HAL/S-360
USER'S MANUAL
IR-58-2

October 8, 1973

Approved by:

[Signature]
Dr. F.H. Martin
Shuttle Program Manager

Approved by:

[Signature]
Daniel J. Lickly
HAL Language/Compiler Dept. Head

Prepared by:

R.E. Kole
L.A. Meuse
FOREWORD

This document was prepared for Rockwell International Corporation, Space Division, under Purchase Order #M3W8XMX-483000, and is submitted in fulfillment of PDRD No. IM011-A of PDRL No. IM-M009723 Rev. A dated 5/11/73.

This version of the User's Manual corresponds to release 4 of the HAL/S-360 compiler system.
TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose of This Manual
1.2 Scope of This Manual

2. RUNNING A HAL/S PROGRAM

2.1 Communication with OS/360 - Job Control Language

2.1.1 Introduction
2.1.2 The Catalogued Procedure
2.1.3 The Optional Parameters: OPTION and RUNPARM
2.1.4 Specifying the Source Language Input
2.1.5 Specifying the Standard Execution-Time Input
2.1.6 Specifying the Standard Execution Time Output
2.1.7 Specifying Additional Execution-Time JCL
2.1.8 A typical Run Submission

2.2 Compiler Outputs

2.2.1 Source Listing
2.2.2 Tables
2.2.3 Summaries
2.2.4 Diagnostics

2.3 Subsequent Steps

2.3.1 Link Step
2.3.2 The Execution Step

2.4 Creating and Running Program Complexes
2.4.1 Introduction 2-7
2.4.2 The Form of Compools and Comsubs 2-8
2.4.3 Compiling a Program Complex 2-9

3. COMPILATION LISTINGS 3-1

3.1 General Description 3-1
3.2 Formats 3-1
  3.2.1 Headings 3-1
  3.2.2 Statement Number 3-1
  3.2.3 Line Type 3-2
  3.2.4 Source Field 3-2
  3.2.5 Current Scope Field 3-2
  3.2.6 Information Field 3-2
3.3 The Output Writer 3-3
  3.3.1 Concept 3-3
  3.3.2 Auto-Indentation 3-3
    3.3.2.1 Declaration Statements 3-4
    3.3.2.2 Labels 3-4
    3.3.2.3 Scope 3-4
    3.3.2.4 IF Statements 3-4
    3.3.2.5 DO Groups 3-5
    3.3.2.6 Continuations 3-5
    3.3.2.7 Page Boundaries 3-5
    3.3.2.8 First Lines 3-5
  3.3.3 Multi-line Expansion and Annotation 3-6
    3.3.3.1 Overpunches 3-6
    3.3.3.2 Array and Structure Notation 3-6
    3.3.3.3 Subscripts and Exponents 3-7
    3.3.3.4 REPLACE'd Symbols 3-7
3.3.4 Comments 3-7
  3.3.4.1 Comment Cards and Directive Cards 3-7
  3.3.4.2 In-Line Comments 3-7
3.4 Block Summaries 3-9
  3.4.1 Concept 3-9
3.4.2 Information Provided

3.5 Program Layout Summary

3.6 Symbol & Cross Reference Listing
   3.6.1 "DCL" Field
   3.6.2 "NAME" Field
   3.6.3 "TYPE" Field
   3.6.4 "ATTRIBUTE & CROSS REFERENCE" Field

3.7 Macro Table

3.8 Optional Unformatted Listing
   3.8.1 Format

3.9 Additional Information

3.10 Phase II Listing

4. DEBUGGING AIDS

4.1 Compilation Errors
   4.1.1 Message Format
   4.1.2 Classification Scheme
   4.1.3 Error Severity
   4.1.4 Phase I Error Summary
   4.1.5 Phase II Errors

4.2 Execution Errors
   4.2.1 Introduction
   4.2.2 GO TO Action
   4.2.3 SYSTPM Action
   4.2.4 IGNORE Action
   4.2.5 The Error Summary
4.3 Execution Dumps and Traces

4.3.1 What is Available

4.3.1.1 TRACE
4.3.1.2 DUMP
4.3.1.3 Location of TRACE and DUMP output

4.3.2 Invoking the TRACE Facility

4.3.2.1 Compile Time Parameters
4.3.2.2 Execution Time Parameters

4.3.3 TRACE Output

4.3.4 Invoking the DUMP Facility

4.3.4.1 Compile Time Parameters
4.3.4.2 Execution Time Parameters

4.3.5 DUMP Output

4.3.6 Abnormal Termination

5. HAL/S REALTIME PROGRAMS
(to be supplied)

6. HAL/S CHARACTERISTICS SPECIFIC TO THE 360

6.1 Introduction

6.2 Compile Time Characteristics

6.2.1 Character Set
6.2.2 Internal Table Capacities
6.2.3 Data Type Size Limitations
6.2.4 Program Organization Limits
6.2.5 Input/Output Statements
6.2.6 Program Naming Convention
6.2.7 The INCLUDE Compiler Directive

6.3 Execution Time Characteristics

6.3.1 Input/Output
APPENDICES

A.  COMPIL-E-TIME JCL OPTIONS  A-1
B.  EXECUT-E-TIME JCL OPTIONS  B-1
C.  PROTOTYPE CATALOGUE PROCEDURES  C-1
D.  COMPIL TME ERROR MESSAGES  D-1
E.  EXECUTI-TME ERRORS  E-1
F.  USER ABEND CODES  F-1
G.  NAMES RESERVED FOR HAL/S PROGRAMS  G-1
H.  COMPILR DIRECTIVES  H-1
1. INTRODUCTION

1.1 Purpose of This Manual

This manual will provide the information needed by a programmer to compile and execute a HAL/S program. It also provides a detailed discussion of the printed matter that will be produced as a result of the compilation and execution of a HAL/S program. This manual is not a guide to the HAL/S language. It is intended as a reference document to be used in the process of getting HAL/S programs compiled and debugged on the IBM/360. A knowledge of the HAL/S language syntax and programming techniques is presumed in some of the discussions.

1.2 Scope of This Manual

The succeeding sections of this document present a system guide for all phases in the development of a successful HAL/S program. Topics range from operating system communication to interpretation of debugging aids. A final section presents features of the HAL/S programming system that have specific System/360 dependencies.
2. RUNNING A HAL/S PROGRAM

2.1 Communication with OS/360 - Job Control Language

2.1.1 Introduction

All communication between the programmer and the operating system of the host computer must be done through Job Control Language (JCL). This section will present the basic JCL that must be provided to invoke HAL/S. A detailed discussion of JCL is not attempted. The intent is to give first-time and average users sufficient information to begin running. A more detailed description of the HAL/S JCL is available in Appendix C. That description is written for persons experienced in handling JCL and therefore does not "teach" the use of JCL.

2.1.2 The Catalogued Procedure

Because JCL is a complex language, the operating system (OS/360) allows for the grouping and saving of whole blocks of JCL. Such a saved block of JCL is known as a catalogued procedure. When this facility is used, the programmer need only submit a minimum of his own JCL to make a run. The descriptions that follow presume the existence of a catalogued procedure that will compile, load, and execute a HAL/S program. A listing of a prototype catalogued procedure (HALSCLG) is presented in Appendix C. Any JCL modifications that are desired may be made in the standard manner described in the IBM publication:

IBM System/360 Operating System:  
Job Control Language User's Guide  
Order #GC28-6703

The user calls in the catalogued procedure by referencing it by name on an EXEC card as follows:

//ANYNAME EXEC HALSCLG

This card is sufficient to call in the catalogued JCL and begin execution of the steps in the compilation process. If no other information is supplied on this EXEC card, all options available to the programmer for specifications will default to that set of options saved in the catalogued procedure. The user may change or add to these options by specifying more information on the EXEC card.
2.1.3 The Optional Parameters: OPTION and RUNPARM

The HAL/S compiler has various options that the user may specify through JCL. These options invoke functions in both the compilation and execution steps of a HAL/S job. The catalogued procedure HALSCLG allows the specification of these options with any of the keyword parameters OPTION or RUNPARM. These parameters are code on the EXEC card as follows:

    //ANYNAME EXEC HALSCLG,OPTION='??',RUNPARM='??'

The OPTION parameter is put into the PARM field of the compilation step and is available to the compiler for interpretation and action (valid options are listed in appendix A). The RUNPARM parameter is similarly made available to the HAL/S execution-time monitoring system.

2.1.4 Specifying the Source Language Input

The user must identify, through JCL, the location of the source program that he wants compiled. The typical input is from punched cards. The compiler reads the source input from the DD card named SYSIN. This card is not supplied in the catalogued procedure since the user must do the specification. For card input the specification would be:

    //HAL.SYSIN DD *
      .
      .
    Source cards
      .
      .
    /*
where the * on the DD card indicated input to follow. For source images saved on some other medium, the HAL.SYSIN DD card must still be included, but the specification on the card must correctly identify the source file. Refer to the IBM JCL manuals for the techniques needed.

2.1.5 Specifying the Standard Execution-Time Input

The catalogued procedure makes the assumption that the primary data input to the running HAL/S program will be made via sequential input file #5. This means that the catalogued procedure supplies a DD card with the name CHANNEL5. The use of the HAL/S statement

```
READ(5) <specification list>;
```

causes data to be read from the data set defined by the CHANNEL5 DD card. The catalogued procedure is organized to associate the following JCL cards with CHANNEL5.

```
//GO.SYSIN DD *
  
  
  Data cards
  
  
  /*

This DD card may alternately be defined in any suitable manner to reflect the location of the desired input data.

2.1.6 Specifying the Standard Execution Time Output

The catalogued procedure provides the necessary JCL to direct the results of the following HAL/S statement to a line printer:

```
WRITE(6) <specification list>;
```

The JCL statement responsible is the

```
//CHANNEL6 DD ... 
```

statement.

2.1.7 Specifying Additional Execution-Time JCL

The HAL/S programmer may reference 10 separate
sequential files with HAL/S I/O statements. In HAL/S statements of the form

```
READ
WRITE (n) <specification listing>
READALL
```

\( n \) may vary from 0 through 9. These statements cause the requested I/O operation to occur on data defined by JCL cards of the following form:

```
//CHANNELn DD <appropriate specification>
```

where \( n \) is the same as in the HAL/S I/O statement. Cards of this form may be added to the JCL brought in from the catalogued procedure by following the rules described in the IBM JCL publications.

2.1.8 A Typical Run Submission

The following JCL is an example of a typical user run. The user has his HAL/S program and his execution-time data on punched cards. In addition his program contains a WRITE(7) ... statement that he wishes to direct to a card punch.

```
1 //ANYNAME1 JOB <installation dependent parameters>
2 //TRYHAL EXEC HALSCLG,OPTION='LISTING2'
3 //HAL.SYSIN DD *
   ...
4 Source Program
   ...
5 /*
6 //GO.CHANNEL7 DD SYSOUT=B
7 //GO.SYSIN DD *
   ...
8 Data Cards
   ...
9 /*
```

Comments on individual lines in the example:

1 To identify himself to the operating system and give pertinent accounting information, the user must supply a JOB card. The form of this card is
This is the EXEC card that causes the JCL saved in catalogued procedure HALSCLG to be read by the operating system. The label TRYHAL is optional and if included may be any 1 to 8 character name beginning with a letter. If omitted, at least one blank must separate the // and the word EXEC. Following the name of the catalogued procedure, the user has coded some optional parameters as specified in Section 2.1.3 of this document. The OPTION keyword shown causes the string 'LISTING2' to be available to the compiler. The compiler recognizes this as a directive to produce an auxiliary source listing (See Sec. 3.8).

This card identifies the primary compiler input as cards immediately following.

The source cards follow.

The /* delineates the end of the in-line source cards and indicates a return to JCL card processing.

The CHANNEL7 DD card defines the destination of the HAL/S program's references to device #7 as system output class B (SYSOUT=B). At a typical 360 installation this class refers to the card punch.

GO.SYSIN defines the input data set associated with HAL/S references to device #5.

Note: Because of the way the HALSCLG catalogued procedure is written, this could also have been specified as:

//GO.CHANNEL5 DD *

The data cards to be read from channel 5 come next.

The /* terminates the input card data and indicates a return to JCL processing. No special end-of-job JCL indication is needed. The operating system will determine the job boundaries by the occurrence of subsequent JOB cards.
2.2 Compiler Outputs

2.2.1 Source Listing

As a result of the compilation process, a listing of the user-supplied source code is printed by the compiler. This primary listing has been formatted by the compiler to conform with standard output rules. The primary listing is always produced and is written to the data set defined by the SYSPRINT card.

The HAL/S compiler operates as two separate phases: Phase I or the syntax analysis phase and Phase II or the code generation phase. Each phase produces some informational and diagnostic output which together make-up the primary source listing.

An optional unformatted listing is available. The user must specify the LISTING2 option in the OPTION field of the EXEC card which invokes the HAL/S catalogued procedure.

The formats of both of these listings are discussed in Section 3.

2.2.2 Tables

In addition to reproducing the HAL/S language source code, the compiler also prints various tables that contain information of interest to the programmer. The tables include the Symbol Table & Cross Reference Table, giving name, type and usage information of identifiers, and the Macro Table giving a summary of replaced names. See Sections 3.6 and 3.7 for descriptions of these tables.

2.2.3 Summaries

The HAL/S compiler produces summaries of programmer actions taken within a particular program block at the close of that block and a quick-reference program layout description at the end of the compiled program. See Sections 3.3 and 3.4.

2.2.4 Diagnostics

The compiler produces error messages when syntax errors or other abnormal conditions occur. These error messages are interlisted with the source listing. An
error summary is provided at the end of the Phase I listing. Section 4 discusses compile time diagnostics in detail.

2.3 Subsequent Steps

2.3.1 Link Step

After an object file has been produced as described above, it must be further processed into a form suitable for loading and execution. This process includes the resolution of any references to HAL/S library routines and the generation of appropriately sized work areas required by the HAL/S programs at run time. These tasks are accomplished in the second step of HALSCLG. This step invokes a program known as HALLINK, a HAL/S compiler system program which performs all necessary functions. The HALLINK program dynamically invokes the System/360 Linkage Editor as part of its operation.

The printed matter generated by this step in the HALSCLG procedure appears in three parts:

1) A standard output produced by the Linkage Editor which may consist of a module map and size statistics. Descriptions of this listing may be found in the appropriate IBM system manual:

IBM System/360 Operating System
Linkage Editor and Loader
Form C28-6538

2) A HALLINK listing which documents the tree structure of all HAL/S modules involved in the link edit.

3) A second standard linkage editor listing as described above. This listing will incorporate changes made to the module structure by the HALLINK program. This second link editor listing is the one corresponding to the final load module produced by this step.

The HALLINK step puts its final result on a direct access device suitable for subsequent loading and execution. The load module thus produced requires a third step to be executed.

2.3.2 The Execution Step

The execution of a compiled HAL/S program may produce both user defined output and system diagnostic output. The user output occurs as a result of HAL/S I/O statements. The system diagnostic output can occur as a
result of execution errors detected by the system or as a result of user requests for dynamic dumps and traces.

2.4 Creating and Running Program Complexes

2.4.1 Introduction

Section 2.1 has explained how to run a self-contained HAL/S program. However, the form of the language allows a HAL/S program to use data external to itself (COMPOOLS), and to call external procedures or functions (COMSUBS). A HAL/S program and the compools and comsubs it uses are collectively known as a PROGRAM COMPLEX. This section explains how to create and run a program complex.

2.4.2 The Form of Compools and Comsubs

This subsection briefly recapitulates the forms taken by compools and comsubs in the HAL/S language. Both compools and comsubs are treated by the HAL/S compiler as independently compilable entities in the same way as a program.

The form of a compool is illustrated by the following example:

```
DATA: COMPOOL;
    DECLARE S SCALAR INITIAL(2.5);
    DECLARE I INTEGER INITIAL(5);
CLOSE DATA;
```

Fig. 2.1

The form of a typical comsub is illustrated by the following example:

```
ROUTINE: PROCEDURE(X);
    DECLARE X SCALAR;
    WRITE(6) X;
CLOSE ROUTINE;
```

Fig. 2.2
A program using data in the compool DATA and calling the comsub ROUTINE must contain the appropriate matching templates for them. Such a program is illustrated by the following example:

```
DATA: EXTERNAL COMPOOL;
   DECLARE S SCALAR INITIAL(2.5);
   DECLARE I INTEGER INITIAL(5);
CLOSE DATA;

ROUTINE: EXTERNAL PROCEDURE(X);
   DECLARE X SCALAR;
CLOSE ROUTINE;

TEST: PROGRAM;
   CALL ROUTINE(I/S);
CLOSE TEST;
```

Fig. 2.3

The HAL/S language of course also allows comsubs themselves to access compool data and/or other comsubs.

2.4.3 Compiling a Program Complex

To compile a program complex, the program module, and the compool and comsub modules must each be compiled separately (although in any order). The compilation of each module is carried out in exactly the same way. For each compilation, a catalogued procedure should be used which incorporates JCL to enable the object module produced to be saved (e.g. the HALSC catalogued procedure in Appendix C). When all object modules have been created, they are linked together (and with the runtime library) by using the HALLINK program (e.g. via the HALSL catalogued procedure listed in Appendix C.). The resulting load module is then executed as described previously for a simple program.

Each compilation also produces symbolic data used as a run-time debugging aid. In compiling and executing simple programs using the catalogued procedure HALSCLG, this symbolic data is written on a member of a temporary PDS passed to the load step of the JCL. These are two methods for insuring that the symbolic data for all modules of a program complex are correctly made available at execution time.

(i) The symbolic data for each module is saved on a different member of a PDS common to all compilations. In the execution step this dataset should be specified on the HALSYMB DD card. (See catalogued procedure in Appendix C.)
(ii) The symbolic data for each module is saved on a different PDS (each of which will therefore only have one member). In the execution step the HALSYM DD cards should specify the catenation of all the PDS's used.
3. COMPILATION LISTINGS

3.1 General Description

The listings produced by the HAL/S compiler are designed to document the actions taken by the compiler in the generation of an executable form of the user's source program. The user's code is reproduced in an annotated form and, optionally, in its original form. All tables and error messages generated by the compiler are also considered part of the documentation and are described in the following sections.

3.2 Formats

The numbered notes in the following discussion refer to an example of a HAL/S source listing shown in Figure 31.

3.2.1 Headings

A one-line page header (1) begins every page of the listing. It contains compiler version identification and page number within the listing.

On page one of the listing, the date and time of generation of the compiler are printed, followed by the date and time at which the current compilation was begun (2).

Following any header information on each page of the listing, a field description line is printed (3). This line breaks the page into columns, the contents of which are described below.

3.2.2 Statement Number

The statement number field (4) headed by the title "_STMT" contains the compiler-assigned sequence number for each HAL/S statement. This field is filled in for each M-line in the source listing. The "_STMT" field for E-line and S-line entries as well as for comment cards and compiler directives is left blank.

Note that the statement number is associated with a complete HAL/S statement, not with the physical number of M-lines. Thus, if a HAL/S statement spans several M-lines, the same statement number will appear on each
M-line.

3.2.3 Line Type

The STMT field is followed by a blank and then by a single character field (5) used to indicate the type of source line. HAL/S has multi-line subscripting and exponentiation capabilities. Such multiple line use is identified for easier reading.

The compiler places an indicator of line type in this one character field. The possible values of this field are:

- C = Comment line
- D = Compiler directive line
- E = Exponent line
- S = Subscript line
- M = Main line

These values correspond generally to the card types (punched in column 1) of the user's source cards.

3.2.4 Source Field

The next field on the page (6) is centered under the title "SOURCE". This field contains the actual HAL/S language text. The field is delimited by vertical bars at either side. The field is 100 characters wide and is filled by reformatted source text, complete with compiler-supplied annotation.

3.2.5 Current Scope Field

Following the source delimiter (!) at the right of the SOURCE field, is the variable length field headed by "CURRENT SCOPE" (7). This is an information field which contains the name of the HAL/S program block to which the current source line belongs. This field applies only to C, D, and M lines.

3.2.6 Information Field

To the right of the "CURRENT SCOPE" field is another variable length area that is used by the compiler to supply additional information. This information may be any compiler generated comments regarding the current line. This field is applicable only to M-lines.
3.3 The Output Writer

3.3.1 Concept

The HAL/S compiler has been designed to provide standard, automatic annotation of its output listing to enhance the readability of HAL/S source code. The HAL/S system allows each programmer to enter programs in a free-form input consistent with individual coding preferences. The compiler then edits the input during compilation into the standard form so that all program listings will observe the same coding rules.

Since HAL/S is a block-oriented language, the logical indenting of program blocks can do a great deal to enhance understanding of program structure. The programmer can do this indenting himself, but the problems of inserting new indentation levels into existing code often cause considerable wasted time in re-punching existing lines to maintain consistency. HAL/S frees the programmer from this task by completely regenerating the indentation scheme each time the program is compiled. Thus, the indentation is always complete and reflects the total program structure.

Although HAL/S source input is in the form of 80 column card images, the compiler treats the input as a continuous stream of information, with only the statement-delimiting semicolons to indicate statement boundaries. Each statement is stored internally until its semicolon is found. Then, with a complete statement in hand, the HAL/S output writer completely reformats the source. The reformatting includes referencing the symbol table to obtain the types of any variables in the statement so that the characteristic HAL/S overpunch mark may be supplied by the compiler. The reformatting also includes expansion of single line input to the full HAL/S multi-line form. Finally, the resulting multi-line, annotated statement is indented to the proper level determined by the line's relative position in the program. The statement is then placed on the output listing, using as many E-M-S groups as necessary to contain it.

The specific conventions imposed on the output listing are detailed in the following sections.

3.3.2 Auto-Indentation

References in this section are to Figure 3.1.
3.3.2.1 Declaration Statements. The output writer breaks up declaration statements in a way which provides maximum visibility of the intent of the declaration. The word DECLARE is aligned at the current indent-level. If the DECLARE has factored attributes, the attributes are placed on the same line as the DECLARE and a new line begun (8). If no factors are present, variable names follow the DECLARE (9). Lists of variables without individual attributes are placed on the same line (10). The occurrence of a variable with attributes causes that variable to appear on a line by itself with its attributes (11). Any lines created after the DECLARE line are indented one indent level.

Structure declarations (12) are reformatted into the commonly used form. Each level of the structure is placed on a separate line with indenting appropriate to the level number.

3.3.2.2 Labels. All statement labels (13) are right justified against the statement to which they apply. The statement itself is placed at the proper indent level before the label is applied. If the label will not fit on the same line as the statement body because of the indent location, it is placed on a separate M-line preceding the statement body (14).

3.3.2.3 Scope Changes. Whenever a PROGRAM, PROCEDURE, TASK, FUNCTION, or UPDATE block is encountered, the statement is placed in the output listing at the current level and then the indent level is increased one increment (15). All statements within the block follow the normal indenting rules relative to the block level indentation. Thus, all statements within a block are indented farther than the block definition.

When the corresponding CLOSE statement is found, the CLOSE statement is output at the same level as the block definition statement and the indent level is reset to its value before the block was entered (16).

3.3.2.4 IF Statements. The IF ... THEN part of the statement is placed on the listing at the current indent level and the indent level is increased one increment. The "true part" of the statement is placed on the next level at the new indent level and the level is decremented one level (17).
If an ELSE clause is present, the "ELSE" gets a new line at the current level, which is the same level as the "IF", and the indent level is incremented. The "false part" of the statement is placed on the next line at the new level and then the level is decremented one level (18).

3.3.2.5 DO Groups. All types of DO groups receive the same treatment. The statement containing the DO is placed in the listing at the current indent level and the level is incremented (19). All statements in the range of the DO are indented relative to this new level. The END closing the group is placed at the same level as the DO (20).

The DO CASE statement obeys the same indent rules as other DO statements, but some additional notation is supplied by the output writer. The first M-line of each case is annotated in the information field beyond the current scope notation with a message of the form "CASE n" where n is the current case number (21). If the current case is really a "case within a case", (i.e., a nested DO CASE is in effect), the notation is "CASE a.b ... n" where the a.b ... indicates the structure of the case statements in the sense of: case n within case b within case a ... Also, the END associated with the DO CASE statement receives the additional information "DO CASE END" to help associate it with its group head (22).

3.3.2.6 Continuations. A reformatted E-M-S group may not fit on the printed page in a single group after the indentation rules have been applied. If this is the case, the output writer breaks the statement into as many E-M-S groups as necessary (23). The break never splits an identifier or keyword. A literal character string may be broken.

3.3.2.7 Page Boundaries. The output writer never places pieces of an E-M-S group on separate pages; i.e. an E-line at the bottom of one page and the corresponding M-line on the next. A page eject will always be performed before such a group is written.

3.3.2.8 The first line of any PROCEDURE, FUNCTION, PROGRAM, TASK, or UPDATE block always begins on a new page.
3.3.3 Multi-line Expansion and Annotation

References are to Figure 3.1.

3.3.3.1 Overpunches. If, after having all pertinent subscripting applied, a variable name is of a type for which HAL/S has defined an overpunch character, the output writer supplies that character on an E-line above the variable name. The overpunch character is centered over the name.

The characters available are "*", "-", ".", ",", "+" for matrix, vector, bit, character, and structure data types respectively. The mark supplied is determined from the totally subscripted form of the variable. Thus the overpunch may be changed by subscripting. For example, an element of a matrix is a scalar. Therefore, a matrix name subscripted down to a particular element receives no overpunch (24). Similarly, a matrix variable subscripted to a particular row of the matrix receives a vector mark (25).

3.3.3.2 Array and Structure Notation. Variables which are structure terminals or which are arrayed may have additional annotation supplied. If a particular use of such a variable has multiple copies due to structure and/or array properties and if those multiple copies have not been subscripted away in the particular use of the variable, the variable is enclosed in the appropriate marks. Multiple copies due to arrayness receive brackets ("[" and "]") while multiple copies due to structure copies receive enclosing braces ("{|" and "}|"). (26)

As with variable typing (see Sec.3.3.3.1), it is possible to subscript away the "arrayness" or "structureness" of a variable and thus have no special annotation appear.

Note that the array and structure notation characters are available only on certain print trains such as the IBM TN chain. If a particular installation does not have such a print chain, the notation will probably appear as blanks (unprinting characters).
STMT
1 M1 DEMO:
1 M1 PROGRAM;

C1 THIS IS A DEMONSTRATION PROGRAM TO SHOW THE LISTING PRODUCED BY
C1 THE HAL/S-360 COMPILER
C1
2 M1 REPLACE PRINTER BY "6";
3 M1 DECLARE INTEGER INITIAL(1),
3 M1 A, B, C;
4 M1 DECLARE D, F, G;
5 M1 DECLARE E VECTOR(4);
6 M1 DECLARE H, I, J;
6 M1 K ARRAY(5) MATRIX(3, 4);
6 M1 L, M, N;
6 M1 O SCALAR;
7 M1 STRUCTURE AA:
7 M1 1 BB,
7 M1 2 CC MATRIX(4, 3),
7 M1 1 DD,
7 M1 2 EE ARRAY(4) MATRIX(3, 4):
8 M1 STRUCTURE QQ:
0 M1 1 RR,
8 M1 2 STRUCTURE AA,
8 M1 2 SS CHARACTER(5);
9 M1 DECLARE MY_STRUCTURE QQ-STRUCTURE(5):

Fig. 3.1 The HAL/S Source Listing
PROC1:  
PROCEDURE:
  DECLARE A INTEGER;
  IF A = B THEN
    DO;
  LABEL1: B = C;
  END;
  ELSE
    A = C;
  END
  CLOSE PROC1;

BLOCK SUMMARY

OUTER VARIABLES REFERENCED:
B, C

OUTER VARIABLES CHANGED:
B
STMT
18 M1 DO FOR C = 1 TO 100;
19 M1 D = K 24 ;
S1 C:2,3
20 M1 26 END;
21 M1 DO CASE A:
22 M1 A = B;
S1
23 M1 E = K 25 ;
S1 B:A,*
24 M1 END:
25 M1 [K] = 0;
26 M1 WRITE(PRINTER) MY_STRUCTURE.BB.BB.CC 3:1 TO 3,* [MY_STRUCTURE.BB.DD.EE] 2:2 TO 4:
S1 D, E, F, G, H
26 M1 23 I, J, [K], L, M, N, O;
27 M1 CLOSE DEMO;

Fig. 3.1 (con't)
3.3.3.3 Subscripts and Exponents. HAL/S allows the user to supply source statements in single or multi-line format. The output writer expands all source to full multi-line format before printing.

During the expansion process, any unnecessary subscript or exponent grouping parentheses are removed. These grouping parentheses are often needed in single line input to show the extent of a subscript or exponent field.

Subscripts applied to variables on an exponent line, and exponents applied to variables on a subscript line, are left in single line format since multiple line expansion would produce an ambiguous listing. Also, overpunch characters are not supplied for variables on exponent or subscript lines.

The multiple lines are indicated by "E" and "S" in the line type field of the listing (see 3.2.3). As many E or S lines as needed to contain the expanded source are generated.

3.3.3.4 REPLACE'd Symbols. Any symbols which are defined as replaced names in REPLACE statements are underlined by the output writer in the source listing (27).

3.3.4 Comments

3.3.4.1 Comment Cards and Directive Cards. All comment cards (C in column one) and directive cards (D in column 1) are transferred unchanged to the output listing. Groups of comment cards are separated by a blank line from other lines and single spaced within the group.

3.3.4.2 In-Line Comments. Comments appearing on the M-lines of input source cards in the form of

/* ... Comment ... */

are collected by the output writer and placed into the output listing after the statement has been processed.

The collected comments are placed on the M-line of the statement if possible. If there is not room, spill-over can occur onto as many S-lines as are necessary. Multiple comments on a single source statement are collected together and printed as one
comment. If the size of the total comment text for any one HAL/S statement reaches 256, additional comment text is ignored and a warning issued.
3.4 Block Summaries

3.4.1 Concept

The HAL/S compiler provides a summary of action taken within a program block at the close of the particular block. The blocks for which summaries are given are PROGRAM, TASK, FUNCTION and UPDATE. When the matching CLOSE to such a block is found, the summary is issued and the listing of the program resumes with a skip to the new page.

3.4.2 Information Provided

Information contained in block summaries consists of lists of labels or variable names used in various contexts within the block. The title "BLOCK SUMMARY" begins the list. For all potentially summarized contexts within the block, a descriptive heading is printed followed by the list of names involved. The headings and their meanings are listed below.

a) PROCEDURES CALLED - A list of PROCEDURE names which appeared as the object of a CALL statement somewhere in the current block. The names noted here have been defined as labels of PROCEDURE statements within the current program but outside this local block.

b) FUNCTIONS INVOKED - A list of user defined functions which were referenced within the current block. Only functions defined outside the current block are tabulated.

c) ERRORS SENT - A list of the HAL/S error numbers which are sent explicitly by a SEND ERROR statement somewhere in the current block. This is the only summary which does not involve a list of identifier names.

d) PROGRAMS SCHEDULED - A list of independent programs scheduled in the current block via the HAL/S SCHEDULE statement.

e) TASKS SCHEDULED - A list of tasks appearing as the object of a SCHEDULE statement somewhere in the current block.

f) PRIORITIES UPDATED - A list of task identifiers referenced by an UPDATE PRIORITY statement within the current block.
EVENTS SIGNALLED - A list of event variables declared outside the current scope and appearing in the current block as the object of a SIGNAL statement.

EVENTS WAITED FOR - A list of event variables declared outside the current block and appearing in the block as terms in an event expression.

OUTER VARIABLES REFERENCED - A list of data names declared at a level global to the current block which are referenced but not changed by the current block. Contexts in which such variables are tabulated this way include appearance in expressions, output lists, CALL statement parameter lists, etc.

OUTER VARIABLES CHANGED - A list of data names declared at a level global to the current block which are assigned new values within the current block. Contexts interpreted this way include appearance on the left-hand side of an assignment, input lists, CALL statement ASSIGN lists, etc.

OUTER REPLACE DEFINITIONS USED - A list of identifiers declared by appearance in a REPLACE statement in a scope global to the current block, which were used within the current block.

IMPLICITLY DECLARED VARIABLES - A list of identifiers which had not been defined before their use in the current block and which therefore had default declarations assumed for them.

Note that in all categories except c and l, only variables which have a NEST level less than that of the current block are included, as the block summary is an indication of the impact of this block outside of its own local variables or sub blocks. A block which is completely self-contained, no matter how complex, will not have a block summary issued.

The order of names in any list within the block summary indicates the order of occurrence of first usages of these names in the block within the identified context.

The block summary for any block will not duplicate the information in the block summary for any block nested within it (i.e. if Procedure A contains Procedure B which schedules Program C, Procedure B's block summary will indicate that Program C was called, but Procedure A's will not).
3.5 Program Layout Summary

Immediately preceding the Symbol Table printout at the CLOSE of the HAL/S program, there is a program layout map, indicating the wa in which PROGRAMS, TASKS, PROCEDURES, FUNCTIONS, and UPDATE blocks were defined. The indent level in this printout indicates the nesting level definition of the block shown. This serves to give a quick overview of the program structure. Such a listing can be of assistance not only as a documentation aid, but also as a guide to locating the definition of procedures and functions which have been diagnosed as undefined by the compiler.

3.6 Symbol & Cross Reference Table Listing

The symbol and cross reference table printed at the end of a HAL/S compilation listing provides a detailed accounting of all programmer-defined symbols. The table listing is organized into two parts: a structure template listing and an alphabetized total listing.

Any structure templates defined in the compilation appear first in the symbol and cross reference table. The template names appear in alphabetical order. The body of each template (i.e., the levels defined under the template name) is listed under the template name in the order of definition. This ordering provides a quick reference to the organization of the structure template. Special action is taken in the "NAME" and "TYPE" fields (described below) to highlight the template organization.

Following the listing of any templates, an alphabetized listing of all programmer-defined symbols is printed. Symbols previously listed as elements of a structure template are included in this list. However, the list is completely alphabetized and template organization is not shown. When a particular symbol is independently defined in more than one name scope, the symbol is multiply listed in order of definition. Figure 3.2 illustrates the form of a HAL/S compiler symbol and cross reference table.

3.6.1 "DCL" Field

The "DCL" field is used to list the compiler-assigned statement number at which the identifier was first declared. For explicit declarations, this number will point to a DECLARE statement somewhere in the program. For implicit declarations, this number is the statement at which the identifier was first used in the program.

3.6.2 "NAME" Field
The "NAME" field lists the symbolic name of the programmer-defined symbol. The width of this field is determined by the length of the longest symbol in the compilation. An asterisk preceding the name indicates that the variable was implicitly defined.

Within the first part of the symbol table listing which contains structure template names and their organizations as described in Section 3.6, all parts of the template body have their names indented one space from the structure template name under which they are defined. In this same template description section, all nested template references are indicated as "**" in the "NAME" field since there is no programmer-defined symbol available.

3.6.3 "TYPE" Field

The "TYPE" field describes the type of each programmer-defined identifier. This field will contain one of the following descriptions:

INTEGER
SCALAR
n - VECTOR
n X m MATRIX
BIT(n)
CHARACTER(n)
EVENT
PROGRAM
PROCEDURE
STATEMENT LABEL
STRUCTURE
STRUCTURE(n)
STRUCTURE NODE
TEMPLATE REFERENCE
REPLACE MACRO

The values of n and m indicate size or dimensionality in their particular contexts.

Additional information may appear in the type field. If the identifier has an array specification, the word "ARRAY" will follow the basic type specification as in:

"INTEGER ARRAY"

If the identifier is a function name, the type specification will be followed by the word "FUNCTION" as in:

"3 - VECTOR FUNCTION"

Within the first part of the symbol table listing
which shows the organization of structure templates, the "TYPE" field contains additional information to indicate the hierarchical relationships which exist within the structure templates. For each symbol which is part of a structure template, the "level number" of the symbol is printed in front of the type information for the symbol.

3.6.4 "ATTRIBUTES & CROSS REFERENCE" Field

This field contains all declared attributes of the symbol, lists all cross reference information, and contains comments about the identifier's use.

Attributes: This part of the field lists all data declaration attributes of variables or labels. In addition, special information about structure template elements is provided. The possible attributes which may appear are:

ARRAY(n,n,n)
SINGLE
DOUBLE
LOCKED
DENSE
ALIGNED
ASSIGN PARM
INPUT PARM
AUTOMATIC
STATIC
LATCHED
PROCESS EVENT
INITIAL
CONSTANT
ACCESS
REENTRANT
EXCLUSIVE
EXTERNAL
<template name>-STRUCTURE

Elements of structure templates have their full attributes listed in the first part of the symbol table listing where they are shown under their appropriate template name. In the alphabetized second part of the listing, these structure template elements will appear again but their attributes will not be repeated. Instead the notice:

"*** SEE STRUCTURE TEMPLATE <template name>"

will appear directing attention to the hierarchical template listing of the structure template whose name is given.

The attributes of a REPLACE MACRO entry will contain only the message:

"MACRO-TEXT INDEX=<number>"
Fig. 3.2 The HAL/S Program Layout and Symbol & Cross Reference Table Listing
### SYMBOL & CROSS REFERENCE TABLE LISTING:

(CROSS REFERENCE FLAG KEY: 4 = ASSIGNMENT, 2 = REFERENCE, 1 = SUBSCRIPT USE, 0 = DEFINITION)

<table>
<thead>
<tr>
<th>DCL</th>
<th>NAME</th>
<th>TYPE</th>
<th>ATTRIBUTES &amp; CROSS REFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>AA</td>
<td>STRUCTURE TEMPLATE</td>
<td>ALIGNED XREF: 0 0007 2 0008</td>
</tr>
<tr>
<td>7</td>
<td>BB</td>
<td>1 STRUCTURE NODE</td>
<td>ALIGNED XREF: 0 0007 POSSIBLY NOT USED</td>
</tr>
<tr>
<td>7</td>
<td>CC</td>
<td>2 4 X 3 MATRIX</td>
<td>SINGLE, ALIGNED XREF: 0 0007 2 0014 2 0026 POSSIBLY NOT ASSIGNED</td>
</tr>
<tr>
<td>7</td>
<td>DD</td>
<td>1 STRUCTURE NODE</td>
<td>ALIGNED XREF: 0 0007 POSSIBLY NOT USED</td>
</tr>
<tr>
<td>8</td>
<td>EE</td>
<td>2 3 X 4 MATRIX ARRAY</td>
<td>ARRAY(4), SINGLE, ALIGNED XREF: 0 0007 4 0014 2 0026</td>
</tr>
<tr>
<td>8</td>
<td>QQ</td>
<td>STRUCTURE TEMPLATE</td>
<td>ALIGNED XREF: 0 0008 2 0009</td>
</tr>
<tr>
<td>8</td>
<td>PR</td>
<td>1 STRUCTURE NODE</td>
<td>ALIGNED XREF: 0 0008 POSSIBLY NOT USED</td>
</tr>
<tr>
<td>8**</td>
<td>2</td>
<td>TEMPLATE REFERENCE</td>
<td>AA-STRUCTURE, ALIGNED XREF: 0 0008</td>
</tr>
<tr>
<td>8</td>
<td>SS</td>
<td>2 CHARACTER(5)</td>
<td>ALIGNED XREF: 0 0008 POSSIBLY NOT USED</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>INTEGER</td>
<td>SINGLE, ALIGNED, STATIC, INITIAL XREF: 0 0003 2 0021 4 0022 1 0023</td>
</tr>
<tr>
<td>11</td>
<td>A</td>
<td>INTEGER</td>
<td>SINGLE, ALIGNED, STATIC XREF: 0 0011 2 0012 4 0016</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>INTEGER</td>
<td>SINGLE, ALIGNED, STATIC, INITIAL XREF: 0 0003 2 0012 4 0013 2 0022 1 0023</td>
</tr>
<tr>
<td>7</td>
<td>BB</td>
<td>STRUCTURE NODE</td>
<td>**** SEE STRUCTURE TEMPLATE AA</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>INTEGER</td>
<td>SINGLE, ALIGNED, STATIC, INITIAL XREF: 0 0003 2 0013 2 0016 4 0018 1 0019</td>
</tr>
<tr>
<td>7</td>
<td>CC</td>
<td>4 X 3 MATRIX</td>
<td>**** SEE STRUCTURE TEMPLATE AA</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>SCALAR</td>
<td>SINGLE, ALIGNED, STATIC XREF: 0 0004 4 0019 2 0026</td>
</tr>
<tr>
<td>7</td>
<td>DD</td>
<td>STRUCTURE NODE</td>
<td>**** SEE STRUCTURE TEMPLATE AA</td>
</tr>
<tr>
<td>1</td>
<td>DEMO</td>
<td>PROGRAM</td>
<td>XREF: 0 0001</td>
</tr>
<tr>
<td>5</td>
<td>E</td>
<td>4 - VECTOR</td>
<td>SINGLE, ALIGNED, STATIC XREF: 0 0005 4 0023 2 0026</td>
</tr>
<tr>
<td>7</td>
<td>EE</td>
<td>3 X 4 MATRIX ARRAY</td>
<td>**** SEE STRUCTURE TEMPLATE AA</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>SCALAR</td>
<td>SINGLE, ALIGNED, STATIC XREF: 0 0004 2 0026 **** ERROR **** REFERENCED BUT NOT ASSIGNED</td>
</tr>
<tr>
<td>4</td>
<td>G</td>
<td>SCALAR</td>
<td>SINGLE, ALIGNED, STATIC XREF: 0 0004 2 0026 **** ERROR **** REFERENCED BUT NOT ASSIGNED</td>
</tr>
<tr>
<td>6</td>
<td>H</td>
<td>SCALAR</td>
<td>SINGLE, ALIGNED, STATIC XREF: 0 0006 2 0026 **** ERROR **** REFERENCED BUT NOT ASSIGNED</td>
</tr>
<tr>
<td>6</td>
<td>I</td>
<td>SCALAR</td>
<td>SINGLE, ALIGNED, STATIC XREF: 0 0006 2 0026 **** ERROR **** REFERENCED BUT NOT ASSIGNED</td>
</tr>
<tr>
<td>6</td>
<td>J</td>
<td>SCALAR</td>
<td>SINGLE, ALIGNED, STATIC XREF: 0 0006 2 0026 **** ERROR **** REFERENCED BUT NOT ASSIGNED</td>
</tr>
<tr>
<td>6</td>
<td>K</td>
<td>3 X 4 MATRIX ARRAY</td>
<td>APSAY(5), SINGLE, ALIGNED, STATIC XREF: 0 0006 2 0019 2 0023 4 0025 2 0026</td>
</tr>
<tr>
<td>6</td>
<td>L</td>
<td>SCALAR</td>
<td>SINGLE, ALIGNED, STATIC XREF: 0 0006 2 0026 **** ERROR **** REFERENCED BUT NOT ASSIGNED</td>
</tr>
<tr>
<td>13</td>
<td>LABEL1</td>
<td>STATEMENT LABEL</td>
<td>XREF: 0 0013 NOT REFERENCED</td>
</tr>
<tr>
<td>6</td>
<td>M</td>
<td>SCALAR</td>
<td>SINGLE, ALIGNED, STATIC XREF: 0 0006 2 0026 **** ERROR **** REFERENCED BUT NOT ASSIGNED</td>
</tr>
<tr>
<td>9</td>
<td>MY_STRUCTURE</td>
<td>STRUCTURE(5)</td>
<td>QQ-STRUCTURE, ALIGNED, STATIC XREF: 0 0009 6 0014 2 0026</td>
</tr>
<tr>
<td>6</td>
<td>N</td>
<td>SCALAR</td>
<td>SINGLE, ALIGNED, STATIC XREF: 0 0006 2 0026 **** ERROR **** REFERENCED BUT NOT ASSIGNED</td>
</tr>
<tr>
<td>6</td>
<td>O</td>
<td>SCALAR</td>
<td>SINGLE, ALIGNED, STATIC XREF: 0 0006 2 0026 **** ERROR **** REFERENCED BUT NOT ASSIGNED</td>
</tr>
<tr>
<td>2</td>
<td>PRINTER</td>
<td>REPLACE MACRO</td>
<td>**** SEE STRUCTURE TEMPLATE QQ</td>
</tr>
<tr>
<td>10</td>
<td>PROC1</td>
<td>PROCEDURE</td>
<td>MACRO-TEXT INDEX=1 XREF: 0 0002 2 0026</td>
</tr>
<tr>
<td>8</td>
<td>EE</td>
<td>STRUCTURE NODE</td>
<td>**** SEE STRUCTURE TEMPLATE QQ</td>
</tr>
<tr>
<td>8</td>
<td>SS</td>
<td>CHARACTER(5)</td>
<td>**** SEE STRUCTURE TEMPLATE QQ</td>
</tr>
</tbody>
</table>

### MACRO TEXT LISTING:

Fig. 3.2 (con't)
which directs the reader to the appropriate entry in the Macro Text Listing for a definition of the replace text.

Cross References: The remainder of the line, following any attributes and the word "XREF: ", is devoted to a list of all references to the identifier in the format:

N XXXX

where XXXX is a four digit specification of the line number of the HAL/S statement containing the identifier. N is a cross reference flag which specifies the identifiers usage:

Flag Code: N Use of Identifier
0 Definition
1 Subscript
2 Reference
3 Subscript and reference
4 Assignment
5 Subscript and assignment
6 Reference and assignment
7 Reference, assignment, and subscript

Comments: Following the last reference listed for each variable, the compiler may insert a comment about the variable usage. Possible comments are:

NOT USED if the identifier appears in a DECLARE statement but is neither referenced or assigned.

NOT REFERENCED if the variable is assigned a value but never used in any reference context.

NOT ASSIGNED if the variable value is referenced but never appears in any context where it can receive a value. This is obviously an error situation involving the use of an uninitialized variable. Therefore, the Symbol Table & Cross Reference listing contains the error message: "****ERROR**** REFERENCED BUT NOT ASSIGNED" to call attention to this situation.

3.7 Macro Table

Following the cross references, there will be a MACRO TEXT listing if any REPLACE definitions appear in the program.
1 M1 DEMO:PROGRAM;
   C1 THIS IS A DEMONSTRATION PROGRAM TO SHOW THE LISTING PRODUCED BY
   C1 THE HAL/S-360 COMPILER
   C1
2 M1 REPLACE PRINTER BY "S":
3 M1 DECLARE INTEGER INITIAL(1),A,B,C;
4 M1 DECLARE D,F,G;
5 M1 DECLARE Z VECTOR(4);
6 M1 DECLARE H,I,J,K ARRAY(5) MATRIX(3,4),L,M,N,O SCALAR;
7 M1 STRUCTURE AA:
   7 M1 1 BB,2 CC MATRIX(4,3), 1 DD,2 EE ARRAY(4) MATRIX(3,4)
   7 M1 :
8 M1 STRUCTURE QQ:
   8 M1 1 FF,
   8 M1 2 STRUCTURE AA,
   8 M1 2 SS CHARACTER(5);
9 M1 DECLARE MY_STRUCTURE QQ-STRUCTURE(5):
10 M1 PROC1:PROCEDURE;
11 M1 DECLARE A INTEGER;
12 M1 IF A=B THEN DO;
13 M1 LABEL1:B=C;
14 M1 MY_STRUCTURE.BB.DD.EE$$(3:2,*):$
15 M1 MY_STRUCTURE.BB.BB.CC$$(*,2);
16 M1 END;
17 M1 ELSE A=C;
18 M1 CLOSE PROC1;
19 M1 DO FOR C = 1 TO 100;
20 M1 D=K$$(C:2,3);
21 M1 """" END:

Fig. 3.3 The HAL/S Unformatted Source Listing and Additional PHASE I Information
CALLS TO SCAN = 297
CALLS TO IDENTIFY = 80
NUMBER OF REDUCTIONS = 603
MAX STACK SIZE = 8
MAX IND. STACK SIZE = 9
END IND. STACK SIZE = 1
MAX EXT_ARRAY INDEX = 6
XPEF LIST ENTRIES = 69
STATEMENT COUNT = 27
NUMBER OF SYMBOLS = 30
MINOR COMPACTIFS = 2
MAJOR COMPACTIFS = 0
MAX NESTING DEPTH = 2
PFFE STRING AREA = 26477


38 CARDS WERE PROCESSED.
ERROR OF SEVERITY 1: ONE OR MORE VARIABLES WERE REFERENCED BUT NOT ASSIGNED; SEE CROSS REFERENCE TABLE.

TOTAL ELAPSED TIME IN PHASE 1 0:0:8.02.
ELAPSED PHASE 1 SET UP TIME 0:0:0.25.
ELAPSED PHASE 1 COMPILING TIME 0:0:7.08.
ELAPSED PHASE 1 CLEAN-UP TIME 0:0:0.69.
PROCESSING RATE: 322 CARDS PER MINUTE.
21 M1 DO CASE A;
22 M1 A=B;
23 M1 E=K$(B:A,*);
24 M1 END;
25 M1 K=0;
26 M1 WRITE(PRINTER) MY_STRUCTURE.BB.CCS(3;1 TO 3,*),MY_STRUCTURE.RR.DD.EE
26 M1 $(2;2 TO 4:),D,E,F,G,H,I,J,K,L,M,N,O;
27 M1 CLOSE DEMO;
Fig. 3.3 contains a MACRO TEXT listing. The LOC field refers to the TYPE field in the symbol table listing for REPLACE class variables, and the TEXT indicates the string which was substituted whenever the identifier was encountered. REPLACE definitions are cross-referenced as well as any identifiers which may occur within replacement string.

3.8 Optional Unformatted Listing

The primary source listing generated by the HAL/S compiler is completely reformatted by the output writer. As a result, the source text has lost its relationship to the original card image input.

In order to give the programmer a correlation between the primary source listing and his original input, the HAL/S compiler will supply an optional unformatted source listing. A sample of such a listing is shown in Figure 4.

This option is requested by including the word "LISTING2" in the OPTION field of the JCL (see Section 2.1.3).

3.8.1 Format

The optional listing contains card images as read from the SYSIN DD card. These images receive no compiler-generated annotation. Each card image is bracketed by vertical bars (|).

To the left of each image, the compiler prints the card type identification from column one of the input card. The HAL/S statement number of the first statement included on the card is also printed to the left of the card type. This statement number is the same as the one assigned to the expanded source by the compiler in the primary source listing.

To the right of the card image, the compiler places a sequential card number, one number per card, which simply indicates a card's relative position in the input deck.

Current Scope information is supplied in the same manner as in the primary listing. No other information is put into the auxiliary listing. This listing does not receive any error indication messages.

3.9 Additional Information

Immediately following the Macro Text Listing, the compiler prints a list of internal statistics for use in compiler development. These statistics can be ignored by the
programmer.

3.10 Phase II Listing

The Phase II listing of the HAL/S Compiler identifies itself and lists current date and time before entering into the code generation process (see fig. 3.4).

If the 'OPTIONS=' field in the EXEC statement of the JCL specifies the 'LIST' option, a listing of all control section names needed by the compiled program is produced (see fig. ). The format of this listing is typical of IBM model 360 compilers and translators.

Following this, a completely formatted object module listing will be produced (see fig. 3.4). This listing shows the complete code generated for each control section in both hexadecimal and pseudo-assembler formats. This is listed after a line in the following format:

```
ST#n    EQU   *
```

where n is the compiled HAL/S statement number, and 'ST#n' is in the label field. The label field is also used to indicate both HAL/S labelnames and internal branch points in the same format as the HAL/S statement number indicator. The comments field gives information about the symbolic operand referenced by the instruction.

Following this is a map of relocation information included in the produced object module.

Regardless of whether 'LIST' was specified or not, a listing of several performance statistics is printed next. These are included to aid in compiler generation, and are of no concern to the programmer.
HAL/S COMPILER PHASE 2 -- VERSION OF OCTOBER 1, 1973. CLOCK TIME = 12:43:52.98

HAL/S PHASE 2 ENTERED OCTOBER 9, 1973. CLOCK TIME = 11:53:33.65

Fig. 3.4 The HAL/S PHASE II Listings
<table>
<thead>
<tr>
<th>ESDID</th>
<th>NAME</th>
<th>TYPE</th>
<th>LENGTH</th>
<th>BLOCK NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>demo</td>
<td>0000</td>
<td>00218</td>
<td>demo</td>
</tr>
<tr>
<td>0002</td>
<td></td>
<td>0004</td>
<td>00074</td>
<td>proc1</td>
</tr>
<tr>
<td>0003</td>
<td></td>
<td>0004</td>
<td>00005</td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td></td>
<td>0004</td>
<td>00008</td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td>demo</td>
<td>0002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>ioinit</td>
<td>0002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td>h21snp</td>
<td>0002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>eout</td>
<td>0002</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3.4 (con't)
<table>
<thead>
<tr>
<th>LOCCTR</th>
<th>CODE</th>
<th>LABEL</th>
<th>INSN</th>
<th>OPEFANDS</th>
<th>SYMBOLIC OPERAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td></td>
<td></td>
<td>ST#1</td>
<td>EQU *</td>
<td>ESDID= 0001</td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td></td>
<td>$DEMO</td>
<td>CSECT</td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td></td>
<td></td>
<td>DEMO</td>
<td>EQU *</td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td>47F0F010</td>
<td></td>
<td></td>
<td>EQU</td>
<td></td>
</tr>
<tr>
<td>000004</td>
<td>000000290</td>
<td></td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000008</td>
<td>0058</td>
<td></td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>00000A</td>
<td>04C4C5D4</td>
<td></td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>00002E</td>
<td>B6</td>
<td></td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>00002F</td>
<td>00</td>
<td></td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000010</td>
<td>58B8F004</td>
<td></td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000014</td>
<td>9B6AB080</td>
<td></td>
<td></td>
<td>LM</td>
<td></td>
</tr>
<tr>
<td>000018</td>
<td></td>
<td>ST#2</td>
<td></td>
<td>EQU *</td>
<td></td>
</tr>
<tr>
<td>000018</td>
<td></td>
<td>ST#3</td>
<td></td>
<td>EQU *</td>
<td></td>
</tr>
<tr>
<td>000018</td>
<td></td>
<td>$DEMO</td>
<td></td>
<td>CSECT</td>
<td></td>
</tr>
<tr>
<td>000018</td>
<td></td>
<td></td>
<td></td>
<td>EQU *</td>
<td>ESDID= 0004</td>
</tr>
<tr>
<td>000035</td>
<td>0001</td>
<td>ST#3</td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000035</td>
<td>0001</td>
<td>ST#4</td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000035</td>
<td>0001</td>
<td>ST#5</td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000035</td>
<td>0001</td>
<td>ST#6</td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000035</td>
<td>0001</td>
<td>ST#7</td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000035</td>
<td>0001</td>
<td>ST#8</td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000035</td>
<td>0001</td>
<td>ST#9</td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000035</td>
<td>0001</td>
<td>ST#10</td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000018</td>
<td></td>
<td></td>
<td></td>
<td>CSECT</td>
<td>ESDID= 0001</td>
</tr>
<tr>
<td>000018</td>
<td></td>
<td></td>
<td></td>
<td>CSECT</td>
<td>ESDID= 0002</td>
</tr>
<tr>
<td>000218</td>
<td></td>
<td></td>
<td></td>
<td>PROC1</td>
<td></td>
</tr>
<tr>
<td>000218</td>
<td>47F0F010</td>
<td></td>
<td></td>
<td>EQU</td>
<td></td>
</tr>
<tr>
<td>00021C</td>
<td>00000290</td>
<td></td>
<td></td>
<td>BC</td>
<td>15,16 (0,15)</td>
</tr>
<tr>
<td>000220</td>
<td>0058</td>
<td></td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000222</td>
<td>05D7D9D6</td>
<td></td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000226</td>
<td>C3F1</td>
<td></td>
<td></td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>000228</td>
<td></td>
<td>ST#11</td>
<td></td>
<td>EQU *</td>
<td></td>
</tr>
<tr>
<td>000228</td>
<td></td>
<td>ST#12</td>
<td></td>
<td>EQU *</td>
<td></td>
</tr>
<tr>
<td>000228</td>
<td></td>
<td></td>
<td></td>
<td>LH</td>
<td>2,6 (0,10)</td>
</tr>
<tr>
<td>00022C</td>
<td>4920A006</td>
<td></td>
<td></td>
<td>CH</td>
<td>2,2 (0,10)</td>
</tr>
<tr>
<td>000230</td>
<td>476F0068</td>
<td></td>
<td></td>
<td>BC</td>
<td>6,104 (15,0)</td>
</tr>
<tr>
<td>000233</td>
<td></td>
<td>ST#13</td>
<td></td>
<td>EQU *</td>
<td></td>
</tr>
<tr>
<td>000233</td>
<td></td>
<td></td>
<td></td>
<td>EQU *</td>
<td></td>
</tr>
<tr>
<td>000233</td>
<td></td>
<td></td>
<td></td>
<td>LH</td>
<td>2,4 (0,10)</td>
</tr>
<tr>
<td>000238</td>
<td></td>
<td>LABEL1</td>
<td></td>
<td>STH</td>
<td>2,2 (0,10)</td>
</tr>
<tr>
<td>000238</td>
<td></td>
<td></td>
<td></td>
<td>EQU *</td>
<td></td>
</tr>
<tr>
<td>000238</td>
<td></td>
<td></td>
<td></td>
<td>LA</td>
<td>9,1 (0,0)</td>
</tr>
<tr>
<td>00023C</td>
<td>4020A002</td>
<td></td>
<td></td>
<td>ST</td>
<td>9,80 (0,13)</td>
</tr>
<tr>
<td>000240</td>
<td></td>
<td>ST#14</td>
<td></td>
<td>EQU *</td>
<td></td>
</tr>
<tr>
<td>000240</td>
<td>u19000001</td>
<td></td>
<td></td>
<td>LA</td>
<td>9,166 (0,11)</td>
</tr>
<tr>
<td>000244</td>
<td>5090D0050</td>
<td></td>
<td></td>
<td>MM</td>
<td>8,80 (0,13)</td>
</tr>
<tr>
<td>000248</td>
<td>4903A0A6</td>
<td></td>
<td></td>
<td>L</td>
<td>8,166 (0,11)</td>
</tr>
<tr>
<td>000248</td>
<td>C88D0050</td>
<td></td>
<td></td>
<td>MH</td>
<td>2,12 (9,10)</td>
</tr>
<tr>
<td>000250</td>
<td>4C80B0A6</td>
<td></td>
<td></td>
<td>LA</td>
<td>3,56 (8,10)</td>
</tr>
<tr>
<td>000254</td>
<td>4129A0D4</td>
<td></td>
<td></td>
<td>LA</td>
<td>4,12 (0,0)</td>
</tr>
<tr>
<td>000258</td>
<td>4138A038</td>
<td></td>
<td></td>
<td>LA</td>
<td>0,4 (0,0)</td>
</tr>
<tr>
<td>000260</td>
<td>41000004</td>
<td></td>
<td></td>
<td>BAL</td>
<td>14,96 (0,12)</td>
</tr>
<tr>
<td>000264</td>
<td>4520C060</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
HAL/S COMPILATION -- PHASE 2 -- INTERMETRICS, INC.

LOCCTR   CODE     LABEL     INSN     OPERANDS
        000268  41900001  LA       9,1(0,0)
        00026C  5A900050  A        9,80(0,13)
        000270  4990B0A8  CH       9,168(0,11)
        000274  47FC002C  BC       12,44(15,0)
        000278  ST#15    EQU      *            H*5
        00027B  ST#16    EQU      *            000244
        000278  47FP0070  LBL#1    EQU      *            000288 LBL#3
        000280  4820A004  LBL#1    EQU      *            C
        000284  4020A006  LBL#1    EQU      *            A
        000288  ST#17    LBL#4    EQU      *            0000AC LBL#4
        000288  47F0C004  LBL#5    CSECT    CSEID= 0001
        000188  ST#18    EQU      *            C
        000198  41900001  EQU      *            H*100
        0001C4  490A004   EQU      *            H*3
        000200  4990B0A4  EQU      *            H*4
        000204  472F004C  EQU      *            K*28
        00020C  ST#19    EQU      *            D
        000220  4C90B0A2  EQU      *            00001C LBL#6
        000230  4C90B0A0  EQU      *            LBL#6
        000234  89900002  EQU      *            00004C LBL#4
        000238  7829A028  EQU      *            LBL#5
        00023C  7020A008  STE      2,8(0,10)            00007A LBL#12
        000240  ST#20    EQU      *            ERRSND
        000244  41900001  EQU      *            00008E LBL#9
        000248  490A004   EQU      *            A
        00024C  4820A000  EQU      *            B
        000250  5920B098  EQU      *            A
        000254  474F007A  EQU      *            B
        000258  413F00D4  EQU      *            A
        000264  59203000  EQU      *            B
        000268  472F007A  EQU      *            A
        000270  89200002  EQU      *            B
        000274  58823000  EQU      *            A
        000278  07FS     EQU      *            B
        00027A  04E00C1A  EQU      *            00008E LBL#9
        00027E  0D00033A  BAL      14,24(0,12)
        000282  ST#22    EQU      *            A*0000033A
        000288  47FP000E  EQU      *            00008E LBL#9
        00028C  4820A002  EQU      *            B
        000290  4020A000  EQU      *            A
        000294  ST#23    EQU      *            B
        000298  47FP000E  EQU      *            H*3
        00029C  4820A002  EQU      *            B
        0002A0  4020A000  EQU      *            H*3
        0002A4  LBL#15    EQU      *            B
        0002A8  4980A002  EQU      *            H*3
        0002AA  4890A002  EQU      *            B
        0002AE  4C90B0A2  EQU      *            H*3

Fig. 3.4 (con't)
<table>
<thead>
<tr>
<th>LOCCTR</th>
<th>CODE</th>
<th>LABEL</th>
<th>INSN</th>
<th>OPERANDS</th>
<th>SYMBOLIC OPERAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>000164</td>
<td>47CF0134</td>
<td>BC</td>
<td>12,308(15,0)</td>
<td>000134</td>
<td></td>
</tr>
<tr>
<td>000168</td>
<td>7800A008</td>
<td>LE</td>
<td>0,8(0,10)</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>00016C</td>
<td>05EC</td>
<td>BALR</td>
<td>14,12</td>
<td>EOUT</td>
<td></td>
</tr>
<tr>
<td>00016E</td>
<td>00000000</td>
<td>DC</td>
<td>A'000000000'</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>000172</td>
<td>4130A02C</td>
<td>LA</td>
<td>3,44(0,10)</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>000176</td>
<td>1344</td>
<td>SR</td>
<td>4,4</td>
<td>EOUT</td>
<td></td>
</tr>
<tr>
<td>000178</td>
<td>41000001</td>
<td>LA</td>
<td>0,1(0,0)</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>00017C</td>
<td>41100004</td>
<td>LA</td>
<td>1,4(0,0)</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>000180</td>
<td>05EC</td>
<td>BALR</td>
<td>14,12</td>
<td>EOUT</td>
<td></td>
</tr>
<tr>
<td>000182</td>
<td>00000000</td>
<td>DC</td>
<td>A'000000000'</td>
<td>M21SNP</td>
<td></td>
</tr>
<tr>
<td>000186</td>
<td>7800A00C</td>
<td>LE</td>
<td>0,12(0,10)</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>00018A</td>
<td>05EC</td>
<td>BALR</td>
<td>14,12</td>
<td>EOUT</td>
<td></td>
</tr>
<tr>
<td>00018C</td>
<td>00000000</td>
<td>DC</td>
<td>A'000000000'</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>000190</td>
<td>7800A010</td>
<td>LE</td>
<td>0,16(0,10)</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>000194</td>
<td>05EC</td>
<td>BALR</td>
<td>14,12</td>
<td>EOUT</td>
<td></td>
</tr>
<tr>
<td>000196</td>
<td>00000000</td>
<td>DC</td>
<td>A'000000000'</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>00019A</td>
<td>7800A014</td>
<td>LE</td>
<td>0,20(0,10)</td>
<td>J</td>
<td></td>
</tr>
<tr>
<td>00019B</td>
<td>05EC</td>
<td>BALR</td>
<td>14,12</td>
<td>EOUT</td>
<td></td>
</tr>
<tr>
<td>00019D</td>
<td>00000000</td>
<td>DC</td>
<td>A'000000000'</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>0001A0</td>
<td>7800A018</td>
<td>LE</td>
<td>0,24(0,10)</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>0001A4</td>
<td>05EC</td>
<td>BALR</td>
<td>14,12</td>
<td>EOUT</td>
<td></td>
</tr>
<tr>
<td>0001A6</td>
<td>00000000</td>
<td>DC</td>
<td>A'000000000'</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>0001A8</td>
<td>54900001</td>
<td>LA</td>
<td>9,9(0,0)</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>0001AC</td>
<td>4C900009</td>
<td>ST</td>
<td>8,80(0,13)</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>0001C0</td>
<td>49900002</td>
<td>MH</td>
<td>9,156(0,11)</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>0001C4</td>
<td>89900002</td>
<td>SLL</td>
<td>9,2</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>0001C8</td>
<td>41391000</td>
<td>LA</td>
<td>3,12(9,10)</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>0001CC</td>
<td>1844</td>
<td>SR</td>
<td>4,4</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>0001CE</td>
<td>41000003</td>
<td>LA</td>
<td>0,3(0,0)</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>0001D2</td>
<td>41100004</td>
<td>LA</td>
<td>1,4(0,0)</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>0001D6</td>
<td>05EC</td>
<td>BALR</td>
<td>14,12</td>
<td>M21SNP</td>
<td></td>
</tr>
<tr>
<td>0001DA</td>
<td>00000000</td>
<td>DC</td>
<td>A'000000000'</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>0001DC</td>
<td>41900001</td>
<td>LA</td>
<td>9,9(0,0)</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>0001E0</td>
<td>5A900000</td>
<td>A</td>
<td>9,80(0,13)</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>0001E4</td>
<td>4990000A</td>
<td>CH</td>
<td>9,168(0,11)</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>0001E8</td>
<td>47CF01BC</td>
<td>BC</td>
<td>12,44(15,0)</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>0001EC</td>
<td>7800A020</td>
<td>LE</td>
<td>0,32(0,10)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001EC</td>
<td>ST#27</td>
<td>BALR</td>
<td>14,12</td>
<td>EOUT</td>
<td></td>
</tr>
<tr>
<td>0001FA</td>
<td>05EC</td>
<td>DC</td>
<td>A'000000000'</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>0001FC</td>
<td>00000000</td>
<td>LE</td>
<td>0,36(0,10)</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>000200</td>
<td>7800A028</td>
<td>BALR</td>
<td>14,12</td>
<td>EOUT</td>
<td></td>
</tr>
<tr>
<td>000204</td>
<td>05EC</td>
<td>DC</td>
<td>A'000000000'</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>000208</td>
<td>00000000</td>
<td>LE</td>
<td>0,40(0,10)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00020C</td>
<td>05EC</td>
<td>BALR</td>
<td>14,12</td>
<td>EOUT</td>
<td></td>
</tr>
<tr>
<td>00020E</td>
<td>00000000</td>
<td>DC</td>
<td>A'000000000'</td>
<td>EQU</td>
<td></td>
</tr>
<tr>
<td>000210</td>
<td>00000000</td>
<td>BC</td>
<td>15,4(0,12)</td>
<td>STRACE</td>
<td></td>
</tr>
<tr>
<td>000218</td>
<td>47F00004</td>
<td>BC</td>
<td>15,372(0,12)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3.4 (con't)
<table>
<thead>
<tr>
<th>LOCCTR</th>
<th>CODE</th>
<th>LABEL</th>
<th>INSN</th>
<th>OPERANDS</th>
<th>SYMBOLIC OPERAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>000310</td>
<td>00000358</td>
<td>DC</td>
<td>A'00000358'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000314</td>
<td>00000358</td>
<td>DC</td>
<td>A'00000358'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000318</td>
<td>00000358</td>
<td>DC</td>
<td>A'00000358'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00031C</td>
<td>00000358</td>
<td>DC</td>
<td>A'00000358'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000320</td>
<td>00000358</td>
<td>DC</td>
<td>A'00000358'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000324</td>
<td>00000000</td>
<td>DC</td>
<td>A'00000000'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000328</td>
<td>00000001</td>
<td>DC</td>
<td>X'00000001'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00032C</td>
<td>000C</td>
<td>DC</td>
<td>X'0000C'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00032E</td>
<td>0010</td>
<td>DC</td>
<td>X'0010'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000330</td>
<td>0004</td>
<td>DC</td>
<td>X'0004'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000332</td>
<td>0003</td>
<td>DC</td>
<td>X'0003'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000334</td>
<td>0064</td>
<td>DC</td>
<td>X'0064'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000336</td>
<td>00F9</td>
<td>DC</td>
<td>X'00F9'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000338</td>
<td>0005</td>
<td>DC</td>
<td>X'0005'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00033A</td>
<td>1AC3C1E2</td>
<td>DC</td>
<td>X'1AC3C1E2'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00033E</td>
<td>C540E5C1</td>
<td>DC</td>
<td>X'C540E5C1'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000342</td>
<td>D9C9C1C2</td>
<td>DC</td>
<td>X'D9C9C1C2'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000346</td>
<td>D3C540D6</td>
<td>DC</td>
<td>X'D3C540D6'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00034A</td>
<td>E4E340D6</td>
<td>DC</td>
<td>X'E4E340D6'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00034E</td>
<td>C640D9C1</td>
<td>DC</td>
<td>X'C640D9C1'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000352</td>
<td>D5C7C5</td>
<td>DC</td>
<td>X'D5C7C5'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

END

Fig. 3.4 (cont')
HALS COMPILATION -- PHASE 2 -- INTERMETRICS, INC.

RLD POS REP FLAG ADDRESS
0001 0008 C8 000211
0001 0008 C8 000207
0001 0008 08 0001FD
0001 0008 C8 0001F3
0001 0007 C8 0001D9
0001 0008 08 0001B5
0001 0008 08 0001AB
0001 0008 08 0001A1
0001 0008 C8 000197
0001 0008 C8 00018D
0001 0007 08 000163
0001 0003 08 0000FE
0001 0007 C8 000155
0001 0007 08 00012D
0001 0006 08 000119
0001 0001 C8 000D0D
0001 0001 C8 0000D9
0001 0003 08 00007F
0001 0003 C8 000005
0002 0003 C8 00021D
0003 0005 C8 000325
0003 0004 C8 000321
0003 0004 C8 00031D
0003 0004 C8 000319
0003 0004 08 000315
0003 0004 C8 000311

LOC B DISP NAME

UNDER DEMO
000358 A 000 A
00035A A 002 B
00035C A 004 C
000360 A 008 D
000364 A 00C F
000368 A 010 G
000368 A 02C E
00036C A 014 H
000370 A 018 I
000374 A 01C J
000376 A 00C K
000378 A 020 L
00037C A 024 M
000380 A 028 N
000384 A 02C O
000488 A 038 MY_STRUCTURE

UNDER PROC1
000352 A 006 A

233 LINES OF ASSEMBLY CODE CONVERTED
856 BYTES OF PROGRAM, 1544 BYTES OF DATA.

Fig. 3.4 (con't)
<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. Indirect Stack Size</td>
<td>7</td>
</tr>
<tr>
<td>End Indirect Stack Size</td>
<td>1</td>
</tr>
<tr>
<td>Number of Statement Labels Used</td>
<td>18</td>
</tr>
<tr>
<td>Max. Storage Descriptor Stack Size</td>
<td>2</td>
</tr>
<tr>
<td>End Storage Descriptor Stack Size</td>
<td>1</td>
</tr>
<tr>
<td>Number of Minor Compactifies</td>
<td>1</td>
</tr>
<tr>
<td>Number of Major Compactifies</td>
<td>0</td>
</tr>
</tbody>
</table>

End of HAL/S Phase 2 October 9, 1973. Clock Time = 11:53:44.45

Total Elapsed Time in Phase 2: 0:0:10.80
Elapsed Phase 2 Set Up Time: 0:0:0.08
Elapsed Phase 2 Generation Time: 0:0:3.35
Elapsed Phase 2 Clean-Up Time: 0:0:7.37
4. DEBUGGING AIDS

4.1 Compilation Errors

4.1.1 Message Format

When Phase 1 of the HAL/S compiler (the syntax checking phase) detects an error condition, a diagnostic message is placed in the primary source listing at the point of detection. These error messages have the following form:

```
****** c ERROR #n OF SEVERITY s. *****

****** text of message
```

In this message:

c = mnemonic error name uniquely identifying this error message (see Sec. 4.1.2)

n = indication that this is the nth error in the current compilation

s = severity of error (see Sec. 4.1.3)

For error messages other than the first in a given compilation, the following line is placed after all error messages referring to a particular HAL/S statement:

```
****** LAST ERROR WAS DETECTED AT STATEMENT m *****
```

where m is the HAL/S statement number of the most recent previous statement that received an error message.

4.1.2 Classification Scheme

Each error message that may be generated by Phase 1 of the HAL/S compiler has a unique mnemonic designation which appears in the error printout. These mnemonics have been assigned according to general error-type classes. The first letter of each error message mnemonic indicates the major class to which the error belongs (see Appendix D). The second letter, if present, indicates a sub-class further describing the error. These one or two letters
are followed by a number which simply indicates members of a class-subclass group. The table at the beginning of Appendix D shows the meaning of the possible letter combinations.

4.1.3 Error Severity

The severity indication in the error message shows the effect of the error on the compilation process. The possible severities and their effects are as follows:

0 = warning (compilation proceeds normally)
1 = error (compilation proceeds, execution prevented)
2 = severe error (syntax check continues, code generation prevented)

>2 = abortive error (compilation halts immediately)

4.1.4 Phase I Error Summary

Near the end of the Phase I source listing and table printout, a summary of detected errors is printed in the following form.

END OF HAL/S PHASE 1, <date>. CLOCK TIME = <time>
n CARDS WERE PROCESSED.

x ERRORS WERE DETECTED. THE MAXIMUM SEVERITY WAS Y. THE LAST ERROR DETECTED WAS AT STATEMENT z.

***** SUMMARY OF ERRORS DETECTED IN PHASE 1 *****

ERROR #1 AT STATEMENT e1 OF SEVERITY s1
ERROR #2 AT STATEMENT e2 OF SEVERITY s2

4.1.5 Phase II Errors

Phase II of the HAL/S compiler (the code generation phase) may also produce some error messages (see Sec. 3.6). These messages have the form:

*** ERROR #1 DURING CONVERSION OF HAL/S STATEMENT n

text of message

4-2
where \( n \) is the HAL/S compiler-assigned statement number to which the error refers. Following these specific messages, Phase II supplies a disposition message indicating the total effect of all Phase II messages on the compilation.

4.2 Execution Errors

4.2.1 Introduction

This section describes how error handling has been implemented in the HAL/S 360 system. In the HAL/S system every error is assigned a non-zero positive number. There are two classes of errors:

i) system-defined errors, (1-74) which arise as a result of failure during the execution of a user's program, and are signalled internally;

ii) user-defined errors, (75-100) which are signalled by the user through a SEND ERRORn statement.

System-defined errors may also be signalled by the user through a SEND ERRORn statement.

The HAL/S Error Processor processes all errors and determines what action is to be taken. The user can gain control after an error by means of the ON ERRORn GO TO xxx statement, or leave the system in control by default or by means of the ON ERRORn SYSTEM statement. Additionally, the user can specify that the HAL/S Error Processor take no action for a specific error by means of the ON ERRORn IGNORE statement. These three possible actions will be referred to as GO TO action, SYSTEM action, and IGNORE action respectively.

In the following sections, the error format given shows the maximum information printed. If the compile time TRACE option is not in effect then there will be no information on the statement number of the last HAL/S statement to be fully executed.

4.2.2 GO TO Action

(to be supplied later)

4.2.3 SYSTEM Action

One of three possible SYSTEM actions may ensue:
UNLIMITED - The standard fixup for ERRORn is taken and the following message is printed:

***** HAL ERROR n - (message) --> (location)
LAST STATEMENT WAS m
STANDARD FIXUP, EXECUTION RESUMED

where (message) is the error text corresponding to ERRORn. The last HAL/S statement to be fully executed is specified by m and the address of the error is specified by (location). Following this, the Error Processor returns control to the user's program.

LIMITED - The error count for ERRORn is updated. If it equals or exceeds a user-set maximum count (refer to the the ERROPLIM option in Appendix B - note that this is applied to all LIMITED type errors for which SYSTEM action is specified), the following message is printed:

***** HAL ERROR n - (message) --> (location)
LAST STATEMENT WAS m
ERROR COUNT EXCEEDED

and execution of the user's program abnormally terminates.

Otherwise, if the maximum count is not exceeded, the standard fixup for ERRORn is taken and the following message is printed:

***** HAL ERROR n - (message) --> (location)
LAST STATEMENT WAS m
STANDARD FIXUP, EXECUTION RESUMED

and the Error Processor returns control to the user's program.

In both cases, (message) is the text for ERRORn, (location) is the address where the error occurred, and m is the last HAL/S statement to be fully executed.

TERMINATE - The following message is printed:

***** HAL ERROR n - (message) --> (location)
LAST STATEMENT WAS m
SYSTEM ACTION IS TERMINATE

where (message) is the text for ERRORn, (location) is the address where the error occurred, and m is the last HAL/S statement to be fully executed. Following this, execution of
the user's program abnormally terminates.

Note that the form of the SYSTEM action and standard fixup is specified for each error message in Appendix E.

4.2.4 IGNORE Action
(to be supplied later)

4.2.5 The Error Summary

Whether execution terminates normally or abnormally, if there were any errors signalled either internally or by the user, a summary in the following form is printed on the output channel defined by the MCHAN RUNPARM option (see Appendix B):

*** SUMMARY OF ERRORS ***

ERROR n1 OCCURRED r1 TIMES
ERROR n2 OCCURRED r2 TIMES
...

4.3 Execution Dumps and Traces.

4.3.1 What is Available

4.3.1.1 TRACE. For debugging runs, the HAL/S compiler can supply checkpoints in the compiled code at each executable statement. The "TRACE" compiler option provides this facility. Any program compiled with the TRACE option can take advantage of any of these execution-time options:

a) A running trace indicating the current statement number being executed.

b) A selective statement number trace, which can either be used to monitor a program segment or, alternatively, to isolate heavily used program segments such as large DO FOR loops, which could produce a large number of repeated trace points.

c) A back trace of the last N statements executed prior to program termination, where N is specified by the user.
d) A trace of block invocations and returns.

4.3.1.2 DUMP. The HAL/S system provides a readable dump of all program variables upon abnormal termination of any program. Program variable names, rather than machine addresses, are used to identify the items contained in the dump. The declaration type of the variables determines their appearance in the printout: matrices are printed in matrix form, character strings are printed as strings and integers and scalars are printed in decimal form; only bit variables are dumped in hexadecimal notation. This dump can be of significant value in determining the cause of program failure. The features available in using the DUMP facility are as follows:

a) Optional full program dump upon abnormal termination of a program.

b) Optional full program dump given at program termination whether normal or abnormal.

c) A full program variable dump at selected statement numbers.

d) Block label (PROGRAM, TASK, UPDATE, PROCEDURE, FUNCTION) dumps at selected statement numbers.

4.3.1.3 Location of TRACE and DUMP output. The TRACE output is directed to the same HAL/S output channel as error messages. The output normally goes to a printer file (SYSOUT=A) and appears interleaved with any program-induced output on that channel. The output channel number is optionally specified by the MCHAN execution time parameter. The DUMP output is directed to the file designated by the HALDUMP DD card which is normally defined as a printer file (SYSOUT=A). Note that this output is not interlisted with any HAL/S program output.

4.3.2 Invoking the TRACE Facility

4.3.2.1 Compile Time Parameters. To allow a program to use the execution time tracing facilities for statement tracing or back tracing (see appendix B for the run time options needed to invoke these forms of tracing) the program must be compiled with the TRACE option specified in the PARM field of the
compilation-step JCL. When using the catalogued procedure HALSCLG, this is accomplished by coding the option in the option parameter of the invocation step:

```c
//ANYNAME EXEC HALSCLG,OPTION='TRACE',<other compile options>',<other JCL options>
```

4.3.2.2 Execution Time Parameters. A program may specify execution time requests for trace output. The various forms of these TRACE requests are listed in appendix B.

A statement back trace always appears if a program terminates abnormally. The user may optionally call for a back trace on a normal termination by specifying the MSGLEVEL run time option to be greater than 0. The HISTORY option may be used to specify the number of statements to be backtraced.

These requests are placed in the PARM field of the step which executes the HAL/S program. When using the HALSCLG catalogued procedure, this is accomplished by coding the requests in the RUNPARM parameter of the invocation step:

```c
//ANYNAME EXEC HALSCLG,OPTION='TRACE',<other compile options>',RUNPARM='TRACE=1,HISTORY=20,<other execution options>',<other JCL options>
```

4.3.3 TRACE Output

When the TRACE function is triggered by the end of execution of a HAL/S statement for which a trace request was in effect, the following message appears on the defined output channel.

```
STATEMENT n
```

where n is the number of the source statement being traced, and whose execution has just finished.

The form of the statement back trace is illustrated in the following example:

```
STATEMENT HISTORY, FROM OLD TO NEW: 19 24 25 27
```

4.3.4 Invoking the DUMP Facility

4.3.4.1 Compile Time Parameters. To utilize features (a) and (b) of Section 4.3.1.2, no compile time options are required. To utilize features (c) and (d) of that section, the TRACE option must be
specified as noted in Section 4.3.2.1.

4.3.4.2 Execution Time Parameters. The user may request DUMP output in one of three forms as specified in appendix B. These DUMP requests are placed in the PARM field of the step which executes the HAL/S program. The method used to specify these options when invoking the HALSCLG catalogued procedure is identical to that for specifying the TRACE options (see Section 4.3.2.2).

4.3.5 DUMP Output

When the DUMP function is triggered, DUMP output is placed on the file defined by the HALDUMP DD card. The DUMP output for each block of code of the user's program begins on a new page with a header of the form:

VARIABLES DEFINED UNDER <block> <name>

where <block> may be PROGRAM, PROCEDURE, TASK, FUNCTION, UPDATE, or COMPOOL and <name> is the name of the block.

Note that this definition of "variables defined within the block" means only those variables at the next level of the block. Variables defined in blocks which are nested within the current block are not dumped with the outer enclosing block, but instead are dumped with the block in which they are actually declared.

Each variable dumped is identified by a line in the dump of the form:

<name> <type> <other information>

where <name> is the HAL/S name of the variable and <type> is the type of the variable. The <other information> field is filled with any pertinent facts about the named variable such as ARRAY information. A sample dump format is shown in Figure .

The <name> <type> line is followed by the actual value of the variable, printed in the format required by the type and size of each individual variable.

4.3.6 Abnormal Termination

If a HAL/S program terminates abnormally, the following information is output on the error message channel:

4-8
### VARIABLES DEFINED UNDER PROGRAM INVERSER

**B MATRIX (3,3)**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1.5582966E+60</td>
<td>2.5098037E-01</td>
<td>-4.6712099E+27</td>
</tr>
<tr>
<td>2</td>
<td>-1.3303539E+68</td>
<td>-6.7664175E+66</td>
<td>2.5098037E-01</td>
</tr>
<tr>
<td>3</td>
<td>2.5098037E-01</td>
<td>2.5098037E-01</td>
<td>2.5098037E-01</td>
</tr>
</tbody>
</table>

**A MATRIX (3,3)**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**C MATRIX (3,3)**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.5120925E-01</td>
<td>9.4509822E-01</td>
<td>2.5098037E-01</td>
</tr>
<tr>
<td>2</td>
<td>2.5098037E-01</td>
<td>9.7636336E-01</td>
<td>2.5099140E-01</td>
</tr>
<tr>
<td>3</td>
<td>-1.5269125E+60</td>
<td>2.5098037E-01</td>
<td>2.5329500E-01</td>
</tr>
</tbody>
</table>

**M MATRIX (3,4)**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.0000000E+00</td>
<td>3.0000000E+00</td>
<td>3.0000000E+00</td>
<td>1.0000000E+00</td>
</tr>
<tr>
<td>2</td>
<td>3.0000000E+00</td>
<td>3.0000000E+00</td>
<td>3.0000000E+00</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>3.0000000E+00</td>
<td>3.0000000E+00</td>
<td>3.0000000E+00</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**N MATRIX (3,4)**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0000000E+00</td>
<td>2.0000000E+00</td>
<td>3.0000000E+00</td>
<td>4.0000000E+00</td>
</tr>
<tr>
<td>2</td>
<td>4.0000000E+00</td>
<td>4.0000000E+00</td>
<td>1.5000000E+00</td>
<td>1.5000000E+00</td>
</tr>
<tr>
<td>3</td>
<td>1.5000000E+00</td>
<td>-1.1999998E+00</td>
<td>8.7999992E+00</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**MT MATRIX (4,3)**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>0.0</td>
<td>3.0000000E+00</td>
</tr>
<tr>
<td>4</td>
<td>2.5098037E-01</td>
<td>2.5098037E-01</td>
</tr>
</tbody>
</table>

**I INTEGER**

5

**T INTEGER**

1

The following message indicating the occurrence of the dump is output on the error message channel:

**HAL VARIABLE DUMP WRITTEN TO DD CARD HALDUMP**

---

Figure 4-1.
SYSTEM

ABEND CODE nnnn IN <name>, ENTRY=dddddddd

USER

ABEND --> AT ddddd, { +ddd IN 

| LIBRARY(<membername>) 
| INTRINSICS 
| <program block name>

UNKNOWN ADDRESS

LAST STATEMENT WAS n

The above may be followed by TRACE and DUMP information and by a summary of run time errors.

The following comments are useful in interpreting the above information:

a) <name> is the member name of the load module being executed.

b) ENTRY=dddddddd is the absolute address of the entry point into the module.

c) ABEND --> AT ddddd gives the absolute address where execution terminated abnormally.

d) UNKNOWN ADDRESS appears if the ABEND address does not lie within the bounds of the area occupied by the program.

e) +ddd indicates the ABEND address relative to the start of the program segment specified.

f) LIBRARY(<membername>) appears if the abnormal termination was in the runtime library.

g) INTRINSICS appears if the abnormal termination was in an intrinsic function.

h) If the abnormal termination was in the object code of the user program, then <program block name> gives the name of the block where the ABEND occurred.

i) Appendix F gives a list of user abend codes. System abend codes are those characteristic of OS/360.
5. HAL/S REALTIME PROGRAMS

(to be supplied)
7 characters of the HAL/S name. Therefore, care should be exercised in naming programs so that unique names will be generated.

b) Any occurrence of the underscore character (\_\) in the first 7 characters of a PROGRAM, PROCEDURE, FUNCTION, TASK, or COMPOOL is eliminated. The resulting characters are joined together to produce the name given to the control section which is created, (e.g. A-B-C becomes ABC). An additional character ($) is placed on the front of the resultant name to form the final name.

6.2.7 The INCLUDE Compiler Directive

Use of the INCLUDE compiler directive to access a symbolic library has been implemented in the following way. The name found on the INCLUDE directive such as:

```
INCLUDE COMPOOLA
```

will be used to search the symbolic library. The name must be a 1 to 8 character string beginning with a letter.

The symbolic library must be defined by the following JCL:

```
//HAL.INCLUDE DD DSN=<library name>,<other parameters>
```

The <library name> must be the data set name of the library. The library data set must have partitioned organization and must be referred to in the JCL by data set name only (DSN=HAL.LIB), not as data set and member name (DSN=HAL.LIB(COMPOOLA)).

The library thus defined is searched for a member whose name is exactly the name found on the INCLUDE directive. The library must contain 80 character card images but may be blocked in any legal manner. Two or more libraries may be concatenated in the JCL if they conform to the standard JCL rules for for such concatenation.

6.3 Execution Time Characteristics

6.3.1 Input/Output

a) The I/O device defined to be the error message channel (CHANNEL6 is the default) is forced to have record format PA or FBA.
b) Numeric formatting of output items is done as follows:

integers - an 11-character field is printed with the number right justified. A floating minus sign is added if the number is negative.

single precision scalars - a 14-character field is printed as follows:

\[ sx.xxxxxxxxxE+txx \]

where \( s \) is a blank or minus sign
\( x \) is a single digit 0 to 9
\( t \) is a plus or minus sign

double precision scalars - a 23 character field is generated as follows:

\[ sx.xxxxxxxxxxxxxxxxxxEtxx \]

c) If not specified through JCL, the HAL/S I/O routines will assume a logical record length of 80 for unpaged output devices and a logical record length of 133 for paged output devices. The first character of the 133 character paged output record is assumed to be an ASA carriage control character.
APPENDIX A.

Compile-Time JCL Options

The following is a list of options which may be coded in the "OPTION=" field of the JCL invoking the HAL/S compiler.

- LISTNG2
  - Causes unformatted source listing to be generated

- DUMP
  - Requests the compiler to produce a memory dump if certain internal compiler errors occur

- PAGES=n
  - Sets the maximum page number to be allowed in generation of the primary compilation listing to n. The default is 250 pages.

- LIST
  - Produces an assembly listing from PHASE II of the compiler

- TRACF
  - Causes the generation of statement tracing code in the object module
APPENDIX B.

Execution-Time JCL Options

The following is a list of options which may be coded in the "RUNPARM=" field of the JCL invoking the HAL/S compiler. This information is directed to the "PARM" field of the execution step.

**TAB=n** : Specifies the automatic spacing between elements on output. The default value is 5.

**LINES=n** : Specifies the number of lines per page of paged output. The default value is 55 lines.

**HISTORY=n** : Specifies the number of HALS statements recorded in the circular history buffer (minimum is 10) The default value is 100 statements.

**MCHAN=n** : Specifies the channel number (DD card CHANNELn) on which error and trace messages appear. The valid range is 0-9; CHANNEL6 is the default.

**TRACE=0** : Specifies no tracing (default option)

**TRACE=1** : Statement tracing
(Note that this option is valid only if 'TRACE' was specified on the compilation)

**TRACE=2** : Procedure & function call and return tracing

**TRACE=3** : Statement, procedure & function call, and return tracing

**DUMP=0** : No HAL/S variables dumped

**DUMP=1** : HAL/S variables dumped on ABEND (default option)

**DUMP=2** : same as DUMP=1 but also on normal end (Note that DD cards HALSYM and HALDUMP are required)
MSGLEVEL=0 : no backtrace on normal termination (this is the default value)

MSGLEVEL=1 : backtrace occurs on normal termination

ERRORLIM=n : specifies the maximum number of execution time errors allowed. The default value is 10. The maximum is 254; if 255 is specified, the SYSTEM action will be UNLIMITED.
APPENDIX C.

Prototype Catalogued Procedures

This Appendix shows five examples of the form of catalogued procedures needed to compile, link edit and execute a HAL/S program. The five procedures perform varying degrees of processing. The procedure names and their uses are:

- HALSC: Compile a HAL/S program
- HALSCL: Compile and link a HAL/S program
- HALSCLG: Compile, link, and execute a HAL/S program
- HALSL: Link a previously compiled HAL/S program
- HALSLG: Link and execute a previously compiled HAL/S program

The most complex procedure, HALSCLG, is described line by line.

The following comments apply to the prototype catalogued procedure HALSCLG listed in this appendix. The comments generally apply equally to either of the other procedures.

Line 10000 -  

This PROC statement names the procedure and defines the symbolic parameters. The OPTION and RUNPARAM parameters are the means of supplying optional information to the compiler and run-time system respectively.

Lines 10100 - 10200  

The name of the compilation step is HAL. The name of the actual program to be executed is MONITOR. MONITOR handles all compiler/CS interfaces and also performs the actual loading and overlaying of the two phases of the compiler. The compiler requires a 300K region. A larger region may be specified. The compiler will always
use all the memory it is given. A larger region will generally result in smaller compilation times. A default time limit of 1 minute is supplied. This is sufficient for most average size HAL/S programs (approx. 300 HAL/S statements).

The PARM field contains the compile-time options. The OPTION field receives any user-specified options.

Line 10300 -

The STEPLIB DD card specifies the location of the load module library which contains the module MONITOR needed to run the compiler. This card may define any direct access library which contains the proper module or may be deleted at installations where the module has been made part of the system library (SYS1.LINKLIB).

Lines 10400 -
10500

The PROGRAM DD card defines the phases of the compiler that are to be used. The two phases are specified as concatenated DD cards. The first DD card specifies the name and location of the syntax analysis phase. The second DD card specifies the name and location of the code generation phase. These two data sets have DCB's of the form:

$$DCB=(RECFM=P,LRECL=7200,BLKSIZE=7200)$$

and may reside on direct access or magnetic tape. It is recommended, however, that direct access be used.

Line 10600 -

The SYSPRINT DD card defines the primary listing data set. This is generally assigned to a system output class, but may be associated with any sequential data set with the proper characteristics. The record format must be FA or FBA with a logical record length of 133 and any appropriate block size. If not supplied in the JCL, the DCB will default to

$$DCB=(RECFM=FBA,LRECL=133,BLKSIZE=7182)$$

C-2
The LISTING2 DD card defines the secondary listing data set. It may define a system output class or any sequential data set. The DCH requirements are the same as for SYSPRINT.

The OUTPUT3 DD card defines the data set which is to receive object code which is produced by the compiler. In the prototype procedure, this data set is given a temporary Name (E&HALOBJ) and is passed (DISP=(MOD,PASS)) to subsequent steps.

Since this data set contains card images, it must have a logical record length of 80 and a record format of F or FB. The blocking factor may be any legal multiple of 80. If no DCB information is supplied in the JCL, the default will be DCB=(RECFM=FB,LRECL=80,BLKSIZ=7200). This DD card may specify a direct access, magnetic tape, or unit record device.

The OUTPUT5 DD card defines the data set which will receive the DUMP/TRACE mapping file. This file allows the execution-time DUMP and TRACE facilities to associate memory locations with HAL/S variable names. For the compile-load-go type of run, this data set is generally a temporary one which exists only for the duration of the job. However, if the results of a given compilation are saved to be executed many times without recompilation, the DUMP/TRACE mapping file should be saved as well.

The OUTPUT5 DD card must define a partitioned data set. The SPACE parameters used in the prototype procedure are more than adequate to contain the mapping file of any one HAL/S program. If a permanent file is to be used to save the mapping files of several HAL/S programs, the SPACE requirements should be based on the expectation of one 40 byte logical record per symbol table entry.

The OUTPUT5 DD card is accessed during the code generation phase of the compilation.
One member of the partitioned data set is created for a compilation unit. The name of the member is the first 8 characters of the name of the Program or Procedure being compiled, padded with blanks if necessary.

If a member with the desired name does not exist at compile-time, one is created. If a member with the desired name already exists, it is replaced by the new member.

The use of a permanent partitioned data set makes it possible to maintain a "library" of mapping files, with the member names uniquely specifying the HAL/S compilations to which the maps may be applied.

The format of the file must have a logical record length of 40 and a record format of PB. If DCB information is omitted, the default is

\[ DCB=(RECFM=PB,LRECL=40,BLKSIZE=400) \]

**Line 11300** - The ERROR DD card defines the partitioned data set which contains the error message texts used by the syntax analysis phase of HAL/S. This data set is supplied with the compiler and, being a partitioned data set, must reside on a direct access volume.

**Lines 11400 - 11600** The FILE1, FILE2 and FILE3 DD cards specify work files. These files are used for interphase communication. The device may be either direct access or magnetic tape. Space equivalent to approximately 60 tracks on a 2314 should be available for each DD card. The DCB is internally specified as

\[ DCB=(RECFM=F,LRECL=7200,BLKSIZE=7200) \]

**Line 11700** - The SYSUDUMP DD card specifies the location to which an abend dump is to be sent in the event that the compiler terminates abnormally.

**Lines 20000 - 21200** These lines define the data sets necessary to allow the external references in the
object program to be resolved and prepare the program for loading into memory. The JCL is the standard required by the OS/360 Linkage Editor plus that which is necessary for execution of the HALLINK processing program. Details of the use of individual DD cards may be found in the IBM Linkage-Editor and Loader manual. The only feature of this JCL which is peculiar to a HAL/S compilation is the form of the SYSLIB DD card and the addition of the STACKOBJ and TEMPLoad DD cards. These two additional DD cards should be used as specified in the procedures. They serve as intermediate work data sets for the HALLINK program.

The SYSPRINT DD card should always have a BLKSIZE parameter specified in the JCL. The BLKSIZE may be any value which is acceptable to the OS/360 Linkage Editor.

The HAL/S runtime library package must be the primary source of modules to resolve external references. Any references which cannot be resolved from the HAL/S library will be resolved from any libraries catenated to the first DD card.

If the user has any HAL/S program references to user-written non-HAL/S subprograms or to previously compiled HAL/S programs, it is at this point that the programs must be supplied to the linking process. This may be done through the use of a catenated DD card specifying a user library, or by direct user input of control cards to the link editor.

**Lines 30000 - 30200**

These cards specify the execution of the load module created in the LKED step. The parameter 'RUNPARM' causes any user-supplied runtime parameter to be made available to the running program.

**Line 30300 -**

CHANNEL6 DD card is supplied in the cataloqed procedure because device 6 is commonly used as an output unit. It is also supplied because the HAL/S execution-time error handling system requires CHANNEL6 for message listings. The user may specify (via the RUNPARM keyword) that the runtime
error messages be sent to a different channel.

Line 30400 -

The HALDUMP DD card defines the data set to receive the DUMP outputs that result from user requests or abnormal termination. The data set attributes are internally defined to be

\[ 
DCB=(RECFM=FBA,LRECL=120,BLSIZE=7200) 
\]

Line 30500 -

The HALSYMB DD card defines the partitioned data set which contains the DUMP/TRACE mapping file for the program being executed. In the case of the prototype procedure, this is the temporary data set created by the compilation step as OUTPUT5.

The discussion of the OUTPUT5 DD card above mentions the use of a permanent library of mapping files. Such a library would be identified in the HALSYMB DD card.

Line 30600 -

The CHANNEL5 DD card has been supplied in the catalogued procedures because it is common practice to assume the device 5 will be used for input. The use of the DDNAME parameter on the CHANNEL5 DD card allows the user to specify either

```
//GO.CHANNEL5 DD <parameters>
```

or

```
//GO.SYSIN DD <parameters>
```

to define the run-time input.

Line 30700 -

The SYSUDUMP DD card specifies the location of a hexadecimal memory dump in the event of a program abnormal termination.

**Other User-Defined Execution-Time JCL**

The user may require additional execution time JCL to service C-6
his program's requirements:

Additional HAL/S data sets (other than the CHANNEL5 and CHANNEL6 supplied in the procedure). These are data sets with names CHANNELn where n is in the range 0 to 9. These data sets are referenced by HAL/S I/O statements specifying the proper n as their device number (e.g. WRITE(8) references DD card CHANNEL8).

The HAL/S I/O routines make some assumptions about the characteristics of the data sets which are allocated to the various CHANNELn DD cards. These assumptions, along with a description of devices and organizations supported by HAL/S are detailed below.

The user may supply any, all, or none of the DCB attributes for CHANNELn DD cards. The record formats which are acceptable to HAL/S are:

F FB FA FBA FBSA
V VB VA VBA
U UA

Machine carriage control is acceptable but subject to interpretation described below.

General Rules Used by HAL/S to Create DCB Attributes

a. BLKSIZE not supplied (by JCL or data set). The maximum block size of the particular device is found. The BLKSIZE is set to the largest multiple of LRECL which is less than or equal to this block size. Note that for tapes this maximum size is 32767 bytes which would require a sizeable buffer area to be taken out of main storage.

b. LRECL not supplied. For PRINT files the LRECL defaults to 133. For non-PRINT files the default is 80.

c. RECFM not supplied. For PRINT files the RECFM defaults to FBA. For non-PRINT files the default is FB.

General Rules Governing HAL/S Sequential Output

a. If the mode is PRINT and the DCB RECFM specifies "A", carriage control characters will be automatically
generated. For "M" on PAGED devices, it is the user's responsibility to see that the first character of each output line has the proper control characters. For UNPAGED devices, control characters are not allowed.

b. Variable length record specification will cause records to be written in variable format. However, the records will actually be all the same length (LRECL).

c. Format U records (undefined record format) will be written in the proper form, but all records will be the same length (LRECL).

General Rules Governing HAL/S Sequential Input

a. Carriage control characters encountered during input will be available to the programmer; i.e. scanning of the input will begin at the carriage control character.

b. Variable length records may be read. The 4 byte descriptor field of each record will not be available to the programmer. The effective length of a variable record will be "length read minus 4" (subject to further modification due to carriage control).
The Catalogued Procedure HALSCL

//HALSCL PROC OPTION
//HAL EXEC PGM=MONITOR,REGION=300K,TIME=1,
// PARM='S O P T I O N '
//STEPLIB DD DISP=SHR,DSN=HALS360.MONITOR
//PROGRAM DD DISP=SHR,DSN=HALS360.PHASE1
// DD DISP=SHR,DSN=HALS360.PHASE2
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA, LRECL=133, BLKSIZE=3458)
//LISTING2 DD SYSOUT=A,DCB=(RECFM=FBA, LRECL=133, BLKSIZE=3458)
//OUTPUT3 DD UNIT=SYSDA,DISP=(MOD, PASS), SPACE=(CYL, (1, 1)),
// ECH=(RECFM=FB, LRECL=80, BLKSIZE=400),
// DSN=6&HALOBJ
//OUTPUT5 DD DISP=(MOD, PASS), DSN=6&HALSYM3, SPACE=(TRK, (2, 2, 1)),
// DBC=(RECFM=FB, LRECL=40, BLKSIZE=400),UNIT=SYSDA
//ERROR DD DISP=SHR,DSN=HALS360.ESE08LIB
//FILE1 DD UNIT=SYSDA,SPACE=(CYL, (3))
//FILE2 DD UNIT=SYSDA,SPACE=(CYL, (3))
//FILE3 DD UNIT=SYSDA,SPACE=(CYL, (3))
//SYSDUMP DD SYSOUT=A
//LKD EXEC PGM=HALLINK, REGION=100K, PARM=(LIST, MAP),
// COND=(0, LT, HAL)
//STEPLIB DD DISP=SHR,DSN=HALS360.MONITOR
//SYSPPINT DD SYSOUT=A,DCB=BLKSIZE=1573
//SYSLIB DD DSN=HALS360.FUNTIB,DISP=SHR
//SYSLIN DD DISP=(OLD, DELETE), DSN=6&HALOBJ
// DD DDNAME=SYSLN
//SYSLMOD DD DSN=6&HALMOD(GO),DISP=(, PASS),UNIT=SYSDA,
// SPACE=(CYL, (1, 1, 1))
//SYSLIB DD SPACE=(CYL, (1, 1)),UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD))
//STACKOBJ DD SPACE=(TRK, (5, 10)),UNIT=SYSDA
//TEMPLOAD DD DISP=(NEW, PASS), DSN=6&TEMP(TEMPNAME),
// SPACE=(TRK, (20, 20, 1)),UNIT=SYSDA
The Catalogued Procedure HALSC

//HALSC   PROC OPTION=
//HAL    EXEC PGM=MONITOR, REGION=300K, TIME=1,
//       PARM='&OPTION'
//STEPLIB DD DISP=SHR,DSN=HALS000.MONITOR
//PROGRAM DD DISP=SHR,DSN=HALS000.PHASE1
//       DD DISP=SHR,DSN=HALS000.PHASE2
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA, LRECL=133, LRECL=6553)
//LISTING DD SYSOUT=A,DCB=(RECFM=FBA, LRECL=133, LRECL=6553)
//OUTPUT3 DD UNIT=SYSDA,DISP=(MOD,PASS),SPACE=(CYL,(1,1)),
//       DCL=(RECFM=FB, LRECL=80, PLSIZE=400),
//SYSLIB DD SYSOUT=A
//OUTPUT5 DD DISP=(MOD,PASS),DSN=SYS000.LIBRARY,SPACE=(TPK,(2,2,1)),
//       DCL=(RECFM=FB, LRECL=40, PLSIZE=400),UNIT=SYSDA
//ERROR DD DISP=SHR,DSN=HALS000.ERRORLIB
//FILE1 DD UNIT=SYSDA,SPACE=(CYL,(3))
//FILE2 DD UNIT=SYSDA,SPACE=(CYL,(3))
//FILE3 DD UNIT=SYSDA,SPACE=(CYL,(3))
//SYSLIB   DD SYSOUT=A
The Catalogued Procedure HALSL

//HALSL PROC
//LISP EXEC PGM=HALLINK,REGION=100K,PARM=(LIST,HAP) 00010000
//STEP 1 DD DISP=SHR,DCN=HALC300,MONITOR 00020000
//SYSPUT DD SYSCNT=A,DDN=SYSMKSIZE=1573 00030000
//SYSLIB DD DSNAME=HALC300,RECL=LIB,DISP=SHR 00040000
//SYSLIB DD DFPNAME=SYSLIB 00050000
//SYSLINK DD DSNAME=HALLink,COPY=(0),DISP=(,PASS),UNIT=SYSDA, 00060000
// SPACE=(CYL,(1,1,1)) 00070000
//SYSLINK DD SPACE=(CYL,(1,1)),UNIT=(SYSDA,SEP=(SYSLIB,SYSLINK)) 00080000
//STACK DD SPACE=(TRK,(5,10)),UNIT=SYSDA 00090000
//TEMPLOAD DD DISP=(NEW,PASS),DSNAME=SYSTEM(TEMPSPACE), 00100000
// SPACE=(TRK,(20,20,1)),UNIT=SYSDA 00110000

00120000
The Catalogued Procedure \texttt{HALSLG}

```plaintext
//HALSLG PROC RUNPARM=
//LKEEP EXEC PC=HALLINK,REGION=100K,PAP=(LIST,PAP)
//STEPLIB DD DISP=SHR,DCB=HALS350,MONITOR
//SYSPRINT DD SYSOUT=A,DCB=DSLETSIZE=1573
//SYSLIB DD DSNAME=HALS350,REPLIB,DISP=SIR
//SYSLIB DD DDNAME=SYSIN
//SYS FORM DD DSNAME=HALSLDFORM,DISP=(,PASS),UNIT=SYSDA,
// SPACE=(CYL,(1,1,1))
//SYSUT1 DD SPACE=(CYL,(1,1,1)),UNIT=(SYSDA,SEP=(SYSLIB,SYSLIB))
//STACKDR DD SPACE=(TRK,(5,19)),UNIT=SYSDA
//TEMPLOAD DD DISP=(CSD PASS),DCB=DDNAME(TEMPNAME),
// SPACE=(TRK,(20,10,1)),UNIT=SYSDA
//GO EXEC PGM=\texttt{LKEEP},SYSLIB,REGION=100K,
// COMP=(4,LT,LKEEP)
//PART='DADHPARH'
//CHANNEL6 DD SYSOUT=A
//HALPSLP DD SYSOUT=A
//HALSYSDD DD DUMMY
//CHANNELS DD DDNAME=SYSIN
//SYSUDUMP DD SYSOUT=A
```

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

PROC OPTION='RUNPARM=
//HAL EXEC PGM=MONITOR, REGION=300K, TIME=1,
// PARM='6 OPTION'
//STEPLIB DD DISP=SHF, DSN=HALS360.MONITOR
//PROGRAM DD DISP=SHF, DSN=HALS360.PHASE1
// DD DISP=SHF, DSN=HALS360.PHASE2
//SYSPRINT DD SYSPUT=A, DBC=(RECFM=FBA, LRECL=133, BLKSIZE=3456)
//LISTING DD SYSPUT=A, DBC=(RECFM=FBA, LRECL=133, BLKSIZE=3456)
//OUTPUT DD UNIT=SYSDA, DISP=(MOD, PASS), SPACE=(CYL, (1, 1)),
// ECB=(RECFM=FIB, LRECL=80, BLKSIZE=400),
// DSN=6EHALOJB
//OUTPUT DD DISP=(MOD, PASS), DSN=6EHALSYMB, SPACE=(TRK, (2, 2, 1)),
// ECB=(RECFM=FIB, LRECL=80, BLKSIZE=400), UNIT=SYSDA
//ERROR DD DISP=SHF, DSN=HALS360.ERRLIB
//FILE DD UNIT=SYSDA, SPACE=(CYL, (3))
//FILE2 DD UNIT=SYSDA, SPACE=(CYL, (3))
//FILE3 DD UNIT=SYSDA, SPACE=(CYL, (3))
//SYSUDUMP DD SYSPUT=A
//LKD EXEC PGM=HALLINK, REGION=100K, PARM='LIST, MAP',
// COND=(0, LT, HAL)
//STEPLIB DD DISP=SHF, DSN=HALS360.MONITOR
//SYSPRINT DD SYSPUT=A, DBC=BLKSIZE=1573
//SYSLIB DD DSN=HALS360.RUNLIB, DISP=SHF
//SYSLIN DD DISP=(OLD, DELETE), DSN=6EHALOJB
//SYSMCLDD DD DSN=6EHALMOD(GO), DISP=(PASS), UNIT=SYSDA,
// SPACE=(CYL, (1, 1, 1))
//SYSUT1 DD SPACE=(CYL, (1, 1, 1)), UNIT=(SYSDA, SEP=(SYSLIN, SYSMCLD))
//STACKOBJ DD SPACE=(TRK, (5, 10)), UNIT=SYSDA
//TEMPLOAD DD DISP=(NEW, PASS), DSN=6ESTEMP(TEMPNAME),
// SPACE=(TRK, (20, 20, 1)), UNIT=SYSDA
//GO EXEC PGM=XLKD, SYSMCLD, REGION=100K,
// COND=((0, LT, HAL), (4, LT, XLKD)),
// PARM='6 RUNPARM'
//CHANNEL6 DD SYSPUT=A
//HALDUMP DD SYSPUT=A
//HALSYMDD DD DSN=6EHALSYMB, DISP=(OLD, DELETE)
//CHANNEL5 DD DDNAME=SYSDA
//SYSUDUMP DD SYSPUT=A
Appendix D.

Compile Time Error Messages

A complete list of compile time error messages is presented here. The first table gives the mnemonic naming scheme used to identify the class-subclass structure of the Phase I error messages. The complete list of Phase I errors are presented next. Phase II errors are listed last. These errors do not have a mnemonic naming scheme and are simply listed with their severities. The occurrences of double question marks (??) in the text of the messages listed here indicate positions at which text specific to each actual error will be inserted (e.g. a variable name may be inserted to make a clear identification of the error source).
Error Classifications

Note: "b" denotes a blank.

CLASS A: ASSIGNMENT STATEMENTS
A APPEAR ASSIGNMENT
V COMPLEX VARIABLE ASSIGNMENT
b MISCELLANEOUS ASSIGNMENT

CLASS B: COMPILER TERMINATION
A VALIDATE BLOCK SIZE
N NAME SCOPE MISTAKES
S STACK SIZE LIMITATIONS
T TABLE SIZE LIMITATIONS
X COMPILER ERRORS
b MISCELLANEOUS

CLASS C: COMPARISONS
b GENERAL COMPARISONS

CLASS D: DECLARATION ERRORS
A ATTRIBUTE LIST
C STORAGE CLASS ATTRIBUTE
D DIMENSION
F FUNCTION DECLARATION
I INITIALIZATION
L LOCKING ATTRIBUTE
Q STRUCTURE TEMPLATE TREE ORGANIZATION
S FACTORED/UNFACTORED SPECIFICATION
T TYPE SPECIFICATION
U UNDECLARED DATA
b MISCELLANEOUS
CLASS E: EXPRESSIONS

A  ARRAYNESS
B  BIT STRING EXPRESSIONS
C  CROSS PRODUCT
D  DCT PRODUCT
L  LIST EXPRESSIONS
M  MATRIX EXPRESSIONS
O  OUTER PRODUCT
V  VECTOR EXPRESSIONS
b  MISCELLANEOUS EXPRESSIONS

CLASS F: FORMAL PARAMETERS & ARGUMENTS

D  DIMENSION AGREEMENT
N  NUMBER OF ARGUMENTS
S  SUBBIT ARGUMENTS
T  TYPE AGREEMENT

CLASS G: STATEMENT GROUPINGS (DO GROUPS)

B  BIT TYPE CONTROL EXPRESSION
C  CONTROL EXPRESSION
E  EXIT/REPEAT STATEMENTS
L  END LABEL
V  CONTROL VARIABLE

CLASS I: IDENTIFIERS

L  LENGTH
R  REPLACED IDENTIFIERS
S  QUALIFIED STRUCTURE NAMES

CLASS L: LITERALS

B  BIT STRING
C  CONVERSION TO INTERNAL FORMS
F  FORMAT OF ARITHMETIC LITERALS
S  CHARACTER STRING

CLASS M: MULTILINE FORMAT

C  OVERPUNCH CONTEXT
F  E-LINE
O  OVERPUNCH USE
S  S-LINE
b  COMMENTS
CLASS P: PROGRAM CONTROL & INTERNAL CONSISTANCE
C COMPOOL BLOCKS
D DATA DEFINITION
E EXTERNAL TEMPLATES
F FUNCTION RETURN EXPRESSIONS
L LABELS
M MULTIPLE DEFINITIONS
P BLOCK DEFINITION
S PROCEDURE/FUNCTION TEMPLATES
T TASK DEFINITIONS
U CALLS FROM UPDATE BLOCKS
D MISCELLANEOUS

CLASS Q: SHAPING FUNCTIONS
A ARRAYNESS
D DIMENSION INFORMATION
S SUBSCRIPTS
X ARGUMENT TYPE

CLASS R: REAL TIME STATEMENTS
E ON/SEND ERROR STATEMENTS
T TIMING EXPRESSIONS
U UPDATE BLOCKS

CLASS S: SUBSCRIPT USAGE
C SUBSCRIPT COUNT
P PUNCTUATION
Q PRECISION QUALIFIER
R RANGE OF SUBSCRIPT VALUES
S STRUCTURE SUBSCRIPTING
T SUBSCRIPT TYPE
V VALIDITY OF USAGE

CLASS T: I/O STATEMENTS
C CONTROL
D DEVICE NUMBER
b MISCELLANEOUS

CLASS U: UPDATE BLOCKS
I IDENTIFIER USAGE
P PROGPHA BLOCKS
T I/O
CLASS V: COMPIL-TIME EVALUATIONS

A  ARITHMETIC OPERATIONS
C  CATENATION OPERATIONS
D  UNDEFINED VARIABLES
E  UNCOMPUTABLE EXPRESSIONS
F  FUNCTION EVALUATION

CLASS X: IMPLEMENTATION DEPENDENT FEATURES

D  DEVICE DIRECTIVE
I  INCLUDE DIRECTIVE
U  UNKNOWN OR INVALID DIRECTIVE
ERROR MESSAGES FOR MAJOR CLASSIFICATION A

CLASSIFICATION "A" ERRORS ARE RELATED TO ASSIGNMENT STATEMENTS

AA1 - SEVERITY 1
ARRAYNESS OF LEFT HAND SIDE OF ASSIGNMENT DOES NOT MATCH THAT OF RIGHT HAND SIDE

AA2 - SEVERITY 1
ARRAYNESS OF ?? IS INCONSISTENT WITH THAT OF OTHER LEFT HAND SIDE VARIABLES

AA3 - SEVERITY 1
ARRAYNESS OF ?? DISAGREES WITH ARRAYNESS OF ITS SUBSCRIPTING

AV1 - SEVERITY 1
TYPE OF RIGHT HAND SIDE EXPRESSION IS ILLEGAL FOR ASSIGNMENT INTO ??

AV2 - SEVERITY 1
MATRIX DIMENSIONS DISAGREE ACROSS ASSIGNMENT

AV3 - SEVERITY 1
VECTOR LENGTHS DISAGREE ACROSS ASSIGNMENT

AV4 - SEVERITY 1
TREE ORGANIZATIONS DO NOT MATCH ACROSS ASSIGNMENT

A1 - SEVERITY 1
ILLEGAL ASSIGNMENT TO CONSTANT OR PARAMETER ??
ERROR MESSAGES FOR MAJOR CLASSIFICATION B
CLASSIFICATION "B" ERRORS RESULT FROM ABORTIVE COMPILER FAILURES

BB1  -SEVERITY 2
     INTERMEDIATE CODE STORAGE OVERFLOW: ERROR SCAN CONTINUING

BN1  -SEVERITY 3
     MAX NAME SCOPE NESTING DEPTH EXCEEDED

BS1  -SEVERITY 1
     MAXIMUM DEPTH OF DO...END GROUP NESTING EXCEEDED

BS2  -SEVERITY 2
     INDIRECT PARSE STACK SIZE EXCEEDED

BS3  -SEVERITY 3
     PARSE STACK OVERFLOW

BS4  -SEVERITY 2
     CURRENT ARRAYNESS STACK SIZE EXCEEDED

BS5  -SEVERITY 1
     MAXIMUM FUNCTION NESTING DEPTH EXCEEDED

BT1  -SEVERITY 3
     SYMBOL TABLE OVERFLOW

BT3  -SEVERITY 3
     LITERAL TABLE DATA OVERFLOW

BT4  -SEVERITY 3
     LITERAL TABLE STRING OVERFLOW

BT5  -SEVERITY 3
     MACRO TABLE OVERFLOW

BT7  -SEVERITY 2
     INITIAL LIST STORAGE CAPACITY EXCEEDED

BX1  -SEVERITY 2
     SYT_CLASS = 0 FOR ??

BX2  -SEVERITY 2
     FUNC_TOKEN = 0

BX4  -SEVERITY 3
     TOO MANY BUILT-IN FUNCTIONS

BX5  -SEVERITY 2
     EXT_ARRAY OVERFLOW

B1   -SEVERITY 3
     INSUFFICIENT CORE AVAILABLE
ERROR MESSAGES FOR MAJOR CLASSIFICATION C
CLASSIFICATION "C" ERRORS DEAL WITH COMPARISONS

C1  -SEVERITY 1
     ?? COMPARISONS MAY ONLY BE = OR ≠

C2  -SEVERITY 1
     ARRAYED COMPARISONS ARE RESTRICTED TO = OR ≠

C3  -SEVERITY 1
     TREE ORGANIZATIONS OF STRUCTURES COMPARED DO NOT MATCH
ERROR MESSAGES FOR MAJOR CLASSIFICATION D
CLASSIFICATION "D" ERRORS ARE RELATED TO DATA DECLARATIONS

DA0  -SEVERITY 1
CONFLICTING ATTRIBUTE SPECIFIED WITH THE LATCHED ATTRIBUTE

DA1  -SEVERITY 1
ILLEGAL ATTRIBUTE SPECIFIED FOR BIT DATA TYPE

DA10 -SEVERITY 1
ILLEGAL ATTRIBUTE SPECIFIED FOR STRUCTURE DATA TYPE

DA11 -SEVERITY 1
ILLEGAL ATTRIBUTE SPECIFIED IN PROGRAM/TASK DECLARATION

DA2  -SEVERITY 1
ILLEGAL ATTRIBUTE SPECIFIED FOR CHARACTER DATA TYPE

DA20 -SEVERITY 1
ILLEGAL ATTRIBUTE SPECIFIED FOR MINOR STRUCTURE ??

DA21 -SEVERITY 1
AN ARRAY SPECIFICATION IS NOT ALLOWED FOR THE MINOR STRUCTURE ??

DA22 -SEVERITY 1
NO ATTRIBUTES MAY BE SPECIFIED ON A NESTED STRUCTURE TEMPLATE REFERENCE

DA23 -SEVERITY 1
ILLEGAL ATTRIBUTE FOR THE STRUCTURE TERMINAL ??

DA3  -SEVERITY 1
ILLEGAL ATTRIBUTE SPECIFIED FOR MATRIX DATA TYPE

DA4  -SEVERITY 1
ILLEGAL ATTRIBUTE SPECIFIED FOR VECTOR DATA TYPE

DA5  -SEVERITY 1
ILLEGAL ATTRIBUTE FOR SCALAR DATA TYPE

DA6  -SEVERITY 1
ILLEGAL ATTRIBUTE SPECIFIED FOR INTEGER DATA TYPE

DA9  -SEVERITY 1
ILLEGAL ATTRIBUTE SPECIFIED FOR EVENT DATA TYPE

DC1  -SEVERITY 1
ILLEGAL STATIC/AUTO SPECIFICATION FOR ??

DC2  -SEVERITY 1
ILLEGAL STATIC OR AUTO SPEC AT COMPOOL OR PROGRAM LEVEL FOR ??

DC3  -SEVERITY 1
ILLEGAL STATIC OR AUTO SPEC FOR THE PARAMETER ??

DC4  -SEVERITY 1
FACTORED AND NON-FACTORED TYPE SPECIFICATION FOR ?? DISAGREE
THE NON-FACTORED TYPE SPECIFICATION WILL BE USED

DD1  -SEVERITY 1
ILLEGAL ARRAY DIMENSION SPECIFICATION
-SEVERITY 1
**MUST BE A PARAMETER FOR USE OF UNKNOWN ARRAY SIZE NOTATION**

-SEVERITY 1
**ILLEGAL STRUCTURE DIMENSION SPECIFICATION**

-SEVERITY 2
**TOO MANY DIMENSIONS IN ARRAY**

-SEVERITY 1
**INVALID MATRIX DIMENSION SPECIFICATION; A DIMENSION OF 3 IS ASSUMED**

-SEVERITY 1
**INVALID VECTOR LENGTH SPECIFICATION; A 3-VECTOR IS ASSUMED**

-SEVERITY 1
**ONLY SINGLE DIMENSION ARRAYS MAY USE THE * TO DENOTE UNKNOWN LENGTH**

-SEVERITY 1
**A * MAY NOT BE USED TO SPECIFY VECTOR LENGTH; A 3-VECTOR IS ASSUMED**

-SEVERITY 1
**A * MAY NOT BE USED TO SPECIFY A MATRIX DIMENSION; A DIMENSION OF 3 IS ASSUMED**

-SEVERITY 1
**THE FUNCTION ?? MAY NOT BE DECLARED IN A COMPOOL**

-SEVERITY 1
**ILLEGAL ATTRIBUTE FOR THE FUNCTION ??**

-SEVERITY 1
**THE FUNCTION ?? MAY NOT HAVE AN INITIAL/CONSTANT SPECIFICATION**

-SEVERITY 1
**REPEAT FACTOR IN INITIALIZATION HAS NO LEGAL VALUE COMPUTABLE AT COMPILE TIME**

-SEVERITY 1
**TOO MANY ELEMENTS SUPPLIED IN INITIAL LIST FOR ??**

-SEVERITY 1
**IMPLIED NUMBER OF ELEMENTS IN INITIAL LIST EXCEEDS COMPILER LIMIT**

-SEVERITY 1
**EXPRESSION IN INITIAL LIST IS NOT COMPUTABLE AT COMPILE TIME**

-SEVERITY 1
**INITIALIZATION OF ?? HAS ILLEGAL TERMINATING * : NUMBER OF INITIAL VALUES MATCHES TOTAL NUMBER OF ELEMENTS**

-SEVERITY 1
**TOO FEW ELEMENTS SUPPLIED IN INITIAL LIST FOR ??**

-SEVERITY 1
**ILLEGALLY-TYPED INITIAL VALUE--INITIALIZATION OF ?? EXPECTS A VALUE OF CHARACTER TYPE**
DD10  -SEVERITY 1
      ** MUST BE A PARAMETER FOR USE OF UNKNOWN ARRAY SIZE NOTATION

DD11  -SEVERITY 1
      ILLEGAL STRUCTURE DIMENSION SPECIFICATION

DD3   -SEVERITY 2
      TOO MANY DIMENSIONS IN ARRAY

DD4   -SEVERITY 1
      INVALID MATRIX DIMENSION SPECIFICATION; A DIMENSION OF 3 IS ASSUMED

DD5   -SEVERITY 1
      INVALID VECTOR LENGTH SPECIFICATION; A 3-VECTOR IS ASSUMED

DD6   -SEVERITY 1
      ONLY SINGLE DIMENSION ARAYS MAY USE THE * TO DENOTE UNKNOW LENGTH

DD7   -SEVERITY 1
      A * MAY NOT BE USED TO SPECIFY VECTOR LENGTH; A 3-VECTOR IS ASSUMED

DD8   -SEVERITY 1
      ** MUST BE A PARAMETER FOR USE OF UNKNOWN STRUCTURE COPY NOTATION

DD9   -SEVERITY 1
      A * MAY NOT BE USED TO SPECIFY A MATRIX DIMENSION; A DIMENSION OF 3 IS ASSUMED

DF1   -SEVERITY 1
      THE FUNCTION ** MAY NOT BE DECLARED IN A COMPOOL

DF2   -SEVERITY 1
      ILLEGAL ATTRIBUTE FOR THE FUNCTION **

DF3   -SEVERITY 1
      THE FUNCTION ** MAY NOT HAVE AN INITIAL/CONSTANT SPECIFICATION

DI1   -SEVERITY 1
      REPEAT FACTOR IN INITIALIZATION HAS NO LEGAL VALUE COMPUTABLE AT
      COMPILCE TIME

DI10  -SEVERITY 1
      TOO MANY ELEMENTS SUPPLIED IN INITIAL LIST FOR **

DI2   -SEVERITY 1
      IMPLIED NUMBER OF ELEMENTS IN INITIAL LIST EXCEEDS COMPILER LIMIT

DI3   -SEVERITY 1
      EXPRESSION IN INITIAL LIST IS NOT COMPUTABLE AT COMPILCE TIME

DI4   -SEVERITY 1
      INITIALIZATION OF ?? HAS ILLEGAL TERMINATING *:
      NUMBER OF INITIAL VALUES MATCHES TOTAL NUMBER OF ELEMENTS

DI5   -SEVERITY 1
      TOO FEW ELEMENTS SUPPLIED IN INITIAL LIST FOR ??

DI6   -SEVERITY 1
      ILLEGALLY-TYPED INITIAL VALUE--INITIALIZATION OF ?? EXPECTS
      A VALUE OF CHARACTER TYPE
-SEVERITY 1
ILLEGALLY-TYPED INITIAL VALUE--INITIALIZATION OF ?? EXPECTS A VALUE OF BIT TYPE

-SEVERITY 1
ILLEGALLY-TYPED INITIAL VALUE--INITIALIZATION OF ?? EXPECTS A VALUE OF INTEGER OR SCALAR TYPE

-SEVERITY 1
THE DECLARATION OF ?? HAS BOTH FACTORED AND UNFACTORED INITIAL/CONSTANT ATTRIBUTES; THE UNFACTORED ATTRIBUTES WILL BE USED

-SEVERITY 1
ILLEGAL TO SPECIFY LOCKING ATTRIBUTE AT CURRENT NEST LEVEL

-SEVERITY 1
THE LOCKED ATTRIBUTE MAY NOT BE USED IN CONJUNCTION WITH THE CONSTANT ATTRIBUTE

-SEVERITY 1
FIRST NODE DECLARED IN TEMPLATE MUST BE AT LEVEL 1

-SEVERITY 1
ILLEGAL SEQUENCE OF LEVEL NUMBERS IN TEMPLATE

-SEVERITY 1
STRUCTURE FUNCTION BEARS SAME NAME AS TEMPLATE SPECIFYING ITS TREE ORGANIZATION

-SEVERITY 1
STRUCTURE ?? CANNOT BE MADE UNQUALIFIED - SPECIFIED OR INDIRECTLY REFERENCED STRUCTURE TEMPLATE IS ALREADY USED BY AN UNQUALIFIED STRUCTURE

-SEVERITY 1
STRUCTURE ?? CANNOT BE MADE UNQUALIFIED - SPECIFIED OR INDIRECTLY REFERENCED STRUCTURE TEMPLATE IS NOT DECLARED IN SAME NAME SCOPE

-SEVERITY 1
INVALID BIT-LENGTH SPECIFICATION

-SEVERITY 1
FACTORED AND NON-FACTORED STRUCTURE TEMPLATE REFERENCES DISAGREE; NON-FACTORED REFERENCE WILL BE USED

-SEVERITY 1
INPUT/ASSIGN PARAMETERS OF CHARACTER TYPE CAN ONLY BE GIVEN A * LENGTH SPECIFICATION

-SEVERITY 1
INVALID CHAR-LENGTH SPECIFICATION

-SEVERITY 1
A * IS AN ILLEGAL CHARACTER LENGTH SPECIFICATION; A LENGTH OF 8 IS ASSUMED

-SEVERITY 1
A * IS AN ILLEGAL BIT LENGTH SPECIFICATION; A LENGTH OF 1 IS ASSUMED

-SEVERITY 1
FACTORED AND NON-FACTORED BIT SIZE SPECIFICATION FOR ?? DISAGREE THE NON-FACTORED SPECIFICATION WILL BE USED

-SEVERITY 1
FACTORED AND NON-FACTORED CHARACTER LENGTH SPECIFICATION FOR ?? DISAGREE THE NON-FACTORED SPECIFICATION WILL BE USED
ERROR MESSAGES FOR MAJOR CLASSIFICATION E
CLASSIFICATION "E" ERRORS DEAL WITH EXPRESSIONS

EA1
- SEVERITY 1
ARRAYNESS OF ?? IS INCONSISTENT WITH CURRENT ARRAYNESS OF EXPRESSION

EB1
- SEVERITY 0
RESULT OF BIT CATENATION WILL BE LEFT TRUNCATED TO MAXIMUM BIT LENGTH

EB2
- SEVERITY 1
LABEL ?? USED IN BIT OR EVENT EXPRESSION WAS NOT A PROGRAM OR TASK EVENT

EC1
- SEVERITY 1
CROSS PRODUCT MUST BE BETWEEN THREE DIMENSIONAL VECTORS

EC2
- SEVERITY 1
CROSS PRODUCT * USED WITHOUT A VECTOR AFTER IT

EC3
- SEVERITY 1
CROSS PRODUCT * USED WITHOUT A VECTOR BEFORE IT

ED1
- SEVERITY 1
DOT PRODUCT . USED WITHOUT A VECTOR AFTER IT

ED2
- SEVERITY 1
DOT PRODUCT . USED WITHOUT A VECTOR BEFORE IT

EL1
- SEVERITY 1
ONLY ARITHMETIC CONVERSION FUNCTIONS MAY POSSESS ARGUMENTS WITH REPEAT FACTORS

EL2
- SEVERITY 1
REPEITION FACTOR OF EXPRESSION MUST BE AN UNARRAYED INTEGER OR SCALAR
EXPRESSIBLE COMPUTABLE AT COMPILE TIME

EM1
- SEVERITY 1
DIMENSIONS OF MATRIX OPERANDS IN EXPRESSION DISAGREE

EM2
- SEVERITY 1
MATRIX ARITHMETIC TYPE CANNOT BE CONVERTED TO A CHARACTER STRING

EM3
- SEVERITY 1
MATRIX-MATRIX MULTIPLICATION DIMENSION DISAGREEMENT

EM4
- SEVERITY 1
INVERSE OF NON-SQUARE MATRIX ATTEMPTED

EO1
- SEVERITY 1
ILLEGAL PRODUCT: OUTER PRODUCT TIMES A VECTOR

EO2
- SEVERITY 1
A PRODUCT INVOLVING BOTH CROSS AND OUTER PRODUCTS IS INDICATED.
USE MORE PARENTHESIS.

EO3
- SEVERITY 1
A PRODUCT INVOLVING BOTH DOT AND OUTER PRODUCTS IS INDICATED.
USE MORE PARENTHESIS.

EV1
- SEVERITY 1
LENGTHS OF VECTOR OPERANDS IN EXPRESSION DISAGREE
EV2  -SEVERITY 1
   MATRIX-VECTOR MULTIPLICATION DIMENSION DISAGREEMENT

EV3  -SEVERITY 1
   VECTOR-MATRIX MULTIPLICATION DIMENSION DISAGREEMENT

EV4  -SEVERITY 1
   VECTOR MAY NOT HAVE AN EXPONENT

EV5  -SEVERITY 1
   VECTOR ARITHMETIC TYPE CANNOT BE CONVERTED TO A CHARACTER STRING

E1   -SEVERITY 1
   DIVIDOPS MAY ONLY BE OF INTEGER OR SCALAR TYPE

E2   -SEVERITY 1
   MATRIX MUST HAVE AN EXPONENT OF INTEGER TYPE KNOWN AT COMPILE TIME

E3   -SEVERITY 1
   EXPONENT MUST BE A SINGLE VALUED QUANTITY

E4   -SEVERITY 1
   DOT OR CROSS PRODUCT SYMBOL (•, OR ×) USED IN A PRODUCT NOT INVOLVING VECTOPS

E6   -SEVERITY 1
   INCOMPATIBLE ARITHMETIC OPERAND TYPES IN EXPRESSION
ERROR MESSAGES FOR MAJOR CLASSIFICATION F
CLASSIFICATION "F" ERRORS DEAL WITH FORMAL PARAMETERS AND ARGUMENTS

FD1 - SEVERITY 1
MATRIX DIMENSIONS OF ARGUMENT AND CORRESPONDING FORMAL PARAMETER DO NOT AGREE

FD2 - SEVERITY 1
VECTOR LENGTHS OF ARGUMENT AND CORRESPONDING FORMAL PARAMETER DO NOT AGREE

FD3 - SEVERITY 1
TREE ORGANIZATIONS OF STRUCTURE ARGUMENT AND CORRESPONDING FORMAL PARAMETER ARE NOT IDENTICAL

FD4 - SEVERITY 1
ARRAYNESS OF FUNCTION ARGUMENT DOES NOT MATCH CURRENT ARRAYNESS OF EXPRESSION CONTAINING THE INVOCATION

FD5 - SEVERITY 1
ARRAYNESS OF ARGUMENT AND CORRESPONDING FORMAL PARAMETER ARE NOT IDENTICAL

FD6 - SEVERITY 1
ARRAYNESS OF ARGUMENT OF ?? FUNCTION IS NOT A SQUARE MATRIX

FN1 - SEVERITY 1
FUNCTION ?? WAS INVOKED WITH TOO FEW ARGUMENTS

FN2 - SEVERITY 1
FUNCTION ?? WAS INVOKED WITH TOO MANY ARGUMENTS

FN3 - SEVERITY 2
?? WAS USED MORE THAN ONCE AS A PARAMETER

FN4 - SEVERITY 1
?? FUNCTION HAS INCORRECT NUMBER OF ARGUMENTS

FS1 - SEVERITY 1
AN ASSIGN ARGUMENT OF A PROCEDURE CALL MAY NOT BE A SUBBIT PSEUDO VARIABLE

FT1 - SEVERITY 1
TYPE OF FUNCTION ARGUMENT IS INCOMPATIBLE WITH TYPE OF CORRESPONDING FORMAL PARAMETER

FT2 - SEVERITY 1
?? FUNCTION HAS AN ARGUMENT OF INCORRECT TYPE

FT3 - SEVERITY 1
ILLEGAL TYPE FOR THE FUNCTION ??

FT4 - SEVERITY 1
THE SIZE SPECIFICATION FOR THE RETURN TYPE OF FUNCTION ?? DISAGREES WITH THE PREVIOUSLY DECLARED SIZE

FT5 - SEVERITY 1
FUNCTION ?? MAY NOT POSSESS MULTIPLE STRUCTURE COPIES

FT6 - SEVERITY 1
THE STRUCTURE TEMPLATE INDICATED IN THE TYPE SPECIFICATION OF FUNCTION ?? DISAGREES WITH THE TEMPLATE USED IN A PREVIOUS DECLARATION

FT7 - SEVERITY 1
CONFLICTING SINGLE/DOUBLE SPECIFICATION FOR THE FUNCTION ??
ERROR MESSAGES FOR MAJOR CLASSIFICATION G
CLASSIFICATION "G" ERRORS DEAL WITH STATEMENT GROUPINGS (DO STATEMENTS)

GB1 - SEVERITY 1
BIT EXPRESSION IN ?? CLAUSE MUST BE BOOLEAN

GC1 - SEVERITY 1
CONTROL EXPRESSION IN A DO CASE MUST BE OF UNARRAYED INTEGER OR SCALAR TYPE

GC2 - SEVERITY 1
BIT EXPRESSION IN WHILE OR UNTIL CLAUSE MAY NOT BE ARRAYED

GC3 - SEVERITY 1
CONTROL EXPRESSIONS IN A DO FOR MUST BE OF UNARRAYED INTEGER OR SCALAR TYPE

GE1 - SEVERITY 1
EXIT STATEMENT IS NOT IN AN ENCLOSING DO...END GROUP

GE2 - SEVERITY 1
REPEAT STATEMENT IS NOT IN AN ENCLOSING DO FOR...END OR DO WHILE/UNTIL...END GROUP

GE3 - SEVERITY 1
EXIT CAUSES ILLEGAL BRANCHING OUT OF CODE BLOCK DEFINITION

GE4 - SEVERITY 1
REPEAT CAUSES ILLEGAL BRANCHING OUT OF CODE BLOCK DEFINITION

GL1 - SEVERITY 1
LABEL AFTER END STATEMENT DOES NOT MATCH DO STATEMENT LABEL

GL2 - SEVERITY 1
LABEL IS THE DESTINATION OF A GO TO PROM OUTSIDE THE ENCLOSING DO...END GROUP

GL3 - SEVERITY 1
GO TO STATEMENT CAUSES A BRANCH INTO A DO...END GROUP

GV1 - SEVERITY 1
CONTROL VARIABLE IN A DO FOR MUST BE OF UNARRAYED INTEGER OR SCALAR TYPE
ERROR MESSAGES FOR MAJOR CLASSIFICATION I
CLASSIFICATION "I" ERRORS ARE RELATED TO IDENTIFIERS

IL1  -SEVERITY 1
     IDENTIFIER NAME MAY NOT END WITH AN UNDERSCORE CHARACTER

IL2  -SEVERITY 1
     NAME TOO LONG - TRUNCATED

IP1  -SEVERITY 1
     ILLEGAL REPLACEMENT FOR LOCAL NAME: ??

IR10 -SEVERITY 3
     MAXIMUM NUMBER OF PARAMETERS FOR SOURCE MACRO DEFINITION EXCEEDED

IR3  -SEVERITY 1
     MACRO EXPANSION TOO LONG

IR5  -SEVERITY 1
     DUPLICATE REPLACE FOR ??

IR6  -SEVERITY 1
     MACRO NAME ?? NOT DEFINED

IR7  -SEVERITY 2
     REPLACE PARAMETER STRING TOO LONG; REPLACE NOT PERFORMED

IR8  -SEVERITY 2
     INCORRECT NUMBER OF PARAMETERS FOR MACRO CALL; REPLACEMENT NOT PERFORMED

IR9  -SEVERITY 3
     MACRO EXPANSION STACK OVERFLOW; RECURSIVE DEFINITION LIKELY

IS1  -SEVERITY 1
     ILLEGAL CONSTRUCTION OF QUALIFIED STRUCTURE NAME
ERROR MESSAGES FOR MAJOR CLASSIFICATION L
CLASSIFICATION "L" ERRORS RELATE WITH LITERALS

LB1 - SEVERITY 1
BIT CONSTANTS MAY NOT BE LONGER THAN 32 BITS

LB2 - SEVERITY 1
DECIMAL BIT CONSTANT MUST SPECIFY OR IMPLY A REPETITION FACTOR OF 1

LB3 - SEVERITY 1
ILLEGAL DECIMAL STRING IN DECIMAL BIT CONSTANT

LB4 - SEVERITY 1
ILLEGAL CHARACTER IN DECIMAL BIT CONSTANT

LB5 - SEVERITY 1
ILLEGAL CHARACTER IN BINARY BIT CONSTANT

LB6 - SEVERITY 1
ILLEGAL CHARACTER IN OCTAL BIT CONSTANT

LB7 - SEVERITY 1
ILLEGAL CHARACTER IN HEXADECIMAL BIT CONSTANT

LB8 - SEVERITY 1
REPETITION FACTOR OF A BIT LITERAL MUST BE GREATER THAN ZERO

LC2 - SEVERITY 1
?? NOT EXPRESSES INTERNALLY

LF1 - SEVERITY 1
ILLEGAL NUMERIC LITERAL CONSTRUCTION

LF2 - SEVERITY 1
ONLY ONE DECIMAL POINT ALLOWED

LF3 - SEVERITY 1
TOO MANY SIGNIFICANT DIGITS - 74 ALLOWED

LF5 - SEVERITY 1
EXPOENT INDICATOR BUT NO EXPONENT DIGITS

LS1 - SEVERITY 0
CHARACTER STRING TOO LONG - TRUNCATED TO 255 CHARACTERS

LS2 - SEVERITY 1
REPETITION FACTOR OF A CHARACTER LITERAL IS NOT GREATER THAN ZERO
EPROP MESSAGES FOR MAJOR CLASSIFICATION M
CLASSIFICATION "M" ERRORS DEAL WITH MULTI-LINE FORMATS

MC1  - SEVERITY 1
      ILLEGAL CONTEXT FOR OVERPUNCH

MC2  - SEVERITY 1
      OVERPUNCH ILLEGAL ON FUNCTION NAMES

MC3  - SEVERITY 1
      OVERPUNCH ILLEGAL ON REPLACED NAME

MC4  - SEVERITY 0
      OVERPUNCH NOT VALID ON RESERVED WORD ??

MC5  - SEVERITY 0
      OVERPUNCH ILLEGAL ON PARAMETER ??

MC6  - SEVERITY 0
      OVERPUNCH ILLEGAL IN DECLARATION OF ??

ME1  - SEVERITY 3
      EXPONENT STRING OVERFLOW

ME2  - SEVERITY 1
      E-LINE CHARACTER MORE THAN ONE LINE ABOVE PRECEDING CHARACTER

ME3  - SEVERITY 1
      E-LINE OVERLAPS M-LINE

ME4  - SEVERITY 1
      OVERLAPPING E-LINE CHARACTERS

MO1  - SEVERITY 0
      OVERPUNCH ILLEGAL

MO2  - SEVERITY 1
      INVALID OVERPUNCH ON ??

MO3  - SEVERITY 0
      MULTIPLE OVERPUNCHES NOT VALID - FIRST ACCEPTED

MO4  - SEVERITY 1
      USER SUPPLIED OVERPUNCH CHARACTER NOT VALID - IGNORED

MS1  - SEVERITY 3
      SUBSCRIPT STRING OVERFLOW

MS2  - SEVERITY 1
      S-LINE CHARACTER MORE THAN ONE LINE LOWER THAN PRECEDING CHARACTER

MS3  - SEVERITY 1
      S-LINE OVERLAPS M-LINE

MS4  - SEVERITY 1
      OVERLAPPING S-LINE CHARACTERS

M1   - SEVERITY 0
      ILLEGAL CARD TYPE - CHANGED TO A COMMENT

M2   - SEVERITY 1
      INVALID SEQUENCE OF CARD TYPES

M3   - SEVERITY 0
      COMMENT LONGER THAN 256 CHARACTERS - HAS BEEN TRUNCATED
ERROR MESSAGES FOR MAJOR CLASSIFICATION P
CLASSIFICATION "P" ERRORS INDICATE FLOW CONTROL PROBLEMS

PC1  - SEVERITY 1
COMPOOL BLOCK CONTAINS STATEMENT(S) OTHER THAN DECLARATIONS

PC2  - SEVERITY 1
COMPOOL TEMPLATE CONTAINS STATEMENT(S) OTHER THAN DECLARATIONS

PD1  - SEVERITY 1
ILLEGAL DECLARATION FOR THE PARAMETER ??

PE1  - SEVERITY 1
EXTERNAL TEMPLATES MUST NOT APPEAR WITHIN A BLOCK DEFINITION

PE2  - SEVERITY 1
EXTERNAL TEMPLATES MUST NOT BE PLACED AFTER A BLOCK DEFINITION

PF1  - SEVERITY 1
RETURN FROM FUNCTION BLOCK MUST BE FOLLOWED BY AN EXPRESSION

PF2  - SEVERITY 1
RETURN MAY ONLY BE FOLLOWED BY AN EXPRESSION IN A FUNCTION BLOCK

PF3  - SEVERITY 1
EXPRESSION TO BE RETURNED MAY NOT POSSESS ARRAYNESS

PF4  - SEVERITY 1
ILLEGAL TYPE CONVERSION OF RETURNED EXPRESSION REQUIRED

PF5  - SEVERITY 1
MATRIX DIMENSIONS OF FUNCTION DISAGREE WITH THOSE OF RETURN EXPRESSION

PF6  - SEVERITY 1
VECTOR LENGTH OF FUNCTION DISAGREES WITH THAT OF RETURN EXPRESSION

PF7  - SEVERITY 1
TREE ORGANIZATION OF FUNCTION DOES NOT MATCH THAT OF RETURN EXPRESSION

PL1  - SEVERITY 1
THE FUNCTION ?? HAS BEEN DECLARED BUT NOT DEFINED

PL2  - SEVERITY 2
?? IS A DUPLICATE LABEL

PL3  - SEVERITY 1
LABEL ON CLOSE DOES NOT MATCH BLOCK DEFINITION LABEL: ??

PL4  - SEVERITY 1
FUNCTION LABEL CONFLICT

PL5  - SEVERITY 1
LABEL ?? IS NOT DEFINED WITHIN THE CURRENT SCOPE

PL6  - SEVERITY 1
THE PROCEDURE/TASK ?? HAS BEEN CALLED/SCHEDULED BUT NOT DEFINED

PM1  - SEVERITY 2
DUPLICATE DEFINITION FOR ??
PM3  - SEVERITY 1
     EARLIER DEFINITION OVERRIDDEN FOR ??

PM4  - SEVERITY 0
     OUTER DEFINITION OVERRIDDEN FOR ??

PP1  - SEVERITY 1
     A ?? DEFINITION MUST BE THE OUTERMOST BLOCK DEFINITION

PP2  - SEVERITY 1
     BLOCK DEFINITION IS NOT THE FIRST OUTERMOST BLOCK DEFINITION

PP3  - SEVERITY 1
     A ?? DEFINITION CANNOT BE AN OUTERMOST BLOCK DEFINITION

PP4  - SEVERITY 1
     NO BLOCK DEFINITIONS WERE ENCOUNTERED IN COMPILATION

PS1  - SEVERITY 1
     EXTERNAL PROCEDURE/FUNCTION TEMPLATE CONTAINS STATEMENT(S)
     OTHER THAN DECLARATIONS

PS2  - SEVERITY 1
     ONLY PROCEDURES OR FUNCTIONS MAY BE DESIGNATED ??

PS3  - SEVERITY 1
     ONLY PROGRAM, PROCEDURE AND FUNCTION BLOCKS MAY POSSESS THE ACCESS ATTRIBUTE

PT1  - SEVERITY 1
     TASK DEFINITIONS MAY ONLY APPEAR WITHIN AN IMMEDIATELY ENCLOSING
     PROGRAM DEFINITION

PU3  - SEVERITY 1
     INVOCATIONS IN AN UPDATE BLOCK OF PROCEDURES OR USER FUNCTIONS DEFINED
     OUTSIDE THE BLOCK ARE ILLEGAL

P1   - SEVERITY 1
     END-OF-FILE AT INVALID POINT IN SOURCE TEXT

P3   - SEVERITY 0
     BLOCK SUMMARY TABLE OVERFLOW

P4   - SEVERITY 2
     CONFLICTING USE OF ??

P5   - SEVERITY 1
     TOO MANY MACRO EXPANSIONS FOR ??

P6   - SEVERITY 0
     PROGRAM LAYOUT TABLE EXCEEDED

P8   - SEVERITY 1
     THE FOLLOWING SYMBOL IS SYNTACTICALLY ILLEGAL IN THE CONTEXT USED: ??
     ERROR RECOVERY MAY CAUSE SUBSEQUENT SPURIOUS ERRORS

D-21
ERROR MESSAGES FOR MAJOR CLASSIFICATION Q
CLASSIFICATION "Q" ERRORS DEAL WITH SHAPING FUNCTIONS

QA1 - SEVERITY 1
ARRAYNESS OF SINGLE ARGUMENT OF INTEGER/SCALAR CONVERSION FUNCTION DOES NOT MATCH THAT OF EXPRESSION CONTAINING FUNCTION

QA2 - SEVERITY 1
ARRAYNESS OF RESULT OF INTEGER/SCALAR CONVERSION FUNCTION IS UNCOMPUTABLE

QA3 - SEVERITY 1
SPECIFIED ARRAYNESS OF INTEGER/SCALAR CONVERSION FUNCTION IS INCONSISTENT WITH NUMBER OF DATA ELEMENTS SUPPLIED IN ARGUMENT LIST

QA4 - SEVERITY 1
ARRAYNESS OF RESULT OF INTEGER/SCALAR CONVERSION FUNCTION DOES NOT MATCH THAT OF EXPRESSION CONTAINING FUNCTION

QD1 - SEVERITY 1
DIMENSIONS OF VECTOR/MATRIX CONVERSION FUNCTION DO NOT AGREE WITH THE NUMBER OF DATA ELEMENTS SUPPLIED IN THE ARGUMENT LIST

QD2 - SEVERITY 1
BIT OR CHARACTER CONVERSION FUNCTION MAY ONLY HAVE ONE ARGUMENT

QS1 - SEVERITY 1
COLONS AND SEMICOLONS MAY NOT APPEAR IN SUBSCRIPT OF CONVERSION FUNCTIONS

QS10 - SEVERITY 1
BIT OR CHARACTER CONVERSION FUNCTION MAY ONLY HAVE ONE SUBSCRIPT

QS11 - SEVERITY 1
SUBBIT CONVERSION FUNCTION MAY ONLY HAVE ONE SUBSCRIPT

QS12 - SEVERITY 1
COLONS AND SEMICOLONS MAY NOT APPEAR IN THE SUBSCRIPT OF A SUBBIT PSEUDO-VARIABLE

QS13 - SEVERITY 1
SUBSCRIPT OF A SUBBIT PSEUDO-VARIABLE MAY NOT CONTAIN A PRECISION MODIFIER

QS2 - SEVERITY 1
MATRIX CONVERSION FUNCTION DOES NOT HAVE TWO SUBSCRIPTS

QS3 - SEVERITY 1
VECTOR CONVERSION FUNCTION DOES NOT HAVE ONE SUBSCRIPT

QS4 - SEVERITY 1
INTEGER OR SCALAR CONVERSION FUNCTION HAS MORE THAN MAXIMUM PERMITTED NUMBER OF SUBSCRIPTS

QS5 - SEVERITY 1
SUBSCRIPT OF ARITHMETIC CONVERSION FUNCTION IS NOT A SINGLE INDEX

QS6 - SEVERITY 1
SUBSCRIPT OF ARITHMETIC CONVERSION FUNCTION MAY NOT CONTAIN # VALUES

QS7 - SEVERITY 1
SUBSCRIPT OF ARITHMETIC CONVERSION FUNCTION MUST BE AN UNARRAYED INTEGER/SCALAR EXPRESSION COMPUTABLE AT COMPILTime
- SEVERITY 1
VALUE OF SUBSCRIPT OF ARITHMETIC CONVERSION FUNCTION LIES OUTSIDE LEGAL RANGE

- SEVERITY 1
SUBSCRIPT OF BIT OR CHARACTER CONVERSION FUNCTION MAY NOT CONTAIN A PRECISION QUALIFIER

- SEVERITY 1
CONVERSION FUNCTIONS MAY NOT HAVE ARGUMENTS OF STRUCTURE TYPE

- SEVERITY 1
MATRIX/VECTOR CONVERSION FUNCTIONS MAY NOT HAVE ARGUMENTS OF BIT TYPE

- SEVERITY 1
MATRIX/VECTOR CONVERSION FUNCTIONS MAY NOT HAVE ARGUMENTS OF CHARACTER TYPE

- SEVERITY 1
MATRIX OR VECTOR ARGUMENT IS ILLEGAL IN BIT OR CHARACTER CONVERSION FUNCTION

- SEVERITY 1
CHARACTER CONVERSION FUNCTION WITH RADIX DOES NOT HAVE ARGUMENT OF BIT TYPE

- SEVERITY 1
BIT CONVERSION FUNCTION WITH RADIX DOES NOT HAVE ARGUMENT OF CHARACTER TYPE

- SEVERITY 1
IN AN ASSIGNMENT CONTEXT THE ARGUMENT OF A SUBBIT PSEUDO-VARIABLE MAY NOT ITSELF BE A SUBBIT PSEUDO-VARIABLE

- SEVERITY 1
ARGUMENT OF ILLEGAL TYPE IN SUBBIT PSEUDO-VARIABLE
ERROR MESSAGES FOR MAJOR CLASSIFICATION R
CLASSIFICATION "R" ERRORS ARE RELATED TO REAL-TIME STATEMENT ERRORS

RE1 - SEVERITY 1
ILLEGAL FORM OR VALUE OF ON ERROR SUBSCRIPT

RE2 - SEVERITY 1
ILLEGAL FORM OR VALUE OF SEND ERROR SUBSCRIPT

RE3 - SEVERITY 1
TOO MANY ON ERROR STATEMENTS ACTIVE

RT1 - SEVERITY 1
SCHEDULE STATEMENT CONTAINS AN ILLEGAL FORM OF ?? TIMING EXPRESSION

RT10 - SEVERITY 1
AN UNLATCHED EVENT MAY NOT BE SET OR RESET

RT2 - SEVERITY 1
WHILE EXPRESSION MAY NOT BE A TIMING EXPRESSION

RT3 - SEVERITY 1
SCHEDULE STATEMENT CONTAINS AN ILLEGAL FORM OF ?? EVENT EXPRESSION

RT4 - SEVERITY 1
?? STATEMENT CONTAINS ILLEGAL PRIORITY EXPRESSION

RT5 - SEVERITY 1
SCHEDULE STATEMENT CONTAINS DUPLICATED AT/IN/ON EXPRESSIONS

RT6 - SEVERITY 1
WAIT STATEMENT CONTAINS ILLEGAL FORM OF ?? EXPRESSION

RT7 - SEVERITY 1
EVENT MUST BE SIGNalled ON/OFF OR ITS BINARY EQUIVALENT

RT8 - SEVERITY 1
AN ARRAYED EVENT MAY NOT BE SIGNalled

RT9 - SEVERITY 1
?? MUST BE A PROGRAM OR PROCEDURE IN ORDER TO BE SCHEDULED

RU1 - SEVERITY 1
SIGNAL STATEMENTS ARE THE ONLY REAL-TIME STATEMENTS WHICH MAY APPEAR INSIDE AN UPDATE BLOCK
ERROR MESSAGES FOR MAJOR CLASSIFICATION S
CLASSIFICATION "S" ERRORS INDICATE INCORRECT SUBSCRIPT USAGE

SC1 - SEVERITY 1
?? HAS TOO MANY STRUCTURE SUBSCRIPTS

SC2 - SEVERITY 1
?? HAS TOO MANY ARRAY SUBSCRIPTS

SC3 - SEVERITY 1
?? HAS TOO FEW ARRAY SUBSCRIPTS

SC4 - SEVERITY 1
?? HAS TOO MANY COMPONENT SUBSCRIPTS

SC5 - SEVERITY 1
?? HAS TOO FEW COMPONENT SUBSCRIPTS

SP1 - SEVERITY 1
SUBSCRIPTING CONTAINS MORE THAN ONE LIST OF STRUCTURE SUBSCRIPTS

SP2 - SEVERITY 1
SUBSCRIPTING CONTAINS MORE THAN ONE LIST OF ARRAY SUBSCRIPTS

SP3 - SEVERITY 1
SUBSCRIPT CONTAINS LEADING COLON, OR A COLON PRECEDED BY A SEMICOLON,
COLON, OR COMMA

SP4 - SEVERITY 1
SUBSCRIPT CONTAINS LEADING SEMICOLON, OR A SEMICOLON PRECEDED BY A SEMICOLON,
COLON, OR COMMA

SP5 - SEVERITY 1
SUBSCRIPT CONTAINS A LEADING COMMA, OR A COMMA PRECEDED BY A SEMICOLON, COLON,
OR COMMA

SP6 - SEVERITY 1
SUBSCRIPT IS EMPTY OR CONTAINS A TRAILING COMMA

SQ1 - SEVERITY 1
?? IS OF INCORRECT TYPE TO POSSESS A PRECISION QUALIFIER

SQ2 - SEVERITY 1
SUBSCRIPTED VARIABLE ?? MAY NOT POSSESS A PRECISION MODIFIER

SQ3 - SEVERITY 1
?? IS IN AN ASSIGNMENT CONTEXT AND THEREFORE MAY NOT POSSESS A PRECISION QUALIFIER
§R1  -SEVERITY 1
SIZE OF PARTITION IN A SUBSCRIPT OF ?? WAS UNKNOWN

§R2  -SEVERITY 1
SIZE OF PARTITION IN SUBSCRIPT OF ?? IS EITHER LESS THAN 2 OR
PRODUCED AN INDEX VALUE GREATER THAN THE MAXIMUM ALLOWABLE

§R3  -SEVERITY 1
THE VALUE OF A SUBSCRIPT OF ?? WAS GREATER THAN THE CORRESPONDING DIMENSION

§R4  -SEVERITY 1
THE VALUE OF A SUBSCRIPT OF ?? WAS LESS THAN 1
SS1 -SEVERITY 1
IN SUBSCRIPT OF ?? ONLY TRAILING ASTERISKS MAY BE OMITTED

ST1 -SEVERITY 1
A SUBSCRIPT OF ?? WAS NOT OF INTEGER OR SCALAR TYPE

SV1 -SEVERITY 1
SUBSCRIPTING OF ?? IS ILLEGAL IN CONTEXT OF USE AS AN ASSIGN ARGUMENT

SV2 -SEVERITY 0
USER SUPPLIED OVERPUNCH, NOT CONSISTENT WITH SUBSCRIPTING FOR VARIABLE ??

SV3 -SEVERITY 1
?? MAY NOT POSSESS SUBSCRIPTS
ERROR MESSAGES FOR MAJOR CLASSIFICATION T
CLASSIFICATION "T" ERRORS DEAL WITH INPUT/OUTPUT STATEMENTS

TC1  - SEVERITY 1
      ARGUMENT OF I/O CONTROL FUNCTION MUST BE OF UNARRAYED INTEGER OR SCALAR TYPE

TD1  - SEVERITY 1
      I/O DEVICE NUMBER IS NOT IN RANGE 0 THROUGH ??

TD2  - SEVERITY 1
      RECORD ADDRESS IS NOT AN UNARRAYED INTEGER OR SCALAR

T1   - SEVERITY 1
      VARIABLE IN READALL IS NOT OF CHARACTER TYPE

T2   - SEVERITY 1
      VARIABLE IN READ MAY NOT BE OF EVENT TYPE

T3   - SEVERITY 1
      VARIABLE IN READ/READALL MAY NOT BE A SUBBIT PSEUDO-VARIABLE

T4   - SEVERITY 1
      A FILE STATEMENT MAY NOT READ INTO A SUBBIT PSEUDO-VARIABLE
ERROR MESSAGES FOR MAJOR CLASSIFICATION U
CLASSIFICATION "U" ERRORS DEAL WITH UPDATE BLOCKS

UI1  - SEVERITY 1
      ?? MAY ONLY BE USED WITHIN AN UPDATE BLOCK

UI2  - SEVERITY 1
      UPDATE BLOCK DEFINITION MAY NOT APPEAR INSIDE AN UPDATE BLOCK

UP1  - SEVERITY 1
      THE PROCEDURE, TASK, OR PROGRAM ?? MAY NOT BE INVOKED WITHIN THE 
      CURRENT UPDATE BLOCK

UP2  - SEVERITY 1
      UPDATE BLOCKS MAY NOT CONTAIN RETURN STATEMENTS

UP3  - SEVERITY 1
      THE FUNCTION ?? MAY NOT BE INVOKED WITHIN THE CURRENT UPDATE BLOCK

UT1  - SEVERITY 1
      I/O STATEMENTS ARE ILLEGAL INSIDE UPDATE BLOCKS
ERROR MESSAGES FOR MAJOR CLASSIFICATION V
CLASSIFICATION "V" ERRORS ARE RELATED TO COMPILE-TIME VARIABLE ERRORS

VA1 - SEVERITY 1
COMPILETIME INTEGER/SCALAR ADDITION FAILED

VA2 - SEVERITY 1
COMPILETIME INTEGER/SCALAR SUBTRACTION FAILED

VA3 - SEVERITY 1
COMPILETIME INTEGER/SCALAR MULTIPLICATION FAILED

VA4 - SEVERITY 1
COMPILETIME INTEGER/SCALAR DIVISION FAILED

VA5 - SEVERITY 1
COMPILETIME INTEGER/SCALAR EXPONENTIATION FAILED

VC1 - SEVERITY 0
COMPILETIME CATENATION PRODUCED TOO LONG A CHARACTER STRING - TRUNCATED TO 255 CHARACTERS

VD1 - SEVERITY 1
THE VARIABLE ?? IS USED IN A COMPILETIME EXPRESSION BUT IS NOT PREVIOUSLY DEFINED

VE1 - SEVERITY 1
AN EXPRESSION NOT COMPUTABLE AT COMPILETIME HAS BEEN USED IN A CONTEXT WHERE A VALUE MUST BE KNOWN

VF1 - SEVERITY 1
COMPILETIME EVALUATION OF ?? FUNCTION FAILED

D-30
ERROR MESSAGES FOR MAJOR CLASSIFICATION X
CLASSIFICATION "X" ERRORS DEAL WITH IMPLEMENTATION DEPENDENT FEATURES

XD1 - SEVERITY 0
UNINTELLIGIBLE INFORMATION IN DEVICE DIRECTIVE

XD2 - SEVERITY 9
DUPLICATE DEVICE DIRECTIVE FOR CHANNEL ??

XD3 - SEVERITY 0
DEVICE DIRECTIVE DOES NOT CONTAIN A VALID CHANNEL INDICATION

XD4 - SEVERITY 0
CHANNEL NUMBERS MUST BE IN RANGE 0 TO 9

XI1 - SEVERITY 0
NESTED INCLUDE DIRECTIVES NOT ALLOWED

XI2 - SEVERITY 0
INCLUDE DIRECTIVE DOES NOT CONTAIN A NAME

XI3 - SEVERITY 0
?? NOT IN INCLUDE LIBRARY

XU1 - SEVERITY 0
D CARD CONTAINS UNKNOWN DIRECTIVE
Phase II ERRORS

SEVERITY 2
DATA STORAGE CAPACITY EXCEEDED

SEVERITY 2
INDIRECT STACK OVERFLOW

SEVERITY 2
CHARACTER LITERAL BUFFER OVERFLOW

SEVERITY 1
LITERAL PROCESSING FAILURE

SEVERITY 2
CONSTANT TABLE OVERFLOW

SEVERITY 1
TOO MANY EXTERNAL NAMES

SEVERITY 1
STORAGE DESCRIPTOR STACK OVERFLOW

SEVERITY 1
EXCEEDED TEMPORARY SPACE

SEVERITY 0
INDEX STACK USAGE INCONSISTANT

SEVERITY 2
STATEMENT LABELS ALL IN USE

SEVERITY 1
SIZE CONFLICT ON VECTOR/MATRIX PARAMETER

SEVERITY 1
ASSIGN PARAMETER NOT SYMBOL

SEVERITY 1
DATA TYPE CONFLICT ON PARAMETER

SEVERITY 1
ARRAYNESS CONFLICT ON PARAMETER

SEVERITY 1
ARRAY SIZE CONFLICT ON PARAMETER
SEVERITY 1
NOT ASSIGN PARAMETER

SEVERITY 2
SUBPROGRAM STACK OVERFLOW

SEVERITY 1
STATEMENT CONTAINS PHASE I ERROR

SEVERITY 0
UNMATCHED DO CASE ENDING

SEVERITY 0
UNMATCHED CASE LABEL

SEVERITY 0
UNMATCHED DO WHILE ENDING

SEVERITY 0
UNMATCHED DO FOR ENDING

SEVERITY 1
LEVEL MISMATCH ON PROC/FUNC/IO ARGUMENT

SEVERITY 1
LEVEL MISMATCH ON SHAPING FUNCTION

SEVERITY 1
UNIMPLEMENTED FEATURE OF HALS CALLED FOR

SEVERITY 1
REAL TIME COMPILER OPTION NOT SPECIFIED
APPENDIX E.

Execution-time Errors

The following tables indicate runtime error conditions which may occur during execution of a HAL/S-360 program. The tables list any standard fixups performed by the runtime system. The form of the system action taken is indicated by the following code:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>UNLIMITED</td>
</tr>
<tr>
<td>L</td>
<td>LIMITED</td>
</tr>
<tr>
<td>T</td>
<td>TERMINATE</td>
</tr>
<tr>
<td>Error Number</td>
<td>Message</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>EXPONENT OVERFLOW</td>
</tr>
<tr>
<td>2</td>
<td>EXPONENT UNDERFLOW</td>
</tr>
<tr>
<td>3</td>
<td>SCALAR DIVISION BY ZERO</td>
</tr>
<tr>
<td>4</td>
<td>EXPONENTIATION OF ZERO TO POWER &lt;= 0</td>
</tr>
<tr>
<td>5</td>
<td>SQUARE ROOT HAS ARG &lt; 0</td>
</tr>
<tr>
<td>Error Number</td>
<td>Message</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>6</td>
<td>EXP FUNCTION HAS ARG &gt; 174.673</td>
</tr>
<tr>
<td>7</td>
<td>LOG FUNCTION HAS ARG &lt;= 0</td>
</tr>
<tr>
<td>8</td>
<td>SIN OR COS FUNCTION HAS</td>
</tr>
<tr>
<td></td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>${1.126E15}$</td>
</tr>
<tr>
<td></td>
<td>PI</td>
</tr>
<tr>
<td>9</td>
<td>SIN OR COSH FUNCTION HAS</td>
</tr>
<tr>
<td></td>
<td>ARG &gt; 175,366</td>
</tr>
<tr>
<td>10</td>
<td>ARCSIN OR ARCCOS FUNCTION HAS</td>
</tr>
<tr>
<td></td>
<td>$</td>
</tr>
<tr>
<td>Error Number</td>
<td>Message</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>11</td>
<td>TAB FUNCTION HAS</td>
</tr>
<tr>
<td></td>
<td>$</td>
</tr>
<tr>
<td>12</td>
<td>TAN FUNCTION TOO CLOSE TO SINGULARITY</td>
</tr>
<tr>
<td>13</td>
<td>CASE VARIABLE OUT OF RANGE</td>
</tr>
<tr>
<td>14</td>
<td>CLOSE REACHED ON FUNCTION</td>
</tr>
<tr>
<td>15</td>
<td>SCALAR TOO LARGE FOR INTEGER CONVERSION</td>
</tr>
<tr>
<td>Error Number</td>
<td>Message</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>16</td>
<td>INTEGER DIVISION BY ZERO</td>
</tr>
<tr>
<td>17</td>
<td>ILLEGAL CHARACTER SUBSCRIPT</td>
</tr>
<tr>
<td>18</td>
<td>BAD LENGTH IN LJUST OR RJUST</td>
</tr>
<tr>
<td>19</td>
<td>MOD DOMAIN ERROR</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>CHARACTER TO SCALAR CONVERSION</td>
</tr>
<tr>
<td>Error Number</td>
<td>Message</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>21</td>
<td>CHARACTER TO SCALAR CONVERSION DURING INPUT</td>
</tr>
<tr>
<td>22</td>
<td>CHARACTER TO INTEGER CONVERSION</td>
</tr>
<tr>
<td>23</td>
<td>CHARACTER TO INTEGER CONVERSION DURING INPUT</td>
</tr>
<tr>
<td>24</td>
<td>NEGATIVE BASE IN EXPONENTIATION</td>
</tr>
<tr>
<td>25</td>
<td>VECTOR/MATRIX DIVISION BY ZERO</td>
</tr>
<tr>
<td>Error Number</td>
<td>MESSAGE</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>26</td>
<td>ILLEGAL BIT STRING DURING INPUT</td>
</tr>
<tr>
<td>27</td>
<td>ARG OF INVERSE IS SINGULAR</td>
</tr>
<tr>
<td>28</td>
<td>ARG OF UNIT FUNCTION IS NULL VECTOR</td>
</tr>
<tr>
<td>29</td>
<td>ILLEGAL BIT STRING</td>
</tr>
<tr>
<td>30</td>
<td>UNUSED</td>
</tr>
<tr>
<td>Error Number</td>
<td>Message</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>31</td>
<td>SYNAE ERROR: xxx</td>
</tr>
<tr>
<td>32</td>
<td>MISSING DD CARD - CHANNELN</td>
</tr>
<tr>
<td>33</td>
<td>PRINT ON INPUT CHANNEL N or INPUT ON PRINT CHANNEL N</td>
</tr>
<tr>
<td>34</td>
<td>ILLEGAL SKIP COUNT ON CHANNEL n</td>
</tr>
<tr>
<td>35</td>
<td>MARGIN VIOLATION ON CHANNEL n</td>
</tr>
<tr>
<td>Error Number</td>
<td>Message</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>36</td>
<td>ILLEGAL PAGE COUNT ON CHANNEL n</td>
</tr>
<tr>
<td>37</td>
<td>ILLEGAL LINE COUNT ON CHANNEL n</td>
</tr>
<tr>
<td>38</td>
<td>ILLEGAL NUMERIC FIELD ON CHANNEL n</td>
</tr>
<tr>
<td>39</td>
<td>ILLEGAL BIT OR CHARACTER STRING</td>
</tr>
<tr>
<td>Error Number</td>
<td>Message</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>40 through</td>
<td>END OF FILE ON CHANNEL n</td>
</tr>
<tr>
<td>49</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>ERROR IN HAL SOURCE</td>
</tr>
<tr>
<td>51</td>
<td>UNUSED</td>
</tr>
<tr>
<td>52</td>
<td>INACTIVE PROCESS IN UPDATE</td>
</tr>
<tr>
<td></td>
<td>PRIORITY STATEMENT</td>
</tr>
<tr>
<td>53</td>
<td>INACTIVE PROCESS IN TERMINATE</td>
</tr>
<tr>
<td></td>
<td>STATEMENT</td>
</tr>
<tr>
<td>Error Number</td>
<td>Message</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>54</td>
<td>INACTIVE PROCESS IN CANCEL STATEMENT</td>
</tr>
<tr>
<td>55</td>
<td>PROCESS ALREADY ACTIVE IN SCHEDULE STATEMENT</td>
</tr>
<tr>
<td>56</td>
<td>SUBSCRIPT OUT OF RANGE</td>
</tr>
<tr>
<td>57</td>
<td>PROCESS NOT A DEPENDENT IN CANCEL STATEMENT</td>
</tr>
<tr>
<td>58</td>
<td>PROCESS NOT A DEPENDENT IN TERMINATE STATEMENT</td>
</tr>
</tbody>
</table>
APPENDIX F.

User Abend Codes During Execution

0  Missing DD card for message channel.
    The HAL/S channel specified for writing error and
    trace messages had no assigned DD card. (This may
    also appear as SYSTEM ABEND CODE 0)

1  HAL/S error with SYSTEM=T.
    A run time error causing abnormal termination
    occurred. A specific message preceding the ABEND
    information explains the error.

2  HAL/S error with SYSTEM=L and error count exceeded.
    A run time error occurred and the specified maximum
    error count for that error was thereby exceeded.

3  Invalid error recursion.
    An error condition arose while processing a run time
    error.

4  unused

5  unused

6  I/O mode conflict.
    Input asked for when I/O processor expected output or
    vice versa.

7  Illegal I/O channel or mode.
    I/O asked for on illegal channel, or I/O mode was
    illegal.

9  Program interrupt in non-HAL/S environment.
    Registers 12, 15 not set as expected on program
    interrupt.

10 Program interrupt from convert-to-binary (CVB) instruction.

11 Illegal HAL/S error number.
User Abend Codes During Compilation

100 Unable to open one of the files: PROGRAM, SYSIN, OR SYSPRINT
200 Unexpected end of file while reading in the XPL program
300 Synad error while reading in the XPL program
400 XPL program won't fit in the amount of memory available
500 Invalid service code from the XPL program
600 Printed-page limit exceeded
700 Linked programs specified different size common areas
800 Synad error on output file
900 Invalid output file specified
1000 Synad error on input file
1100 Linking process overlayed common string area
1200 End of file error on input file
1300 Impossible to move the common strings up during linking
1400 Invalid input file specified
1500 Unknown request by 'MONITOR' func
1600 Unknown do in 'MONITOR' request
1700 Directory error on PDS
1800 Synad error on output PDS file
1900 Invalid member name specified
2000 Synad error on direct access file
2100 Attempt to read from an input PDS without issuing the "FIND" MONITOR request first
2200 End of file error on direct access file
2300  Invalid member to be found
2400  Synad error on PDS input file
2500  File blocking specification err
3000  MON#9/10 error or misaligned # 5
4000  XPL program called exit to force an abend (and a possible core dump)
### APPENDIX G

List of names reserved for HAL/S programs only.

| ARCCOS | EXP | M14SN |
| ARCSIN | FLUSH | M14S3 |
| ARCTAN | GETSEED | M15DN |
| BAKTSPACE | HALDUMP | M15SN |
| CIN | HALPRINT | M20DNP |
| CINDEX | HALSTART | M20SNP |
| CINP | HALSYS | M21DNP |
| CIJUST | HDOPEN | M21SNP |
| COLUMN | HDPEAD | OUTPUT |
| COS | HDWRITE | PAGE |
| COSH | HIM | PROGIN |
| COT | HSDUMP | RANDOM |
| COUT | IIN | RANDOMG |
| CPASP | INPUT | SETSEED |
| CPASP | IOINII | SIN |
| CPASP | IOUT | SINH |
| CRJUST | ITOC | SKIP |
| CTOD | ITOTHEI | SKIPIN |
| CTOE | LINE | SKIPOUT |
| CTOE | LOG | SQRT |
| CTRIM | MM6DN | STMTRACE |
| DARCCOS | MM6D3 | TAB |
| DARCSIN | MM6SN | TAN |
| DCOS | MM6S3 | TENSTBL |
| DCOSH | MOMSTACK | VM6DN |
| DCOT | MSGJOIN | VM6SN |
| DEXP | MV6DN | VO6DN |
| DIN | MV6SN | VO6D3 |
| DLOG | M1DNP | VO6SN |
| DOUT | M1DSNP | VO6S3 |
| DSIN | M1SNP | VO6S3 |
| DSINH | M1SNP | VV6DN |
| DSCRT | M11DN | V10DN |
| DTAH | M11SN | V10D3 |
| DTCC | M12DN | V10SN |
| DTOTHE | M12D3 | V10S3 |
| DTOTHE | M12SN | V9DN |
| EIN | M12S3 | V9D3 |
| EOHT | M13DN | V9SN |
| EFORMON | M13D3 | V9S3 |
| ETOCC | M13SN | WHERE |
| ETOHEE | M13S3 | |
| ETOHEE | M14DN | |
| ETOHEE | M14D3 | |
APPENDIX H.

Compiler Directives

The following compiler directives have been defined for the
HAI/S-360 compiler.

a) The DEVICE directive has the form:

D DEVICE CHANNEL=n <option>

This directive sets the mode of the specified channel
(referred to via the CHANNELn DD card) to the mode
indicated by the <option>. The <option> may be "PAGED",
"UNPAGED", or null (in which case UNPAGED is assumed).

b) The INCLUDE directive has the form:

D INCLUDE <name> <option>

This directive names a member of an include library as
defined in section 6.2.7. The <option> may be "NOLIST" or
null. The "NOLIST" option indicates that the included
text is not to be listed.

II-1