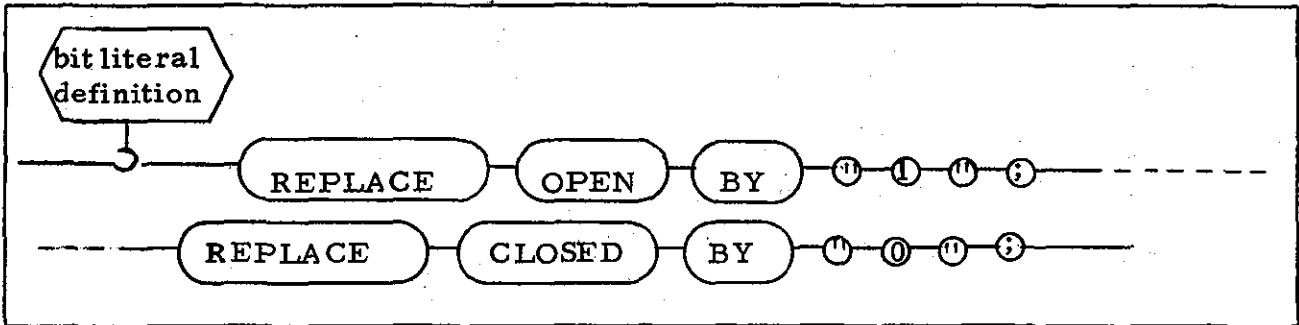
1. This <label> shall be an entry point into the RTL to return as its value the value of the specified HAL/SM <event var>.
2. This <bit var> shall be the HAL/S variable corresponding to the HAL/SM <event var>.

3.1.40 OPEN and CLOSE Literals

HAL/SM Syntax



HAL/S Code Template



Note

1. The <bit literal definition> shall be inserted in a COMPOOL template at the beginning of each HAL/S source module produced by the preprocessor.

3.2 HALLINK

This subsection discusses the differences between the HAL/SM and the HAL/S-360 HALLINK programs. The following modifications shall be made to the HALLINK and HALLKED modules of the HALLINK program:

- o Force the suppression of the HALMAP CSECT in the output load module.
- o Verify that the flight computer timing simulation "cost-use" arrays have been suppressed during compilation.
- o Verify that there are no nested HAL/S TASK's in the output load module.

3.3 RTL

This subsection enumerates the new and modified modules in the HAL/SM RTL. Each numbered paragraph discusses a single RTL interface or functional module and describes its characteristics in terms of calling sequence, parameter lists, standard linkage conventions (see Reference 4), function, and/or MOSS interface. For new modules (i.e., those not in the HAL/S-360 RTL) all of the above mentioned facets are discussed (where they are pertinent). For modified modules only those facets of the module which require modification are discussed.

3.3.1 ON ERROR Interface

The ON ERROR routines shall be additions to the RTL.

Calling Sequence (Form 1)

```
CALL RTL_ON_ERROR1(p1, p2, p3, p4);
```

Parameter List

- p₁: error group number.
- p₂: error number within the group.
- p₃: the error action indicator specified as:
- 0 - SYSTEM, take the standard recovery action.
 - 1 - IGNORE, do not attempt recovery.
- p₄: the flag variable to be signaled on occurrence of the error.

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL ON ERROR (Form 1) routine shall establish the error information specified in the parameter list in this program's error environment stack. The error information shall be used by the error monitor to determine the actions to take when an error occurs.

Calling Sequence (Form 2)

```
CALL RTL_ON_ERROR2(p1, p2, p3);
```

Parameter List

- p₁: error group number.
- p₂: error number within the group.
- p₃: address (label) of the statement to be executed when the error occurs.

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL ON ERROR routine (Form 2) shall establish the error information specified in the parameter list in this program's error environment stack. The error information shall be used by the error monitor to determine the actions to take when an error occurs.

3.3.2 OFF ERROR Interface

The OFF ERROR routine shall be an addition to the RTL.

Calling Sequence

```
CALL RTL_OFF_ERROR(p1, p2);
```

Parameter List

p₁: error group number.

p₂: error number within the group.

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL OFF ERROR routine shall establish the error information specified in the parameter list in the program's error environment stack. The error information shall be used by the error monitor to determine the actions to be taken when an error occurs.

3.3.3 SIGNAL Interface

This RTL routine shall replace the current HAL/S signal routine.

Calling Sequence

CALL RTL_SIGNAL(p_1, p_2, \dots, p_n);

Parameter List

Each parameter in the list shall be the same type. The parameter specification shall be repeated as required.

p_1 : the flag variable to be signaled.

Linkage Convention

NONHAL(1) linkage.

Function

The RTL SIGNAL routine shall set up the parameter list for the MOSS event set interface (see Reference 7, Section 5.3.4) and issue the SVC instruction to invoke that service.

3.3.4 RESET Interface

This RTL routine shall replace the current HAL/S reset routine.

Calling Sequence

```
CALL RTL_RESET(p1, p2, . . . , pn);
```

Parameter List

Each parameter in the list shall be the same type. The parameter specification shall be repeated as required.

p₁: the event variable to be reset.

Linkage Convention

NONHAL(1) linkage.

Function

The RTL RESET routine shall set up the parameter list for the MOSS event delete interface (see Reference 7, Section 5.3.2) and issue the SVC instruction to invoke that service.

3.3.5 CANCEL Interface

This RTL routine shall replace the current HAL/S CANCEL routine. No new syntax has been added to support the CANCEL interface, however, the function has changed.

Calling Sequence

```
CALL RTL_CANCEL(p1);
```

Parameter List

p₁: the task descriptor of the task to be canceled. This parameter shall be optional; if omitted, the requesting task shall be canceled.

Linkage Convention

NONHAL(1) linkage.

Function:

The RTL CANCEL routine shall set up the parameter list for the MOSS task CANCEL interface (see Reference 7, Section 5.2.6) and issue the SVC instruction to invoke that service.

Upon return from MOSS, the return code shall be examined for the two abnormal conditions:

- o the specified task was not in the proper state, and
- o the specified task was not periodic and was already in execution.

HAL/S run time errors shall be generated corresponding to the type of error encountered.

3.3.6 TERMINATE Interface

This RTL interface shall replace the current HAL/S TERMINATE process routines - TERMIN, TERMINT, TERMPCB. The TERMINATE statement shall invoke either the TERMINATE TASK or the TERMINATE JOB routine as described below.

Calling Sequence (TERMINATE TASK)

```
CALL RTL_TTERM;
```

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL TASK TERMINATE routine shall invoke the MOSS TERMINATE interface (see Reference 7, Section 5.2.3) via an SVC instruction.

Calling Sequence (TERMINATE JOB)

```
CALL RTL_JTERM;
```

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL JOB TERMINATE routine shall invoke the MOSS JOB TERMINATE interface (see Reference 7, Section 5.1.3) via an SVC instruction.

3.3.7 ABORT Interface

The ABORT interface shall invoke either the ABORT job or the ABORT task routine described below. These routines are additions to the RTL.

Calling Sequence (ABORT JOB)

CALL RTL_JABORT;

Linkage Convention

NONHAL (1) linkage.

Function

The RTL job ABORT routine shall invoke the MOSS job ABORT interface (see Reference 7, Section 5.1.4) via an SVC instruction.

Calling Sequence (ABORT TASK)

CALL RTL_TABORT (p_1, p_2, \dots, p_n);

Parameter List

Each parameter in the list shall be the same type. The parameter specification shall be repeated as required.

p_1 : the task descriptor of the task to be aborted. This parameter shall be optional; if omitted, the requesting task shall be aborted.

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL task ABORT routine, shall set up the parameter list (if any) for the MOSS task ABORT interface (see Reference 7, Section 5.2.4) and issue the SVC instruction to invoke that service.

3.3.8 LOG Interface

The LOG routine is an addition to the RTL.

Calling Sequence

CALL RTL_LOG (p₁);

Parameter List

p₁: the character expression to be written to the MOSS system log.

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL LOG routine shall set up the parameter list for the MOSS write to log interface (see Reference 7, Section 5.4.4) and issue the SVC instruction to invoke that service.

Upon return to MOSS, the return code shall be examined to determine if the message exceeded 126 bytes. If so, a HAL/S run time error shall be generated within the library.

3.3.9 UNLOCK Interface

The UNLOCK routine is an addition to the RTL.

Calling Sequence

```
CALL RTL_UNLOCK (p1, p2, ..., pn);
```

Parameter List

Each parameter in the list shall be the same type. The parameter specification shall be repeated as required.

p₁: the load module descriptor that identifies the module to be unlocked.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL UNLOCK routine shall set up to the parameter list for the MOSS UNLOCK interface (see Reference 7, Section 5.4.2) and issue the SVC instruction to invoke that service.

3.3.10 LOAD Interface

The LOAD routine is an addition to the RTL.

Calling Sequence

CALL RTL_LOAD (p₁);

Parameter List

p₁: the job descriptor of the job to be loaded.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL job LOAD routine shall set up the parameter list for the MOSS job LOAD interface (see Reference 7, Section 5.1.1) and issue the SVC instruction to invoke that service.

Upon return from MOSS, the return code shall be examined to determine if either of the following abnormal conditions occurred:

- o the job descriptor was invalid, or
- o the job was not in the proper state for loading.

If either of these conditions is present, the appropriate run time error shall be generated within the library.

3.3.11 INITIATE Interface

The INITIATE routine is an addition to the RTL.

Calling Sequence

CALL RTL_INIT (p₁);

Parameter List

p₁: the job descriptor of the job to be initiated.

Linkage Convention

NONHAL, (1) linkage.

Function

The RTL job INITIATE routine shall set up the parameter list for the MOSS job INITIATE interface (see Reference 7, Section 5.1.2) and issue the SVC instruction to invoke that service.

Upon return from MOSS, the return code shall be examined to determine if the following conditions occurred:

- o the specified job was not in External Paging Memory (EPM), or
- o the specified job was not in the proper state for job initiation.

If either of these conditions is present, the appropriate run time error shall be generated within the library.

3.3.12 DELETE Interface

The DELETE interface shall invoke either the DELETE job or DELETE task routines described below. These routines are additions to the RTL.

Calling Sequence (JOB DELETE)

CALL RTL_JDEL;

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL job DELETE routine shall invoke the MOSS job DELETE interface (see Reference 7, Section 5.1.5) via an SVC instruction.

Calling Sequence (TASK DELETE)

CALL RTL_TDEL;

Linkage Convention

Normal HAL/S-360 Linkage.

Function

The RTL task DELETE routine shall invoke the MOSS task DELETE interface (see Reference 7, Section 5.2.5) via an SVC instruction.

3.3.13 SCHEDULE Interface

This RTL routine shall replace the current HAL/S SCHEDULE routine.

Calling Sequence

CALL RTL_SCHEDULE (p₁, p₂, p₃, p₄, p₅, p₆, p₇, p₈, p₉);

Parameter List

p₁: a bit vector indicating the presence and format of the optional parameters p₃-p₉. The format of the bit vector and the meaning of a "one" in the corresponding position in the vector are:

<u>Bit Number</u>	<u>Meaning</u>
0	p ₃ , the schedule parameter is present.
1, 2	A 2-bit value indicating the presence and format of p ₄ , the precondition time delay as follows: 0 - not present 1 - millisecond time interval delay 2 - GMT time value delay 3 - MET time value delay
3	p ₅ , p ₆ , p ₇ ; the event expression parameters are present.
4	p ₈ , the time interval delay after events are considered parameter is present.
5	p ₉ , the repetition time delay for cyclic tasks is present.

p₂: the task descriptor of the task being scheduled.

p₃: the character schedule parameter to be passed to the task being scheduled.

p₄: the time delay to be applied before considering any of the other conditional scheduling parameters. The time delay value may be a millisecond time interval, a GMT time value, or a MET time value as indicated by p₁.

- p_5 : an array of event variables whose logical combination, as indicated by p_6 , must be satisfied before the task is scheduled.
- p_6 : an array of logical operators defining the logical expression of event variables as given in p_5 . The event expression shall be represented in Reverse Polish form.
- p_7 : the number of event variables in the array given in p_5 .
- p_8 : the millisecond time value which must expire after the event expression is satisfied before the task is scheduled.
- p_9 : the millisecond time period between successive scheduling for a periodic task.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL SCHEDULE routine shall set up the parameter list as received in the format required by the MOSS task SCHEDULE interface (see Reference 7, Section 5.2.1) and issue the SVC instruction to invoke that service.

Upon return from MOSS, the return code shall be examined to determine if the parameter data was too large for the receiving task. If so, a HAL/S run time error shall be generated within the library.

3.3.14 WAIT Interface

The WAIT routine shall replace the current HAL/S WAIT routine - WAIT, WAITDEP, and WAITFOR.

Calling Sequence

CALL RTL_WAIT (p₁, p₂, p₃, p₄, p₅, p₆)

Parameter List

p₁: a bit variable indicating the presence and format of the optional parameters p₂-p₆. The meaning of each bit position is given below:

<u>Bit Number</u>	<u>Meaning</u>
0, 1	A 2-bit value indicating the presence and format of the precondition time delay. 0 - time delay not given 1 - time delay given for milliseconds time interval 2 - time delay given for GMT time value 3 - time delay given for MET time value
2	An event expression (p ₃ , p ₄ , and p ₅) is present.
3	A post-condition time interval delay in milliseconds is present.

p₂: the time value to delay before considering any other WAIT conditions. The format of this value shall be indicated by p₁.

p₃: an array of event variables whose logical combination, as indicated by p₄, must be satisfied before the task continues.

p₄: an array of logical operators defining the logical expression of event variables given in p₃. The event expression shall be represented in Reverse Polish form.

p₅: the number of event variables in the array given in p₃.

p_6 : the millisecond time value which must expire after the event expression is satisfied before the task is allowed to continue.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL WAIT routine shall set up the parameter list for the MOSS task SUSPEND interface (see Reference 7, Section 5.2.2) and issue the SVC instruction to invoke that service.

3.3.15 ALERT Interface

The ALERT routine is an addition to the RTL.

Calling Sequence

CALL RTL_ALERT (p₁, p₂, p₃, p₄, p₅, p₆)

Parameter List

- p₁: the bit variable corresponding to the HAL/SM event variable to be ALERTed.
- p₂: a bit vector indicating presence and format of the optional parameters p₃-p₅. The meaning of each bit position is given below:

Bit Number

Meaning

0, 1, 2

A 3-bit value indicating the presence and format of p₄, the event descriptor.

0 - event is task termination; p₃ is present.

1 - event is a program flag; p₃ is present.

2 - event is an ERP condition; p₃ and p₄ are present.

4 - event is a MET clock condition; p₃ is absent, p₄ and p₅ are present.

5 - event is GMT clock condition; p₃ is absent, p₄ and p₅ are present.

- p₃: the event descriptor specified as a task descriptor, flag variable, or an ERP descriptor.
- p₄: a number giving the relational operators for ERP and clock events. The values of this parameter are given in Reference 7, Section 5.3.1. When this parameter indicates an "is between" or "is not between" relation, p₆ is present.
- p₅: the value to which the ERP or clock specified is to be compared, as indicated by the relation given in p₄.
- p₆: the second value to be used in the comparisons "is (not) between," as indicated by p₄.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL ALERT routine shall set up the parameter list for the MOSS event ALERT interface (see Reference 7, Section 5.3.1) and issue the SVC instruction to invoke that service.

3.3.16 AVERAGE Interface

The AVERAGE AI routine is an addition to the RTL.

Calling Sequence

CALL RTL_AVERAGE (p₁, p₂, p₃)

Parameter List

- p₁: the number of readings to be averaged.
- p₂: the ERP designator to be averaged.
- p₃: the variable in which the result is to be stored.

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL AVERAGE AI routine shall set up the parameter list for the MOSS read ERP interface (see Reference 7, Section 5.6.2). The parameter list shall indicate the read AI AVERAGED option and the address of the RTL's I/O error processing routine.

3.3.17 READ ERP Interface

The READ ERP routine is an addition to the RTL.

Calling Sequence

CALL RTL_SENSE ($P_1, P_2, P_3, \dots, P_n, P_{n+1}, \dots, P_m$)

Parameter List

- P_1 : a number whose value indicates the presence of the READ AI deltas option as follows:
- 0 - AI deltas option not requested.
 - 1 - AI deltas option requested.
- P_2 : a number whose value indicates the number of ERP's to be read.
- $P_3 - P_n$: each of these parameters shall be an ERP designator to be read. All of the ERP's given must be of the same type.
- $P_{n+1} - P_m$: each of these parameters shall be a variable in which the results of the READ operation for each of the ERP designators in $p_2 - p_n$ is to be placed. The number of parameters in this group shall be the same as in the group $p_3 - p_n$.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL READ ERP routine shall set up the parameter list for the MOSS READ ERP interface (see Reference 7, Section 5.6.2) and invoke that service via an SVC. The parameter list shall be paired for each ERP designator and corresponding variable data area. The address of the RTL's I/O error processing routine shall be included as a parameter for each ERP designator.

3.3.18 ISSUE Interface

The ISSUE routine is an addition to the RTL.

Calling Sequence

CALL RTL_ISSUE (p_1, p_2, \dots, p_n)

Parameter List

p_1 : the RO ERP designator to which the issue applies.

p_2-p_n : each of these parameters shall be the same type - either

- o variables of the same type.
- o bit literals, or
- o numbers.

The collection of parameters p_2-p_n shall form the data to be issued to the RO.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL ISSUE routine shall collect the data in each of the input parameters p_2-p_n and form a contiguous data area. The address of this data area, its length, the ERP designator, and the address of the RTL's I/O error processing routine shall be passed in a parameter list to the MOSS WRITE ERP interface (see Reference 7, Section 5.6.3) via an SVC instruction.

3.3.19 SET DISCRETE Interface

The SET DISCRETE routine is an addition to the RTL.

Calling Sequence

CALL RTL_SETD ($p_1, \dots, p_n, p_{n+1}, p_{n+2}, \dots, p_m, p_{m+1}$)

Parameter List

- p_1 : this parameter shall specify the number of DO ERP descriptors to follow.
- p_2 - p_n : each parameter shall be of the same type, DO ERP descriptor, and shall indicate the discretets to be set.
- p_{n+1} : this parameter shall specify the number of value parameters to follow.
- p_{n+2} - p_m : each parameter shall be of the same type, either:
- o variable of the same type,
 - o bit literals, or
 - o numbers.
- Each parameter gives the value to set to the corresponding DO defined in p_1 - p_n .
- p_{m+1} : the time value in milliseconds that the DO is to be pulsed. This parameter is only allowable when p_2 and p_{n+2} specify a single DO to be pulsed.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL SET DISCRETE routine shall pair each DO ERP designator and its corresponding value in a parameter list for the MOSS WRITE ERP interface (see Reference 7, Section 5.6.3). If a single DO to be pulsed is specified, the parameter list shall include the time value parameter. The address of the RTL's I/O error processing routine shall be included as a parameter for all requests. The WRITE ERP service shall be invoked via an SVC instruction when the parameter list is complete.

3.3.20 APPLY ANALOG Interface

The APPLY ANALOG routine is an addition to the RTL. Multiple forms of the calling sequence and parameter list are shown to simplify the explanations due to special cases which may be generated.

Calling Sequence (APPLY ANALOG - DELTAS)

CALL RTL_APPLY (p₁, p₂, p₃, p₄)

Parameter List

p₁: an indicator which specifies the special option selected.

1 - DELTAS

p₂-p₃: the two values to be applied as deltas.

p₄: the AI ERP designator to which the delta values given in p₂ and p₃ are to be applied.

Calling Sequence (APPLY ANALOG - PULSED)

CALL RTL_APPLY (p₁, p₂, p₃, p₄)

Parameter List

p₁: special option indicator:

2 - PULSED

p₂: the value to be applied to the AO.

p₃: the AO ERP designator to be pulsed.

p₄: the time value indicating the length of the pulse.

Calling Sequence (APPLY ANALOG - RAMPED)

CALL RTL_APPLY (p₁, p₂, p₃, p₄, p₅)

Parameter List

p₁: special option indicator:

3 - RAMPED

- P₂: the initial value to be applied to the AO.
- P₃: the AO ERP designator to be ramped.
- P₄: the time value for each increment of the ramp.
- P₅: the maximum (or minimum for negative ramping) to which the AO is to be ramped.

Calling Sequence (APPLY ANALOG - No Special Options)

CALL RTL_APPLY (P₁, P₂, P₃, ..., P_n, P_{n+1}, ..., P_m)

Parameter List

- P₁: special option indicator:
0 - no special options
- P₂: the number of multiple ERP designators being supplied.
- P₃-P_n: each parameter of this group shall be the value to be applied to the AO's specified in parameters P_{n+1}-P_m.
- P_{n+1}-P_m: each parameter of this group shall be an AO ERP designator to be applied.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL APPLY ANALOG routine shall set the parameter list as indicated for each of the cases given above for the MOSS WRITE ERP interface (see Reference 7, Section 5.6.3). The address of the RTL's I/O error processing routine shall be included in the parameter list. The WRITE ERP interface shall be invoked via an SVC instruction.

3.3.21 WRITE TO OPERATOR Interface

The WRITE TO OPERATOR routine is an addition to the RTL.

Calling Sequence

CALL RTL_WTO (P₁, P₂, P₃)

Parameter List

- P₁: a bit variable indicating whether the reply option has been selected:
- 0 - no reply, P₃ is present.
 - 1 - reply expected, P₃ is present.
- P₂: the character expression to be displayed to the operator.
- P₃: the character variable in which any reply is to be stored.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL WRITE TO OPERATOR routine shall examine P₁ to determine if the reply option has been selected. If no, P₂ and its length shall be passed to the MOSS WRITE TO OPERATOR interface (see Reference 7, Section 5.4.5) via an SVC instruction. If the reply option has been selected, P₂, its length, and P₃ shall be passed to the MOSS WRITE TO OPERATOR with REPLY interface (see Reference 7, Section 5.4.6).

A run time error shall be generated if the output message exceeds 100 bytes for either case.

3.3.22 DISPLAY CONTROL Interface

The DISPLAY CONTROL routine is an addition to the RTL.

Calling Sequence

CALL RTL_DISCON (p₁, p₂, p₃, p₄)

Parameter List

- p₁: the control function indicator whose value shall be interpreted as follows:
- 0 - page select
 - 1 - clear page
 - 2 - video 1
 - 3 - video 1/stroke
 - 4 - video 2
 - 5 - video 2/stroke
 - 6 - allocate page
 - 7 - deallocate page
- p₂: a bit vector whose value shall indicate the presence of p₃ and p₄, the CRT, and page number.
- 0 - p₃ and p₄ are absent
 - 1 - p₃ and p₄ are present
- p₃: the CRT number to which the control function shall be directed.
- p₄: the page number to which the control function shall be directed.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL DISPLAY CONTROL routine shall examine the control function indicator, p₁. If the allocate or deallocate page functions have been selected,

the MOSS allocate or deallocate console page interface shall be invoked with the CRT and page number parameters. To perform the other functions, MOSS DISPLAY CONTROL interfaces shall be invoke the appropriate parameters.

3.3.23 DISPLAY DATA Interface

The DISPLAY DATA interface shall invoke either the DISPLAY BACKGROUND, DISPLAY ADDON, UPDATE BACKGROUND, DISPLAY TYPE1 PROGRAM, or UPDATE TYPE1 DATA routines described below. These routines are additions to the RTL.

Calling Sequence (DISPLAY BACKGROUND)

CALL RTL_DISBACKD (P₁, P₂, P₃, P₄, P₅, P₆)

Parameter List

- P₁: the program identifier by which the data set containing the background message is identified.
- P₂: the message identification number which specifies the record number of the message within the data set.
- P₃: the blink status option specification.
- P₄: the bit indicator which indicates the CRT number and page number to follow.
- P₅: the CRT number (if specified).
- P₆: the page number (if specified).

Linkage Convention

NONHAL (1) linkage.

Function

The RTL DISPLAY BACKGROUND routine shall set up the parameter list for the MOSS DISPLAY BACKGROUND (from disk) interface and issue the DISBACKD SVC. (This SVC obtains the indicated message from the C&D message library and transmits it to the indicated C&D Console page.)

An I/O ERROR EXIT routine shall be provided in the RTL routine and the entry point to this code shall be provided in the SVC parameter list.

Calling Sequence (DISPLAY ADDON)

CALL RTL_DISADD (p₁, p₂, p₃, p₄, p₅, ..., p_i, p_j, p_k, p_m);

Parameter List

- p₁: the program identifier by which the data set containing the ADDON message is identified.
- p₂: the message identification number which specifies the record number of the message within the data set.
- p₃: the blink status option specification.
- p₄: the numbers of pairs, each containing a variable control word and an associated variable used to describe the ADDON variable data.
- p₅-p_i: (where $i = 4 + 2 * p_4$): the pairs of variable control word with associated variable described above. The sequence is: variable control word, variable, variable control word, variable, ...
- p_j: (where $j = i + 1$): the bit indicator which indicates that a CRT number and page number follow.
- p_k: (where $k = i + 2$): the CRT number (if specified).
- p_m: (where $m = i + 3$): the page number (if specified).

Linkage Convention

NONHAL (1) linkage.

Function

The RTL DISPLAY ADDON routine shall convert any variable data to ASCII format, set up the parameter list for the MOSS DISPLAY ADDON (from disk) interface, and issue the DISADD SVC. (This SVC obtains the indicated ADDON message from the C&D message library, updates any variable fields specified, and transmits it to the indicated C&D Console page.)

An I/O ERROR EXIT routine shall be provided in this RTL routine and an entry point to this code shall be provided in the SVC parameter list.

Calling Sequence (UPDATE BACKGROUND)

CALL RTL_UPDISM (p₁, p₂, ..., p_i, p_j, p_k, p_m, p_n)

Parameter List

- p₁: the number of pairs, each containing a variable control word and an associated variable to be updated.
- p₂-p_i: (where i = 2+2*p₂): the pairs of variable control word and associated variable described above. The sequence shall be variable control word, variable, variable control word, variable, ...
- p_j: (where j = i+1): the blink status option specification.
- p_k: (where k = i+2): the bit indicator which indicates that a CRT number and page number follow.
- p_m: (where m = i+3): the CRT number (if specified).
- p_n: (where n = i+4): the page number (if specified).

Linkage Convention

NONHAL (1) linkage.

Function

The RTL UPDATE BACKGROUND routine shall construct the update text from the variable control words and variables, convert the text to ASCII, set up the parameter list for the MOSS UPDATE BACKGROUND interface, and issue the UPDISM SVC. (This SVC displays the supplied text to the indicated C&D Console page as an UPDATE message.)

An I/O ERROR EXIT routine shall be provided in this RTL routine and an entry point to this code shall be provided in the SVC parameter list.

Calling Sequence (DISPLAY TYPE1 PROGRAM)

CALL RTL_T1TRANS (p₁, p₂, p₃, p₄)

Parameter List

- p₁: the TYPE1 PROGRAM identified in the TYPE1 PROGRAM library.

- p_2 : the bit indicator which indicates that a CRT number and page number follow.
- p_3 : the CRT number (if specified).
- p_4 : the page number (if specified).

Linkage Convention

NONHAL (1) linkage.

Function

The RTL DISPLAY TYPE1 PROGRAM routine shall set up the parameter list for the MOSS TYPE1 PROGRAM transmission interface and issue the T1 TRANS SVC. (This SVC obtains the indicated TYPE1 PROGRAM from the TYPE1 PROGRAM library and transmits it to the indicated C&D Console and page.)

An I/O ERROR EXIT routine shall be provided in this RTL routine and an entry point to this code shall be provided in the SVC parameter list.

Calling Sequence (UPDATE TYPE1 DATA)

CALL RTL_T1DATA ($p_1, p_2, p_3, \dots, p_i, p_j, p_k, p_m$);

Parameter List

- p_1 : the TYPE1 PROGRAM identifier.
- p_2 : the number of TYPE1 UPDATE variables.
- p_3-p_i : (where $i = 3+p_2$): the list of variable data.
- p_j : (where $j = i+1$): the bit indicator which indicates that a CRT number and page number follow.
- p_k : (where $k = i+2$): the CRT number (if specified).
- p_m : (where $m = i+3$): the page number (if specified).

Linkage Convention

NONHAL (1) linkage.

Function

The RTL UPDATE TYPE1 DATA routine shall set up the parameter list for the MOSS TYPE1 DATA transmission interface and issue the T1DATA SVC. (This SVC transmits the specified TYPE1 UPDATE data to the indicated C&D Console page.)

An I/O ERROR EXIT routine shall be provided in this RTL routine and an entry point to this code shall be provided in the SVC parameter list.

3.3.24 REQUEST KEYBOARD Interface

The REQUEST KEYBOARD routine is an addition to the RTL.

Calling Sequence

CALL RTL_RQST_KYBD (p₁, p₂)

Parameter List

- p₁: the character variable into which the KEYBOARD message is to be stored.
- p₂: a bit variable to be set if a message was returned. If this parameter is absent, the requesting task shall be delayed until a KEYBOARD message is available.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL REQUEST KEYBOARD routine shall invoke the MOSS REQUEST KEYBOARD interface with the character data area address and an indicator for automatic wait, if specified, as parameters. If the post-variable p₂ is specified, the routine shall set this bit variable as indicated by the return code from MOSS.

The address of the RTL's I/O error processing routine shall also be included as a parameter to the MOSS interface.

3.3.25 SELECT Interface

The SELECT routine is an addition to the RTL.

Calling Sequence

CALL RTL_SELECT (p₁, p₂, ..., p_n);

Parameter List

The parameter list shall be formed of repeated pairs of parameters of the formats given below:

- p₁: the resource descriptor to be selected. The descriptor shall be one of the following formats:
- o Data module descriptor,
 - o Program module descriptor, or
 - o SPIO I/O channel or file number.
- p₂: a bit vector indicating the access rights requested for the resource indicated in p₁. The format of the bit vector shall be the same as the MOSS SELECT interface (see Reference 7, Section 5.5.1).

Linkage Convention

NONHAL (1) linkage.

Function

The RTL SELECT routine shall set up the parameter list for the MOSS SELECT interface and issue the SVC instruction to invoke that service.

3.3.26 RELEASE Interface

The RELEASE routine is an addition to the RTL.

Calling Sequence

CALL RTL_RLSE (p₁, p₂, ..., p_n);

Parameter List

Each parameter in the list shall be of the same format.

p₁: the resource descriptor of the resource to be released.
The descriptor shall be in one of the following formats:

- o Data module descriptor,
- o Program module descriptor, or
- o SPIO I/O channel number.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL RELEASE routine shall set up the parameter list for the MOSS RELEASE interface and issue the SVC instruction to invoke that service.

3.3.27 CHANNEL CONTROL Interface

This interface is an addition to the RTL.

Calling Sequence

CALL RTL_CHAN (p₁, p₂, p₃);

Parameter List

- p₁: the channel number of the SPIOS I/O channel being manipulated.
- p₂: the function indicator which selects the function to be performed as follows:
- 1 - backspace,
 - 2 - space,
 - 3 - rewind,
 - 4 - unload,
 - 5 - end file, and
 - 6 - close file.
- p₃: the value giving the number of files to be backspaced or forward spaced. This parameter shall only be provided for the space and backspace options.

Linkage Convention

NONHAL (1) linkage.

Function

The RTL CHANNEL CONTROL routine shall set up the parameter list for the MOSS CONTROL interface (see Reference 7, Section 5.7.4), and issue the SVC instruction to invoke that service. The parameter list shall include the address of the RTL's I/O error processing routine.

3.3.28 CRITICAL SECTION Interface

The CRITICAL SECTION interface shall consist of two routines, one executed at the beginning of the critical block, the other at the end of the critical block. The CRITICAL SECTION routines are additions to the RTL.

Calling Sequence

CALL RTL_GO_CRITICAL

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL BEGIN CRITICAL SECTION routine shall invoke the MOSS CRITICAL SECTION interface (see Reference 7, Section 5.4.7) with the enter critical processing mode parameter.

Calling Sequence (END CRITICAL)

CALL RTL_STOP_CRITICAL

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL END CRITICAL SECTION routine shall invoke the MOSS CRITICAL SECTION interface with the exit critical processing mode parameter.

3.3.29 EVENT VARIABLE Interface

The EVENT VARIABLE interface shall be an RTL function routine which shall return the value of the event variable. This routine is an addition to the RTL.

Calling Sequence

RTL_TEST (p₁);

Parameter List

p₁: the event variable to be tested.

Linkage Convention

Normal HAL/S-360 linkage.

Function

The EVENT VARIABLE function routine shall examine the event variable value to determine if it is zero or an MOSS provided event variable descriptor. If the value is zero, the function routine shall return a zero value. If the value is an event variable descriptor, the descriptor shall be passed to the MOSS EVENT TEST interface (see Reference 7, Section 5.3.3) via an SVC. The return code provided by MOSS indicates the value of the event variable which shall be returned by the FUNCTION routine.

3.3.30 TIME Interface

This RTL routine shall replace the current HAL/S clock time routine.

Calling Sequence

TIME (p₁);

Parameter List

p₁: an integer indicating the type of time value desired.
The permissible values are:

0 - MET time.

1 - GMT time.

Return Value: a double precision floating point number equal to the value of the specified clock time in milliseconds.

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL TIME FUNCTION shall set up the parameter list in the format required by the MOSS TIME REQUEST interface (see Reference 7, Section 5.4.1), and issue the SVC to invoke that service. The returned time shall then be converted to the above specified form and returned to the caller.

3.3.31 DATE Interface

This RTL routine shall replace the current HAL/S DATE routine.

Calling Sequence

DATE

Parameter List

Return Value: an integer equal to the value of the "days" fields of the GMT clock.

Linkage Convention

Normal HAL/S-360 linkage.

Function

The RTL DATE function shall set up the parameter list in the format required by the MOSS TIME REQUEST interface (see Reference 7, Section 5.4.1), and issue the SVC to invoke that service. The required field shall then be extracted from the returned value, converted to the above specified form, and returned to the caller.

3.3.32 SPIOs Interfaces

The I/O processing routines and macros in the HAL/S library shall be modified to utilize MOSS SPIOs services. These I/O routines currently use the OS/360 I/O services. All format conversion, line, and page manipulation logic shall remain relatively unchanged after the modification.

The following routines and macros shall be modified:

<u>Routines</u>	<u>Macros</u>
COLUMN	FCBDEF
FILEIN	FILEDEF
INPUT	HXCTL
IOINIT	
LINE	
OUTPUT	
PAGE	
SKIP	

The major impact of the modification shall consist of replacing the OS/360 I/O service requests with equivalent MOSS I/O service requests. The SUMC-S MOSS Application Program Design Specification (Reference 7, Section 5.7.1) describes the MOSS I/O service requests which shall be used. These include:

- o READ,
- o WRITE,
- o CONTROL, and
- o CLOSE.

The modified routines shall use the MOSS I/O ERROR EXIT routine conventions to handle I/O errors. Uncorrectable errors shall be communicated to the HAL/S error monitor to coordinate with the user defined error environment.

3.3.33 Interrupt and Error Handling Modules

The modules PROGINT, ERRORMON, and ERRGRP are involved in the processing of program interrupts and all HAL/SM errors. These modules shall be modified to operate within the MOSS environment.

The major changes to these modules shall be in how errors are recognized (not in how they are processed). A forced-end routine (see Reference 7, Section 6) shall replace the HAL/S-360 STAE routine, and a program exception routine shall replace the HAL/S-360 SPIE routine. I/O errors shall be recognized by the I/O modules via I/O ERROR EXIT routines specified in the individual I/O SVC's and the specific errors shall be issued to ERRORMON through the normal and error intrinsic in HALSYS.

In addition, all modules which detect and send errors may require change to renumber the errors they send in their invocations of ERRORMON.

3.3.34 Miscellaneous Modules

The module HALSTART is concerned with initiation and normal termination of a HAL/S-360 program complex. This module shall be modified to delete certain run time JCL options which are not supported and to use MOSS interfaces to properly release resources (e.g., close any open data sets).

The HALSYS module and macro contain common status and linkage information which is required by the RTL and compiler emitted code. This module and macro shall be modified to reflect the reorganization of data and modules within the library resulting from other specific changes.

3.3.35 LOCK Interface

The current HAL/S LOCK routine shall be modified to operate in the MOSS multi-tasking environment. The current HAL/S library only simulates a multi-tasking environment for the user programs, thus, conflicts due to multiple simultaneous executions of the LOCK routine in support of HAL/S update blocks cannot occur. In the MOSS environment, however, the HAL/SM library is operating in a true multi-tasking mode and it must coordinate multiple use of the LOCK routine.

The MOSS SELECT and RELEASE interfaces (see Reference 7, Section 5.5) shall be used by the LOCK routine to ensure mutually exclusive accesses to the lock group variable. The function of the LOCKGRP variable for coordinating accesses to individual data items by the update blocks shall remain unchanged. The MOSS EVENT CONTROL interfaces (see Reference 7, Section 5.3), shall be used to suspend and signal the LOCK routine (in support of a specific task) when access to individual data items, as indicated by the LOCKGRP variable, are locked or freed.

3.3.36 RTL Modules to be Deleted

The following HAL/S-360 RTL modules and macros shall be deleted from the HAL/SM RTL either because their functions are not supported within the HAL/SM system or because they have been replaced by equivalent modules or MOSS services.

Modules:

BAKTRACE
CANCEL
CLOKTIME
CLOSEHAL
DATE
DISPATCH
DUMPHAL
ENTERTQE
EVENTENQ
EXCLUDE
EXECTRCE
FORMATDA
HALSIM
SCHEDULE
SDL DUMMY
SDLSTACK
SET
SIGNAL
SVBLOCK
SVBTOC
SVDTOC
SVETOC
SVITOC
SVPMSG
SVSIGNAL
SVSTOP
SVTDEQ
SVTENQ
SVTIME
SVVSTP
TERMIN
TERMINT
TERMPCB

Macros:

COUNT
EVXQDEF
HALSIM
PCBDEF
SDLCALL
STPGEN
TIMQDEF
TRACEX

Modules:

TIMECANC
TIMEINT
TIMENQ
UPPRIO
WAIT
WAITDEF
WAITFOR
WHERE

3.4 User

This subsection discusses the user interface to the HAL/SM system in terms of OS and MOSS JCL general requirements. The specific definition of the complete user interface shall be defined in a "HAL/SM User's Manual" which shall be published by the implementors.

3.4.1 OS JCL Procedures

Cataloged procedures shall be provided to perform HAL/SM pre-processing, compilation, and linking in a manner similar in form and intent to the HAL/S prototype cataloged procedures (see Reference 5) HALSC, HALSCL, and HALSL. When HAL/SM modules have been properly compiled and linked they must be put in a form suitable for processing by the MOSS Linker. A utility and procedures for use shall be supplied with the MOSS system. Standard IBM utility programs may be used to place the C&D Display Message Data Set on a medium suitable for transfer to the SUMC-S.

3.4.2 MOSS JCL

MOSS jobs containing HAL/SM software shall follow all the normal conventions for MOSS JCL (see Reference 7, Section 3). In addition, certain conventions must be followed in specifying the following items:

- o Job names.
- o Task names.
- o Program flag names.
- o Critical sections.
- o Forced-end routines.
- o Error processing routines.
- o File names.

4. RESTRICTIONS AND LIMITATIONS

Any programming system has certain characteristics which are specific to its implementation. In the particular case of HAL/SM, the decision to adapt the HAL/S-360 compiler system using a preprocessor and a modified run time library shall have an impact on the usability of the system which needs to be clearly recognized. This section briefly discusses some of the effects which are apparent at this time.

4.1 HAL/S Dependencies

The ground rules of this implementation require that the HAL/S-360 compiler remain unmodified. This has both advantages and disadvantages. On the positive side this means that any upgrades to the compiler which would improve its efficiency, accuracy, or correctness can easily be incorporated in the HAL/SM system. On the other hand, these same upgrades may change the syntax and/or semantics of the HAL/S language and, thereby, make it unusable within the HAL/SM system. The modifications made to HALLINK and the RTL may also make it impractical to utilize future upgrades to these programs as well.

4.2 Separation of Host and Target Machines

The host system for the HAL/SM language processor system shall be an IBM S-360/370 series machine running under some version of OS/MVT, as mentioned previously. The target system shall be the SUMC-S running under MOSS. These two systems are quite different in concept and, in addition, they shall most likely be separated physically and managerially. This means that the HAL/SM programmer must be familiar with two different systems and sets of operating procedures before he can use the HAL/SM language. He must also deal with the cumbersome and time consuming problem of transferring his program between the two systems for every compile and test sequence during program development. One possible remedy to this problem might be a software emulator for MOSS running on the S/360.

4.3 Diagnostic Capabilities

The ground rules of this implementation require that no processing be performed in the preprocessor which can be performed by the compiler. This implies that the preprocessor shall not perform a complete syntax analysis. The major drawback to this approach is that only a subset of the syntax errors will be diagnosed by the preprocessor and related directly to the HAL/SM program. All remaining syntax errors are caught by the compiler which

can only state the diagnostic information in terms of the transformed HAL/S text. The programmer is forced to examine listings from both language processors, be familiar with both languages, and be able to relate errors in the HAL/S program to the corresponding HAL/SM construct and determine where the error lies there. Since some HAL/SM constructs require considerable manipulation, and some transformations are performed independent of context, the required correction may not always be readily apparent. A possible solution might be to require the preprocessor to perform a complete syntax analysis and verification.

4.4 RTL Reentrancy

The HAL/S-360 RTL is presently not reentrant. The disadvantage here is that each task must have a private copy for its own use. Under MOSS, a reentrant RTL might be placed in the job (or system) common area of the address space and shared by all HAL/SM tasks within the job (or system) at a considerable savings in memory requirements and execution time (due to reduced paging). However, making the RTL reentrant would probably require a major (if not complete) rewrite of this software.

REFERENCES

1. "HAL/SM Language Specification," M&S Computing, Inc., NASA/MSFC Contract NAS8-26990, Report No. 75-0043, November 21, 1975.
2. "HAL/S Language Specification," Intermetrics, Inc., NASA/JSC Contract NAS9-13864, Document No. IR-61-5, November 22, 1974.
3. "HAL/S-360 Compiler System Functional Specification," Intermetrics, Inc., Rockwell International Corporation, purchase order #M3V8XMX-483000, PDRL#IM004, July 13, 1974.
4. "HAL/S-360 Compiler System Specification," Intermetrics, Inc., NASA/JSC Contract NAS9-13864, Document No. IR 60-3, January 10, 1975.
5. "HAL/S-360 User's Manual, Intermetrics, Inc., NASA/JSC Contract NAS9-13864, Document No. IR-58-8, February 20, 1975.
6. "SUMC-S MOSS Detailed Functional Specifications," RCA-ATL, NASA/MSFC Contract NAS8-29072, Document No. SUMC-S-R-SP-001-01.0, August 5, 1974.
7. "SUMC-S MOSS Application Program Design Specification," RCA-ATL, NASA/MSFC Contract NAS8-29072, Document No. SUMC-S-R-SP-004-00.0, October 4, 1974.
8. "Higher Order Language (HOL) Preprocessor Requirements Specification Document," IBM-FSD, NASA/MSFC Contract NAS8-30604, Document No. 74W-00252, Revision 1, January 30, 1975.
9. "The Programming Language HAL - A Specification," Intermetrics, Inc., Document No. MSC-01846, June, 1971.

