Program Product

IBM System/360 Operating System Assembler H Messages

Program Number 5734-AS1

This book describes the assembly error diagnostic messages and the abnormal assembly termination messages issued by Assembler H. Assembler H is an assembler language processor for the IBM System/360 Operating System. It performs high-speed assemblies on an IBM System/360 Model 40 or higher and on an IBM System/370 Model 145, 155, or 165 with at least 256K bytes of main storage.

This book is intended for all Assembler H programmers. It should be used in conjunction with the Operating System Assembler Language manual, Order Number GC28-6514; the Assembler H Language Specifications, Order Number GC26-3771; and the Assembler H Programmer's Guide, Order Number SC26-3759.

For each error message, this book describes the number and text of the message, the explanation of the error, the assembler's handling of the error, and the programmer's response to correct the error. The introduction to this book describes the format, content, and placement of the error messages.

IBM

First Edition (June, 1970)

This edition, with Technical Newsletter SN33-8093, applies to version 2 of the IBM System/360 Operating System Assembler H Program Product 5734-AS1 and to all subsequent versions until otherwise indicated in new editions or Technical Newsletters. Changes to the text, and small changes to illustrations, are indicated by a vertical line to the left of the change; changed or added illustrations are denoted by the symbol • to the left of the caption.

Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/360 SRL Newsletter, Order No. GN20-0360 for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Nordic Laboratory, Publications Development, Box 962, S-18109 Lidingö 9, Sweden.

© Copyright International Business Machines Corporation 1970

IBM Technical Newsletter

Re: Order No. SC26-3770-0

This Newsletter No. SN33-8093

Date February 15, 1971

Previous Newsletter Nos. None

IBM SYSTEM/360 OPERATING SYSTEM ASSEMBLER H MESSAGES

©IBM Corp. 1970

This Technical Newsletter, a part of version 2 of IBM System/360 Operating System, Assembler H Program Product provides replacement pages IBM System/360 Operating System Assembler H Messages, Order Number SC26-3770-0. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are listed below:

Cover, ii
5,6
9-12
15,16

A change to the text or a small change to an illustration is indicated by a vertical line to the left of the change; a changed or added illustration is denoted by the symbol • to the left of the caption.

Summary of Amendments

Minor errors and omissions are corrected throughout the manual and information on IBM System/370 has been added.

Note: File this cover letter at the back of the manual to provide a record of changes.

IBM Nordic Laboratory, Technical Communications, Box 962, S-181 09 Lidingö 9, Sweden

© IBM Corp. 1971

PRINTED IN U.S.A.

PREFACE

This manual describes the Assembler H error diagnostic messages and abnormal termination messages. Assembler H is a high performance assembler language processor for the IBM System/360 Operating System.

This manual has three sections:

- Introduction
- Assembly Error Diagnostic Messages
- Assembly Abnormal Termination Messages

The Introduction describes the format, contents, and placement of messages -- both in macro definitions and in the source program. The other two sections describe the messages themselves. All messages appear in the assembler listing. One message, IEV999, also appears on the system console device.

This book is intended for all Assembler H programmers. To use it, you should be familiar with the Assembler H language and operating procedures described in the following books:

IBM System/360 Operating System Assembler Language, Order Number GC28-6514.

The Assembler Language manual contains the basic assembler and macro assembler specifications, except those unique to Assembler H.

IBM System/360 Operating System Assembler H Language Specifications, Order Number GC26-3771.

The Assembler H Language Specifications describes the language features that are available with Assembler H. It is supplemental to the Assembler Language manual listed above.

IBM System/360 Operating System Assembler H Programmer's Guide, Order Number SC26-3759.

The Assembler H Programmer's Guide contains detailed information about using Assembler H. It includes assembler options, cataloged job control language procedures, assembler listing and output descriptions, and information on programming techniques and considerations.

IBM System/360 Operating System Assembler H System Information, Order Number SC26-3768.

The System Information manual contains the performance estimates and storage estimates for Assembler H. It also describes how to add the assembler to the Operating System (system generation).

This page intentionally left blank.

CONTENTS

INTRODUCTION..... 1
Assembly Error Diagnostic Messages..... 1
  Message Not Known..... 2
Abnormal Assembly Termination Messages..... 2
Message Descriptions..... 2
  Message Number and Text..... 3
  Explanation of Message..... 3
  Assembler Action..... 3
  Programmer Response..... 3
  Severity Code..... 4

ASSEMBLY ERROR DIAGNOSTIC MESSAGES..... 5

ASSEMBLY ABNORMAL TERMINATION MESSAGES..... 51

This page intentionally left blank.

ASSEMBLY ERROR DIAGNOSTIC MESSAGES

Assembler H prints most error messages in the listing immediately following the statements in error. It also prints the total number of flagged statements and their line numbers in the Diagnostic Cross Reference section at the end of the listing.

The messages do not follow the statement in error when:

- Errors are detected during editing of macro definitions read from a library. A message for such an error appears after the first call in the source program to that macro definition. You can, however, bring the macro definition into the source program with a COPY statement. The editing error messages will then be attached to the statements in error.

- Errors are detected by the lookahead function of the assembler. (Lookahead scans, for attribute references, statements after the one being assembled.) Messages for these errors appear after the statements in which they occur. The messages may also appear at the point where lookahead was called.

- Errors are detected on conditional assembly statements during macro generation or MHELP testing. Such a message follows the most recently generated statement or MHELP output statement.

A typical error diagnostic message is:

```
IEV057  ***ERROR*** UNDEFINED OPERATION CODE -- xxxxx
```

The term ***ERROR*** is part of the message if the severity code is 8 or greater. The term **WARNING** is part of the message if the severity code is 0 or 4.

A copy of a segment of the statement in error, represented above by xxxxx, is appended to the end of many messages. Normally this segment, which can be up to 16 bytes long, begins at the bad character or term. For some errors, however, the segment may begin after the bad character or term. The segment may include part of the remarks field.

If a diagnostic message follows a statement generated by a macro definition, the following items may be appended to the error message:

- The number of the model statement in which the error occurred, or the first five characters of the macro name.

- The SET symbol, parameter number, or value string associated with the error.

Note: references to macro parameters are by number (such as PARAM008) instead of name. The first seven numbers are always assigned for the standard system parameters as follows:

```
PARAM000 = &SYSNDX
PARAM001 = &SYSECT
PARAM002 = &SYSLOC
PARAM003 = &SYSTIME
PARAM004 = &SYSDATE
PARAM005 = &SYSPARM
PARAM006 = Name Field Parameter
```

Then the keyword parameters are numbered in the order defined in the macro definition, followed by positional parameters. When there are no keyword parameters in the macro definition, PARAM007 refers to the first positional parameter.

If a diagnostic message follows a conditional assembly statement in the source program, the following items will be appended to the error message:

- The word "OPENC"
- The SET symbol or value string associated with the error

Several messages may be issued for a single statement or even for a single error within a statement. This happens because each statement is usually evaluated on more than one level (for example, term level, expression level, and operand level) or by more than one phase of the assembler. Each level or phase can diagnose errors; therefore, most or all of the errors in the statement are flagged. Occasionally, duplicate error messages may occur. This is a normal result of the error detection process.

MESSAGE NOT KNOWN

The following message may appear in a listing:

IEVnnn ***ERROR*** MESSAGE NOT KNOWN -- xxxxxxxxxx

The statement preceding this message contains an error but the assembler routine which detected the error issued the number (IEVnn) of a nonexistent error message to the assembler's message generation routine. The segment of the statement in error may be appended to the message. If you can correct the error, this statement will assemble correctly. However, there is a bug in the error detection process of the assembler. Save the output and the source deck from this assembly and report the problem to your IBM customer engineer.

ABNORMAL ASSEMBLY TERMINATION MESSAGES

Whenever an assembly cannot be completed, Assembler H provides a message and, in some cases, a specially formatted dump for diagnostic information. This may indicate an assembler malfunction or it may indicate a programmer error. The statement causing the error is identified and, if possible, the assembly listing up to the point of the error is printed. The messages in this book will give enough information to correct the error and reassemble your program or to determine that the error is an assembler malfunction. The Assembler H Program Logic Manual, Order Number LY26-3760, gives a complete explanation of the abnormal termination dump.

MESSAGE DESCRIPTIONS

Each message entry in this book has five sections:

- Message Number and Text
- Explanation of Message

- Assembler Action
- Programmer Response
- Severity Code

MESSAGE NUMBER AND TEXT

Only the message number and the major fixed portion of the message text (For example, UNDEFINED OPERATION CODE in the sample message above) are included in the message description. Any abbreviations in actual message text are spelled out in full in the book. Unused message numbers account for the gaps in the message number sequence. No messages are defined for numbers, such as IEV005, not included in this section.

EXPLANATION OF MESSAGE

There may be more than one explanation for some messages because they are generated by different sections of the assembler. Several of the assembler termination messages have identical explanations. In these cases, each message precedes the explanation.

ASSEMBLER ACTION

This section tells how the assembler handles statements with errors. A machine instruction is assembled as all zeros. An assembler instruction is usually ignored; it is printed but has no effect on the assembly. Many assembler instructions, however, are partially processed or processed with a default value.

For some instructions, the operands preceding the operand in error or every operand except the operand in error are processed. For example, if one of several operands on a DROP statement is a symbol which has not been equated to a register number, only that operand is ignored. All the correctly specified registers are correctly processed.

For some assembler statements, especially macro prototype and conditional assembly statements, the operand or term in error is given a default value. Thus the statement will assemble completely, but will probably cause incorrect results if the program is executed.

PROGRAMMER RESPONSE

Many errors have specific or probable causes. In such a case, the programmer response section gives specific steps for fixing the error. Most messages, however, have too many possible causes (from keypunch error to wrong use of the statement) to list. The programmer response for these error messages does not give specific directions. The cause of most such errors can be determined from the message text and the explanation.

SEVERITY CODE

The severity code indicates the seriousness of the error. The severity codes and their meanings are shown below:

Severity Code	Explanation
0	No errors detected
4	Minor errors detected; successful program execution is probable
8	Errors detected; unsuccessful program execution is possible
12	Serious errors detected; unsuccessful program execution is probable
16	Critical errors detected; normal execution is impossible
20	I/O error from which the system could not recover occurred during assembly, or data sets are missing; assembly terminated

This code is the return code issued by the assembler when it returns control to the Operating System. The IBM-supplied cataloged procedures include a COND parameter on the linkage edit and execution steps. The COND parameter prevents execution of these steps if the return code from the assembler is 8 or greater. Thus errors with ***ERROR*** in the message prevent the assembled program from link editing or executing. Errors with **WARNING** in the message do not.

ASSEMBLY ERROR DIAGNOSTIC MESSAGES

IEV001 OPERATION-CODE NOT ALLOWED TO BE GENERATED

Explanation: An attempt was made to produce a restricted operation code by variable symbol substitution. Restricted operation codes are:

ACTR	AGO	AGOB	AREAD
AIF	AIFB	ANOP	SETA
COPY	REPRO	ICTL	SETB
MACRO	MEND	MEXIT	SETC
GBLA	GBLB	GBLC	
LCLA	LCLB	LCLC	

Assembler Action: The statement is ignored.

Programmer Response: If you want a variable operation code, use AIF to branch to the correct unrestricted statement.

Severity Code: 8

IEV002 GENERATED STATEMENT TOO LONG. STATEMENT TRUNCATED

Explanation: The statement generated by a macro definition is more than 864 characters long.

Assembler Action: The statement is truncated; the leading 864 characters are retained.

Programmer Response: Shorten the statement.

Severity Code: 12

IEV003 UNDECLARED VARIABLE SYMBOL. DEFAULT=0, NULL, OR TYPE=U

Explanation: A variable symbol in the operand field of the statement has not been declared (defined) in the name field of a SET statement, in the operand field of a LCL or GBL statement, or in a macro prototype statement.

Assembler Action: The variable symbol is given a default value as follows:

SETA = 0
SETB = 0
SETC = null (empty) string

The type attribute (T') of the varaible is given a default value of U (undefined).

Programmer Response: Declare the variable before you use it as an operand.

Severity Code: 8

IEV004 DUPLICATE SET SYMBOL DECLARATION. FIRST IS RETAINED

Explanation: A SET symbol has been declared (defined) more than once. A SET symbol is declared when it is used in the name field of a SET statement, in the operand field of a LCL or GBL statement, or in a macro prototype statement.

Assembler Action: The value of the first declaration of the SET symbol is used.

Programmer Response: Eliminate the incorrect declarations.

Severity Code: 8

IEV007 PREVIOUSLY DEFINED SEQUENCE SYMBOL

Explanation: The sequence symbol in the name field has been used in the name field of a previous statement.

Assembler Action: The first definition of the sequence symbol is used; this definition is ignored.

Programmer Response: Remove or change one of the sequence symbols.

Severity Code: 12

IEV008 PREVIOUSLY DEFINED SYMBOLIC PARAMETER

Explanation: The same variable symbol has been used to define two different symbolic parameters.

Assembler Action: When the parameter name (the variable symbol) is used inside the macro definition, it will refer to the first definition of the parameter in the prototype. However, if the second parameter defined by the variable symbol is a positional parameter, the count of positional operands will still be increased by one. The second parameter can then be referred to only through use of &SYSLIST.

Programmer Response: Change one of the parameter names to another variable symbol.

Severity Code: 12

IEV009 SYSTEM VARIABLE SYMBOL ILLEGALLY RE-DEFINED

Explanation: A system variable symbol has been used in the name field of a macro prototype statement. The system variable symbols are:

&SYSECT	&SYSDATE
&SYSLIST	&SYSLOC
&SYSNDX	&SYSPARM
&SYSTIME	

Assembler Action: The name parameter is ignored. The name on a corresponding macro instruction will not be generated.

Programmer Action: Change the parameter to one which is not a system variable symbol.

Severity Code: 12

IEV011 INCONSISTENT GLOBAL DECLARATIONS. FIRST IS RETAINED

Explanation: A global SET variable symbol has been defined in more than one macro definition or in a macro definition and in the source program, and the two definitions are inconsistent in type or dimension.

Assembler Action: The first definition encountered is retained.

Programmer Response: Assign a new SET symbol or make the definitions compatible.

Severity Code: 8

IEV012 UNDEFINED SEQUENCE SYMBOL. MACRO ABORTED

Explanation: A sequence symbol in the operand field is not defined; that is, it is not used in the name field of a model statement.

Assembler Action: Exit from the macro definition.

Programmer Response: Define the sequence symbol.

Severity Code: 12

IEV013 ACTR COUNTER EXCEEDED

Explanation: The conditional assembly loop counter (set by an ACTR statement) has been decremented to zero. The ACTR counter is decremented by one each time an AIF or AGO branch is executed successfully. The counter is halved for most errors encountered by the macro editor phase of the assembler.

Assembler Action: A macro expansion is terminated. If the ACTR statement is in the source program, the assembly is terminated.

Programmer Response: Check for an AIF/AGO loop or another type of error. (You can use the MHELP facility, described in the Assembler H Programmer's Guide, Order Number SC26-3759, to trace macro definition logic.) If there is no error, increase the initial count on the ACTR instruction.

Severity Code: 12

IEV017 UNDEFINED KEYWORD PARAMETER. DEFAULT TO POSITIONAL INCLUDING KEYWORD

Explanation: A keyword parameter in a macro call is not defined in the corresponding macro prototype statement.

Note: This message may be generated by a valid positional parameter that contains an equals sign.

Assembler Action: The keyword (including the equals sign and value) is used as a positional parameter.

Programmer Response: Define the keyword in the prototype statement.

Severity Code: 4

IEV018 DUPLICATE KEYWORD IN MACRO CALL. LAST VALUE IS USED

Explanation: A keyword operand occurs more than once in a macro call.

Assembler Action: The latest value assigned to the keyword is used.

Programmer Response: Eliminate one of the keyword operands.

Severity Code: 12

IEV020 ILLEGAL GBL OR LCL STATEMENT

Explanation: A global (GBL) or local (LCL) declaration statement does not have an operand.

Assembler Action: The statement is ignored.

Programmer Response: Remove the statement or add an operand.

Severity Code: 8

IEV021 ILLEGAL SET STATEMENT

Explanation: The operand of a SETB statement is not 0, 1, or a SETB expression enclosed in parentheses.

Assembler Action: The statement is ignored.

Programmer Response: Correct the operand or delete the statement.

Severity Code: 8

IEV024 INVALID VARIABLE SYMBOL

Explanation: One of these errors has occurred:

- A symbolic parameter or a SET symbol is not an ampersand followed by 1 to 62 alphameric characters, the first being alphabetic.
- A created SET symbol definition is not a valid SET symbol expression enclosed in parentheses.

Assembler Action: The statement is ignored.

Programmer Response: Supply a valid symbol or expression.

Severity Code: 8

IEV025 INVALID MACRO PROTOTYPE OPERAND

Explanation: The format of the operand field of a macro prototype statement is invalid. For example, two parameters are not separated by a comma or a parameter contains an invalid character.

Assembler Action: The operand field of the prototype is ignored.

Programmer Response: Supply a valid operand field.

Severity Code: 12

IEV026 MACRO CALL OPERAND TOO LONG. 255 LEADING CHARACTERS DELETED

Explanation: An operand of a macro instruction is more than 255 characters long.

Assembler Action: The leading 255 characters are deleted.

Programmer Response: Limit the operand to 255 characters, or break it down into two or more operands.

Severity Code: 12

IEV027 EXCESSIVE NUMBER OF OPERANDS

Explanation: One of the following errors has occurred:

- More than 240 positional and/or keyword operands have been explicitly defined in a macro prototype statement.
- There are more than 255 operands in a DC, DS, or DXD statement.

Assembler Action: The excess parameters are ignored.

Programmer Response: For a DC, DS, or DXD statement, use more than one statement. For a macro prototype statment, delete the extra operands and use &SYSLIST to access the positional operands, or redesign the macro definition.

Severity Code: 12

IEV028 INVALID DISPLACEMENT

Explanation: One of the following errors has occurred:

- The displacement field of an explicit address is not an absolute value within the range 0 through 4095.
- The displacement field of an S-type address constant is not an absolute value within the range 0 through 4095.

Assembler Action: The statement or constant is assembled to zero.

Programmer Response: Correct the displacement or supply an appropriate USING statement containing an absolute first operand prior to this statement.

Severity Code: 8

IEV029 INCORRECT REGISTER OR MASK SPECIFICATION

Explanation: The value specifying a register or a mask is not an absolute value within the range 0 through 15; an odd register is used where an even register is required; or a register is used where none can be specified.

Assembler Action: For machine instructions and S-type address constants, the statement or constant is assembled to zero. For USING and DROP statements, the invalid register operand is ignored.

Programmer Response: Specify a valid register.

Severity Code: 8

IEV030 INVALID LITERAL USAGE

Explanation: A literal is used in an assembler instruction, another literal, or a field of a machine instruction where it is not permitted.

Assembler Action: An assembler instruction containing a literal is generally ignored and another message, relative to the operation code of the instruction, appears. A machine instruction is assembled to zero.

Programmer Response: If applicable, replace the literal with the name of a DC statement.

Severity Code: 8

IEV031 INVALID IMMEDIATE FIELD

Explanation: The value of an immediate operand of a machine instruction requires more than one byte of storage (exceeds 255) or the value of the immediate operand exceeds 9 on an SRP instruction.

Assembler Action: The instruction is assembled to zero.

Programmer Response: Use a valid immediate operand, or specify the immediate information in a DC statement or a literal and change the statement to a non-immediate type.

Severity Code: 8

IEV032 RELOCATABLE VALUE FOUND WHERE ABSOLUTE VALUE REQUIRED

Explanation: A relocatable or complex relocatable expression is used where an absolute expression is required.

Assembler Action: A machine instruction is assembled to zero. In a DC, DS, or DXD statement, the operand in error and the following operands are ignored.

Programmer Response: Supply an absolute expression or term.

Severity Code: 8

IEV033 ALIGNMENT ERROR

Explanation: An address referenced by this statement might not be aligned to the proper boundary for this instruction; for example the data referenced by a load instruction (L) may be on a half word boundary, or the address might depend upon an index register.

Assembler Action: The instruction is assembled as written.

Programmer Response: Correct the operand if it is in error. If you are using a System/360 model which does not require alignment or you wish to supress alignment checking for some other reason, you can specify PARM='NOALGN' in the EXEC card. If a particular statement is correct, you can supress this message by writing the statement with an absolute displacement and an explicit base register, as in this example:
L 1,SYM-BASE(,2)

Severity Code: 4

IEV034 ADDRESSABILITY ERROR

Explanation: The address referenced by this statement does not fall within the range of a USING statement, or a base register is specified along with a relocatable displacement.

Assembler Action: The instruction is assembled to zero.

Programmer Response: Insert appropriate USING statement prior to this statement. Otherwise, check this statement for a misspelled symbol, an unintended term or symbol in an address expression, or a relocatable symbol used as a displacement.

Severity Code: 8

IEV035 INVALID DELIMITER

Explanation: (1) A required delimiter in a DC, DS, or DXD statement is missing or appears where none should be; the error may be any of these:

- A quote with an address constant.
- A left parenthesis with a non-address constant.
- A constant field not started with a quote, left parenthesis, blank, or comma.
- An empty constant field in a DC.
- A missing comma or right parenthesis following an address constant.

- A missing subfield right parenthesis in an S-type address constant.
- A missing right parenthesis in a constant modifier expression.

(2) A parameter in a macro prototype statement was not followed by a valid delimiter -- comma, equals sign, or blank.

Assembler Action: The operand or parameter in error and the following operands or parameters are ignored.

Programmer Response: Supply a valid delimiter.

Severity Code: 12

IEV036 REENTRANT CHECK FAILED

Explanation: A machine instruction which might store data into a control section or common area when executed has been detected. This message is generated only when reentrant checking is requested by PARM='RENT' on the EXEC job control card.

Assembler Action: The statement is assembled as written.

Programmer Response: If you want reentrant code, correct the instruction. Otherwise, you can suppress reentrant checking by using PARM='NORENT' on the EXEC job control card.

Severity Code: 4

IEV037 ILLEGAL SELF-DEFINING VALUE

Explanation: A decimal, binary (B), hexadecimal (X), or character (C) self-defining term contains invalid characters or is in illegal format.

Assembler Action: In the source program, the operand in error and the following operands are ignored. In a macro definition, the entire statement is ignored.

Programmer Response: Supply a valid self-defining term.

Severity Code : 8

IEV038 OPERAND VALUE FALLS OUTSIDE OF CURRENT SECTION/LOCTR

Explanation: An ORG statement specifies a location outside of the control section or the LOCTR in which the ORG is used. Note that ORG cannot force a change to another section or LOCTR.

Assembler Action: The statement is ignored.

Programmer Response: Change the ORG statement if it is wrong. Otherwise, insert a CSECT, DSECT, COM, or LOCTR statement to set the location counter to the proper section before the ORG statement is executed.

Severity Code: 12

IEV039 LOCATION COUNTER ERROR

Explanation: The location counter has exceeded $2^{24}-1$, the largest address which can be contained in three bytes. This occurrence is called location counter wraparound.

Assembler Action: The location counter is four bytes long (only three bytes appear in the listing and the object deck). The overflow is carried into the high-order byte and the assembly continues. However, the resulting code will probably not execute correctly.

Programmer Response: The probable cause is a high ORG statement value or a high START statement value. Correct the value or split up the control section.

Severity Code: 12

IEV040 MISSING OPERAND

Explanation: This statement requires an operand and none is present.

Assembler Action: A machine instruction is assembled to zero. An assembler instruction is ignored.

Programmer Response: Supply the missing operand.

Severity Code: 12

IEV041 TERM EXPECTED. TEXT IS UNCLASSIFIABLE

Explanation: One of these errors has occurred:

- A term was expected, but the character encounted is not one that starts a term (letter, number, =, +, -, *).
- A letter and a quote did not introduce a valid term; the letter is not L, C, X, or B.

Assembler Action: Another message will accompany an assembler statement. A machine instruction will be assembled to zero.

Programmer Response: Check for missing punctuation, a wrong letter on a self-defining term, a bad attribute request, a leading comma, or a dangling comma. Note that the length attribute is the only one accepted here. If a scale, type, or integer attribute is needed, use a SETA statement and substitute the variable symbol where the attribute is needed.

Severity Code: 8

IEV043 PREVIOUSLY DEFINED SYMBOL

Explanation: The symbol in a name field or in the operand field of an EXTRN statement was defined (used as a name or an EXTRN operand) in a previous statement.

Assembler Action: The name or EXTRN operand of this statement is ignored. The following operands of an EXTRN will be processed. The first occurrence of the symbol will define it.

Programmer Response: Correct a possible spelling error or change the symbol.

Severity Code: 8

IEV044 UNDEFINED SYMBOL

Explanation: A symbol in the operand field has not been defined, that is, used in the name field of another statement or the operand field of an EXTRN.

Assembler Action: A machine instruction or an address constant is assembled to zero. In a DC, DS, or DXD statement or in a duplication-factor or length-modifier expression, the operand in error and the following operands are ignored. Any other instruction is ignored entirely.

Programmer Response: Define the symbol or remove the references to it.

Severity Code: 8

IEV045 REGISTER NOT PREVIOUSLY USED

Explanation: A register specified in a DROP statement has not been previously specified in a USING statement.

Assembler Action: Registers not currently active are ignored.

Programmer Response: Remove the unreferenced registers from the DROP statement. You can drop all active base registers at once by specifying DROP with a blank operand.

Severity Code: 4

IEV046 FLAG BYTE OPERAND IS NOT A MULTIPLE OF 8

Explanation: Bits 37-39 of a Channel Command Word specified by a CCW statement are not all zero.

Assembler Action: The CCW is assembled to zero.

Programmer Response: Set bits 37-39 to zero to suppress message during next assembly.

Severity Code: 8

IEV047 SEVERITY CODE TOO LARGE

Explanation: The severity code (first operand) of an MNOTE statement is not * or an unsigned decimal number from 0 to 255.

Assembler Action: The statement is printed in standard format instead of MNOTE format. The MNOTE is given the severity code of this message.

Programmer Response: Choose a severity code of * or a number less than 255, or check for a generated severity code.

Severity Code: 8

IEV048 ENTRY ERROR

Explanation: One of the following errors was detected in the operand of an ENTRY statement:

- Duplicate symbol (previous ENTRY).
- Symbol defined in a DSECT or COM section.
- Symbol defined by a DXD statement.
- Undefined symbol.
- Symbol defined by an absolute or complex relocatable EQU statement.

Assembler Action: The External Symbol Dictionary output is supressed for the symbol.

Programmer Response: Define the ENTRY operand correctly.

Severity Code: 8

IEV049 ILLEGAL RANGE ON ISEQ

Explanation: If this message is accompanied by another, this one is advisory. If it appears by itself, it indicates one of the following errors:

- An operand value is less than 1 or greater than 80 or the second operand (rightmost column to be checked) is less than the first operand (leftmost column to be checked).
- More or fewer than two operands are present, or an operand is null (empty).
- An operand expression contains an undefined symbol.
- An operand expression is not absolute.
- The statement is too complex. For example, it may have forward references or cause an arithmetic overflow during evaluation.
- The statement is circularly defined.

Assembler Action: Sequence checking is stopped.

Programmer Response: Supply valid ISEQ operands. Also, be sure that the cards following this statement are in order; they have not been sequence checked.

Severity Code: 4

IEV050 ILLEGAL NAME FIELD. NAME DISCARDED

Explanation: One of these errors has occurred:

- The name field of a macro prototype statement contains an invalid symbolic parameter (variable symbol).
- The name field of a COPY statement in a macro definition contains an entry other than blank or a valid sequence symbol.

Assembler Action: The invalid name field is ignored.

Programmer Response: Correct the invalid name field.

Severity Code: 8

IEV051 ILLEGAL STATEMENT OUTSIDE A MACRO DEFINITION

Explanation: A MEND, MEXIT, or AREAD statement appears outside a macro definition.

Assembler Action: The statement is ignored.

Programmer Response: Remove the statement or, if a macro definition is intended, insert a MACRO statement.

Severity Code: 8

IEV052 CARD OUT OF SEQUENCE

Explanation: Input sequence checking, under control of the ISEQ assembler instruction, has determined that this statement is out of sequence. The sequence number of the statement is appended to the message.

Assembler Action: The statement is assembled normally. However, the sequence number of the next statement will be checked relative to this statement.

Programmer Response: Put the statements in proper sequence. If you want a break in sequence, put in a new ISEQ statement with a high sequence number. ISEQ always resets the sequence number; the card following the ISEQ is not sequence checked.

Severity Code: 12

IEV053 BLANK SEQUENCE FIELD

Explanation: Input sequence checking, controlled by the ISEQ assembler statement, has detected a statement with a blank sequence field. The sequence number of the last numbered statement is appended to the message.

Assembler Action: The statement is assembled normally. The sequence number of the next statement will be checked relative to the last statement having a non-blank sequence field.

Programmer Response: Put the proper sequence number in the statement or discontinue sequence checking over the blank statements by means of an ISEQ statement with a blank operand.

Severity Code: 4

IEV054 ILLEGAL CONTINUATION CARD

Explanation: A statement has more than 10 cards or end-of-input has been encountered when a continuation card was expected.

Assembler Action: The cards already read are processed as is. If the statement had more than 10 cards, the next card is treated as the beginning of a new statement.

Programmer Response: In the first case break the statement into two or more statements. In the second case, ensure that a continued statement does not span the end of a library member. Check for lost cards or an extraneous continuation punch.

Severity Code: 8

IEV055 RECURSIVE COPY

Explanation: A nested COPY statement (COPY within another COPY) attempted to copy a library member already being copied by a higher level COPY within the same nest.

Assembler Action: This COPY statement is ignored.

Programmer Response: Correct the operand of this COPY if it is wrong or rearrange the nest so that the same library member is not copied by COPY statements at two different levels.

Severity Code: 12

IEV057 UNDEFINED OPERATION CODE

Explanation: One of the following errors has occurred:

- The operation code of this statement is not a valid machine or assembler instruction or macro name.
- In an OPSYN statement, this operand symbol is undefined or illegal or, if no operand is present, the name field symbol is undefined.

Assembler Action: The statement is ignored. Note that OPSYN does not search the macro library for an undefined operand.

Programmer Response: Correct the statement. In the case of an undefined macro instruction, the wrong data set may have been specified for the macro library. In the case of OPSYN, a previous OPSYN or macro definition may have failed to define the operation code.

Severity Code: 8

IEV059 ILLEGAL ICTL

Explanation: An ICTL statement has one of the following errors:

- The operation code was created by variable symbol substitution.
- It is not the first statement in the assembly.
- The value of one or more operands is incorrect.
- An operand is missing.
- An invalid character is detected in the operand field.

Assembler Action: The ICTL statement is ignored. Assembly continues with standard ICTL values.

Programmer Response: Correct or remove the ICTL. The begin column must be 1-40; the end column must be 41-80 and at least five greater than the begin column; and the continue column must be 2-40.

Severity Code: 16

IEV060 COPY CODE NOT FOUND

Explanation: (1) If this message is on a COPY statement and no text is printed with it, one of the following occurred:

- The library member was not found.
- The lookahead phase previously processed the COPY statement and did not find the library member, the copy was recursive, or the operand contains a variable symbol.

(2) If this message is not on a COPY statement, but has a library member name printed with it, the lookahead phase of the assembler could not find the library member because the name is undefined or contains a variable symbol.

Assembler Action: The COPY statement is ignored; the library member is not copied.

Programmer Response: Check that the correct macro library was assigned or check for a possible misspelled library member name. If the library member may be read by the lookahead phase of the assembler, do not make the library member name a variable symbol.

Severity Code: 12

IEV061 SYMBOL NOT NAME OF DSECT OR DXD

Explanation: The operand of a Q-type address constant is not a symbol or the name of a DSECT or DXD statement.

Assembler Action: The constant is assembled to zero.

Programmer Response: Supply a valid operand.

Severity Code: 8

IEV062 ILLEGAL OPERAND FORMAT

Explanation: One of the following errors has occurred:

- DROP or USING -- more than 16 registers were specified in the operand field.
- PUSH or POP -- an operand does not specify a PRINT or USING statement.
- PRINT -- an operand specifies an invalid print option.
- MNOTE -- the syntax of the severity code (first operand) is invalid.

Assembler Action: The first 16 registers in a DROP or USING statement are processed. The operand in error and the following operands of a PUSH, POP, or PRINT statement are ignored.

Programmer Response: Supply a valid operand field.

Severity Code: 8

IEV063 NO ENDING APOSTROPHE

Explanation: The quote terminating an operand is missing, or the standard value of a keyword parameter of a macro prototype statement is missing.

Assembler Action: The operand or standard value in error is ignored. If the error is in a macro definition model statement, the entire statement is ignored.

Programmer Response: Supply the missing quote.

Severity Code: 8

IEV064 FLOATING POINT CHARACTERISTIC OUT OF RANGE

Explanation: A converted floating-point constant is too large or too small for the CPU. The allowable range is 7.2×10^{75} to 5.3×10^{-77}.

Assembler Action: The constant is assembled to zero.

Programmer Response: Check the characteristic (exponent), exponent modifier, scale modifier, and mantissa (fraction) for validity. Remember that a floating-point constant is rounded, not truncated, after conversion.

Severity Code: 12

IEV065 UNKNOWN TYPE

Explanation: An unknown constant type has been used in a DC or DS statement or in a literal.

Assembler Action: The operand in error and the following operands are ignored.

Programmer Response: Supply a valid constant. Look for an incorrect type code or incorrect syntax in the duplication factor.

Severity Code: 8

IEV066 RELOCATABLE Y-TYPE CONSTANT

Explanation: This statement contains a relocatable Y-type address constant. A Y-constant is only two bytes long, so addressing errors will occur if this program is loaded at a main storage address greater than 32K (32,768).

Assembler Action: The statement is assembled as written.

Programmer Response: If this program will not be loaded at a main storage address greater than 32K, you can leave the Y-constant.

Severity Code: 4

IEV067 ILLEGAL DUPLICATION FACTOR

Explanation: One of the following errors has occurred:

- A literal has a zero duplication factor.
- The duplication factor of a constant is greater than $2^{24}-1$.
- A duplication factor expression of a constant is invalid.

Assembler Action: The operand in error and the following operands of a DC, DS, or DXD statement are ignored. The statement containing the literal is assembled to zero.

Programmer Response: Supply a valid duplication factor. If you want a zero duplication factor, write the literal as a DC statement.

Severity Code: 12

IEV068 LENGTH ERROR

Explanation: One of the following error has occurred:

- The length modifier of a constant is wrong.
- The C, X, B, Z, or P-type constant is too long.
- An operand is longer than $2^{24}-1$ bytes.
- A relocatable address constant has an illegal length.
- The length field in a machine instruction is invalid or out of the permissible range.

Assembler Action: The operand in error and the following operands of the DC, DS, or DXD statement are ignored, except that an address constant with an illegal length is truncated. A machine instruction is assembled to zero.

Programmer Response: Supply a valid length.

Severity Code: 12

IEV070 SCALE MODIFIER ERROR

Explanation: A scale modifier in a constant is used illegally, is out of range, or is relocatable, or there is an error in a scale modifier expression.

Assembler Action: If the scale modifier is out of range, it defaults to zero. Otherwise, the operand in error and the following operands are ignored.

Programmer Response: Supply a valid scale modifier.

Severity Code: 8

IEV071 EXPONENT MODIFIER ERROR

Explanation: The constant contains multiple internal exponents, the exponent modifier is out of range or relocatable, or the sum of the exponent modifier and the internal exponent is out of range.

Assembler Action: If the constant contains multiple internal exponents, the operand in error and the following operands are ignored. Otherwise, the exponent modifier defaults to zero.

Programmer Response: Change the exponent modifier or the internal exponent.

Severity Code: 8

IEV072 DATA ITEM TOO LARGE

Explanation: A Y-type address constant is larger than $2^{15}-1$ or smaller than -2^{15}, or the value of a decimal constant is greater than the number of bits (integer attribute) allocated to it.

Assembler Action: The constant is truncated. The high-order bits are lost.

Programmer Response: Supply a smaller scale modifier or a longer constant.

Severity Code: 8

IEV073 PRECISION LOST

Explanation: The scale modifier of a floating-point number was large enough to shift the entire fraction out of the converted constant.

Assembler Action: The constant is assembled with an exponent but with a zero mantissa (fraction).

Programmer Response: Change the scale modifier or use a longer constant. For example, use a D-type constant instead of an E-type constant.

Severity Code: 8

IEV074 ILLEGAL SYNTAX IN EXPRESSION

Explanation: An expression has two terms or two operators in succession, or invalid or missing characters or delimiters.

Assembler Action: In a DC, DS, or DXD statement the operand in error and the following operands are ignored. In a macro definition, the entire statement is ignored. A machine instruction is assembled to zero.

Programmer Response: Check the expression for keypunch errors, or for missing or invalid terms or characters.

Severity Code: 8

IEV075 ARITHMETIC OVERFLOW

Explanation: The intermediate or final value of an expression is not within the range -2^{31} through $2^{31}-1$.

Assembler Action: A machine instruction is assembled to zero. An assembler instruction is ignored.

Programmer Response: Change the expression.

Severity Code: 8

IEV076 STATEMENT COMPLEXITY EXCEEDED

Explanation: The complexity of this statement caused the assembler's expression evaluation work area to overflow.

Assembler Action: A machine instruction is assembled to zero. An assembler instruction is ignored.

Programmer Response: Reduce the number of terms, levels of expressions, or references to complex relocatable EQU names.

Severity Code: 8

IEV077 CIRCULAR DEFINITION

Explanation: The value of a symbol in an expression is dependent on itself, either directly or indirectly, via one or more EQU statements. For example,

```
A EQU B
B EQU C
C EQU A
```

A is circularly defined.

Assembler Action: The value of the EQU statement defaults to the current value of the location counter. All other EQU statements involved in the circularity are defaulted in terms of this one.

Programmer Response: Supply a correct definition.

Severity Code: 8

IEV079 ILLEGAL PUSH-POP

Explanation: More POP assembler instructions than PUSH instructions have been encountered.

Assembler Action: This POP instruction is ignored.

Programmer Response: Eliminate a POP statement or add another PUSH statement.

Severity Code: 8

IEV080 STATEMENT IS UNRESOLVABLE

Explanation: A statement cannot be resolved because it contains a complex relocatable expression or because the location counter has been circularly defined.

Assembler Action: The statement is ignored.

Programmer Response: Untangle the forward references or check the complex relocatable EQU statements.

Severity Code: 8

IEV081 CREATED SET SYMBOL EXCEEDS 63 CHARACTERS

Explanation: A SET symbol created by variable symbol substitution is longer than 63 characters (including the ampersand as the first character).

Assembler Action: If the symbol is in the operand field of a SET, AIF, or AGO statement, its value is set to zero or null, and the type attribute is set to undefined (U). If the symbol is in the operand field of a GBL or LCL statement or the name field of a SET statement, the macro is aborted.

Programmer Response: Shorten the symbol.

Severity Code: 8

IEV082 CREATED SET SYMBOL IS NULL

Explanation: A SET symbol created by variable symbol substitution is null (empty string).

Assembler Action: If the symbol is in the operand field of a SET, AIF, or AGO statement, its value is set to zero or null, and the type attribute is set to undefined (U). If the symbol is in the operand field of a GBL or LCL statement or the name field of a SET statement, the macro is aborted.

Programmer Response: Supply a valid symbol.

Severity Code: 8

IEV083 CREATED SET SYMBOL IS NOT A VALID SYMBOL

Explanation: A SET symbol created by variable symbol substitution or concatenation does not consist of an ampersand followed by up to 62 alphameric characters, the first of which is alphabetic.

Assembler Action: If the symbol is in the operand field of a SET, AIF, or AGO statement, its value is set to zero or null, and the type attribute is set to undefined (U). If the symbol is in the operand field of a GBL or LCL statement or the name field of a SET statement, the macro is aborted.

Programmer Response: Supply a valid symbol.

Severity Code: 8

IEV084 GENERATED NAME FIELD EXCEEDS 63 CHARACTERS. DISCARDED

Explanation: The name field on a generated statement is longer than 63 characters.

Assembler Action: The name field is not generated. The rest of the statement is assembled normally.

Programmer Response: Shorten the generated name to 63 characters or fewer.

Severity Code: 12

IEV085 GENERATED OPERAND FIELD IS NULL

Explanation: The operand field of a generated statement is null (empty).

Assembler Action: The statement is assembled as though no operand were specified.

Programmer Response: Provide a non-empty operand field. If you want the statement assembled with no operand, substitute a comma rather than leave the operand blank.

Severity Code: 0

IEV086 MISSING MEND GENERATED

Explanation: A macro definition, appearing in the source program or being read from a library by a macro call or a COPY statement, ends before a MEND statement is encountered to terminate it.

Assembler Action: A MEND statement is generated. The portion of the macro definition read in will be processed.

Programmer Response: Insert the MEND statement if it was left out. Otherwise, check if all the macro definition is on the library.

Severity Code: 12

IEV087 GENERATED OPERATION CODE IS NULL

Explanation: The operation code of a generated statement is null (blank).

Assembler Action: The generated statement is printed but not assembled.

Programmer Response: Provide a valid operation code.

Severity Code: 12

IEV088 UNBALANCED PARENTHESES IN MACRO CALL OPERAND

Explanation: Excess left or right parentheses occur in an operand (parameter) of a macro call statement.

Assembler Action: The parameter corresponding to the operand in error is given a null (empty) value.

Programmer Response: Balance the parentheses.

Severity Code: 8

IEV089 ARITHMETIC EXPRESSION CONTAINS ILLEGAL DELIMITER OR ENDS PREMATURELY

Explanation: An arithmetic expression contains an invalid character or an arithmetic subscript ends without sufficient right parentheses.

Assembler Action: The statement is ignored.

Programmer Response: Supply a valid expression.

Severity Code: 8

IEV090 EXCESS RIGHT PARENTHESIS IN MACRO CALL OPERAND

Explanation: A right parenthesis without a corresponding left parenthesis was detected in an operand of a macro instruction.

Assembler Action: The excess right parenthesis is ignored. The macro expansion may be incorrect.

Programmer Response: Insert the proper parenthesis.

Severity Code: 8

IEV091 SETC OR CHARACTER RELATIONAL OPERAND OVER 255 CHARACTERS. TRUNCATED TO 255 CHARACTERS

Explanation: The value of the operand of a SETC statement or the character relational operand of an AIF statement is longer than 255 characters.

Assembler Action: The first 255 characters are used.

Programmer Response: Shorten the SETC expression value or the operand value.

Severity Code: 8

IEV092 SUBSTRING EXPRESSION 1 POINTS PAST STRING END. DEFAULT=NULL

Explanation: The first arithmetic expression of a SETC substring points beyond the end of the expression character string.

Assembler Action: The substring is given a null value.

Programmer Response: Supply a valid expression.

Severity Code: 0

IEV093 SUBSTRING EXPRESSION 1 LESS THAN 1. DEFAULT=1

Explanation: The first arithmetic expression of a SETC substring is less than one; that is it points before the expression character string.

Assembler Action: The value of the expression is set to one.

Programmer Response: Supply a valid expression.

Severity Code: 4

IEV095 SUBSTRING EXPRESSION 2 LESS THAN 1. DEFAULT = NULL

Explanation: The second arithmetic expression of a SETC substring is less than or equal to zero.

Assembler Action: No characters (a null string) from the substring character expression are used.

Programmer Response: Supply a valid expression.

Severity Code: 4

IEV096 UNSUBSCRIPTED SYSLIST. DEFAULT=SYSLIST(1)

Explanation: The system variable symbol, &SYSLIST, is not subscripted. &SYSLIST(n) refers to the nth positional parameter in a macro instruction. Note that N'&SYSLIST does not have to be subscripted.

Assembler Action: The subscript defaults to one so that the first positional parameter will be referred to.

Programmer Response: Supply an appropriate subscript.

Severity Code: 8

IEV097 INVALID ATTRIBUTE REFERENCE TO SETA OR SETB SYMBOL. DEFAULT=U OR 0

Explanation: A type (T'), length (L'), scaling (S'), integer (I'), or defined (D') attribute refers to a SETA or SETB symbol.

Assembler Action: The attributes are set to default values: T'=U, L'=0, S'=0 I'=0 and D'=0

Programmer Response: Change or remove the attribute reference.

Severity Code: 8

IEV098 ATTRIBUTE REFERENCE TO INVALID SYMBOL. DEFAULT=U OR 0

Explanation: An attribute attempted to reference an invalid symbol. (A valid symbol is 1 to 63 alphameric characters, the first of which is alphabetic.)

Assembler Action: For a type (T') attribute, default to U. For all other attributes, default to 0.

Programmer Response: Supply a valid symbol.

Severity Code: 8

IEV099 I ATTRIBUTE REFERENCE IS NOT FOR COMPUTATIONAL TYPE OF CONSTANT. DEFAULT=0

Explanation: An integer (I') attribute references a symbol whose type is other than floating-point (E,D,L), decimal (P,Z), or binary (H,F).

Assembler Action: The integer attribute defaults to zero.

Programmer Response: Remove the integer attribute reference or change the constant type.

Severity Code: 4

IEV100 SUBSCRIPT LESS THAN 1. DEFAULT TO SUBSCRIPT = 1.

Explanation: The subscript of a subscripted SET symbol in the name field of a SET statement, the operand field of a GBL or LCL statement, or an &SYSLIST statement is less than 1.

Assembler Action: The subscript defaults to 1.

Programmer Response: Supply the correct subscript.

Severity Code: 8

IEV101 SUBSCRIPT LESS THAN 1. DEFAULT TO VALUE=0 OR NULL

Explanation: The subscript of a SET symbol in the operand field is less than 0.

Assembler Action: The subscript is set to 0.

Programmer Response: Supply a valid subscript.

Severity Code: 8

IEV102 ARITHMETIC TERM IS NOT SELF-DEFINING TERM. DEFAULT=0

Explanation: A SETC term or expression used as an arithmetic term is not a self-defining term.

Assembler Action: The value of the SETC term or expression is set to zero.

Programmer Response: Make the SETC a self-defining term, such as C'A', X'1EC', B'1101', or 27. Note that the C, X, or B and the quotes must be part of the SETC value.

Severity Code: 8

IEV103 MULTIPLICATION OVERFLOW. DEFAULT PRODUCT=1

Explanation: A multiplication overflow occurred in a macro definition statement.

Assembler Action: The value of the expression up to the point of overflow is set to one; evaluation is resumed.

Programmer Response: Change the expression so that overflow does not occur; break it into two or more operations, or regroup the terms by parentheses.

Severity Code: 8

IEV105 ARITHMETIC EXPRESSION TOO COMPLEX

Explanation: An arithmetic expression in a macro definition statement caused an overflow because it is too complex; that is, it has too many terms and/or levels.

Assembler Action: The assembly is terminated.

Programmer Response: Simplify the expression or break it into two or more expressions.

Severity Code: 20

IEV106 WRONG TARGET SYMBOL TYPE. VALUE LEFT UNCHANGED

Explanation: The SET symbol in the name field does not match its declared type (does not match the operation code): SETA, SETB, or SETC.

Assembler Action: The statement is ignored.

Programmer Response: Make the declaration agree with the SET statement type. If you want to store across types, store first into a SET symbol of matching type.

Severity Code: 8

IEV107 INCCNSISTENT DIMENSICN ON TARGET SYMBOL. SUBSCRIPT IGNORED OR 1 USED

Explanation: The SET symbol in the name field is dimensioned (subscripted), but was not declared in a GBL or LCL statement as dimensioned, or vice versa.

Assembler Action: The subscript is ignored or a subscript of 1 is used, in accordance with the declaration.

Programmer Response: Make the declaration and the usage compatible. Note that you can declare a SET symbol as dimensioned by using it, subscripted, in the name field of a SET statement.

Severity Code: 8

IEV108 INCONSISTENT DIMENSION ON SET SYMBOL REFERENCE. DEFAULT = 0, NULL, OR TYPE = U

Explanation: A SET symbol in the operand field is dimensioned (subscripted), but was not declared in a GBL or LCL statement as dimensioned, or vice versa.

Assembler Action: A value of zero or null is used for the subscript. If the type attribute of the SET symbol is being requested, it is set to U.

Programmer Response: Make the declaration and the usage compatible. Note that you can declare a SET symbol as dimensioned by using it, subscripted, in the name field of a SET statement.

Severity Code: 8

IEV109 MULTIPLE OPERANDS FOR UNDIMENSIONED SET SYMBOL. GETS LAST OPERAND

Explanation: Multiple operands were assigned to an undimensioned (unsubscripted) SET symbol.

Assembler Action: The SET symbol is given the value of the last operand.

Programmer Response: Declare the SET symbol as dimensioned or assign only one operand to it.

Severity Code: 8

IEV110 LIBRARY MACRO 1ST STATEMENT NOT --MACRO--OR COMMENT

Explanation: A statement other than a comment statement preceeded a MACRO statement in a macro definition read from a library.

Assembler Action: The macro definition is not read from the library. A corresponding macro call cannot be processed.

Programmer Response: Ensure that the library macro definition begins with a MACRO statement preceded (optionally) by comment statements only.

Severity Code: 12

IEV111 INVALID AIF OR SETB OPERAND FIELD

Explanation: The operand of an AIF or SETB statement either does not begin with a left parenthesis or is missing altogether.

Assembler Action: The statement is ignored.

Programmer Response: Supply a valid operand.

Severity Code: 12

IEV112 INVALID SEQUENCE SYMBOL

Explanation: One of the following errors has occurred:

- A sequence symbol doesn't begin with a period followed by one to 62 alphameric characters, the first being alphabetic.
- A sequence symbol in the name field was created by substitution.

Assembler Action: The sequence symbol in the name field is ignored. A sequence symbol in the operand field of an AIF or AGO statement causes the entire statement to be ignored.

Programmer Response: Supply a valid sequence symbol.

Severity Code: 12

IEV113 CONTINUE COLUMN BLANK

Explanation: A SET symbol declaration in a GBL or LCL statement began with an ampersand in the end column (normally column 71) of the previous card, but the continue column (normally column 16) of this card is blank.

Assembler Action: This card and any following cards of the statement are ignored. Any SET symbols appearing entirely on the previous card(s) are processed normally.

Programmer Response: Begin this card in the continuation column.

Severity Code: 12

IEV114 INVALID COPY OPERAND

Explanation: The operand of a COPY statement is not a symbol of one to eight alphameric characters, the first being alphabetic.

Assembler Action: The COPY statement is ignored.

Programmer Response: Supply a valid operand.

Severity Code: 12

IEV115 COPY OPERAND TOO LONG

Explanation: The symbol in the operand field of a COPY statement is more than eight characters long.

Assembler Action: The COPY statement is ignored.

Programmer Response: Supply a valid operand.

Severity Code: 12

IEV116 ILLEGAL SET SYMBOL

Explanation: A SET symbol in the operand field of a GBL or LCL statement or in the name field of a SET statement does not consist of an ampersand followed by one to 62 alphameric characters, the first being alphabetic.

Assembler Action: The invalid SET symbol and all following SET symbols in a GBL or LCL statement are ignored. The entire SET statement is ignored.

Programmer Response: Supply a SET symbol.

Severity Code: 8

IEV117 ILLEGAL SUBSCRIPT

Explanation: The subscript following a SET symbol contained unbalanced parentheses or an invalid arithmetic expression.

Assembler Action: This statement is ignored.

Programmer Response: Supply an equal number of left and right parentheses or a valid arithmetic expression.

Severity Code: 8

IEV118 SOURCE MACRO ENDED BY --MEND-- IN COPY CODE

Explanation: A library member, being copied by a COPY statement within a macro definition, contained a MEND statement. This terminated the definition.

Assembler Action: The MEND statement is ignored. No more COPY code is read. The statements brought in before the end of the COPY code are processed. The macro definition is resumed with the statement following the COPY statement.

Programmer Response: Make sure that each library member to be used as COPY code contains balanced MACRO and MEND statements.

Severity Code: 12

IEV119 TOO FEW MEND STATEMENTS IN COPY CODE

Explanation: A macro definition is started in a library member brought in by a COPY statement and the COPY code ends before a MEND statement is encountered.

Assembler Action: A MEND statement is generated to terminate the macro definition. The statements brought in before the end of the COPY code are processed.

Programmer Response: Check to see if part of the macro definition was lost. Also, ensure that each macro definition to be used as COPY code contains balanced MACRO and MEND statements.

Severity Code: 12

IEV120 EOD WHERE CONTINUE CARD EXPECTED

Explanation: An end-of-data occurred when a continuation card was expected.

Assembler Action: The portion of the statement read in is assembled. The assembly is terminated if the end-of-data is on SYSIN. If a library member is being copied, the assembly continues with the statement after the COPY statement.

Programmer Response: Check to determine whether any statements were omitted from the source program or from the COPY code.

Severity Code: 12

IEV121 INSUFFICIENT CORE FOR EDITOR WORK AREA

Explanation: The macro editor module of the assembler cannot get enough main storage for its work areas.

Assembler Action: The assembly is terminated.

Programmer Response: Split the assembly into two or more parts or give the macro editor more working storage. This can be done by increasing the region size for the assembler, decreasing blocking factor or blocksize on the assembler data sets, or a combination of both.

Severity Code: 12

IEV122 ILLEGAL OPERATION CODE FORMAT

Explanation: The operation code is not followed by a blank or is missing altogether, or the first card of a continued source statement is missing.

Assembler Action: The statement is ignored.

Programmer Response: Ensure that the statement has a valid operation code and that all cards of the statement are present.

Severity Code: 12

IEV123 VARIABLE SYMBOL TOO LONG

Explanation: A SET symbol, symbolic parameter, or sequence symbol contains more than 62 characters following the ampersand or period.

Assembler Action: This statement is ignored.

Programmer Response: Shorten the variable symbol or sequence symbol.

Severity Code: 12

IEV124 ILLEGAL USE OF PARAMETER

Explanation: A symbolic parameter was used in the operand field of a GBL or LCL statement or in the name field of a SET statement. In other words, a variable symbol has been used both as a symbolic parameter and as a SET symbol.

Assembler Action: The statement is ignored.

Programmer Response: Change the variable symbol to one which is not a symbolic parameter.

Severity Code: 12

IEV125 ILLEGAL MACRO NAME - MACRO UNCALLABLE

Explanation: The operation code of a macro prototype statement is not a valid symbol; that is, one to 63 alphameric characters, the first alphabetic.

Assembler Action: The macro definition is edited. However, since the macro name is invalid, the macro cannot be called.

Programmer Response: Supply a valid macro name.

Severity Code: 12

IEV126 LIBRARY MACRO NAME INCORRECT

Explanation: The operation code of the prototype statement of a library macro defintion is not the same as the operation code of the macro instruction (call). Library macro definitions are located by their member names. However, the assembler compares the macro instruction with the macro prototype.

Assembler Action: The macro defintion is edited using the operation code of the prototype statement as the macro name. Thus, the definition cannot be called by this macro instruction.

Programmer Response: Ensure that the member name of the macro defintion is the same as operation code of the prototype statement. This will usually require listing the macro definition from the library.

Severity Code: 12

IEV127 ILLEGAL USE OF AMPERSAND

Explanation: One of the following errors has occurred:

- An ampersand was found where all substitution should have already been performed.
- The standard value of a keyword parameter in a macro prototype statement contained a single ampersand or a string of ampersands whose length was odd.
- An unpaired ampersand occurred in a character (C) constant.

Assembler Action: In a macro prototype statement, all information following the error is ignored. In other statements, the action depends on which field the error occurred in. If the error occurred in the name field, the statement is processed without a name. If the error occurred in the operation code field, the statement is ignored. If the error occurred in the operand field, another message is issued to specify the default. However, if the error occurred in a C-type constant, the operand in error and the following operands are ignored.

Programmer Response: Ensure that ampersands used in keyword standard values or in C-type constants occur in pairs. Also, avoid substituting an ampersand into a statement unless there is a double ampersand.

Severity Code: 12

IEV128 EXCESS RIGHT PARENTHESIS

Explanation: An unpaired right parenthesis has been found.

Assembler Action: A machine instruction is assembled to zero. An assembler instruction is ignored and an additional message relative to the statement type appears. However, if the error is in the standard value of a keyword on a macro prototype statement, only the the operands in error and the following operands are ignored.

Programmer Response: Make sure that all parentheses are paired.

Severity Code: 12

IEV129 INSUFFICIENT RIGHT PARENTHESES

Explanation: An unpaired left parenthesis has been found. Note that parentheses must balance at each comma in a multiple operand statement.

Assembler Action: A machine instruction is assembled to zero. An assembler instruction is ignored and an additional message relative to the statement type will appear. However, if the error is in the standard value of a keyword on a macro prototype statement, only the operands in error and the following operands are ignored.

Programmer Response: Make sure that all parentheses are paired.

Severity Code: 12

IEV130 ILLEGAL ATTRIBUTE REFERENCE

Explanation: One of the following errors has occurred:

- The symbol following a D, I, L, S, or T attribute reference is not a valid variable symbol or ordinary symbol.
- The symbol following a K or N attribute reference is not a valid ordinary symbol.
- The quote is missing from a T attribute reference.

Assembler Action: The statement is ignored.

Programmer Response: Supply a valid attribute reference.

Severity Code: 12

IEV131 PARENTHESIS NESTING DEPTH EXCEEDS 255

Explanation: There are more than 255 levels of parentheses in a SETA expression.

Assembler Action: The statement is ignored.

Programmer Response: Rewrite the SETA statement using several statements to regroup the sub-expressions in the expression.

Severity Code: 12

IEV132 INVALID SETB EXPRESSION

Explanation: A SETB expression in the operand field of a SETB statement or an AIF statement does not consist of valid character relational expressions, arithmetic relational expressions, and single SETB symbols, connected by logical operators.

Assembler Action: The statement is ignored.

Programmer Response: Supply a valid SETB expression.

Severity Code: 12

IEV133 ILLEGAL SUBSTRING REFERENCE

Explanation: A substring expression following a SETC expression does not consist of two valid SETA expressions separated by a comma and enclosed in parentheses.

Assembler Action: The statement is ignored.

Programmer Response: Supply a valid substring expression.

Severity Code: 12

IEV134 INVALID RELATIONAL CPERATCR

Explanaticn: Characters cther than EQ, NE, LT, GT, LE, or GE are used in a SETB expression where a relational operator is expected.

Assembler Action: The statement is ignored.

Programmer Response: Supply a valid relational operator.

Severity Code: 12

IEV135 INVALID LOGICAL OPERATOR

Explaration: Characters other than ANL, OR, or NOT are used in a SETB expressicn where a logical operator is expected.

Assembler Action: The statement is ignored.

Prcgrammer Response: Supply a valid logical operator.

Severity Code: 12

IEV136 ILLEGAL LOGICAL/RELATIONAL OPERATOR

Explanation: Characters other than a valid logical or relaticnal operatcr are used in a SETB expression where a lcgical or relaticnal cperator is expected.

Assembler Action: The statement is ignored.

Programmer Response: Supply a valid logical or relational cperator.

Severity Code: 12

IEV137 ILLEGAL SETC EXPRESSION

Explanation: A SETC expression is not a valid type attribute (T') reference or a valid character expression enclosed in quotes.

Assembler Action: The statement is ignored.

Programmer Response: Supply a valid expression.

Severity Code: 12

IEV139 EOD DURING REPRO PROCESSING

Explanation: A REPRO statement was immediately followed by an end-of-data sc that nc valid card cculd be punched. The REPRO is either the last card of source input or the last card cf a COPY member.

Assembler Action: The REPRO statement is ignored.

Programmer Response: Remove the REPRO or ensure that it is followed by a card to be punched.

Severity Code: 12

IEV140 END CARD MISSING

Explanation: End-of-file on the source input data set occurred before an END statement was read. One of the following has occurred:

- The END statement was omitted or misspelled.
- The END operation code was changed or deleted by OPSYN or by definition of a macro named END. The lookahead phase of the assembler marks what it thinks is the END statement. If an OPSYN statement or a macro definition redefines the END statement, premature end-of-input may occur because the assembler will not pass the original END statement.

Assembler Action: An END statement is generated. It is assigned a statement number but not printed. If any literals are waiting, they will be processed as usual following the END statement.

Programmer Action: Check for lost cards. Supply a valid END statement; or, if you use OPSYN to define another symbol as END, place it prior to possible entry into the lookahead phase.

Severity Code: 4

IEV141 BAD CHARACTER IN OPERATION CODE

Explanation: The operation code contains a non-alphameric character, that is, a character other than A-Z, 0-9, $, #, or @. Embedded blanks are not allowed.

Assembler Action: The statement is ignored.

Programmer Response: Supply a valid operation code. If the operation code is formed by variable symbol substitution, check the statements leading to substitution.

Severity Code: 8

IEV142 OPERATION CODE NOT COMPLETE ON FIRST CARD

Explanation: The entire name and operation code, including a trailing blank, is not contained on the first card (before the continue column -- usually column 72) of a continued statement.

Assembler Action: The statement is ignored.

Programmer Response: Shorten the name and/or the operation code or simplify the statement by using a seperate SETC statement to create the name or operation code by substitution.

Severity Code: 8

IEV143 BAD CHARACTER IN NAME FIELD

Explanation: The name field contains a non-alphameric character, that is, a character other than A-Z, 0-9, $, #, or @.

Assembler Action: If possible, the statement is processed without a name. Otherwise, it is ignored.

Programmer Response: Put a valid symbol in the name field.

Severity Code: 8

IEV144 BEGIN-TO-CONTINUE COLUMNS NOT BLANK

Explanation: On a continuation card, one or more columns between the begin column (usually column 1) and the continue column (usually column 16) are not blank.

Assembler Action: The extraneous characters are ignored.

Programmer Response: Check whether the operand started in the wrong column or whether the preceding card contained an erroneous continue punch.

Severity Code: 8

IEV145 OPERATOR, RIGHT PARENTHESIS, OR END-OF-EXPRESSION EXPECTED

Explanation: One of the following has occurred:

- A letter, number, equals sign, quote, or undefined character occurred following a term where a right parenthesis, an operator, a comma, or a blank ending the expression was expected.
- In an assembler instruction, a left parenthesis followed a term.

Assembler Action: A machine instruction is assembled to zero. An assembler instruction is ignored and another message, relative to the operation code, is issued.

Programmer Response: Check for an omitted or mispunched operator. Subscripting is not allowed on this statement.

Severity Code: 8

IEV146 SELF-DEFINING TERM TOO LONG OR VALUE TOO LARGE

Explanation: A self-defining term is longer than four bytes, (8 hexadecimal digits, 32 bits, or 4 characters), or the value of a decimal self-defining term is greater than $2^{31}-1$.

Assembler Action: A machine instruction is assembled to zero. An assembler instruction is ignored. However, another message, relative to the operation code, is issued.

Programmer Response: Reduce the size or value of the self defining term or specify it in a DC statement.

Severity Code: 8

IEV147 SYMBOL TOO LONG, OR 1ST CHARACTER NOT A LETTER

Explanation: A symbol does not begin with a letter or is longer than 63 characters.

Assembler Action: If the symbol is in the name field, the statement is processed as unnamed. If the symbol is in the operand field, an assembler operation or a macro definition model statement is ignored and a machine operation is assembled to zero.

Programmer Response: Supply a valid symbol.

Severity Code: 8

IEV148 SELF-DEFINING TERM LACKS ENDING QUOTE OR HAS BAD CHARACTER

Explanation: A hexadecimal or binary self-defining term contains an invalid character or is missing the final quote.

Assembler Action: A machine operation is assembled to zero. An assembler operation is ignored and another message, relative to the operation code, is issued.

Programmer Response: Correct the invalid term.

Severity Code: 8

IEV149 LITERAL LENGTH EXCEEDS 256 CHARACTERS, INCLUDING EQUAL SIGN

Explanation: A literal is longer than 256 characters.

Assembler Action: The instruction is assembled to zero.

Programmer Response: Shorten the literal or change it to a DC statement.

Severity Code: 8

IEV150 SYMBOL HAS NON-ALPHAMERIC CHARACTER OR INVALID DELIMITER

Explanation: The first character following a symbol is not a valid delimiter, (plus sign, minus sign, asterisk, slash, left or right parentheses, comma, or blank).

Assembler Action: A machine operation is assembled to zero. An assembler operation is ignored, and another message, relative to this operation code, is issued.

Programmer Response: Ensure that the symbol does not contain a non-alphameric character or that it is followed by a valid delimiter.

Severity Code: 8

IEV151 LITERAL EXPRESSION MODIFIERS MUST BE ABSOLUTE AND PREDEFINED

Explanation: The duplication factor or length modifier in a literal is not (1) a self-defining term or (2) an expression using self-defining terms or previously-defined symbols.

Assembler Action: The statement is assembled to zero.

Programmer Response: Supply a valid self-defining term or ensure that symbols appear in the name field of a previous statement.

Severity Code: 8

IEV152 EXTERNAL SYMBOL TOO LONG OR UNACCEPTABLE CHARACTER

Explanation: One of the following errors has occurred:

- An external symbol is longer than 8 characters. An external symbol might be the name of a CSECT, START, DXD, or COM statement, or the operand of an ENTRY or EXTRN statement or a Q-type or V-type address constant.
- The operand of an ENTRY or EXTRN statement of a Q-type or V-type address constant is an expression instead of a single term, or contains a bad character.

Assembler Action: The symbol does not appear in the External Symbol Dictionary. If the error is in the name field, an attempt is made to process the statement as unnamed. If the error is in the operand field, the bad operand is ignored and, if possible, the following operands are processed. A bad constant is assembled to zero.

Programmer Response: Supply a shorter name or replace the expression with a term.

Severity Code: 12

IEV153 START STATEMENT ILLEGAL - CSECT ALREADY BEGUN

Explanation: A START statement occured after the beginning or a control section.

Assembler Action: The statement is processed as a CSECT statement; any operand is ignored.

Programmer Response: Ensure that the START precedes all machine instructions and any assembler instruction, such as EQU, that initiates a control section. If you want EQU statements before the START, place them in a dummy section (DSECT).

Severity Code: 12

IEV154 OPERAND MUST BE ABSOLUTE, PREDEFINED SYMBOLS. SET TO 0

Explanation: The operand on a START or MHELP statement is invalid. If there is another message with this statement, this message is advisory. If this message appears alone, it indicates one of the following:

- There is a location counter reference (*) in a START operand.
- An expression does not consist of absolute terms and/or predefined symbols.

- The statement is too complex. For example, it may have too many forward references or cause arithmetic overflow during evaluation.
- The statement is circularly defined.
- A relocatable term is multiplied or divided.

Assembler Action: The operand of the statement is treated as zero.

Programmer Response: Correct the error if it exists. Note than paired relocatable symbols in different LOCTRs, even though in the same CSECT or DSECT, are not valid where an absolute, predefined value is required.

Severity Code: 8

IEV155 PREVIOUS USE OF SYMBOL IS NOT THIS SECTION TYPE

Explanation: The name on a CSECT, DSECT, COM, or LOCTR statement has been used previously, on a different type of statement. For example, the name on a CSECT has been used before on a statement other than CSECT, such as a machine instruction or a LOCTR.

Assembler Action: This name is ignored and the statement is processed as unnamed.

Programmer Response: Correct the misspelled name or change the name to one that does not conflict.

Severity Code: 12

IEV156 ONLY ORDINARY SYMBOLS, SEPERATED BY COMMAS, ALLOWED

Explanation: The operand field of an ENTRY or EXTRN statement contains a symbol which does not consist of one to eight alphameric characters, the first being alphabetic, or the operands are not separated by a comma.

Assembler Action: The operand in error is ignored. If other operands follow, they will be processed normally.

Programmer Response: Supply a correct symbol or insert the missing comma. If you want an expression as an ENTRY statement operand (such as SYMBOL+4), use an EQU statement to define an additional symbol.

Severity Code: 12

IEV157 OPERAND MUST BE A SIMPLY-RELOCATABLE EXPRESSION

Explanation: If there is another message with this statement, this message is advisory. If this message appears alone, the operand of an ORG or END statement is not a simply relocatable expression, is too complex, or is circularly defined. The error may also be that the END operand symbol is not in a CSECT.

Assembler Action: An ORG statement is ignored. The operand of an END statement is ignored.

Programmer Response: If an error exists, supply a correct expression. Note that paired relocatable symbols in different LOCTRs, even though in the same CSECT or DSECT, may cause circular definition when used in an ORG statement.

Severity Code: 12

IEV158 OPERAND 1 EXPRESSION IS DEFECTIVE. SET TO *

Explanation: The first operand of an EQU statement is defective. If another message appears with this statement, this message is advisory. If this message appears alone, one of the following errors has occurred:

- The statement is too complex. For example, it has too many forward references or causes an arithmetic overflow during evaluation.
- The statement is circularly defined.
- The statement contains a relocatable term which is multiplied or divided.

Assembler Action: The symbol in the name field is equated to the current value of the location counter (*), and operands 2 and 3 or the statement, if present, are ignored.

Programmer Response: If an error exists, supply a correct expression for operand 1 of the statement.

Severity Code: 8

IEV159 OPERANDS MUST BE ABSOLUTE, PROPER MULTIPLES OF 2 OR 4

Explanation: The combination of operands of a CNOP statement is not one of the following valid combinations:

0,4	2,4
0,8	2,8
4,8	6,8

Assembler Action: The statement is ignored. However, the location counter is adjusted to a halfword boundary.

Programmer Response: Supply a valid combination of CNOP operands.

Severity Code: 12

IEV161 ONLY ONE TITLE CARD MAY HAVE A NAME FIELD

Explanation: More than one TITLE statement has a name field. The named TITLE statement need not be the first one in the assembly, but it must be the only one named.

Assembler Action: The name on this TITLE statement is ignored. The name used for deck identification is taken from the first named TITLE statement encountered.

Programmer Response: Delete the unwanted name.

Severity Code: 4

IEV162 PUNCH OPERAND EXCEEDS 80 COLUMNS. IGNORED

Explanation: A PUNCH statement attempted to punch more than 80 characters into a card.

Assembler Action: The statement is ignored. The card is not punched.

Programmer Response: Shorten the operand to 80 characters or fewer or use more than one PUNCH statement.

Severity Code: 12

IEV163 OPERAND NOT PROPERLY ENCLOSED IN QUOTES

Explanation: The operand of a PUNCH or TITLE statement does not begin with a quote, or the operand of a PUNCH, MNOTE, or TITLE statement does not end with a quote, or the ending quote is not followed by a blank.

Assembler Action: The statement is ignored.

Programmer Response: Supply the missing quote. Be sure than a quote to be punched as data is represented as two quotes.

Severity Code: 8

IEV164 VACUOUS OPERAND - CARD NOT PUNCHED

Explanation: A PUNCH statement does not have any characters between its two quotes, or a quote to be punched as data is not represented by two quotes.

Assembler Action: The statement is ignored.

Programmer Response: Correct the operand, if you want to "punch" a blank card, the operand of the PUNCH statement should be a blank enclosed in quotes.

Severity Code: 12

IEV165 UNEXPECTED NAME FIELD

Explanation: The assembler operation has a name and the name field should be blank.

Assembler Action: The name is equated to the current value of the location counter (*). However, if no control section has been started, the name is equated to zero.

Programmer Response: Remove the name. Check that the period was not omitted from a sequence symbol.

Severity Code: 4

IEV166 SEQUENCE SYMBCL TOO LONG

Explanation: A sequence symbol contains more than 62 characters following the period.

Assembler Action: If the sequence symbol is in the name field, the statement is processed without a name. If it is in the operand field of an AIF or AGO statement, the entire statement is ignored.

Programmer Response: Shorten the sequence symbol.

Severity Code: 12

IEV167 REQUIRED NAME MISSING

Explanation: This statement requires a name and has none. The name field may be blank because an error occurred during an attempt to create the name by substitution or because a sequence symbol was used as the name.

Assembler Action: The statement is ignored.

Programmer Response: Supply a valid name or ensure that a valid name is created by substitution. If a sequence symbol is needed, put it on an ANOP statement ahead of this one and put a name on this statement.

Severity Code: 8

IEV168 UNDEFINED SEQUENCE SYMBCL

Explanation: The sequence symbol in the operand field of an AIF or AGO statement outside a macro definition is not defined; that is, it does not appear in the name field of an appropriate statement.

Assembler Action: This statement is ignored; assembly continues with the next statement.

Programmer Response: If the sequence symbol is misspelled or omitted, correct it. Note that when the sequence symbol is not previously defined, the assembler looks ahead for the definitions. The lookahead stops when an END statement or an OPSYN equivalent is encountered. Be sure that OPSYN statements and macro definitions which redefine END precede possible entry into lookahead.

Severity Code: 16

IEV170 INTERLUDE ERROR-ICGGING CAPACITY EXCEEDED

Explanation: The table which the interlude phase of the assembler uses to keep track of the errors it detects is full. This does not stop error detection by other phases of the assembler.

Assembler Action: If there are additional errors, normally detected by the interlude phase, in other statements either before or after this one, they will not be flagged. Statement processing depends on the type of error.

Programmer Response: Correct the indicated errors and run the assembly again to diagnose any further errors.

Severity Code: 12

IEV171 STANDARD VALUE TOO LONG

Explanation: The standard (default) value of a keyword parameter on a macro prototype statement is longer than 255 characters.

Assembler Action: The parameter in error and the following parameters are ignored.

Programmer Response: Shorten the standard value.

Severity Code: 12

IEV172 NEGATIVE DUPLICATION FACTOR. DEFAULT = 1

Explanation: The duplication factor of a SETC statement is negative.

Assembler Action: The duplication factor is given a default value of 1.

Programmer Response: Supply a positive duplication factor.

Severity Code: 8

IEV173 DELIMITER ERROR, EXPECT BLANK

Explanation: Another character, such as a comma or a quote, is used where a blank (end of operand) is required.

Assembler Action: A machine instruction is assembled to zero. An ORG statement is ignored. For an EQU or END statement, the invalid delimiter is ignored and the operand is processed normally. For a CNOP statement, the location counter is aligned to a halfword boundary.

Programmer Response: Replace the invalid delimiter with a blank. Look for an extra operand or a missing left parenthesis.

Severity Code: 12

IEV174 DELIMITER ERROR, EXPECT BLANK OR COMMA

Explanation: Another character, such as a quote or ampersand, is used where a blank or a comma is required.

Assembler Action: A machine instruction is assembled to zero. For a USING or DROP statement, the invalid delimiter is ignored and the operand is processed normally.

Programmer Response: Replace the invalid delimiter with a blank or a comma. Look for an extra operand or a missing left parenthesis.

Severity Code: 12

IEV175 DELIMITER ERROR, EXPECT COMMA

Explanation: Another character, such as a blank or a parenthesis, is used where a comma is required.

Assembler Action: A machine instruction is assembled to zero. For a CNOP statement, the location counter is aligned to a halfword boundary.

Programmer Response: Replace the invalid delimiter with a comma. Be sure each expression is syntactically correct and no parentheses are omitted.

Severity Code: 12

IEV176 DELIMITER ERROR, EXPECT COMMA OR LEFT PARENTHESIS

Explanation: Ancther character, such as a blank or a right parenthesis, is used in a machine instruction where a comma or a left parenthesis is required.

Assembler Action: The machine instruction is assembled to zero.

Programmer Response: Replace the invalid delimiter with a comma or a left parenthesis. Look for invalid syntax or invalid base or length fields on the first operand.

Severity Code: 12

IEV177 DELIMITER ERROR, EXPECT BLANK OR LEFT PARENTHESIS

Explanation: Another character, such as a comma or a right parenthesis, is used in a machine instruction when a blank or left parenthesis is required.

Assembler Action: The machine instruction is assembled to zero.

Programmer Response: Replace the invalid delimiter with a blank or a left parenthesis. Look for invalid punctuation or invalid length, index, or base field.

Severity Code: 12

IEV178 DELIMITER ERROR, EXPECT COMMA OR RIGHT PARENTHESIS

Explanation: Another character, such as a blank or a left parenthesis, is used in a machine instruction when a comma or a right parenthesis is required.

Assembler Action: The machine instruction is assembled to zero.

Programmer Response: Replace the invalid delimiter with a comma or a right parenthesis. Look for a missing base field.

Severity Code: 12

IEV179 DELIMITER ERROR, EXPECT RIGHT PARENTHESIS

Explanation: Another character, such as a blank or a comma, is used in a machine instruction when a right parenthesis is required.

Assembler Action: The machine instruction is assembled to zero.

Programmer Response: Replace the invalid delimiter with a right parenthesis. Look for an index field used where it is not allowed.

Severity Code: 12

IEV180 OPERAND MUST BE ABSOLUTE

Explanation: The operand of a SPACE statement or the first, third, or fourth operand of a CCW statement is not an absolute term.

Assembler Action: A SPACE statement is ignored. A CCW statement is assembled to zero.

Programmer Response: Supply an absolute operand. Note that paired relocatable terms may span LOCTRs but must be in the same control section.

Severity Code: 12

IEV181 CCW OPERAND VALUE IS OUTSIDE ALLOWABLE RANGE

Explanation: One or more operands of a CCW statement are not within the following limits:

- 1st operand -- 0-255
- 2nd operand -- 0-16,775,215
- 3rd operand -- 0-255 and a multiple of 8
- 4th operand -- 0-65,535

Assembler Action: The CCW is assembled to zero.

Programmer Response: Supply valid operands.

Severity Code: 12

IEV182 OPERAND 2 MUST BE ABSOLUTE, 1-65536. IGNORED

Explanation: If there is another message with this statement, this message is advisory. If this message appears alone, the second operand of an EQU statement contains one of the following errors:

- It is not a absolute term or expression whose value is within the range of 1 to 65,536.
- It contains a symbol which is not previously defined.

- It is circularly defined.
- It is too complex; for example, it causes an arithmetic overflow during evaluation.

Assembler Action: Operand 2 is ignored and the length attribute of the first operand is used. If the third operand is present, it will be processed normally.

Programmer Response: Correct the error if it exists. Note that paired relocatable symbols in different LOCTRs, even though in the same CSECT, are not valid where an absolute, predefined value is required.

Severity Code: 8

IEV183 OPERAND 3 MUST BE ABSOLUTE, 0-255. IGNORED

Explanation: If there is another message with this statement, this message is advisory. If this message appears alone, the third operand of an EQU statement contains one of the following errors:

- It is not an absolute term or expression whose value is within the range of 0 to 255.
- It contains a symbol which is not previously defined.
- It is circularly defined.
- It is too complex; for example, it causes an arithmetic overflow during evaluation.

Assembler Action: The third operand is ignored and the type attribute of the EQU statement is set to U.

Programmer Response: Correct the error if it exists. Note that paired relocatable symbols in different LOCTRs, even though in the same CSECT, are not valid where an absolute, predefined value is required.

Severity Code: 8

IEV184 COPY DISASTER

Explanation: The assembler copied a library member (executed a COPY statement) while looking ahead for attribute references. However, when the complete text was analyzed, the COPY operation code had been changed by an OPSYN statement or "swallowed" by an AREAD statement, and the COPY should have not been executed. (Lookahead phase ignores OPSYN statements.) This message will follow the first card of the COPY code.

Assembler Action: The library member will be assembled. If it included an ICTL statement, the format of that ICTL will be used.

Programmer Response: Move COPY statements, or OPSYN statements that modify the meaning of COPY, to a point in the assembly prior to possible entry into lookahead mode.

Severity Code: 16

IEV253 TOO MANY ERRORS

Explanation: No more error messages can be issued for this statement because the assembler work area where the errors are logged is full.

Assembler Action: If more errors are detected for this statement, the messages and/or annotated text are discarded.

Programmer Response: Correct the indicated errors and rerun the assembly. If there are more errors on this statement, they will be detected in the next assembly.

Severity Code: 16

IEV254 *** MNOTE ***

Explanation: The text of an MNOTE statement, which is appended to this message, has been generated by your program or by a macro definition or a library member copied into your program. An MNOTE statement enables a source program or macro definition to signal the assembler to generate an error or informational message.

Assembler Action: None.

Programmer Response: Investigate the reason for the MNOTE. Errors flagged by MNOTE will often cause unsuccessful execution of the program.

Severity Code: An MNOTE is assigned a severity code of 0 to 255 by the writer of the MNOTE statement.

This page intentionally left blank.

IEV950 END OF STATEMENT FLAG WAS EXPECTED IN MACRO EDITED TEXT, BUT WAS NOT FOUND - MACRO EDITOR IS SUSPECT

IEV951 THE MACRO GENERATOR HAS ENCOUNTERED UNTRANSLATABLE MACRO EDITED TEXT

IEV952 BAD SET SYMBOL NAME FIELD OR LCL/GBL OPERAND - CHECK THE MACRO EDITED TEXT

IEV953 BAD SUBSCRIPT ON SET SYMBCL - CHECK THE MACRO EDITED TEXT

IEV954 CHARACTER EXPRESSION FOLLOWED BY BAD SUBSCRIPTS - CHECK THE MACRO EDITED TEXT

IEV955 A RIGHT PARENTHESIS WITH NO MATCHING LEFT PARENTHESIS WAS FOUND IN AN EXPRESSION - CHECK THE MACRO EDITED TEXT

IEV956 MULTIPLE SUBSCRIPTS OR BAD SET SYMBOL TERMINATOR - CHECK THE MACRO EDITED TEXT

IEV957 BAD TERMINATOR ON CREATED SET SYMBOL - CHECK THE MACRO EDITED TEXT

IEV958 BAD TERMINATOR ON PARAMETER - CHECK THE MACRO EDITED TEXT

IEV959 UNEXPECTED END OF DATA ON H-ASSEMBLER WORK FILE (SYSUT1) - INTERNAL CORE MANAGEMENT IS SUSPECT

IEV960 A BAD INTERNAL FILE NUMBER HAS BEEN PASSED TO THE xxxxx INTERNAL CORE MANAGEMENT ROUTINE

IEV961 AN INVALID CORE REQUEST HAS BEEN MADE, OR THE FREE CORE CHAIN POINTERS HAVE BEEN DESTROYED

Explanation: The assembly is terminated because of one of the errors described in IEV950-IEV961. This usually is caused by a bug in the assembler itself. Under certain conditions, however, the assembly can be rerun successfully.

Assembler Action: A special abnormal termination dump (Assembler H Interrupt and Diagnostic Dump) follows the message. Depending on where the error occurred, the assembly listing up to the bad statement may also be produced. The dump usually indicates which statement caused termination. It also may include contents of the assembler registers and work areas and other status information for use by IBM or your assembler maintenance programmers in determining the cause of the termination.

Programmer Response: Check the statement that caused termination. Correct any errors in it or, especially if the statement is long or complex, rewrite it. Reassemble the program; it may assemble correctly. However, even if it reassembles without error, there may be a bug in the assembler. Save the abnormal termination dump, the assembly listing (if one was produced), and the input deck and give them to your IBM customer engineer. Also, if the program assembles correctly, submit a copy of the listing and input deck of the correct assembly. This information may be helpful in diagnosing and fixing the assembler bug. The Assembler H Program Logic Manual, Order Number LY26-3760, contains a complete description of the purpose and format of the assembler's abnormal termination dump.

Severity Code: 20

IEV962 STAE HAS INTERCEPTED A SYSTEM xxx OR A USER xxx ABEND

Explanation: The assembly is terminated by a program interruption (system 0C1-0CF ABEND completion code) or by inclusion of a QUIT1, QUIT2, or QUIT3 instruction in the assembly (user 201-203 abend completion code). The program interruption is usually caused by a bug in the assembler itself. Under certain conditions, however, the assembly can be rerun successfully. The QUIT instructions can be used to diagnose a suspected assembler bug.

Assembler Action: A special abnormal termination dump (Assembler H Interrupt and Diagnostic Dump) follows the message. This is followed by an Operating System ABEND dump (if you included a SYSABEND or SYSUDUMP statement). Depending on where the error occurred, the assembly listing up to the bad statement may also be produced. The dump usually indicates which statement caused termination. It also may include contents of the assembler registers and work areas and other status information for use by IBM or your assembler maintenance programmers in determining the cause of the termination.

Programmer Response: If this is a program interruption, check the statement that caused termination. Correct any errors in it or, especially if the statement is long or complex, rewrite it. Reassemble the program; it may assemble correctly. However, even if the program reassembles without error, there may be a bug in the assembler. Save the abnormal termination dump, the assembly listing (if one was produced), and the input deck and give them to your IBM customer engineer. Also, if the program assembles correctly, submit a copy of the listing and input deck of the correct assembly. This information may be helpful in diagnosing and fixing the assembler bug. The Assembler H Program Logic Manual, Order Number LY26-3760, contains has a complete description of the purpose and format of the assembler's abnormal termination dump.

Severity Code: Not used.

IEV970 STATEMENT COMPLEXITY EXCEEDED, BREAK THE STATEMENT INTO SEGMENTS AND RERUN THE ASSEMBLY

Explanation: The statement is too complex to be evaluated by the macro generator phase of the assembler. It overflowed the evaluation work area of the assembler. Normally, there is no assembler malfunction; the statement can be corrected and the program reassembled successfully.

Assembler Action: A special abnormal termination dump (Assembler H Interrupt and Diagnostic Dump) follows the message. The statement causing termination is SETA, SETB, SETC, AGO, or AIF. The dump does not indicate which statement caused termination; however, it may show the last statement generated in the macro. The dump may also include contents of the assembler registers and work areas and other status information for use by IBM or your assembler maintenance programmers in determining the cause of the termination. However, it will not be needed unless the error persists. This information may be helpful in diagnosing and fixing an assembler bug.

Programmer Response: Check the statement that caused termination. Rewrite the statement or break into two or more statements. Reassemble the program; it should assemble correctly. However, if the error persists, there may be an assembler malfunction. Save the abnormal termination dump, the assembly listing (if one was produced), and the input deck and give them to your IBM customer engineer. The Assembler H Program Logic Manual, Order Number LY26-3760, contains has a complete description of the purpose and format of the assembler's abnormal termination dump.

Severity Code: 20

IEV971 INSUFFICIENT CORE AVAILABLE FOR MACRO EDITOR WORK AREA

IEV972 NO AVAILABLE STORAGE REMAINS - ALLOCATE MORE CORE OR BREAK THE INPUT INTO MULTIPLE ASSEMBLIES

Explanation: The assembler work areas are full and none of the contents can be spilled onto the auxiliary data set (SYSUT1). Note that the load modules and fixed data areas of the assembler require about 96K bytes of main storage. The rest of the assembler's region is used for data set buffers, assembler internal files, and work areas. Some of the internal files, like the symbol table, must remain in main storage throughout the assembly.

Assembler Action: A special abnormal termination dump (Assembler H Interrupt and Diagnostic Dump) follows the message. Depending on where the error occurred, the assembly listing up to the bad statement may also be produced. The dump usually indicates the statement being processed when the assembler ran out of main storage. The other information in the dump, such as register and work area contents, is not needed.

Programmer Response: Increase the region size or split the assembly into two or more assemblies. Complete information on these and other possible remedies, such as decreasing the storage used for data set buffers, is in the "Storage Estimates" chapter of the Assembler H System Information Manual, Order Number SC26-3768.

Severity Code: 20

IEV980 SYSUT1 IS REQUIRED TO BE ASSIGNED TO A DIRECT ACCESS DEVICE, BUT WAS NOT

IEV981 THE DD STATEMENTS FOR SYSIN AND SYSUT1 WERE MISSING OR INVALID

IEV982 THE DD STATEMENT FOR SYSIN WAS MISSING OR INVALID

IEV983 THE DD STATEMENT FOR SYSUT1 WAS MISSING OR INVALID

Explanation: The DD statement for the data sets indicated in IEV980-IEV983 have not been included in the job control language for the assembly job step or it is invalid.

Assembler Action: The assembly is not done because the assembler does not have the required data sets. This message appears alone, without any other abnormal termination dump information.

Programmer Response: Supply a valid DD statement and rerun the assembly. The Assembler H Programmer's Guide, Order Number SC26-3759, describes the assembler data sets and the standard DD statements (in the IBM-supplied cataloged procedures) for them. Be sure to check whether your installation has changed the DDname (for example, SYSUT1 to SYSWORK1) or one or more parameters in the cataloged procedure statement.

Severity Code: 20

IEV998 THE ASSEMBLER COULD NOT RESUME READING A SYSLIB MEMBER BECAUSE IT COULD NOT FIND THE MEMBER AGAIN

Explanation: The assembly is terminated because the assembler cannot find a COPY member that it has already read. This usually is caused by a bug in the assembler itself or by an Operating System I/O error. Under certain conditions, however, the assembly can be rerun successfully.

Assembler Action: A special abnormal termination dump (Assembler H Interrupt and Diagnostic Dump) follows the message. The dump usually indicates which statement caused termination. It also may include contents of the assembler registers and work areas and other status information for use by IBM or your assembler maintenance programmers in determining the cause of the termination.

Programmer Response: Reassemble the program; it may assemble correctly. If it does not reassemble without error, save the abnormal termination dump, the assembly listing (if one was produced), and the input deck and give them to your IBM customer engineer. The Assembler H Program Logic Manual, Order Number LY26-3760, a complete description of the purpose and format of the assembler's abnormal termination dump.

Severity Code: 20

IEV999(I) ASSEMBLY TERMINATED - SYNAD EXIT TAKEN - PERMANENT I/O ERROR ON xxxxx DATA SET

Explanation: The assembly was terminated because of a permanent I/O error on the data set indicated in the message. This is usually caused by a machine or Operating System error. The assembly usually can be rerun successfully. This message will also appear on the console output device.

Assembler Action: A special abnormal termination dump (Assembler H Interrupt and Diagnostic Dump) follows the message. Depending on where the error occurred, the assembly listing up to the bad statement may also be produced. The dump usually indicates which statement caused termination. It also may include contents of the assembler registers and work areas and other status information for use by IBM or your assembler maintenance programmers in determining the cause of the termination.

Programmer Response: If the I/O error is on SYSIN or SYSLIB, you may have concatenated the input or library data sets incorrectly. Make sure that the DD statement for the data set with the largest block size (BLKSIZE) is placed in the JCL before the DD statements of the data sets concatenated to it. Also, make sure that all input or library data sets have the same device class (all DASD or all tape).

Reassemble the program; it may assemble correctly. If it does not reassemble without error, save the abnormal termination dump, the assembly listing (if one was produced), and the input deck and give them to your IBM customer engineer. Also, if the program assembles correctly, submit a copy of the listing and input deck of the correct assembly. The Assembler H Program Logic Manual, Order Number LY26-3760, contains a complete description of the purpose and format of the assembler's abnormal termination dump.

Severity Code: 20

This page intentionally left blank.

YOUR COMMENTS, PLEASE . . .

This publication is one of a series which serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Department 813
112 East Post Road
White Plains, New York
10601

fold fold

IBM System/360 Printed in U.S.A. SC26-3770-0

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM SYSTEM/360 OPERATING SYSTEM
ASSEMBLER H MESSAGES

SC26-3770-0

Please comment on the usefulness and readability of this book, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you want a reply, be sure to give your name and address.

Name _____ Occupation _____

Address _____

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE . . .

This publication is one of a series which serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Department 813
112 East Post Road
White Plains, New York
10601

fold fold

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM System/360 Printed in U.S.A. SC26-3770-0