SC33-0027-2
File No. S360/S370-29

Program Product

OS
PL/I Optimizing Compiler: Messages

Program Numbers 5734-PL1
5734-LM5
(These program products are available as part of composite package 5734-PL3)

IBM

Third Edition (June, 1973)

This is a major revision of, and obsoletes, SC33-0027-1. Changes or additions to the text are indicated by a vertical line to the left of the change.

This edition applies to Version 1, Release 2, Modification 0 of the PL/I Optimizing Compiler, Program Product 5734-PL1, and to all subsequent versions, releases, or modifications until otherwise indicated in new editions or technical newsletters.

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 and System/370 Bibliography, Order No. GA22-6822, and associated technical newsletters, for the editions that are applicable and current.

VSAM material in this publication is for planning purposes only.

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 United Kingdom Laboratories Ltd., Publications Department, Hursley Park, Winchester, Hampshire, England. Comments become the property of IBM.

Preface

This publication lists the compile-time messages from the OS PL/I Optimizing Compiler and the execution-time messages from the OS PL/I Transient Library. Most of these messages are accompanied by additional information intended to illustrate the detected condition and to point to the appropriate corrective action. Also included are the messages that may be produced by the PL/I prompter.

Messages produced by the OS PL/I Optimizing Compiler are listed in Part I. These result from conditions detected during compilation of the PL/I program. The messages are listed in both their short and long forms (except for compiler control messages which do not have a short form). The short form is listed first. However, only one form will be printed on the compiler listing depending on whether the SMESSAGE (short form) or LMESSAGE (long form) compiler option has been specified.

Messages produced by the OS PL/I Transient Library are listed in Part II. These result from conditions detected during execution of the PL/I program.

Messages produced by the PL/I prompter are listed in Part III. These messages result from conditions detected during use of the compiler in conversational mode in a TSO environment.

Associated Publications

OS PL/I Optimizing Compiler:

Design Objectives, Order No. GC33-0013

Specifications, Order No. GC33-0022

General Information, Order No. GC33-0001

Programmer's Guide, Order No. SC33-0006

Execution Logic, Order No. SC33-0025

| *Installation*, Order No. SC33-0026

TSO User's Guide, Order No. SC33-0029

OS PL/I Checkout and Optimizing Compilers:

Keywords and Terminal Commands Reference Card, Order No. GX33-6002

Language Reference Manual, Order No. GC33-0009

Availability of Publications

The availability of a publication is indicated by its *use key*, the first letter in the order number. The use keys are:

G - General: available to users of IBM systems, products, and services without charge, in quantities to meet their normal requirements; can also be purchased by anyone through IBM branch offices.

S - Sell: can be purchased by anyone through IBM branch offices.

Contents

Part I: Compile-time(IEL) Messages

Compile-time messages include compiler control messages, preprocessor messages, and compiler messages.

- Compiler control messages (numbers 0002 through 0049) are written on the first page of the compiler listing; they are mainly concerned with errors detected in the specification of compiler options in the PROCESS statement.

- Preprocessor messages (number 0001 and numbers 0050 through 0229) are written after any listed output from the preprocessor, and, if compilation follows immediately, before any listed output from the compilation.

- Compiler messages (numbers 0230 through 0999) are written as a group following the source program and any other listings produced by the compiler.

Format of Compile-time Messages

Each message has a number of the form IELnnnnI, where "IEL" indicates that the message is an optimizing compiler message and "nnnn" is the number of the message. The final character "I" indicates that the message does not require responsive action.

There are five types of messages: informatory, warning, error, severe error, and unrecoverable error.

An informatory message calls attention to some aspect of the source program that might assist the programmer.

A warning message calls attention to a possible program error or to a potential failure to achieve full optimization. It does not imply a syntactical error in the source program. In addition to alerting the programmer, warning messages may assist him in making the program more efficient.

An error message describes an error that the compiler has corrected and for which the correction is likely to be successful.

A severe error message describes an error that the compiler has attempted to correct, but for which the correction may not be successful. Frequently, the correction consists of ignoring the incorrect section of the statement.

An unrecoverable error message describes an error that cannot be corrected by the compiler. Such errors, when discovered, normally force termination of the compilation. They are usually caused by a compiler, system, or setup error rather than by an error in the source program.

In the list of messages, the symbols I, W, E, S, and U indicate the severity level of each message. Except for compiler control messages, the compiler prints the messages in groups according to these severity levels.

The compiler FLAG option can be used to suppress the listing of messages in the compiler listing. The FLAG option is described in the programmer's guide for this compiler.

Symbols in Messages

Many of the messages reproduced in this publication contain symbols indicating where the compiler will insert information when it prints the message. The symbols used are:

D - An identifier used in the program.

N - A decimal integer.

P - Compiler phase.

T - Text: up to 20 characters derived from the source program.

T_1- Text: up to 20 characters derived from the source program, being the first text insert in the message.

T_2- Text: up to 20 characters derived from the source program, being the second text insert in the message.

Alternative Forms of Messages

Some of the messages may be produced by the compiler in more than one form. Alternative forms of a message include or omit optional phrases. Those messages which can include an optional phrase are listed in this publication with the phrases enclosed in square brackets. For example, message IEL0399I may be printed as:

SEMICOLON ASSUMED. or as: SEMICOLON ASSUMED AFTER T.

This message is listed in this publication as:

IEL0399I E SEMICOLON ASSUMED [AFTER T].

Other phrases which may be included in a message in this way are:

PROLOGUE CODE

STATEMENT IGNORED

RESULTS OF PROLOGUE UNDEFINED

T TO D

The term PROLOGUE refers to the instructions generated by the compiler for a PROCEDURE or BEGIN statement. These instructions perform the housekeeping that is required on entry to a procedure or begin block. Messages with references to the prologue indicate that the compiler has detected the condition which resulted in the message while generating the prologue code.

Conditions detected while generating the prologue code may include items such as the misuse of the INITIAL attribute or of parameters. Consequently, the presence of a reference to the prologue indicates that the error is not contained in the PROCEDURE or BEGIN statement itself, but in some other statement, such as a DECLARE statement, that follows the indicated statement.

Before Calling IBM . . .

Before calling IBM for programming support with regard to a compile-time error, attempt at least one recompilation; if the problem recurs, ensure that the following are available:

1. A listing of the source program.
2. The job stream (source program and job control statements) in machine readable form.

The requirements for problem determination and APAR submission are given in the programmer's guide for this compiler.

Compiler Control Messages and Preprocessor Messages

IEL0001I U PREPROCESSOR ERROR NUMBER N DURING PHASE P.

PREPROCESSOR ERROR NUMBER N DURING PHASE P.

Explanation: An error has occurred during preprocessing. Processing has been terminated. This error is due to a fault in the preprocessor, not the source program.

Programmer Response: Rerun the job, and if the problem recurs, call IBM for programming support. Before calling IBM, refer to the introduction to this part of the publication for details of information that IBM will need in order to diagnose the problem.

IEL0002I U END-OF-FILE ENCOUNTERED ON INPUT FILE DURING COMPILER INITIALIZATION.

Example:

```
     // EXEC IEL0AA
     *PROCESS;
     /*
```

Explanation: The compiler has encountered the end of file for the source program before reading a complete PL/I statement.

Programmer Response: Ensure that the source program immediately follows the EXEC IEL0AA statement. If a PL/I comment is the first statement in the source program, ensure that the "/*" is not in the first two positions of the record (columns 1 and 2) and thereby are assumed to be the job control end-of-file delimiter. If the first statement in the source program is a PROCESS statement, ensure that the terminating semicolon is not in positions 73-80 of the first record.

IEL0003I THE FOLLOWING STRING IS NOT RECOGNIZED AS A VALID OPTION KEYWORD AND IS IGNORED - T.

Example:

```
     * PROCESS ATRIBUTES;
               └--------┘
                    T
```

Explanation: A character string in the PROCESS statement cannot be recognized as a valid keyword. In the above example, ATTRIBUTES is misspelled.

IEL0004I RIGHT PARENTHESIS MISSING IN SPECIFICATION OF FOLLOWING OPTION, BUT OPTION IS ACCEPTED - T.

Example

```
     * PROCESS LINECOUNT (55, SIZE(MAX);
               └-----------┘
                    T
```

IEL0005I THE SPECIFICATION OF THE FOLLOWING OPTION CONTAINS INVALID SYNTAX, DEFAULT ASSUMED FOR T.

Example:

```
     * PROCESS SIZE)80K)...;
               L---J
                 T
```

IEL0006I THE FOLLOWING OPTION IS DELETED, DEFAULT ASSUMED FOR T.

Explanation: The compiler, while processing the PROCESS statement, has encountered an option keyword that was deleted from the compiler at system generation. The default assumed for the option is the default specified for the option at system generation.

Programmer Response: If the option is essential, arrange to have the option restored to the compiler when the system is next generated or use the CONTROL option to restore the option temporarily.

IEL0007I THE FOLLOWING SUB-FIELD OF THE 'CHARSET' OPTION IS INVALID AND IS IGNORED - T.

Example:

```
     *PROCESS...CS(EBC,60)...;
```

IEL0008I THE FOLLOWING SUB-FIELD OF THE 'CHARSET' OPTION IS DELETED, DEFAULT ASSUMED FOR T.

Explanation: The option required has been deleted from use at system-generation.

Programmer Response: To obtain temporary use of this option, specify the CONTROL option with an appropriate password. If the option is required permanently, have the system generated again without deleting the option.

IEL0009I THE FOLLOWING SUB-FIELD OF THE 'TERMINAL' OPTION IS INVALID AND IS IGNORED - T.

Example:

```
     * PROCESS...TERMINAL(CODE,ESD,STORAGE);
```

In this example CODE is the invalid sub-field.

Explanation: The stated subfield is not recognized, because of misspelling, or because of the use of an invalid subfield. In each case, the stated subfield is ignored.

IEL0010I THE FOLLOWING SUB-FIELD OF THE 'TERMINAL' OPTION IS DELETED, DEFAULT ASSUMED FOR T.

Explanation: The option stated in the subfield has been deleted at system generation time. The default value of the subfield is assumed and may or may not be the option requested.

IEL0011I SOURCE MARGINS INCORRECTLY SPECIFIED. DEFAULTS ASSUMED FOR THE 'MARGINS' OPTION.

Example:

```
* PROCESS MARGINS(72,2,1);
```

Explanation: The left hand margin position is to the right of the right hand margin position. The default values assumed will be 2 and 72. The carriage control character position, if specified, is ignored.

IEL0012I CARRIAGE CONTROL CHARACTER OVERLAPS SEQUENCE FIELD OR SOURCE MARGINS. CONTROL CHARACTER IGNORED.

Example:

```
1.    * PROCESS MAR(5,72,73) SEQ(73,80);

2.    * PROCESS MAR(5,72,10);
```

Explanation: The carriage control character position, if used, must be outside the margins or sequence limits. The values of 5 and 72 will be used for the margins, and the carriage control character position ignored.

IEL0013I ARGUMENT NOT WITHIN PERMITTED RANGE. DEFAULT ASSUMED FOR FOLLOWING OPTION - T.

Example:

```
1.  * PROCESS MARGINS(2,103,1)...;

2.  * PROCESS LINECOUNT(0)...;
```

IEL0014I UNMATCHED LEFT PARENTHESIS IN COMPILER OPTIONS SPECIFICATION. SUBSEQUENT OPTIONS IGNORED.

Example:

```
*PROCESS...CHARSET(BCD,LIST,ESD,...;
```

IEL0015I SPECIFIED 'SIZE' OPTION IS LESS THAN MINIMUM REQUIRED BY COMPILER. DEFAULT ASSUMED.

Example:

```
* PROCESS SIZE(40K)...;
```

Explanation: The compiler requires at least 44K bytes of main storage.

IEL0016I SIZE SPECIFICATION TOO BIG. SIZE(MAX) ASSUMED

Example:

```
*PROCESS...SZ(20000K)
```

IEL0018I NAME FIELD TOO LONG. 'NAME' OPTION IGNORED

Explanation: The total number of non-blank characters appearing in the name field of the specified NAME option is too large. Correct the specification and resubmit the job.

IEL0019I 'SIZE' OPTION IGNORED. VALUE IN FIRST MEMBER OF BATCH ASSUMED.

Explanation: It is not possible to alter the amount of main storage to be used by the compiler for the compilation of the second or subsequent external procedures in a batched compilation.

IEL0020I 'DUMP' OPTION NOT SPECIFIED FOR FIRST MEMBER OF BATCH. OPTION IGNORED.

Explanation: It is not possible to specify the use of the compiler DUMP option for the compilation of the second or subsequent external procedures in a batched compilation if the option is not specified for the compilation of the first external procedure in the batch.

IEL0021I DELIMITER ',.' ACCEPTED AS SEMICOLON. OPTION SCANNING TERMINATED.

Example:

```
     * PROCESS NEST,XREF,ATTRIBUTES,.
```

Explanation: When the compiler reads the options in the PROCESS statement, it cannot tell whether the CHARSET(48) or CHARSET(60) option is to be used until it has processed the options. Therefore it assumes that either the 48- or the 60-character sets will be used and will always interpret ',.' as a semicolon.

IEL0023I W NON-BLANK CHARACTERS FOLLOWING SEMICOLON IGNORED.

Explanation: Non-blank characters have been detected following the semicolon in the options list. Any comments and the first statement in the external procedure must follow a PROCESS statement on the following card (or line).

Example:

```
     * PROCESS A,X; P:PROC OPTIONS(MAIN);
```

IEL0024I U SPILL FILE NEEDED BUT DD STATEMENT INCORRECT. COMPILATION TERMINATED.

Explanation: If the spill file cannot be opened, message IEL0026I or message IEL0031I will be produced. If the spill file is needed, message IEL0024I is produced. The compilation may be completed without needing a spill file.

IEL0025I INVALID SYNTAX IN LAST OPTION OF 'PARM' FIELD. PROCESSING OF 'PARM' OPTIONS TERMINATED.

Example:

```
    PARM = TA(KQ
```

Explanation: In the above example, the right parenthesis has been omitted.

IEL0026I THE COMPILER SPILL FILE IS NOT DIRECT ACCESS. COMPILATION WILL TERMINATE IF SPILL FILE NEEDED.

Example:

```
//SYSUT1 DD SYSOUT=A
```

Explanation: Compilation will be terminated if the spill file is needed and it is not on a direct access storage device. Compilation will not be terminated if the spill file is not needed.

IEL0027I U INCORRECT SPECIFICATION OF THE 'CONTROL' OPTION. COMPILATION TERMINATED.

Explanation: Either the CONTROL option has been specified syntactically incorrect or the wrong password has been supplied.

IEL0028I DELIMITER AT START OF STRING 'T' IS INVALID AND IS IGNORED.

Example:

```
* PROCESS 'FLAG(S)...';
```

the quote (') characters are invalid.

```
* PROCESS (FLAG(S)...);
```

the first left parenthesis and the last right parenthesis are invalid.

IEL0030I U THE COMPILER INPUT FILE CANNOT BE OPENED.

Explanation: The compiler input file SYSCIN or SYSIN cannot be opened, possibly because no DD card for the file has been provided. Compilation is terminated.

IEL0031I THE COMPILER SPILL FILE CANNCT BE OPENED. COMPILATION WILL TERMINATE IF SPILL FILE NEEDED.

Explanation: The compiler spill file SYSUT1 cannot be opened, possibly because no DD card has been provided. Compilation will not be terminated if the spill file is not needed.

IEL0032I S THE COMPILER PUNCH FILE CANNOT BE OPENED.

Explanation: The DECK or MDECK option has been requested but SYSPUNCH cannot be opened, possibly because no DD card has been provided. Compilation continues with no punched output.

IEL0033I S THE COMPILER LOAD FILE CANNOT BE OPENED.

Explanation: The OBJECT option list has been specified but SYSLINK cannot be opened, possibly because no DD card has been provided. Compilation continues but subsequent link editing is impossible.

IEL0034I U INSUFFICIENT MAIN STORAGE AVAILABLE. COMPILATION TERMINATED.

Explanation: The compiler has insufficient main storage to complete initialization. Either the region or the partition is below the minimum required, or the buffers allocated to the compiler input/print/load/punch files may be too big. In either case, retry with larger region/partition.

IEL0035I 'NUMBER' OPTION BUT NO 'SEQUENCE'. DEFAULT SEQUENCE ASSUMED.

Example:

```
*PROCESS NUM NSEQ;
```

Explanation: The NUMBER option derives a line number from the sequence number in the position specified in the SEQUENCE option. If this position is not specified, the following position is assumed:

F-format records:	last eight columns
U-format records:	first eight columns
V-format records:	first eight columns

IEL0036I THE FOLLOWING OPTION IS NOT SUPPORTED AND IS IGNORED - T.

Explanation: A valid PL/I option keyword has been specified, but is not supported by this compiler.

IEL0037I S INVALID BLOCKSIZE FOR PUNCH FILE. 80 ASSUMED.

Explanation: The block size specified for the punch file (SYSPUNCH) is not a multiple of 80.

IEL0038I S INVALID BLOCKSIZE FOR LOAD FILE. 80 ASSUMED.

Explanation: The block size specified for the load file (SYSLIN) is not a multiple of 80.

IEL0039I S INVALID FORMAT SPECIFICATION FOR INPUT FILE. U(100) ASSUMED.

Explanation: The record format specified for the input file (SYSCIN or SYSIN) is not supported by the compiler.

IEL0040I MACRO OPTION IMPLIED BY CHAR (48 OR BCD) AND INPUT RECORDS GREATER THAN 80.

Explanation: The macro preprocessor is required if the input records have LRECL >80 and if the source is in BCD or CHAR48.

IEL0041I SEQUENCE FIELD OVERLAPS SOURCE MARGINS. DEFAULT SEQUENCE ASSUMED.

Example:

```
*PROCESS MAR(10,72) SEQ(10,18);
```

Explanation: The source margins need not overlap the position of the sequence number. If they do, the following position for the sequence number is assumed:

```
F-format records:  last eight columns
U-format records:  first eight columns
V-format records:  first eight columns
```

IEL0042I SOURCE MARGINS OVERLAP SEQUENCE FIELD. DEFAULT MARGINS ASSUMED.

Example:

```
*PROCESS MAR(2,80) SEQ(1,8);
```

Explanation: The assumed position of the sequence number, as described in the explanation for message IEL0041I, has failed to prevent overlapping of the sequence number by the source margins. The MARGINS and SEQUENCE options are given the following values:

```
F-format records:  MAR(2,72) SEQ(73,80)
U-format records:  MAR(10,100) SEQ(1,8)
V-format records:  MAR(10,100) SEQ(1,8)
```

IEL0043I 'COUNT' OPTION USED WITH 'NOGOSTMT' OR 'NOGONUMBER' OPTION. 'COUNT' OPTION IGNORED.

Example:

```
*PROCESS CT NUM NGN;
```

Explanation: Statement frequency counting is performed by recording the numbers of statements involved in all branches. With the exception of points of interrupt, all statements that may be involved in branches can be recognized at compile-time.

When a statement number table is not available at execution-time (because NOGOSTMT or NOGONUMBER are in effect) it is impossible to determine the statement number at a point of interrupt. If return is not made to the point of interrupt, the count values will be incorrect.

If NOGOSTMT or NOGONUMBER are not specified explicitly, GOSTMT or GONUMBER (depending on whether STMT or NUMBER have been specified) will be implied by COUNT.

IEL0044I I 'DUMP' OPTION SPECIFIED. COMPILATION SPEED WILL BE DEGRADED SINCE AVAILABLE STORAGE IS REDUCED.

Example:

```
* PROCESS DUMP;
```

Explanation: The DUMP option requires 18K of main storage. Compilation speed is degraded because this 18K cannot be used as working storage, thereby causing more spilling than usual. It is recommended, therefore, that the programmer uses the DUMP option only if necessary.

IEL0045I U I/O ERROR 'T'.

Explanation: The insert 'T' is the information pertinent to the I/O error that is provided by the SYNADAF macro instruction (see the appropriate manual for this compiler for details). Compilation is terminated. If the I/O error is on the print file, "PRINT FILE ERROR" will appear on the operator console.

IEL0047I U COMPILER INITIALIZATION ERROR. COMPILATION TERMINATED.

Explanation: An error has occurred in the compiler initialization phase.

IEL0050I E IDENTIFIER BEGINNING T EXCEEDS N CHARACTERS.

PREPROCESSOR RESTRICTION. IDENTIFIER BEGINNING T IS TOO LONG. TRUNCATED TO FIRST N CHARACTERS.

Example:

```
    %INCLUDE  DECLARATIONS;
    %INCLUDE  X(DECLARATIONS);
```

Explanation: The maximum possible length for an identifier in a %INCLUDE statement is 8 characters. Therefore in the above example, the identifier DECLARATIONS is truncated to DECLARAT.

IEL0051I S NESTING LEVEL FOR '%INCLUDE' STATEMENT EXCEEDS N.

MORE THAN N LEVELS OF NESTING FOR '%INCLUDE' STATEMENT. STATEMENT IGNORED. RERUN WITH 'MACRO' OPTION.

Example:

```
        %INCLUDE A;
In A:   %INCLUDE B;
In B:   %INCLUDE C; (and so on, to a depth greater than 8).
```

Explanation: %INCLUDE statements may not be nested with more than eight levels when using the INCLUDE compiler option.

Programmer Response: The preprocessor, which has no limits on the depth of nesting, should be used by specifying the MACRO compiler option instead of the INCLUDE compiler option.

IEL0052I S '%INCLUDE' MEMBER T NOT FOUND.

'%INCLUDE' MEMBER T NOT FOUND. MEMBER IGNORED.

Example:

```
    %INCLUDE X,Y;
```

Explanation: If X cannot be found in the SYSLIB data set, the member is ignored and processing continues with Y.

IEL0053I S I/O ERROR READING MEMBER T.

I/O ERROR READING MEMBER T. PROCESSING OF MEMBER TERMINATED.

Example:

```
    %INCLUDE X,Y;
```

Explanation: If an I/O error is encountered while including X, processing of X is terminated and an attempt is made to include Y.

IEL0054I S INVALID TEXT BEGINNING T IGNORED.

INVALID TEXT BEGINNING T IN '%INCLUDE' STATEMENT. STATEMENT IGNORED.

Example:

```
    %INCLUDE A*B;
```

Explanation: The syntax of the %INCLUDE statement is incorrect. In the example shown, an identifier is expected.

IEL0055I W '%INCLUDE' FILE D. T ASSUMED.

SEQUENCE POSITIONS NOT SPECIFIED FOR '%INCLUDE' FILE D. T ASSUMED.

Explanation: This message is printed when the record format of the included data set differs from that of SYSIN, no sequence values for this record format were specified at system generation, and the NUMBER option applies.

Programmer Response: If the fix-up is unsatisfactory, change the compiler default options FSEQUEN or VSEQUEN.

IEL0056I W INVALID CARRIAGE CONTROL POSITION IGNORED FOR '%INCLUDE' MEMBER D.

CARRIAGE CONTROL POSITION FOR '%INCLUDE' MEMBER D IS WITHIN SOURCE MARGINS OR SEQUENCE FIELD. IT IS IGNORED.

Example:

```
    SEQUENCE(14,20) MAR(26,50,15);
```

Explanation: The carriage control position specified in the MARGINS option must lie outside the margins and outside any sequence field. In the example shown, the MARGINS statement field will become (26,50,0).

IEL0057I S SEQUENCE AND MARGINS OVERLAP FOR D. T ASSUMED.

SEQUENCE AND MARGINS FIELDS OVERLAP FOR '%INCLUDE' FILE D. T ASSUMED.

Explanation: The MARGINS option is modified if it overlaps the sequence field.

Programmer Response: If the fix-up is unsatisfactory, one or more of the following compiler options will need to be modified: FMARGINS, FSEQUEN, UMARGINS, USEQUEN, MARGINS, SEQUENCE.

IEL0058I S NO 'DD' STATEMENT FOR MEMBER T.

MISSING 'DD' STATEMENT FOR '%INCLUDE' MEMBER T. MEMBER IGNORED.

Example:

```
    %INCLUDE P(MEMBER),DECLS;
```

Explanation: A DD statement for library P in the example shown must be present. If it is not present, this message is issued and the preprocessor proceeds to the next specification, which is DECLS.

Programmer Response: Provide a DD statement with a ddname P with the JCL statements used to perform the compilation.

IEL0059I S I/O ERROR SEARCHING FOR MEMBER T.

I/O ERROR SEARCHING FOR MEMBER T. MEMBER IGNORED.

Example:

```
    %INCLUDE X;
```

Explanation: In the example shown, an I/O error has occurred during an attempt to find member X.

Programmer Response: If the error persists, call IBM for programming support.

IEL0060I S RECORD LENGTH GREATER THAN N FOR MEMBER T.

LOGICAL RECORD LENGTH GREATER THAN N FOR MEMBER T. PROCESSING OF MEMBER TERMIANTED.

Explanation: The maximum permitted logical record length is 100 for F-format data sets and 104 for V-format data sets. For V-format data sets, no message is issued until a record longer than 104 bytes is actually encountered.

Programmer Response: Recreate the data set using permitted logical record length.

IEL0061I E DEFAULT RECORD LENGTH OR BLOCK SIZE ASSUMED FOR MEMBER T.

LOGICAL RECORD LENGTH OR BLOCK SIZE NOT SPECIFIED FOR '%INCLUDE' MEMBER T. DEFAULT ASSUMED.

Explanation: If either or both the logical length and the block size are not specified, the following assumptions are made. For F-format data sets, if neither block size nor record length are specified, a block size of 80 and record length of 400 are assumed; if only one of the two is specified, the value specified is assumed for both. For V-format and U-format data sets, the maximum practicable block size (4260 bytes) is assumed.

Programmer Response: Specify LRECL and BLKSIZE in the DCB parameter in the DD statement for the data set.

IEL0065I E 'RETURNS' ATTRIBUTE ON D IGNORED.

'RETURNS' ATTRIBUTE ON BUILTIN FUNCTION D IGNORED.

Example:

```
    %DECLARE SUBSTR BUILTIN RETURNS(CHAR);
```

Explanation: Data type returned by a built-in function is determined by the language rules.

IEL0066I E 'ENTRY' ATTRIBUTE ON D IGNORED.

'ENTRY' ATTRIBUTE ON BUILTIN FUNCTION D IGNORED.

Example:

```
    %DECLARE INDEX BUILTIN ENTRY;
```

Explanation: The BUILTIN attribute implies the ENTRY attribute.

IEL0067I S D INVALID BUILTIN FUNCTION NAME.

D IS NOT A VALID BUILTIN FUNCTION NAME. REFERENCE WILL END PROCESSING.

Example:

```
    %DECLARE HARRIET BUILTIN;
```

Explanation: Only SUBSTR, INDEX, and LENGTH are permitted built-in function names for the preprocessor.

IEL0068I E DESCRIPTOR LIST AFTER 'ENTRY' IGNORED.

PARAMETER DESCRIPTOR LIST ON 'ENTRY' ATTRIBUTE IGNORED.

Example:

```
    %DECLARE P ENTRY(CHAR,FIXED);

    should be

    %DECLARE P ENTRY;
```

Explanation: The arguments are always converted to the types specified by the PROCEDURE statement.

IEL0069I E 'RETURNS' ATTRIBUTE IGNORED.

'RETURNS' ATTRIBUTE IN 'DECLARE' STATEMENT IGNORED.

Example:

```
    %DECLARE P ENTRY RETURNS(FIXED);

    should be

    %DECLARE P ENTRY;
```

Explanation: The attribute of the value returned by a compile-time procedure is determined by the procedure statement.

IEL0070I S END OF SOURCE TEXT IN STRING.

END OF SOURCE TEXT IN STRING. QUOTE ASSUMED BEFORE END OF SOURCE TEXT.

Explanation: End of source text found while scanning for closing quote character. Check that all quotes are paired.

IEL0071I S NO DELIMITER ON REPLACEMENT VALUE STRING.

REPLACEMENT VALUE CONTAINS NO END OF STRING DELIMITER. DELIMITER ASSUMED AT END OF STRING.

Explanation: An end-of-string delimiter has not been found in a replacement value.

IEL0072I E INVALID CHARACTER IN BIT STRING.

INVALID CHARACTER IN BIT STRING. PROCESSED AS CHARACTER STRING.

IEL0073I S END OF SOURCE TEXT IN COMMENT.

END OF SOURCE TEXT IN COMMENT. COMMENT DELIMITER ASSUMED AT END OF SOURCE TEXT.

Explanation: The end of the source text has been encountered while scanning for an end-of-comment delimiter.

IEL0074I E NO COMMENT DELIMITER IN REPLACEMENT VALUE.

REPLACEMENT VALUE CONTAINS NO END OF COMMENT DELIMITER. COMMENT DELIMITER ASSUMED AT END OF REPLACEMENT VALUE.

Explanation: An end-of-comment delimiter cannot be found in a replacement value.

IEL0075I E INVALID CHARACTER REPLACED BY BLANK.

INVALID CHARACTER REPLACED BY BLANK.

Explanation: An invalid character has been found in the source text.

IEL0076I U BLOCKSIZE FOR '%INCLUDE' D EXCEEDS 400.

BLOCKSIZE OF '%INCLUDE' D EXCEEDS THE DEFAULT MAXIMUM OF 400 ALLOWED WITH THIS SIZE OPTION. PROCESSING TERMINATED.

Explanation: The INCLUDE data set block size can never exceed the spill fill record size, and with small compiler SIZE option values the maximum is 400. The point at which a block size greater than 400 may be used depends on the storage allocation performed at compiler initialization time, but usually a SIZE value of 60K is sufficient.

Programmer Response: Use a large SIZE option value, or recreate the INCLUDE data set with a smaller block size.

IEL0077I E   CONFLICTING USE OF D.

USE OF D IN PROCEDURE ENDING AT THIS LINE CONFLICTS WITH PREVIOUS USE. REFERENCE WILL END PROCESSING.

Example:

```
     %DCL E ENTRY;
     %P: PROCEDURE RETURNS(CHAR);
         E = 3;
     %END;
```

Explanation: An identifier has been used but not declared in a compile-time procedure. The use conflicts with a use or declaration outside the procedure.

IEL0078I E   '%' IN LABEL LIST IGNORED.

'%' IN LABEL LIST IGNORED.

Explanation:

In the statement

```
     % LABEL4: % IF C1 = C2 etc.
```

the second '%' is ignored.

IEL0079I E   NO LABEL BEFORE COLON.

NO LABEL BEFORE COLON. COLON IGNORED.

Example:

```
     %: A = B;
```

Programmer Response: Insert label or remove colon.

IEL0080I S   INVALID TEXT IGNORED FROM T TO SEMICOLON.

INVALID TEXT IGNORED FROM T TO SEMICOLON.

Example:

```
     % GOTOLABEL 2;  should be   % GOTO LABEL2;
```

IEL0081I E   CONFLICTING USE OF D.

CONFLICTING USE OF IDENTIFIER D AS LABEL. REFERENCE WILL END PROCESSING.

Example:

```
     %DCL (A,B,C) CHAR;
     %A: B = C;
```

Explanation: No system action is taken unless a statement which references the identifier is detected.

IEL0082I E   MULTIPLE USE OF D AS LABEL.

D USED AS LABEL MORE THAN ONCE. REFERENCE WILL END PROCESSING.

Example:

```
%L:A = 1;
%L:A = 2;
```

Explanation: No system action is taken unless a statement which references the multiply-defined label is detected.

IEL0083I W   LABELS ON DECLARE STATEMENT.

LABELS ON 'DECLARE' STATEMENT IGNORED.

Example:

```
% LABEL1: DECLARE etc.
```

IEL0084I S   CONFLICTING USE OF D.

USE OF D CONFLICTS WITH PREVIOUS USE AS LABEL.

Example:

```
%L:;
%L = 2;
```

IEL0085I E   NO ATTRIBUTE DECLARED FOR D.

NO ATTRIBUTE DECLARED FOR PARAMETER D IN PROCEDURE ENDING AT THIS LINE. CHARACTER ASSUMED.

Example:

```
%PROC1: PROC (P1,P2,P3) RETURNS(CHAR);
DCL (P1, P2) FIXED;
%END PROC1;
```

IEL0086I E   LABEL D IS UNDEFINED.

LABEL D IS UNDEFINED. REFERENCE WILL END PROCESSING.

Explanation: No system action is taken unless a %GOTO statement that references the undefined label is executed. Check all references to the label, or define it.

IEL0087I E   END OF SOURCE TEXT IN PROGRAM.

END OF SOURCE TEXT BEFORE LOGICAL END OF PROGRAM. '%END' STATEMENT ASSUMED.

Explanation: Check that each %PROCEDURE and %DO statement is matched with a %END statement.

IEL0088I E   D IS UNDEFINED IN PROCEDURE.

LABEL D IS UNDEFINED IN PROCEDURE ENDING AT THIS LINE. REFERENCE WILL END PROCESSING.

Explanation: A label must be defined within the procedure as no transfers out of procedures are allowed.

IEL0089I E   SEMICOLON AFTER 'IF' EXPRESSION.

SEMICOLON TERMINATES 'IF' EXPRESSION. SEMICOLON IGNORED.

Example:

```
%IF P1 = P2;
%THEN C1 = C2;
```

IEL0090I S 'IF' STATEMENT IGNORED.

'IF' EXPRESSION NOT FOLLOWED BY '%' OR 'THEN'. 'IF' STATEMENT IGNORED.

Example:

```
%IF C1 = C2 GOTO etc.
```

IEL0091I E NO '%' BEFORE 'THEN'.

MISSING '%' ASSUMED BEFORE 'THEN' IN '%IF' STATEMENT.

Example:

```
% IF C1 = C2 THEN etc.
```

IEL0092I E NO 'THEN' AFTER '%'

MISSING 'THEN' ASSUMED AFTER '%' IN '%IF' STATEMENT.

Example:

```
%IF C1 = C2
%C2 = C3; etc.
```

IEL0093I E INVALID STATEMENT AFTER '%THEN' OR '%ELSE'.

STATEMENT AFTER '%THEN' OR '%ELSE' NOT A PREPROCESSOR STATEMENT. '%' ASSUMED BEFORE IT.

Example:

```
% IF C1 = C2 % THEN C1 = C3;
```

is incorrect.

Explanation: If the statement in question is not a preprocessor statement, it should be inside a preprocessor do-group.

IEL0094I E MISSING 'THEN' ASSUMED.

MISSING 'THEN' ASSUMED IN 'IF' STATEMENT.

Example:

```
%P: PROC RETURNS (FIXED);

    IF I = 1 GOTO L;
    .
    .
    .
```

IEL0095I E INVALID '%' IGNORED.

INVALID '%' IN PREPROCESSOR PROCEDURE IGNORED.

Example:

```
% PROC1: PROCEDURE RETURNS(CHARACTER);
% DCL etc.
```

Explanation: Statements within preprocessor procedures may not be preceded by %.

IEL0096I W LABELS ON 'ELSE' IGNORED.

LABELS ON 'ELSE' IGNORED.

Example:

```
% LABEL3: ELSE % etc.
```

IEL0097I E NULL STATEMENT ASSUMED.

NO STATEMENT AFTER 'THEN' OR 'ELSE'. NULL STATEMENT ASSUMED.

Example:

```
% IF ....% THEN % ELSE%;
```

IEL0098I E NO 'IF' BEFORE 'ELSE'

NO 'IF' BEFORE 'ELSE'. 'ELSE' IGNORED.

Explanation: An ELSE clause has been found which is not part of an IF statement.

IEL0099I U BLOCKSIZE FOR '%INCLUDE' D EXCEEDS TEXT PAGE SIZE.

BLOCKSIZE OF '%INCLUDE' D EXCEEDS THE TEXT PAGE SIZE ALLOWED WITH THIS SIZE OPTION. PROCESSING TERMINATED.

Explanation: The text page size depends on the size option specified for the compilation. The block size of an INCLUDE data set may not exceed this.

Programmer Response: Specify a large enough SIZE value to ensure that text pages are at least as big as INCLUDE data set blocks _OR_ recreate INCLUDE data set with smaller blocking factor, using a utility (e.g. IEBGENER).

IEL0100I E DUMMY LABEL ASSUMED ON STATEMENT.

NO LABEL ON '%PROCEDURE' STATEMENT. DUMMY LABEL ASSUMED.

Example:

```
%PROC RETURNS(CHAR);
```

Explanation: A %PROCEDURE statement should have a label.

Programmer Response: Insert a label on the PROCEDURE statement.

IEL0101I U MORE THAN N PROCEDURES.

PREPROCESSOR RESTRICTION. MORE THAN N PREPROCESSOR PROCEDURES DEFINED IN A COMPILATION. PROCESSING TERMINATED.

Programmer Response: Reduce the number of preprocessor procedures to within the given limit.

IEL0102I E D PREVIOUSLY DEFINED.

ENTRY NAME D PREVIOUSLY DEFINED. REFERENCE WILL END PROCESSING.

Example:

```
%E: PROC RETURNS(CHAR)
.
.
.

%E: PROC RETURNS(CHAR);
```

Explanation: No action is taken unless the multiply-defined label is referenced by a statement that is executed.

Programmer Response: Change the label on one of the %PROCEDURE statements, or remove one of the procedures.

IEL0103I E INVALID USE OF D.

INVALID USE OF FUNCTION D ON LEFT OF EQUALS SYMBOL. REFERENCE WILL END PROCESSING.

Example:

```
%DCL E ENTRY RETURNS(CHAR);
%E = 'ABC';
```

Explanation: Entry names and built-in function names may not appear on the left hand side of an assignment statement. Execution of such a statement will terminate processing.

IEL0104I E CONFLICTING USE OF D.

CONFLICTING USE OF IDENTIFIER D AS ENTRY NAME. REFERENCE WILL END PROCESSING.

Example:

```
%DCL C CHAR;
%C = C(I);
```

Explanation: An identifier followed by a parenthesis in a preprocessor expression is considered to be an entry name. Execution of such a statement will terminate processing.

IEL0105I E MULTIPLE USE OF D IN PARAMETER LIST.

PARAMETER D APPEARS MORE THAN ONCE IN PARAMETER LIST. AN ARGUMENT CORRESPONDING TO SECOND USE OF PARAMETER WILL NOT BE USED WITHIN PROCEDURE.

Example:

```
%E: PROC(P,P) RETURNS(CHAR);
```

Explanation: The number of parameters to the procedure is not changed, but, within the procedure, references to the multiply-defined parameter will apply to its first use.

IEL0106I S   MORE THAN N PARAMETERS USED.

PREPROCESSOR RESTRICTION. MORE THAN N PARAMETERS USED. REFERENCE WILL END PROCESSING.

Explanation: Processing is ended if a procedure having more than fifteen parameters is referenced by a statement that is executed.

IEL0107I E   MISSING PARAMETER.

MISSING PARAMETER. A CORRESPONDING ARGUMENT WILL NOT BE USED WITHIN PROCEDURE.

Example:

```
%PROCL: PROCEDURE (P1,P2,,P4) RETURNS(CHAR);
```

Explanation: The assumption is made that the omission of the parameter is intentional.

IEL0108I E   PARAMETER T INVALID.

PARAMETER T INVALID. AN ARGUMENT CORRESPONDING TO THE PARAMETER WILL NOT BE USED WITHIN THE PROCEDURE.

Example:

```
%P: PROC(8) RETURNS(CHAR);
```

Explanation: The expected parameter is not an identifier. The parameter is assumed to exist but is not identified within the procedure.

IEL0109I S   T TO NEXT COMMA OR SEMICOLON IGNORED.

INVALID BLANK OR MISSING COMMA IN PARAMETER LIST. TEXT IGNORED FROM T TO NEXT COMMA OR SEMICOLON.

Example:

```
%PROC1: PROC (P1,P2,P3 P4) RETURNS(CHAR);
```

IEL0110I S   RIGHT PARENTHESIS ASSUMED FOR SEMICOLON.

SEMICOLON FOUND IN PARAMETER LIST. RIGHT PARENTHESIS ASSUMED.

Example:

```
%E: PROC (P ;
```

Explanation: A semicolon has been encountered during the scan of an apparent parameter list. A right parenthesis has been inserted before the semicolon.

IEL0111I E INVALID RETURNED VALUE. T TO SEMICOLON IGNORED.

RETURNED VALUE NOT 'FIXED' OR 'CHARACTER'. TEXT IGNORED FROM T TO SEMICOLON.

Example:

```
%E: PROC RETURNS(BIT);
```

Explanation: Returned values may only be FIXED or CHARACTER. CHARACTER is the assumed attribute.

IEL0112I E CHARACTER ASSUMED FOR RETURNED VALUE.

NO ATTRIBUTE FOR RETURNED VALUE. CHARACTER ASSUMED.

Example:

```
%P: PROC;
```

IEL0113I S SEMICOLON MISSING. T TO NEXT SEMICOLON IGNORED.

SEMICOLON MISSING IN '%PROCEDURE' STATEMENT. TEXT IGNORED FROM T TO NEXT SEMICOLON.

Example:

```
%E: PROC RETURNS(FIXED)
C1 = '6';
```

Explanation: The preprocessor did not find a semicolon indicating the end of a %PROCEDURE statement where expected.

IEL0114I E T IS IGNORED.

T IS IGNORED IN '%DEACTIVATE' STATEMENT.

Example:

```
%DEACTIVATE A NORESCAN;
```

Explanation: RESCAN and NORESCAN options are only valid in a %ACTIVATE statement.

IEL0115I E CHARACTER ASSUMED FOR UNDECLARED D.

REFERENCE TO UNDECLARED IDENTIFIER D. CHARACTER ATTRIBUTE ASSUMED.

Example:

```
%DCL (A,B) CHAR C FIXED; (%C=N;);
%D=A||B;
```

Explanation: D is given the attribute CHAR by default.

IEL0116I S '%PROCEDURE' STATEMENT INVALID.

'%PROCEDURE' STATEMENT IN PREPROCESSOR PROCEDURE. TEXT IGNORED TO NEXT PREPROCESSOR '%END' STATEMENT.

Example:

```
          %PROC: PROC;
          PROC6: PROC;
          END PROC6;
           %END;
```

<u>Explanation</u>: Procedures cannot be nested in preprocessor procedures. Other messages may be generated by this error.

IEL0117I S '%PROCEDURE' STATEMENT REPLACED BY NULL.

'%PROCEDURE' STATEMENT IN '%THEN' OR '%ELSE' CLAUSE REPLACED BY NULL STATEMENT. TEXT IGNORED TO NEXT PREPROCESSOR '%END' STATEMENT.

<u>Example</u>:

```
          %IF C1 = C2 %THEN %PROC2: PROCEDURE;
          END PROC2;
          %ELSE %PROC3: PROCEDURE; etc.
```

<u>Explanation</u>: %PROCEDURE statements are not allowed in preprocessor 'THEN' or 'ELSE' clauses. Other messages may be generated by this error.

IEL0118I S '%RETURN' STATEMENT INVALID. IGNORED.

'%RETURN' STATEMENT INVALID OUTSIDE PREPROCESSOR PROCEDURE. STATEMENT IGNORED.

<u>Example</u>:

```
          %RETURN(0);
```

IEL0119I E MISSING PARENTHESIS ASSUMED.

MISSING PARENTHESIS ASSUMED FOR 'RETURN' EXPRESSION.

<u>Example</u>:

```
          %P: PROC FIXED;
          RETURN  6);
          %END;
```

IEL0120I E INVALID TEXT. T TO SEMICOLON IGNORED.

INVALID TEXT AFTER EXPRESSION IN 'RETURN' STATEMENT. TEXT IGNORED FROM T TO SEMICOLON.

<u>Example</u>:

```
          RETURN ('1') IF A = B;
```

<u>Explanation</u>: The RETURN statement has been processed but scan finds text when it expects a semicolon.

IEL0121I S 'GOTO' STATEMENT IGNORED.

NO OPERAND IN 'GOTO'. STATEMENT IGNORED.

<u>Example</u>:

```
          %GOTO;
```

IEL0122I E   CONFLICTING USE OF D.

USE OF IDENTIFIER D IN '%GOTO' STATEMENT CONFLICTS WITH PREVIOUS USE. REFERENCE WILL END PROCESSING.

Example:

```
%P:PROC RETURNS(FIXED);
.
.
.
%GOTO P;
```

IEL0123I S   SEMICOLON MISSING. T TO NEXT SEMICOLON IGNORED.

SEMICOLON MISSING AFTER '%GOTO' STATEMENT. TEXT IGNORED FROM T TO NEXT SEMICOLON.

Example:

```
%GOTO LABEL4 C1 = C2;
```

IEL0124I U   '%GOTO' IS AN INVALID BRANCH.

OPERAND OF '%GOTO' IS LABEL IN ITERATIVE 'DO' OR INCLUDED TEXT. PROCESSING TERMINATED.

Example:

```
% GOTO L1;
.
.
.
% DO I 1 TO N;
%L1:
%END;
```

IEL0125I S   STATEMENT INVALID IN PROCEDURE.

INVALID '%ACTIVATE' OR '%DEACTIVATE' IN PREPROCESSOR PROCEDURE. STATEMENT IGNORED.

Explanation: ACTIVATE and DEACTIVATE statements may not be used in preprocessor procedures.

IEL0126I E   STATEMENT HAS NO OPERAND. IGNORED.

'%ACTIVATE' OR '%DEACTIVATE' HAS NO OPERAND. STATEMENT IGNORED.

Example:

```
%ACTIVATE;
```

IEL0127I E   REDUNDANT COMMA IGNORED.

MISSING OPERAND OR REDUNDANT COMMA IN '%ACTIVATE' OR '%DEACTIVATE". COMMA IGNORED.

Example:

```
%DEACTIVATE C5,, C6;
```

IEL0128I S   INVALID FIELD T IGNORED.

INVALID FIELD T IN '%ACTIVATE' OR '%DEACTIVATE' STATEMENT IS IGNORED.

Example:

```
%ACTIVATE 7TIMES;
```

IEL0129I S   IDENTIFIER D IGNORED.

IDENTIFIER D NOT PROCEDURE OR VARIABLE. IT HAS BEEN IGNORED IN '%ACTIVATE' OR '%DEACTIVATE' STATEMENT.

Example:

```
% DEACTIVATE LABEL4;

 (where LABEL4 is a statement label).
```

IEL0130I S   T TO COMMA OR SEMICOLON IGNORED.

INVALID BLANK OR MISSING COMMA IN '%ACTIVATE' OR '%DEACTIVATE' STATEMENT. TEXT IGNORED FROM T TO COMMA OR SEMICOLON.

Example:

```
%DEACTIVATE C5, C6 C7;
%DEACTIVATE IDENTIFIER;
```

IEL0131I S   NON-ITERATIVE 'DO' ASSUMED.

INVALID SYNTAX IN 'DO' STATEMENT. NON-ITERATIVE 'DO' ASSUMED.

Example:

```
%DO A: = 1 TO 10;
```

IEL0132I W   NO MAXIMUM VALUE FOR 'DO' ITERATION.

NO MAXIMUM VALUE SPECIFIED FOR 'DO' ITERATION.

Example:

```
%DO I = 1 BY 1; etc.
```

Explanation: This warning is given because the program may loop.

Programmer Response: If the program loops, provide an iteration limit or an alternative exit.

IEL0133I E SEMICOLON ASSUMED BEFORE '%'.

MISSING SEMICOLON ASSUMED BEFORE '%'.

Explanation: A percent found in the text has been assumed to signify the start of a new statement.

IEL0134I E SECOND 'TO' REPLACED BY 'BY'.

SECOND 'TO' FOUND IN ITERATION SPECIFICATION OF 'DO' STATEMENT. REPLACED BY 'BY'.

Example:

```
%DO I = 1 TO 10 TO 1;
%DO I = 1 TO 10 TO 1 BY 1;
```

(BY will have been ignored when this message occurs).

IEL0135I E SECOND 'BY' REPLACED BY 'TO'.

SECOND 'BY' FOUND IN ITERATION SPECIFICATION OF 'DO' STATEMENT. REPLACED BY 'TO'.

Example:

```
%DO I = 1 BY 1 BY 10;
```

IEL0136I E SEMICOLON MISSING. T TO NEXT SEMICOLON IGNORED.

MISSING SEMICOLON IN 'DO' STATEMENT. TEXT FROM T TO NEXT SEMICOLON IGNORED.

Example:

```
%DO I = 1 TO 10 BY 1 BY 7;
```

IEL0137I E NULL STATEMENT ASSUMED BEFORE 'END'.

'END' STATEMENT MAY NOT FOLLOW 'THEN' OR 'ELSE'. NULL STATEMENT ASSUMED BEFORE 'END'.

Example:

```
%DO; %IF C1 = C2 %THEN %END;
```

IEL0138I E SEMICOLON MISSING. T TO NEXT SEMICOLON IGNORED.

MISSING SEMICOLON IN 'END' STATEMENT. TEXT FROM T TO NEXT SEMICOLON IGNORED.

Explanation: A '%END' statement must be followed by a semicolon or by a label and a semicolon.

IEL0139I E REDUNDANT '%END' STATEMENT IGNORED.

REDUNDANT '%END' STATEMENT IGNORED.

Explanation: A %END statement is not preceded by a %DO or %PROCEDURE statement that has not already been terminated.

IEL0140I E REFERENCE TO UNKNOWN LABEL IGNORED.

LABEL REFERENCED IN '%END' STATEMENT NOT FOUND. REFERENCE IGNORED.

Explanation: The operand of the %END statement cannot be matched with the label on a %PROCEDURE or %DO statement which has not already got a matching %END statement.

IEL0141I E '%' ASSUMED BEFORE 'END' STATEMENT.

'%' ASSUMED BEFORE 'END' STATEMENT OF PROCEDURE.

Explanation: The END statement is the logical end of the procedure and should be preceded by '%'.

IEL0142I E T NOT A LABEL. IGNORED.

IDENTIFIER T ON '%END' STATEMENT NOT A LABEL. IDENTIFIER IGNORED.

Example:

```
%X = Y + A;
.
.
.
%END A;
```

IEL0143I E NO 'RETURN' STATEMENT IN PROCEDURE.

NO 'RETURN' STATEMENT IN PROCEDURE T. NULL VALUE WILL BE RETURNED.

Explanation: The PL/I language requires the use of a RETURN statement in a preprocessor procedure; a null value is returned if a procedure without a RETURN statement is invoked.

IEL0144I S '%INCLUDE' INVALID IN PROCEDURE.

'%INCLUDE' STATEMENT INVALID IN PREPROCESSOR PROCEDURE. STATEMENT IGNORED.

Example:

```
%PROC1: PROCEDURE (P1, P2) RETURNS(CHAR);
INCLUDE RUBBISH;
%END;
```

IEL0145I E DDNAME TRUNCATED TO N CHARACTERS.

PREPROCESSOR RESTRICTION. DDNAME TRUNCATED TO FIRST N CHARACTERS.

Explanation: The first of a pair of data-set identifiers is a ddname limited to a maximum of 8 characters.

IEL0146I S INVALID FIELD. TEXT IGNORED FROM T.

INVALID FIELD IN '%INCLUDE' STATEMENT. TEXT IGNORED FROM T TO NEXT COMMA OR SEMICOLON.

Example:

```
%INCLUDE 7RECORDS;
```

IEL0147I S STATEMENT HAS NO OPERAND. IGNORED.

'%INCLUDE' STATEMENT HAS NO OPERAND. STATEMENT IGNORED.

Example:

```
%INCLUDE;
```

IEL0148I E MEMBER NAME TRUNCATED TO N CHARACTERS.

PREPROCESSOR RESTRICTION. MEMBER NAME TRUNCATED TO FIRST N CHARACTERS.

Explanation: Only the first 8 characters of a data-set name are used.

IEL0149I E RIGHT PARENTHESIS ASSUMED.

MISSING RIGHT PARENTHESIS ASSUMED AFTER MEMBER NAME.

IEL0150I S BLOCK SIZE TOO LARGE FOR '%INCLUDE' FILE T.

BLOCK SIZE EXCEEDS N FOR '%INCLUDE' FILE T. PROCESSING OF FILE TERMINATED.

Explanation: When the INCLUDE compiler option is used, the maximum block size for an included data set is 4260 bytes.

Programmer Response: Use IEBGENER or a similar utility program to recreate the data set with the permitted block size.

IEL0151I S 'DECLARE' STATEMENT IGNORED.

'DECLARE' STATEMENT INVALID AFTER 'THEN' OR 'ELSE'. STATEMENT IGNORED.

Example:

```
%IF C1 = C2
%THEN %DCL C1 FIXED;
```

Explanation: A DECLARE statement can only appear after THEN or ELSE when inside a DO group.

IEL0152I E STATEMENT HAS NO OPERAND. IGNORED.

'%DECLARE' STATEMENT HAS NO OPERAND. STATEMENT IGNORED.

Example:

```
%DECLARE;
```

IEL0153I S MAXIMUM FACTORING LEVEL IS N.

PREPROCESSOR RESTRICTION. N LEVELS MAXIMUM FOR FACTORING IN 'DECLARE' STATEMENT. TEXT TO NEXT SEMICOLON IGNORED.

Explanation: A DECLARE statement with too many levels of factoring has been detected.

Programmer Response: Subdivide the DECLARE statement into two or more separate statements so that the level of factoring becomes acceptable.

IEL0154I E REDUNDANT COMMA IGNORED.

MISSING OPERAND OR REDUNDANT COMMA IN 'DECLARE' STATEMENT. COMMA IGNORED.

Example:

```
%DCL (C1, C2,, C3) CHAR;
```

IEL0155I E DUMMY IDENTIFIER ASSUMED.

IDENTIFIER MISSING WHERE EXPECTED. DUMMY ASSUMED.

Example:

```
DCL () CHAR;
```

Explanation: The preprocessor expected to find an identifier but found a delimiter.

IEL0156I E MULTIPLE DECLARATION OF D.

MULTIPLE DECLARATION OF IDENTIFIER D. REFERENCE WILL END PROCESSING.

Example:

```
%DCL C CHAR;
%DCL C CHAR;
```

Explanation: An identifier may be declared only once. No action is taken unless the multiply-declared identifier is referenced.

IEL0157I S INVALID SYNTAX. TEXT IGNORED FROM T.

INVALID SYNTAX IN 'DECLARE' STATEMENT. TEXT IGNORED FROM T TO NEXT SEMICOLON.

Example:

```
%DCL 7 FIXED;
```

IEL0158I E LABEL D CANNOT BE DECLARED.

LABEL D CANNOT BE DECLARED. REFERENCE WILL END PROCESSING.

Example:

```
%L: etc.
%DECLARE L FIXED;
```

IEL0159I E REDUNDANT RIGHT PARENTHESIS IGNORED.

REDUNDANT RIGHT PARENTHESIS IGNORED.

Example:

```
%DCL (B1, E2)) FIXED;
```

IEL0160I E T IGNORED.

INVALID ATTRIBUTE T IGNORED.

Example:

```
%DECLARE B BIT CHAR;
```

Explanation: The position in which an attribute is expected contains something other than FIXED, CHARACTER, BUILTIN, ENTRY, or RETURNS.

IEL0161I E RIGHT PARENTHESIS ASSUMED.

MISSING RIGHT PARENTHESIS ASSUMED.

Example:

```
DCL (C1,C2 CHAR;
```

IEL0162I E 'RETURNS' BUT NO 'ENTRY' ATTRIBUTE FOR D.

'RETURNS' BUT NO 'ENTRY' ATTRIBUTE FOR PROCEDURE D IN 'DECLARE' STATEMENT BEGINNING AT OR BEFORE THIS LINE.

Example:

```
     %DCL PROC2 RETURNS(FIXED);
```

Explanation: The identifier is treated as an entry name. The effect of this statement is to activate the entry name. This error will also cause message number IEL0069I to be printed.

IEL0163I E ATTRIBUTE T ASSUMED FOR D.

NO ATTRIBUTES DECLARED FOR IDENTIFIER D. T ASSUMED.

Example:

```
     %DCL A1, A2 CHAR;
```

Explanation: The attribute CHAR is assumed for an identifier declared without attributes, unless the identifier is given previously as a label on a PROCEDURE statement, in which case ENTRY is assumed.

IEL0165I S '%GOTO' D IS AN INVALID BRANCH.

'%GOTO' D IS AN INVALID BRANCH INTO INCLUDED TEXT. EXECUTION WILL END PROCESSING.

Explanation: A source statement module included in the text by a %INCLUDE statement contains a %GOTO statement that refers to a label contained in a source statement module included in the text by a further, nested, %INCLUDE statement.

IEL0168I E LABEL IGNORED.

LABEL INVALID ON LISTING CONTROL STATEMENT. LABEL IGNORED.

Example:

```
     %L: PAGE;
```

Explanation: A listing control statement should not be prefixed by a label.

IEL0169I E CONFLICTING ATTRIBUTE T FOR D IGNORED.

CONFLICTING ATTRIBUTES FOR IDENTIFIER D. ATTRIBUTE T IGNORED.

Example:

```
     %DCL P CHAR RETURNS(CHAR);
```

IEL0170I E CONFLICTING DECLARATION OF D.

DECLARATION OF IDENTIFIER D CONFLICTS WITH PREVIOUS USE. REFERENCE WILL END PROCESSING.

Example:

```
%E: PROC RETURNS(CHAR);
%END;
%DCL E CHAR;
```

IEL0171I E ZERO OPERAND ASSUMED.

MISSING OPERAND. FIXED DECIMAL ZERO ASSUMED.

Example:

```
%A = A +  ;
```

IEL0172I S T REPLACED BY PLUS.

INVALID OPERATOR T REPLACED BY PLUS.

Example:

```
%A = A**2;
```

Explanation: operators "**" and "->" are not allowed in preprocessor statements.

IEL0173I W BLANK ASSUMED AFTER T.

BLANK ASSUMED BETWEEN CONSTANT T AND FOLLOWING LETTER.

Example:

```
%DO A = '1'TO '3';
```

IEL0174I E 'NOT' REPLACED BY 'NE'.

OPERATOR 'NOT' USED AS INFIX OPERATOR. REPLACED BY 'NE'.

Example:

```
%B = (B1¬B2);
```

IEL0175I W TEXT FOLLOWING '%PAGE' IGNORED TO NEXT SEMICOLON.

PREPROCESSOR RESTRICTION. TEXT FOLLOWING '%PAGE' IGNORED TO NEXT SEMICOLON.

Example:

```
%PAGE ('NEW TITLE', 200);
```

Explanation: The preprocessor does not implement the TITLE or page numbering option of the %PAGE listing control statement.

IEL0176I E CONFLICTING USE OF D.

USE OF IDENTIFIER D IN EXPRESSION CONFLICTS WITH PREVIOUS

USE. REFERENCE WILL END PROCESSING.

Example:

```
%LAB:A = LAB + 2;
```

IEL0177I W TEXT FOLLOWS '%PAGE' OR '%SKIP' IN SAME LINE.

PREPROCESSOR RESTRICTION. TEXT IN SAME LINE AS '%PAGE' OR '%SKIP' STATEMENT. PREPROCESSOR INPUT LISTING NOT FORMATTED.

Example:

```
A=B;
%PAGE; A=B;
```

Explanation: The presence of source program text following a listing control statement prevents the preprocessor from formatting the input listing at this point.

IEL0178I S PLUS ASSUMED AS OPERATOR.

MISSING OPERATOR. PLUS ASSUMED.

Example:

```
%C = A   B;
```

IEL0179I S ZERO EXPRESSION ASSUMED.

EXPRESSION MISSING. FIXED DECIMAL ZERO ASSUMED.

Example:

```
%CL = ;
```

IEL0180I S T REPLACED BY ZERO.

INVALID OPERAND T REPLACED BY FIXED DECIMAL ZERO.

Example:

```
%A = B + 1C;
```

IEL0181I E LEFT PARENTHESIS ASSUMED.

MISSING LEFT PARENTHESIS ASSUMED AT BEGINNING OF EXPRESSION.

Example:

```
%F1 = F2 + F3);
```

IEL0182I U REFERENCE TERMINATED PROCESSING.

REFERENCE TO STATEMENT OR IDENTIFIER WHICH IS IN ERROR. PROCESSING TERMINATED.

Explanation: The preprocessor tried to execute a statement or use an identifier which is in error.

Programmer Response: Check the other messages for the error, and correct the program.

IEL0183I W EXCESS ARGUMENTS TO D IGNORED.

TOO MANY ARGUMENTS TO FUNCTION D. EXCESS ARGUMENTS IGNORED.

Example:

```
%DCL E ENTRY(CHAR);
%C = E(A,B);
```

Explanation: There are too many arguments in the procedure reference.

IEL0184I W TOO FEW ARGUMENTS TO D.

TOO FEW ARGUMENTS TO FUNCTION D. NULL STRINGS PASSED AS MISSING ARGUMENTS.

Example:

```
%E: PROCEDURE(P,Q) RETURNS(FIXED);
DECLARE (P,Q) FIXED;
%END;
%C = E(A);
```

Explanation: There are too few arguments in the procedure reference. For a fixed argument the null string will be converted to fixed zero.

|IEL0185I S RECORD LENGTH EXCEEDS N FOR MEMBER T.

LOGICAL RECORD LENGTH GREATER THAN N FOR '%INCLUDE' MEMBER T. RECORD TRUNCATED.

Explanation: For V-format or U-format records, the maximum permitted data length is 100 bytes.

Programmer Response: Recreate the data set with a permitted record length if necessary.

IEL0186I U PROCEDURE D NOT FOUND.

REFERENCED PROCEDURE D NOT FOUND. PROCESSING TERMINATED.

Explanation: An entry declaration statement has been found for a procedure which is not present in the text.

IEL0187I U RECURSIVE USE OF D INVALID.

RECURSIVE USE OF PROCEDURE D INVALID. PROCESSING TERMINATED.

Example:

```
%P: PROCEDURE RETURNS(CHAR);
RETURN (P + 7);
%END;
```

IEL0188I E NULL STRING RETURNED FOR 'SUBSTR'.

TOO FEW ARGUMENTS SPECIFIED FOR BUILTIN FUNCTION 'SUBSTR'. NULL STRING RETURNED.

Example:

```
%S = SUBSTR(A);
```

IEL0189I E EXCESS ARGUMENTS TO T IGNORED.

TOO MANY ARGUMENTS SPECIFIED FOR BUILTIN FUNCTION T. EXCESS ARGUMENTS IGNORED.

Example:

```
%S = SUBSTR(A,B,C,D);
```

IEL0190I E RESULT TRUNCATED TO 5 DIGITS.

FIXED OVERFLOW. RESULT TRUNCATED TO RIGHTMOST 5 DIGITS.

Example:

```
%A = 99999;
%A = A + 3;
```

IEL0191I E ZERO DIVIDE. RESULT SET TO ONE.

ZERO DIVIDE. RESULT SET TO ONE.

Example:

```
%A = 0;
%B = B/A;
```

IEL0192I S END OF SOURCE TEXT IN STATEMENT.

END OF SOURCE TEXT IN STATEMENT. STATEMENT EXECUTION WILL END PROCESSING.

IEL0193I E IDENTIFIER BEGINNING T TRUNCATED.

PREPROCESSOR RESTRICTION. IDENTIFIER BEGINNING T IS TOO LONG. TRUNCATED TO FIRST 31 CHARACTERS.

Explanation: Identifiers may not exceed 31 characters.

IEL0194I S T HAS PRECISION GREATER THAN N.

PREPROCESSOR RESTRICTION. CONSTANT T HAS PRECISION GREATER THAN N. FIXED DECIMAL ZERO ASSUMED.

Example:

```
     %A = 123456;
```

Explanation: The precision of fixed decimal numbers is limited to n digits. The value of zero is assumed for the constant.

IEL0195I E QUESTION MARK IGNORED.

QUESTION MARK IGNORED.

Explanation: Question mark has no syntactical meaning.

IEL0196I S PRECISION OF CONVERTED BIT STRING GREATER THAN N.

PREPROCESSOR RESTRICTION. BIT STRING CONVERTS TO FIXED DECIMAL NUMBER WITH PRECISION GREATER THAN N. RESULT TRUNCATED ON THE LEFT.

Example:

```
     %DECLARE C CHARACTER, F FIXED;
     %C = '1000000000000000000'B;
     %F = (C&C);
```

Explanation: If the bit string has more than 32 bits the last 32 bits are taken for the conversion.

IEL0197I S STRING INVALID FOR CONVERSION.

STRING CONTAINS CHARACTER NOT '1' OR '0' AND CANNOT BE CONVERTED TO BIT STRING. '0' ASSUMED FOR INVALID CHARACTERS.

Example:

```
     %C = 'A';
     %C = (C&(A¬=B));
```

IEL0198I S STRING INVALID FOR CONVERSION.

STRING CANNOT BE CONVERTED TO FIXED DECIMAL. FIXED ZERO RESULT ASSUMED.

Example:

```
     %C = '1B'
     %A = 2 + C;
```

IEL0199I E '%' MISSING ON LISTING CONTROL STATEMENT.

'%' MISSING ON LISTING CONTROL STATEMENT IN COMPILE-TIME PROCEDURE.

Example:

```
%P: PROC RETURNS(CHAR);
    SKIP(2);
%END;
```

Explanation: A listing control statement, even when in a preprocessor procedure, must be preceded by '%'.

IEL0200I U REFERENCE TO D TERMINATED PROCESSING.

IDENTIFIER D WITH CONFLICTING USE OR MULTIPLE DEFINITIONS REFERENCED. PROCESSING TERMINATED.

Explanation: An attempt has been made to execute a statement referencing an improperly defined identifier.

Programmer Response: Check the other messages for the error, and correct the program.

IEL0201I S D IS UNINITIALIZED.

UNINITIALIZED VARIABLE D USED. NULL STRING OR ZERO VALUE GIVEN.

Example:

```
%DECLARE A FIXED;
%B = A;
```

Explanation: Variables should be assigned values before being used.

IEL0202I U 'DD' STATEMENT FOR '%INCLUDE' D MISSING.

'DD' STATEMENT FOR '%INCLUDE' D IS MISSING. PROCESSING TERMINATED.

IEL0203I U I/C ERROR SEARCHING FOR '%INCLUDE' D.

UNRECOVERABLE I/O ERROR SEARCHING FOR '%INCLUDE' MEMBER D. PROCESSING TERMINATED.

IEL0204I U INVALID RECORD FORMAT FOR '%INCLUDE' D.

INVALID RECORD FORMAT SPECIFIED FOR '%INCLUDE' D. PROCESSING TERMINATED.

IEL0205I U '%INCLUDE' D NOT FOUND ON DATA SET.

'%INCLUDE' D MEMBER NOT FOUND ON DATA SET. PROCESSING

TERMINATED.

IEL0206I W RECORD LENGTH ASSUMED EQUAL TO BLOCKSIZE.

RECORD LENGTH NOT SPECIFIED FOR '%INCLUDE' D. ASSUMED EQUAL TO BLOCKSIZE.

IEL0207I W BLOCKSIZE ASSUMED EQUAL TO RECORD LENGTH.

BLOCKSIZE NOT SPECIFIED FOR '%INCLUDE' D. ASSUMED EQUAL TO RECORD LENGTH.

IEL0208I W RECORD LENGTH 80 AND BLOCKSIZE 400 ASSUMED.

RECORD LENGTH AND BLOCKSIZE NOT SPECIFIED FOR '%INCLUDE' D. RECORD LENGTH OF 80 AND BLOCKSIZE OF 400 ASSUMED.

IEL0209I U I/O ERROR IN READING FROM D.

I/O ERROR WHEN READING TEXT INCLUDED FROM FILE D. PROCESSING TERMINATED.

IEL0210I U LEVEL OF NESTING OR REPLACEMENT TOO LARGE.

PREPROCESSOR RESTRICTION. LEVEL OF NESTING OR REPLACEMENT GREATER THAN MAXIMUM. PROCESSING TERMINATED.

Explanation: The level of nesting is calculated by summing the number of current unbalanced left parentheses, the number of current nested DO's, the number of current nested IF's, and the number of current nested replacements. A level of 25 is always acceptable.

IEL0211I U INPUT RECORD LONGER THAN N.

INPUT RECORD LONGER THAN N CHARACTERS. PROCESSING TERMINATED.

Explanation: An input record contains more than 100 characters.

| IEL0212I S INPUT RECORD LENGTH LESS THAN LEFT MARGIN.

LENGTH OF INPUT RECORD LESS THAN LEFT MARGIN OF MARGINS OPTION. RECORD IGNORED.

Explanation: The length of an input record is less than the left margin of the MARGINS specification.

Programmer Response: Check the use of the MARGINS option; check that a short record is intended.

IEL0213I E 'RETURNS(FIXED)' ASSUMED.

DATA ATTRIBUTE IN '%PROCEDURE' STATEMENT IS NOT PARENTHESIZED AND IS NOT PRECEDED BY 'RETURNS'.

'RETURNS(FIXED)' ASSUMED.

Example:

```
     %P: PROC FIXED;
```

IEL0214I E   'RETURNS(CHAR)' ASSUMED.

DATA ATTRIBUTE IN '%PROCEDURE' STATEMENT IS NOT PARENTHESIZED AND IS NOT PRECEDED BY 'RETURNS'. 'RETURNS(CHAR)' ASSUMED.

Example:

```
     %P: PROC CHAR;
```

IEL0215I E   MISSING PARENTHESIS IN D ARGUMENT LIST.

RIGHT PARENTHESIS ASSUMED AT END OF ARGUMENT LIST FOR PROCEDURE D.

Example:

```
     %DCL C CHAR;
     %E: PROC(P) RETURNS(CHAR);
     .
     .
     .
     %END;
     %C = 'E(6';
```

IEL0216I U   INVALID STATEMENT IN D ARGUMENT LIST.

ARGUMENT LIST FOR PROCEDURE D CONTAINS A PREPROCESSOR STATEMENT. PROCESSING TERMINATED.

Example:

```
     %DCL P ENTRY, X FIXED;
     P(%X = 1;X) = 1;
```

Explanation: Preprocessor statements may not be embedded in the argument list of a preprocessor function reference appearing in non-preprocessor text.

IEL0217I E   LEFT PARENTHESIS MISSING FROM D ARGUMENT LIST.

MISSING LEFT PARENTHESIS FROM ARGUMENT LIST FOR PROCEDURE D. PROCEDURE INVOKED WITHOUT ARGUMENTS.

Example:

```
     %DCL E ENTRY(FIXED);
     %A = 'E + 1';
```

Explanation: The argument list is in a source program reference to a preprocessor procedure.

IEL0218I E   D USED FOR REPLACEMENT.

IDENTIFIER HAS MORE THAN N CHARACTERS. REPLACEMENT DONE ON TRUNCATED FORM D.

Explanation: An identifier activated for replacement by the preprocessor has more than the permitted number of characters. Consequently, any replacement will be performed on the given truncated form.

Programmer Response: Modify the program so that the identifier is reduced to an acceptable length or check that the replacement of the truncated form given does not result in further errors.

IEL0219I E   THIRD ARGUMENT OF 'SUBSTR' NEGATIVE.

THIRD ARGUMENT OF BUILTIN FUNCTION 'SUBSTR' NEGATIVE. NULL STRING RETURNED.

IEL0220I E   THIRD ARGUMENT OF 'SUBSTR' TOO LARGE.

THIRD ARGUMENT OF BUILTIN FUNCTION 'SUBSTR' GREATER THAN STRING LENGTH. RETURNED VALUE TRUNCATED AT END OF SOURCE STRING.

IEL0221I E   ARGUMENTS OF 'SUBSTR' TOO LARGE.

THE SUM OF THE SECOND AND THIRD ARGUMENTS OF BUILTIN FUNCTION 'SUBSTR' GREATER THAN STRING LENGTH PLUS ONE. RETURNED VALUE TRUNCATED AT END OF SOURCE STRING.

IEL0222I E   SECOND ARGUMENT OF 'SUBSTR' SET TO ONE.

SECOND ARGUMENT OF BUILTIN FUNCTION 'SUBSTR' LESS THAN ONE. VALUE SET TO ONE.

IEL0223I E   SECOND ARGUMENT OF 'SUBSTR' TOO LARGE.

SECOND ARGUMENT OF BUILTIN FUNCTION 'SUBSTR' GREATER THAN STRING LENGTH. NULL STRING RETURNED.

IEL0224I S   UNINITIALIZED VARIABLE IN ARGUMENT LIST.

UNINITIALIZED VARIABLE USED IN BUILTIN FUNCTION ARGUMENT LIST. NULL STRING ASSUMED.

Explanation: The variable should be initialized before invoking the built-in function. If a FIXED parameter is matched with a null string argument, the parameter will assume a value of zero.

IEL0226I E   RIGHT PARENTHESIS ASSUMED.

RIGHT PARENTHESIS ASSUMED BEFORE SEMICOLON IN '%PROCEDURE' STATEMENT.

Example:

```
%P: PROC RETURNS(CHAR;
```

IEL0227I E LEFT PARENTHESIS ASSUMED.

LEFT PARENTHESIS ASSUMED BEFORE MEMBER NAME.

Example:

```
%INCLUDE MEMBER);
```

IEL0228I E 'LENGTH' INVOKED WITH NO ARGUMENTS.

BUILTIN FUNCTION 'LENGTH' INVOKED WITH NO ARGUMENTS. FIXED ZERO RETURNED.

Example:

```
%A = LENGTH;
```

IEL0229I E 'INDEX' INVOKED WITH LESS THAN TWO ARGUMENTS.

BUILTIN FUNCTION 'INDEX' INVOKED WITH LESS THAN TWO ARGUMENTS. FIXED ZERO RETURNED.

Example:

```
     %A = INDEX ('ABCDE');
or   %A = INDEX;
```

IEL0230I U COMPILER ERROR NUMBER N DURING PHASE P.

COMPILER ERROR NUMBER N DURING PHASE P.

Explanation: An error has occurred during compilation. Processing has been terminated. This error is due to a fault in the compiler, not the source program. A detailed explanation of this message is given in the program logic manual for this compiler.

Programmer Response: Rerun the job, and if the problem recurs, call IBM for programming support. Before calling IBM, refer to the introduction for details of information that IBM will need in order to correct the problem.

IEL0231I U 48-CHARACTER SET RECORD LENGTH LESS THAN THREE.

COMPILER RESTRICTION. 48-CHARACTER SET RECORD LENGTH LESS THAN THREE. COMPILATION TERMINATED.

Explanation: The source margins for the compilation are less than three characters apart. Therefore, composite symbols of the 48-character set, such as "NOT" or "CAT" cannot be accomodated on a single record.

IEL0232I S 'PROCEDURE' ASSUMED AS FIRST STATEMENT.

FIRST STATEMENT NOT 'PROCEDURE'. 'PROCEDURE' STATEMENT ASSUMED.

Explanation: The first statement in a source program must be a PROCEDURE statement.

Programmer Response: The source program should be checked, particularly the control (that is, JCL and *PROCESS) statements and source margins. The source program should be correctly recorded on its input medium. Ensure that a PROCEDURE statement heads the source program.

| IEL0233I E COLON ASSUMED [AFTER T].

T ASSUMED TO BE STATEMENT LABEL. COLON ASSUMED.

Example:

```
X GOTO Y;
```

Explanation: A statement keyword is preceded by a possible label, but no colon is present.

IEL0234I S INVALID SYNTAX. T IGNORED.

STATEMENT BEGINS WITH INVALID SYNTAX. T IGNORED.

IEL0235I S STATEMENT ASSUMED TO BE CONTINUATION OF 'DECLARE'.

STATEMENT BEGINS WITH INVALID SYNTAX. ASSUMED TO BE CONTINUATION OF PRECEDING 'DECLARE' STATEMENT.

Example:

```
DCL A;B,C;
```

Explanation: An unrecognizable statement follows a DECLARE statement and is assumed to be a DECLARE statement also.

| IEL0236I S INPUT RECORD LENGTH LESS THAN LEFT MARGIN.

LENGTH OF INPUT RECORD LESS THAN LEFT MARGIN OF 'MARGINS' OPTION. RECORD IGNORED.

Programmer Response: Check the use of the MARGINS option, and/or that a short record is intentional.

IEL0237I S INVALID CHARACTER [AFTER T]. T IGNORED.

TEXT IN OR FOLLOWING THIS STATEMENT CONTAINS INVALID CHARACTER [AFTER T]. T IGNORED.

Example:

```
CALL E(A,B,?);
```

Explanation: The presence of an invalid character might be detected before the start of a statement. Consequently, the statement number may not be updated. When such an error is detected, the text is ignored from the start of the statement to the invalid character. The remaining characters in the statement will be treated as the complete statement. Consequently, other errors will almost certainly be indicated. These apparent errors will not be indicated if the program is recompiled with the invalid character corrected.

IEL0238I W CHARACTER STRING CONTAINS SEMICOLON.

CHARACTER STRING CONSTANT CONTAINS SEMICOLON.

Example:

```
STRING = 'B = C;';
```

Explanation: A common error is to omit one of a pair of quotation marks round a character string constant. The presence of a semicolon in a constant could be an indication of such an error, although it is not an error in itself.

IEL0239I W COMMENT CONTAINS SEMICCLCN.

COMMENTS IN OR FOLLOWING STATEMENT CONTAIN ONE OR MORE SEMICOLONS.

Example:

```
/* A = B; */
```

Explanation: A common error is to omit the delimiter '*/' after a comment. The presence of a semicolon in a comment could be an indication of such an error, although it is not an error in itself.

IEL0240I S QUOTE ASSUMED [AFTER T].

END OF SOURCE TEXT FOUND WITH UNMATCHED QUOTE. QUOTE ASSUMED [AFTER T].

Explanation: A quotation mark has been omitted causing the latter part of the program to appear as a string constant. A quote has been inserted prior to the first semicolon in this string. Note that statement numbers for statements following the statement in which the unmatched quote appears will not be printed.

Programmer Response: Check whether the quote was omitted or the source program is incomplete.

IEL0241I S 'END' STATEMENT(S) ASSUMED.

END OF SOURCE TEXT FOUND BEFORE LOGICAL END OF PROGRAM. N 'END' STATEMENT(S) ASSUMED.

Explanation: There are insufficient END statements to close all blocks. Any incomplete statements are ignored. Sufficient END statements are assumed in order to give valid nesting.

Programmer Response: Check the program block structure and that the source program is complete.

IEL0242I S STATEMENT TOO LARGE. T TO T IGNORED.

COMPILER RESTRICTION. STATEMENT EXCEEDS MAXIMUM LENGTH. TEXT IGNORED FROM T TO T.

Programmer Response: Divide the statement into two or more statements or remove any superfluous blanks.

IEL0243I S INVALID IDENTIFIER [AFTER T]. T REPLACED BY NULL.

INVALID IDENTIFIER FOLLOWING KEYWORD T. T REPLACED BY NULL STATEMENT.

Example:

1. GOTO *;

2. CALL 1;

3. ON(A

IEL0244I S QUOTE ASSUMED [AFTER T].

STATEMENT LENGTH MORE THAN COMPILER MAXIMUM AND CONTAINS UNMATCHED QUOTE. QUOTE ASSUMED [AFTER T].

Explanation: The compiler has assumed that the statement size appears to be too long because of the omission of a quote, and has assumed a quote prior to a semicolon within the statement. Note that statement numbers for statements following the statement in which the unmatched quote appears will not be printed.

IEL0245I S OPERAND INVALID [AFTER T].

OPERAND MISSING OR INVALID IN EXPRESSION [AFTER T].

Explanation: The compiler action depends on the context of the expression. A further message will indicate the action taken.

Programmer Response: Check for a further message for this statement.

IEL0246I S OPERATOR INVALID [AFTER T].

INVALID USE OF PREFIX OPERATOR [AFTER T].

Example:

```
     A = B + 4 ¬ C;
 └------------┘
         T
```

Explanation: The compiler action depends on the context of the expression. A further message will indicate the action taken.

Programmer Response: Check for the invalid use of an operator.

IEL0247I S INVALID SYNTAX. T REPLACED BY N.

INVALID SYNTAX IN 'IF' STATEMENT EXPRESSION. T HAS BEEN REPLACED BY N.

Example:

```
          T
      ┌----┐
    IF A+,B THEN GO TO LAB;
          ↑
        error
```

Explanation: The reason for the syntax error is diagnosed separately.

IEL0248I W INVALID USE OF '%CONTROL'.

INVALID USE OF '%CONTROL'.

IEL0249I E T SHORTENED TO T.

COMPILER RESTRICTION. IDENTIFIER T TOO LONG. SHORTENED TO T.

Explanation: The identifier is more than 31 characters long. The first 16 and last 15 characters are retained. This may cause the identifier to be no longer unique.

IEL0250I W OPTION T OBSOLETE BUT ACCEPTED.

'ENVIRONMENT' OPTION T IS OBSOLETE BUT IS ACCEPTED.

Example:

```
     Old - DCL F FILE ENV(V(100)...);

     New - DCL F FILE ENV(V BLKSIZE(100)...);
```

IEL0251I S CONSTANT T TOO LONG.

COMPILER RESTRICTION. ARITHMETIC CONSTANT T IS TOO LONG.

Explanation: The number of digits allowed depends on the type of constant, that is, fixed or float. The expression containing the constant is ignored. Further action is indicated by subsequent messagess depending on the context of the expression.

Programmer Response: Check the limits of the arithmetic constant and reduce it to an acceptable size.

IEL0252I S EXPONENT MISSING IN T.

EXPONENT MISSING IN FLOATING POINT CONSTANT T.

Example:

```
     A = 123E * B
```

Explanation: The character E is present but there are no digits following it. The expression containing the constant is ignored. Further action is indicated by subsequent messages depending on the context of the expression.

IEL0253I S   CHARACTER IN T NOT ZERO CR ONE.

CHARACTER IN BINARY CONSTANT T IS NOT ZERO OR ONE.

Explanation: The expression containing the constant is ignored. Further action is indicated by subsequent messages depending on the context of the expression.

Programmer Response: Check for a further message for this statement.

IEL0254I W   BLANK ASSUMED [AFTER T].

NO BLANK BETWEEN CONSTANT AND FOLLOWING LETTER. BLANK ASSUMED [AFTER T].

Example:

```
DCL 1 STRUC, 2 CODE CHAR(3), 2TEXT CHAR(77);
                              ↑
                              T
```

IEL0255I S   EXPONENT OF T TOO LONG.

COMPILER RESTRICTION. EXPONENT OF CONSTANT T TOO LONG.

Explanation: A floating-point constant has an exponent that exceeds the implementation-defined limit. The expression containing the constant is ignored. Further action is indicated by subsequent messages depending on the context of the expression.

Programmer Response: Check for a further message for this statement.

IEL0256I S   NO SIGNIFICANT DIGITS IN T.

CONSTANT T HAS NO SIGNIFICANT DIGITS.

Example:

```
1. .E2

2. .E
```

Explanation: The expression containing the "constant" is ignored. Further action is indicated by subsequent messages depending on the context of the expression.

Programmer Response: Check for a further message for this statement.

IEL0257I S   CHARACTER IN T NOT ZERO OR ONE.

CHARACTER IN BIT STRING CONSTANT T IS NOT ZERO OR ONE.

Explanation: The expression containing the constant is ignored. Further action is indicated by subsequent messages depending on the context of the expression.

Programmer Response: Check for a further message for this statement.

IEL0258I S INVALID PRECISION T IGNORED.

PRECISION SPECIFICATION NOT AN UNSIGNED INTEGER. T IGNORED.

Example:

```
     DCL G FIXED (+ABC) DECIMAL
                 └---┘
                   T
```

IEL0259I S PRECISION TRUNCATED [AFTER T].

SECOND INTEGER MISSING FROM PRECISION SPECIFICATION. PRECISION TRUNCATED [AFTER T] AFTER FIRST INTEGER.

Example:

```
     1. DCL A FIXED (9,X)
        ------------->
              T

     2. B FIXED (3, )
        --------->
           T

     3. C FIXED (4,D FLOAT
        --------->
           T
```

Explanation: The base factor is assumed to be zero.

IEL0260I S INVALID CHARACTER [AFTER T_1]. T_2 IGNORED.

INVALID CHARACTER IN PICTURE [AFTER T_1]. T_2 IGNORED.

Example:

```
          T2
     ┌----------┐
     PIC '99W9'
     -------->
          T1
```

IEL0261I S PARENTHESIS MISSING [AFTER T_1]. T_2 IGNORED.

RIGHT PARENTHESIS MISSING FROM SCALING FACTOR OR REPETITION FACTOR IN PICTURE [AFTER T_1]. T_2 IGNORED.

Example:

```
            T2
      ┌-----------┐
      PIC '99F(2 '
      ---------->
                T1
```

IEL0262I S   INVALID REPETITION FACTOR [AFTER T_1]. T_2 IGNORED.

REPETITION FACTOR NOT AN UNSIGNED INTEGER IN PICTURE [AFTER T_1]. T_2 IGNORED.

<u>Example</u>:

```
           T2
       ┌---------┐
    1. PIC'(+3)9'
       ----->
          T1

           T2
       ┌---------┐
    2. PIC'S(A)9'
       ------>
          T1
```

IEL0263I S   PICTURE INVALID [AFTER T_1]. T_2 IGNORED.

NO CHARACTER FOLLOWS REPETITION FACTOR IN PICTURE [AFTER T_1]. T_2 IGNORED.

<u>Example</u>:

```
         T2
      ┌---------┐
      PIC'59(3)'
      -------->
         T1
```

IEL0264I S   PICTURE INVALID [AFTER T_1]. T_2 IGNORED.

'F' NOT FOLLOWED BY LEFT PARENTHESIS IN PICTURE [AFTER T_1]. T_2 IGNORED.

<u>Example</u>:

```
         T2
      ┌--------┐
      PIC'99F3'
      ------>
        T1
```

IEL0265I S   PICTURE INVALID [AFTER T_1]. T_2 IGNORED.

INVALID SCALING FACTOR IN PICTURE [AFTER T_1]. T_2 IGNORED.

<u>Example</u>:

```
             T2
         ┌---------┐
      1. PIC'99F(*)
         ------->
             T1
```

```
              T2
          r---------┐
       2. PIC'99F( )
          ------->
             T1
```

IEL0266I S STATEMENT INVALID AFTER 'ELSE'.

NON-EXECUTABLE STATEMENT FOLLOWING 'ELSE'. NULL STATEMENT ASSUMED AS 'ELSE' CLAUSE.

Explanation: A null statement is assumed after the word ELSE so that the non-executable statement is no longer the ELSE clause.

Example:

```
IF A THEN B = 3; ELSE DCL C;
```

IEL0267I S STATEMENT INVALID AFTER 'THEN'.

NON-EXECUTABLE STATEMENT FOLLOWING 'THEN'. NULL STATEMENT ASSUMED AS 'THEN' CLAUSE.

Explanation: A null statement is assumed as the THEN clause, forcing the non-executable statement out of the compound IF statement.

Example:

```
IF A THEN FORMAT (A(3))...
```

IEL0268I S REFERENCE TO UNKNOWN LABEL IGNORED.

LABEL REFERENCED BY 'END' STATEMENT CANNOT BE MATCHED. REFERENCE IGNORED.

IEL0269I U TOO MANY 'PROCEDURE' 'BEGIN' AND 'ON' STATEMENTS.

COMPILER RESTRICTION. TOO MANY 'PROCEDURE' 'BEGIN' AND 'ON' STATEMENTS IN THE PROGRAM.

Explanation: The implementation restriction on the number of blocks in a compilation has been exceeded.

Programmer Response: Subdivide the program into two or more procedures for separate compilation, or rewrite it with less blocks.

IEL0270I U 'BEGIN' OR 'PROCEDURE' NESTING EXCEEDS MAXIMUM.

COMPILER RESTRICTION. 'BEGIN' OR 'PROCEDURE' STATEMENT NESTING MORE THAN MAXIMUM LEVEL.

Explanation: The implementation restriction on the level to which blocks may be nested has been exceeded.

Programmer Response: Reorganize the program to contain fewer levels of nested blocks.

IEL0271I S 'THEN' ASSUMED [AFTER T].

KEYWORD 'THEN' ASSUMED [AFTER T] IN 'IF' STATEMENT.

Explanation: The keyword THEN is missing from or incorrectly placed in the IF statement.

Example:

```
1. IF A = B GOTO L;

2. IF B&C IF D&E THEN DO;.....
```

Programmer Response: Check the IF statement.

IEL0272I S INVALID 'ON' UNIT. NULL STATEMENT ASSUMED.

INVALID ON-UNIT SPECIFIED. NULL STATEMENT ASSUMED.

Explanation: The specified statement may be a labeled statement, or an unlabeled statement not permitted as an on-unit. The null statement is assumed as the on-unit, and the text of the invalid on-unit is treated as one or more separate statements.

IEL0273I E PREFIXES ON 'ELSE' ASSUMED TO BE ON NEXT STATEMENT.

PREFIXES ON KEYWORD 'ELSE' ARE ASSUMED TO PRECEDE FOLLOWING STATEMENT.

Example:

```
IF A THEN B = 3;
L: ELSE B = 4;
```

Explanation: Labels and condition prefixes are transferred to the statement following ELSE.

IEL0274I S STATEMENT INVALID AFTER 'THEN'.

STATEMENT MISSING OR INVALID AFTER 'THEN'. NULL STATEMENT ASSUMED AS 'THEN' CLAUSE.

Explanation: No unit has been provided for the THEN clause.

Example:

```
IF A THEN ELSE B = 4;
```

IEL0275I S STATEMENT INVALID AFTER 'ELSE'.

STATEMENT MISSING OR INVALID AFTER 'ELSE'. NULL STATEMENT ASSUMED AS 'ELSE' CLAUSE.

Explanation: No unit has been provided for the ELSE clause.

Example:

```
     IF A THEN IF B THEN C = D; ELSE ELSE E = 4;
```

IEL0276I S 'ELSE' IN INVALID POSITION IGNORED.

KEYWORD 'ELSE' APPEARS IN INVALID POSITION. 'ELSE' IGNORED.

Example:

```
     IF A THEN B = C; D = E; ELSE...
```

Programmer Response: Correct the source program. Check that THEN clauses in nested IF statements are correct.

IEL0277I W 'SYSIN' OR 'SYSPRINT' ASSUMED FOR I/O 'ON' CONDITION.

I/O ON CONDITION HAS NO FILE NAME SPECIFIED. 'SYSIN' OR 'SYSPRINT' ASSUMED.

Explanation: ENDFILE (SYSIN) is assumed for input, and ENDPAGE (SYSPRINT) is assumed for output. All other I/O conditions are ignored and are assumed to be replaced by ON ERROR.

Example:

```
     ON ENDFILE SNAP;
     ON ENDPAGE PUT PAGE;
```

IEL0278I S INVALID CONDITION [AFTER T_1]. T_2 REPLACED BY 'ERROR'.

INVALID 'ON' CONDITION NAME [AFTER T_1]. T_2 REPLACED BY 'ERROR'.

Example:

```
     ON FRED A = B;
  ----> └---------┘
     T1    T2
```

IEL0279I S REDUNDANT COMMA [AFTER T] IGNORED.

MISSING ITEM OR REDUNDANT COMMA IN LIST [AFTER T]. COMMA IGNORED.

Explanation: An expected item has not been found following a left parenthesis or comma in a list, for example: a parameter list, a CHECK list, a FREE statement list, or a data list. The comma is ignored, or the whole list is ignored if it becomes null. Further action in addition to ignoring a null list is indicated by subsequent messages depending on the type of list concerned.

Example:

```
PUT DATA (,B,C);
```

Programmer Response: Correct the source program. Check also for further messages.

IEL0280I E LEFT PARENTHESIS ASSUMED [AFTER T].

LEFT PARENTHESIS ASSUMED [AFTER T].

Explanation: A left parenthesis has been omitted.

Example:

```
DO WHILE X = Y);
```

IEL0281I S ITERATIVE SPECIFICATION INVALID [AFTER T_1].

ITERATIVE SPECIFICATION INVALID [AFTER T_1].

Explanation: The control variable of expression1 is missing.

Example:

```
1. DO 1 TO 3...

2. DO 1,1..

3. DO J = *
```

IEL0282I S EXPRESSION MISSING AFTER 'TO' OR 'BY'.

EXPRESSION FOLLOWING 'TO' OR 'BY' IS MISSING IN 'DO' STATEMENT. NON-ITERATIVE 'DO' ASSUMED.

Example:

```
DO I = 1 BY;
```

IEL0283I S 'RETURN' STATEMENT WITHIN ON UNIT IGNORED.

'RETURN' STATEMENT IS WITHIN ON UNIT. STATEMENT IGNORED.

Example:

```
ON OVERFLOW RETURN;
```

IEL0284I S 'IN' NOT FOLLOWED BY LEFT PARENTHESIS.

KEYWORD 'IN' NOT FOLLOWED BY LEFT PARENTHESIS. 'IN' IGNORED.

<u>Example</u>:

```
1. ALLOCATE X IN A;

2. ALLOCATE A IN B);
```

IEL0285I S LABEL MISSING. DUMMY ASSUMED.

LABEL MISSING FROM 'PROCEDURE' OR 'ENTRY' STATEMENT. ONE HAS BEEN ASSUMED.

IEL0286I S 'ENTRY' IN 'BEGIN' BLOCK IGNORED.

'ENTRY' STATEMENT IS IN A 'BEGIN' BLOCK. STATEMENT IGNORED.

<u>Explanation</u>: The ENTRY statement and its labels are ignored.

<u>Example</u>:

```
B: BEGIN;
E: ENTRY;
END B;
```

IEL0287I S 'IN' OPTION INVALID [AFTER T_1]. T_2 IGNORED.

INVALID 'IN' OPTION [AFTER T_1] IN 'FREE' STATEMENT. T_2 IGNORED.

<u>Example</u>:

```
FREE FRED IN  (25 + AREA);
               ↑
               T1
           |              |
           L--------------J
                  T2
```

IEL0288I S INVALID TEXT. T IGNORED.

INVALID TEXT WITHIN STATEMENT. T IGNORED.

<u>Explanation</u>: Invalid text has been found within a statement, for example, an invalid attribute or option. The text is ignored. Scanning of the source program restarts at the next recognizable item.

IEL0289I S 'END' FOUND BEFORE END OF SOURCE TEXT.

LOGICAL END OF PROGRAM FOUND BEFORE END OF SOURCE TEXT. STATEMENT IGNORED.

Explanation: In order to check the syntax of the whole source text, the END statement which prematurely terminates the program has been ignored. This may cause some extra errors in subsequent PROC, BEGIN, or END statements.

Example:

```
P: PROC OPTIONS (MAIN);
END;    (message produced here)
GOTO LAB;
END;
/*
```

IEL0290I S INVALID OPTION [AFTER T_1]. T_2 IGNORED.

INVALID OR MULTIPLE SPECIFICATION OF OPTION [AFTER T_1]. T_2 IGNORED.

Explanation: The option may:

1. have an invalid argument.

2. be specified more than once.

3. be spelt incorrectly.

4. have no argument.

|IEL0291I E INVALID SYNTAX [AFTER T] IN 'LABEL' ATTRIBUTE.

INVALID SYNTAX FOR LABEL CONSTANT [AFTER T_1]. T_2 IGNORED.

Example:

```
                     T2
                 ┌---------------┐
DCL LAB LABEL(LAB1,6AB2,LAB3);
          -------- >
                  T1
```

Explanation: The compiler has detected an item in the list of label constants which does not begin with an alphabetic character.

Programmer Response: Correct the specification of the label constant.

IEL0292I S LABEL LIST TOO LONG. T IGNORED.

COMPILER RESTRICTION. LABEL PREFIX LIST TOO LONG. LABEL T HAS BEEN IGNORED.

Explanation: The number of label prefixes plus the total number of characters in the label list must not exceed 254. The label prefix list is truncated at the nearest point below the permitted maximum.

Programmer Response: The program should be rewritten with shorter or fewer labels prefixed to this statement. Excess labels may be transferred to an immediately preceding null statement.

IEL0293I S INVALID PREFIX [AFTER T]. T IGNORED.

INVALID CONDITION PREFIX [AFTER T]. T IGNORED.

Explanation: A colon is not present after a prefix list which contains at least one valid prefix condition.

Example:

```
(SUBRG) PROC1: PROCEDURE;
```

IEL0294I E T FOLLOWS LABEL BUT IS ACCEPTED.

CONDITION PREFIX T FOLLOWS LABEL BUT IS ACCEPTED.

Example:

```
L:(FOFL): A = B;
```

Explanation: Condition prefix lists should precede any statement label lists. However, this compiler permits condition prefixes other than "CHECK" and "NOCHECK" to follow any statement labels.

IEL0295I S T FOLLOWS LABEL AND IS IGNORED.

'CHECK' OR 'NOCHECK' CONDITION PREFIX FOLLOWS LABEL. T IGNORED.

Example:

```
L: (CHECK(A),FOFL) : A = B;
    |        |
    L--------J
        T
```

Explanation: The check list should precede a label.

IEL0296I E COLON ASSUMED [AFTER T].

COLON ASSUMED AFTER T. PARENTHESIZED ITEM ASSUMED TO BE CONDITION PREFIX.

Example:

```
(FIXEDOVERFLOW) A=B*C;
```

IEL0297I S ARGUMENT LIST INVALID [AFTER T_1]. T_2 IGNORED.

ARGUMENT LIST MISSING OR INVALID [AFTER T_1]. T_2 IGNORED.

Example:

```
     READ FILE(F) INTO (3);
          -------------↑
                       T1
                |        |
                └--------┘
                         T2
```

IEL0298I E   CONDITION PREFIX INVALID.

CONDITION PREFIX INVALID ON THIS STATEMENT. PREFIX LIST IGNORED.

Explanation: Condition prefix lists are invalid on ENTRY, DECLARE, DEFAULT, and FORMAT statements.

IEL0299I S   FACTORING INVALID [AFTER T].

FACTORING SPECIFIED IN 'ALLOCATE' STATEMENT [AFTER T]. TEXT IGNORED TO NEXT SEMICOLON.

Explanation: No factoring of parentheses or factored attributes are allowed in an ALLOCATE statement. The ALLOCATE statement is ignored and a null statement is assumed.

IEL0300I S   'INITIAL' FACTORING LEVEL [AFTER T] EXCEEDS N.

COMPILER RESTRICTION. FACTORING LEVEL [AFTER T] IN 'INITIAL' EXCEEDS N. ATTRIBUTE IGNORED.

IEL0301I S   SIGN IN T IGNORED.

SIGN IN STRUCTURE LEVEL NUMBER T IGNORED.

Example:

```
     DCL + 1 A,....
```

Explanation: The level number in a DECLARE statement must be an unsigned decimal integer.

IEL0302I S   ZERO [AFTER T] ASSUMED TO BE ONE.

ZERO LEVEL NUMBER [AFTER T] ASSUMED TO BE ONE.

Explanation: The level number in a DECLARE statement must be an unsigned non-zero integer.

IEL0303I S   T IN 'RETURNS' INVALID.

ATTRIBUTE T IN 'RETURNS' INVALID. ATTRIBUTE IGNORED.

IEL0304I S   INVALID SYNTAX [AFTER T_1]. T_2 IGNORED.

INVALID SYNTAX IN ASSIGNMENT STATEMENT [AFTER T_1]. T_2 IGNORED.

__Example__:

```
            T2
        ┌----------┐
     1.  A + B = C;
        --->
            T1

        ┌----┐
     2.  A = ;
        ---->
         T1
```

IEL0305I W INVALID USE OF '%PAGE' OR '%SKIP'.

INVALID USE OF %PAGE OR %SKIP. SOURCE INPUT LISTING NOT FORMATTED.

__Explanation__: The listing control statements must appear between statements and on a separate line from them.

IEL0306I S NO MATCHING FORMAT LIST [AFTER T].

EDIT DATA LIST HAS NO MATCHING FORMAT LIST [AFTER T]. T FORMAT ASSUMED.

__Explanation__: Edit-directed transmission statements require format lists.

IEL0307I S INVALID SYNTAX [AFTER T_1]. T_2 IGNORED.

INVALID SYNTAX IN DATA LIST [AFTER T_1]. T_2 IGNORED.

__Explanation__: The data list has an item missing or has an error in a do-loop specification. The data list is ignored from the invalid item.

IEL0308I S FORMAT LIST INVALID [AFTER T_1]. T_2 IGNORED.

FORMAT ITEM MISSING OR INVALID [AFTER T_1]. T_2 IGNORED.

__Explanation__: The format item has been omitted, has an invalid argument, or is incorrectly spelt. The invalid item is ignored, and the text is scanned for the next item.

IEL0309I W 'FORMAT' STATEMENT HAS NO LABEL.

'FORMAT' STATEMENT HAS NO LABEL.

__Explanation__: A FORMAT statement cannot be referenced without a label.

IEL0311I S   COMMENT DELIMITER ASSUMED [AFTER T].

END OF SOURCE TEXT FOUND WITHIN A COMMENT. COMMENT DELIMITER ASSUMED [AFTER T].

Explanation: A comment delimiter may have been omitted, causing the latter part of the program to appear as a comment. A comment delimiter is inserted at the end of the last source statement.

Programmer Response: Check whether the comment delimiter has been omitted or if the source program is incomplete.

IEL0312I U   NO TEXT IN PROGRAM.

NO TEXT IN PROGRAM.

Example:

```
* PROCESS A,X;
/* PROGRAM STARTS HERE */
.
.
.
```

Explanation: The first comment delimiter is in positions 1 and 2 of the record following the PROCESS statement, and is interpreted as an end-of-file delimiter for the input to the compiler. The compiler has not received any source statements to compile into an object module. Reasons for this include the error shown above, control statements out of sequence, and so on.

IEL0313I S   INVALID KEYWORD [AFTER T_1]. T_2 IGNORED.

INVALID KEYWORD [AFTER T_1] IN REPETITIVE SPECIFICATION. T_2 IGNORED.

Example:

```
PUT LIST((A(I) DO I = 3 IF A>B));
                         ↑
                 |       T1       |
                 L----------------J
                         T2
```

IEL0314I S   END OF SOURCE TEXT FOUND. T IGNORED.

END OF SOURCE TEXT FOUND BEFORE END OF STATEMENT. T IGNORED.

Explanation: This can happen in addition to "end of source text found before logical end of program".

Example:

```
1. DCL A FIXED, B FLOAT, C STATIC (end of file)
```

```
2. A = B; C = D + (end of file)
```

IEL0315I S LABEL T ON 'ON' UNIT IGNORED.

A LABEL ON AN ON UNIT IS INVALID. T IGNORED.

Example:

```
ON OFL L: GOTO LAB;
```

The label L is invalid.

IEL0316I S SEMICOLON ASSUMED [AFTER T].

END OF STATEMENT ASSUMED [AFTER T]. TEXT IGNORED TO NEXT SEMICOLON.

Explanation: A semicolon has not been found where expected after a syntactically correct statement, so one is assumed.

Example:

```
DELAY (25) THEN STOP;
```

IEL0317I S ATTRIBUTE INVALID [AFTER T_1]. T_2 IGNORED.

INVALID ATTRIBUTE SPECIFICATION [AFTER T_1]. T_2 IGNORED.

Example:

```
              T2
          r---------1
DCL JOE FOXED;
------>
   T1
```

IEL0318I S 'DO' IN 'ON' UNIT REPLACED BY 'BEGIN'.

'DO' STATEMENT IS INVALID IN 'ON' UNIT. REPLACED BY 'BEGIN'

Example:

```
ON OFL DO;
PUT SKIP;
END;
```

Explanation: The only valid on-units are single statements or begin blocks.

IEL0319I S MULTIPLE USE OF OPTION. T IGNORED.

STATEMENT USES AN OPTION MORE THAN ONCE. T IGNORED.

Example:

```
                          T
                      ┌--------┐
          DISPLAY(A) EVENT(B) EVENT(C) REPLY(R)'
```

IEL0320I S   NO 'REPLY' OPTION. TEXT [AFTER T] IGNORED.

'DISPLAY' STATEMENT HAS NO 'REPLY' OPTION. TEXT [AFTER T] IGNORED.

Example:

```
          DISPLAY ('HELP') EVENT (E);
          └---------------┘
                 T
```

IEL0321I E   LEFT PARENTHESIS ASSUMED BEFORE EXPRESSION.

MISSING LEFT PARENTHESIS ASSUMED BEFORE EXPRESSION IN 'DELAY' OR 'DISPLAY' STATEMENT.

Example:

```
          DISPLAY MESSAGE);
```

IEL0322I S   INVALID FORMAT ITEM [AFTER T_1]. T_2 IGNORED.

INVALID SPECIFICATION IN FORMAT ITEM [AFTER T_1]. T_2 IGNORED

Example:

```
               T2
          ┌-----------┐
          C(A(3), B(2))
     ------>
        T1
```

IEL0323I E   RIGHT PARENTHESIS ASSUMED [AFTER T].

REPETITIVE SPECIFICATION ENDING AT T IN DATA LIST NOT FOLLOWED BY RIGHT PARENTHESIS. ONE HAS BEEN ASSUMED.

Example:

```
          PUT EDIT ((A(I) DO I = 1 TO 3 ) (F(3));
                                       ↑
                 missing right parenthesis assumed to be here
```

Explanation: Repetitive specifications in data lists must be enclosed in brackets.

IEL0324I S   NESTING LEVEL EXCEEDS N [AFTER T].

COMPILER RESTRICTION. LEVEL OF NESTING EXCEEDS N IN DATA LIST [AFTER T]. STATEMENT IGNORED.

Explanation: If there are no redundant brackets, rewrite the statement within the implementation limits.

IEL0325I S INVALID SYNTAX IN 'CALL' STATEMENT.

INVALID SYNTAX IN 'CALL' STATEMENT. STATEMENT IGNORED.

Example:

```
     CALL (A,B);
```

IEL0326I S 'ENTRY' AND LABEL INSIDE 'DO' IGNORED.

'ENTRY' STATEMENT AND LABEL INSIDE ITERATIVE 'DO' IGNORED.

Explanation: Note that the label is also ignored, so that calls to it will be unresolved.

Example:

```
     DO I = 1 TO 3;
     A(I) = B(I);
     E: ENTRY;
     A(I) = C(I);
     END;
```

IEL0327I S INVALID SYNTAX. T IGNORED .

STATEMENT BEGINS WITH INVALID SYNTAX. T IGNORED.

Example:

```
     1.  13 14) * .¬ |);

         No identifier found in this statement.

     2.  IF A ,GOTO LAB;
               ↑
       'THEN' assumed here

         ",GOTO LAB;" ignored since error follows a fixup.
```

Explanation: Either the statement type could not be identified, or, due to a fixup of an error in the previous statement, recovery was not attempted from the error in the current statement.

IEL0328I S INVALID OPTION [AFTER T]. T IGNORED.

INVALID OPTION [AFTER T] IN 'PROCEDURE' 'BEGIN' OR 'ENTRY' STATEMENT. T IGNORED.

Example:

```
     1. P: PROC MAIN;

     2. B: BEGIN (A,B);
```

IEL0329I S NESTING LEVEL EXCEEDS N [AFTER T].

COMPILER RESTRICTION. LEVEL OF NESTING EXCEEDS N IN FORMAT LIST [AFTER T]. STATEMENT IGNORED

Programmer Response: If there are no redundant brackets, rewrite the statement within the implementation limits.

IEL0330I S NO '=' [AFTER T_1]. T_2 IGNORED.

NO '=' [AFTER T_1] IN REPETITIVE SPECIFICATION. T_2 IGNORED.

Example:

```
PUT LIST ((A(I) DO I TO N))
                   ↑
                 | T1      |
                 L---------J
                     T2
```

IEL0331I S INVALID CONTROL VARIABLE [AFTER T_1]. T_2 IGNORED.

INVALID CONTROL VARIABLE [AFTER T_1] IN REPETITIVE SPECIFICATION. T_2 IGNORED.

Example:

```
PUT LIST ((A(I) DO 3 TO 4))
                  ↑
                 | T1     |
                 L--------J
                     T2
```

IEL0332I S PARENTHESIS NESTING LEVEL EXCEEDS N [AFTER T].

COMPILER RESTRICTION. LEVEL OF PARENTHESIS NESTING GREATER THAN N [AFTER T]. STATEMENT IGNORED.

Example:

```
A(B(C(-----------(A3(B3
                 ↑
                 T
```

IEL0333I U STATEMENT NESTING LIMIT EXCEEDED.

COMPILER RESTRICTION. NESTING LIMIT OF 'PROCEDURE'/ 'BEGIN'/'IF'/'DO' STATEMENTS HAS BEEN EXCEEDED. PROCESSING TERMINATED.

Explanation: The stack containing PROCEDURE, BEGIN, IF, and DO statements and their labels has overflowed.

Programmer Response: Either reduce the number or length of the labels on these statements or restructure the program to reduce the depth of nesting.

IEL0334I S OPTION(S) T MISSING FROM STATEMENT.

OPTION(S) T MISSING FROM RECORD I/O STATEMENT. STATEMENT IGNORED.

Example:

```
READ FILE(F) KEYTO(K); (INTO option missing)
WRITE FILE(F); (FROM option missing)
```

Programmer Response: Ensure that a correct set of options is specified for this statement.

IEL0335I S T IN 'OPTIONS' LIST IGNORED.

INVALID ITEM IN 'OPTIONS' LIST. T IGNORED.

Example:

```
P: PROC OPTIONS(MOAN);
P: PROC OPTIONS (NOMAPIN(3));
```

Programmer Response: Check the list of valid options and their specification.

IEL0336I S VARIABLE MISSING FROM 'LOCATE'.

VARIABLE MISSING FROM 'LOCATE' STATEMENT. STATEMENT IGNORED.

IEL0337I E COLON ASSUMED [AFTER T].

CONDITION PREFIX NOT FOLLOWED BY COLON. COLON ASSUMED [AFTER T].

Example:

```
(STRZ) A = B;
      ↑
colon missing
```

IEL0338I S MULTIPLE 'TO' OR 'BY' [AFTER T_1]. T_2 IGNORED.

MULTIPLE 'TO' OR 'BY' [AFTER T_1] IN REPETITIVE SPECIFICATION. T_2 IGNCRED.

Example:

```
PUT LIST ((A(I) DO I = 1 TO 4 TO 40))
                  ---------------↑
                   |            T1    |
                   L-----------------J
                               T2
```

IEL0339I S FILE OPTION MISSING. T IGNORED.

MISSING FILE OPTION OR REDUNDANT COMMA IN 'OPEN' OR 'CLOSE' STATEMENT. T IGNORED.

Example:

```
     OPEN FILE(F2), FILE(F3) STREAM, OUTPUT;
                                     |    |
                                     L----J
                                        T
```

|IEL0340I S   INVALID SYNTAX [AFTER T_1]. T_2 IGNORED.

INVALID SYNTAX IN 'ENVIRONMENT' OPTION [AFTER T_1]. T_2 IGNORED.

Example:

```
     DCL-----ENV(REROAD,HIGHINDEX(2741))
                   ↑               ↑
                 error           error
```

Explanation: Possible causes for this are:

1. An invalid keyword, or keyword subset has been used (only LEAVE and REREAD are valid in the CLOSE statement).

2. An option has an incorrect or missing argument.

IEL0341I S   INVALID ITEM [AFTER T_1]. T_2 IGNORED.

INVALID ITEM IN PARAMETER LIST [AFTER T_1]. T_2 IGNORED.

Example:

```
                    T2
                  r---1
     P: PROC(P1,  3*P2);
     -----------↑
         T1
```

IEL0342I S   INVALID OPTION [AFTER T]. T IGNORED.

INVALID OPTION IN 'DEFAULT' STATEMENT [AFTER T]. T IGNORED.

Example:

```
     DEFAULT RINGE(A:B) FIXED;
             L----------------J
                     T
```

Explanation: The only valid options of the DEFAULT statement are RANGE and DESCRIPTORS.

IEL0343I S   INVALID IDENTIFIER [AFTER T_1]. T_2 IGNORED.

INVALID IDENTIFIER [AFTER T_1] IN 'RANGE' SPECIFICATION. T_2 IGNORED.

Example:

```
DEFAULT RANGE (A:BC) BINARY;
        --------↑
        |       T1         |
        L------------------J
                T2
```

Explanation: The syntax rules for the RANGE option of the DEFAULT statement are given in the language reference manual for this compiler.

IEL0344I S INVALID IDENTIFIER [AFTER T_1]. T_2 IGNORED.

INVALID IDENTIFIER [AFTER T_1] IN 'GENERIC' SPECIFICATION. T_2 IGNORED.

Example:

```
DCL E  ENTRY GENERIC ( (FLOAT),---
            ---------↑
                     T1
                      |       |
                      L-------J
                         T2
```

Explanation: The syntax rules for the GENERIC attribute are given in the language reference manual for this compiler.

IEL0345I S INVALID 'POSITION' ATTRIBUTE. T IGNORED.

INVALID EXPRESSION IN 'POSITION' ATTRIBUTE. T IGNORED.

Example:

```
DCL P2 DEF P1 POS(3,4);
              |      |
              L------J
                 T
```

IEL0346I S INVALID IDENTIFIER [AFTER T_1]. T_2 IGNORED.

INVALID IDENTIFIER [AFTER T_1] IN NAME LIST. T_2 IGNORED.

Example:

```
ON CHECK(A,3) GOTO LAB;
          ↑↑
         T1T2
```

IEL0347I S INVALID KEYWORD. T IGNORED.

INVALID KEYWORD IN ATTRIBUTE SPECIFICATION IN 'DEFAULT' STATEMENT. T IGNORED.

Example:

```
DEFAULT RANGE(A) VILUE((31,0)) FIXED;
                 |            |
                 L------------J
                      T
```

Explanation: The syntax rules for the DEFAULT statement are given in the language reference manual for this compiler.

IEL0348I S 'WHEN' OPTION MISSING [AFTER T_1]. T_2 IGNORED.

'WHEN' OPTION MISSING [AFTER T_1] IN 'GENERIC' SPECIFICATION. T_2 IGNORED.

Example:

```
DCL E ENTRY GENERIC(E1 IF (FLOAT),...
           ----------↑
                     T1
                     |            |
                     └------------┘
                           T2
```

Explanation: The rules for the GENERIC attribute are given in the language reference manual for this compiler.

IEL0349I S INVALID EXPRESSION [AFTER T_1]. T_2 REPLACED BY 10.

INVALID EXPRESSION [AFTER T_1] IN DIMENSION SPECIFICATION. T_2 REPLACED BY 10.

Example:

```
DCL A(P+Q,P-Q,P/+-):
      ------> ↑
       |T1|  |T2|
       └--┘  └--┘
```

Explanation: The erroneous expression is replaced to ensure that the required number of array dimensions is maintained. Subsequent subscripted references to the array will be correct if this number of dimensions is used.

IEL0350I S INVALID OPTION [AFTER T_1]. T_2 IGNORED.

INVALID OPTION IN 'GET' OR 'PUT' STATEMENT [AFTER T_1]. T_2 IGNORED.

Example:

```
PUT LIST(A) TWICE;
----------↑ └----┘
      T1      T2

GET PAGE DATA(D);
---↑└--┘
  T1 T2
```

Explanation: The option is invalid or inapplicable to this type of statement.

IEL0351I S EXPRESSION INVALID OR MISSING.

EXPRESSION INVALID OR MISSING IN 'DELAY' OR 'DISPLAY' STATEMENT. STATEMENT IGNORED.

<u>Example</u>:

```
1. DELAY;

2. DISPLAY )++;
```

<u>Explanation</u>: If an erroneous expression causes this message to be produced, it will also be indicated by a separate message.

IEL0352I S INVALID OPTION [AFTER T_1]. T_2 IGNORED.

INVALID SPECIFICATION OF OPTION [AFTER T_1]. T_2 IGNORED.

<u>Example</u>:

```
READ FILE(F) FROM(DATA);
           ↑
           T1
             |         |
             L---------J
                  T2
```

IEL0353I E COMMA ASSUMED [AFTER T].

CONSTANT FOUND IN ATTRIBUTE LIST. COMMA ASSUMED [AFTER T].

<u>Example</u>:

```
DCL 1 STRUCT,
    2 FRED (comma missing here)
    3 JOE FLOAT;
```

<u>Explanation</u>: This error, and its correction by the compiler, can only occur where structure levels are used in a DECLARE statement.

IEL0354I S NO IDENTIFIER [AFTER T_1]. T_2 IGNORED.

'DECLARE' 'DEFAULT' OR 'ALLOCATE' DOES NOT HAVE AN IDENTIFIER [AFTER T_1]. T_2 IGNORED.

<u>Example</u>:

```
DCL 1 J, 2 + FIXED, 2 F FLOAT;
    ------↑         |
          T1        |
       |            |
       L------------J
          T2
```

IEL0355I S DATA LIST MISSING [AFTER T].

DATA LIST MISSING [AFTER T]. STATEMENT IGNORED.

Example:

```
     PUT EDIT SKIP(3);
     --------↑
            T
```

Explanation: Only data-directed output statements can be used without a data list.

IEL0356I S INVALID IDENTIFIER [AFTER T_1]. T_2 IGNORED.

INVALID IDENTIFIER [AFTER T_1] IN 'FREE' STATEMENT. T_2 IGNORED.

Example:

```
     FREE A,B, (C.D) IN (AREA);
          ---↑
            T1
            |       |
            L-------J
               T2
```

IEL0357I W TOO FEW PARENTHESES FOR TEXT [AFTER T] TO BE 'DO' SPECIFICATION.

DATA LIST CONTAINS TOO FEW PARENTHESES FOR TEXT [AFTER T] TO BE REPETITIVE SPECIFICATION. ASSUMED TO BE DATA LIST ITEMS.

Example:

```
     PUT DATA (A(I) DO I = 3 TO 4);

     should be:

     PUT DATA ((A(I) DO I = 3 TO 4));

     but is assumed to be:

     PUT DATA (A(I), DO...etc.,
```

Explanation: A repetitive specification must leave extra brackets for each do-group. For example:

```
     PUT LIST (((A(I,J) DO I = 1 TO 2) DO J = 3 TO 4));
```

IEL0358I S NO EXPRESSION [AFTER T_1]. T_2 ASSUMED.

EXPRESSION MISSING FROM FORMAT ITEM [AFTER T_1]. T_2 ASSUMED

Example:

```
     PUT EDIT (A) (F(3),X); (T2=(1))
                ---------↑
                         T1
```

IEL0359I S PREFIX OPTIONS CONFLICT.

PREFIX OPTIONS CONFLICT. THE DISABLING PREFIX HAS BEEN ASSUMED.

Example:

```
     (CONV,OFL,NOCONV): A=B+C;
```

IEL0360I S NO EXPRESSION [AFTER T].

EXPRESSION MISSING FROM 'A' FORMAT ITEM [AFTER T]. 'ERROR' CONDITION WILL BE RAISED ON EXECUTION.

Example:

```
     GET EDIT (P,Q) (F(3),A);
```

Explanation: On input, an edit-directed A-format item must specify the number of characters to be read.

IEL0361I S WRONG NUMBER OF ARGUMENTS [AFTER T_1]. T_2 IGNORED.

WRONG NUMBER OF ARGUMENTS IN 'FORMAT' ITEM [AFTER T_1]. T_2 IGNORED.

Example:

```
     PUT EDIT (A) (F(A,B,3,4), E(3));
```

IEL0362I S COMMA ASSUMED [AFTER T].

COMMA ASSUMED [AFTER T].

Example:

```
     PUT EDIT (A) (B,A X(2));
```

Explanation: A comma is assumed wherever the syntax of a statement requires one in order to be valid.

IEL0363I S PICTURE INVALID [AFTER T]. T IGNORED.

CHARACTER SPECIFICATION [AFTER T] IN PICTURE IS INVALID IN COMPLEX FORMAT ITEM. T IGNORED.

Example:

```
     PUT EDIT (A) (C(F(3),P'99A'));
```

IEL0364I E INVALID SYNTAX [AFTER T].

INVALID SYNTAX IN LISTING CONTROL STATEMENT [AFTER T]. TEXT IGNORED TO NEXT SEMICOLON.

Example:

```
          %SKIP(1
          -------↑
                 T
           A=B;
```

IEL0365I S   INVALID SYNTAX [AFTER T].

INVALID SYNTAX IN 'DECLARE' OR 'DEFAULT' STATEMENT [AFTER T]. STATEMENT IGNORED.

Explanation: A statement beginning with either "DCL(..." or "DEFAULT(..." that is not a DECLARE or DEFAULT statement has been encountered and cannot be compiled.

Programmer Response: Replace the identifier DCL (or DECLARE) or DEFAULT with an identifier that is not a keyword, and recompile the program.

IEL0366I W   STATEMENT NOT SUPPORTED.

STATEMENT IS NOT SUPPORTED AND IS IGNORED.

Example:

```
     HALT;
```

Explanation: The PL/I keywords CHECK, NOCHECK, FLOW, NOFLOW, and HALT are not supported by the optimizing compiler and are ignored if they appear in the source program.

IEL0367I W   INVALID SYNTAX AFTER T. STATEMENT NOT SUPPORTED.

INVALID SYNTAX AFTER T. STATEMENT IS NOT SUPPORTED AND IS IGNORED.

Example:

```
     CHECK(A,2,B)
```

Explanation: The PL/I keywords CHECK and NOCHECK are not recognized by the optimizing compiler.

IEL0368I W   OPTION T NOT SUPPORTED.

OPTION T NOT SUPPORTED. STATEMENT IGNORED.

Example:

```
     PUT ALL;
```

Explanation: The statements PUT ALL, PUT FLOW, and PUT SNAP are not recognized by the optimizing compiler.

IEL0369I W CONDITION NOT SUPPORTED.

CONDITION NOT SUPPORTED. 'ON' STATEMENT AND 'ON' UNIT IGNORED.

Example:

```
     ON ATTENTION BEGIN;
                  GOTO LAB;
                  END;
```

Explanation: The condition ATTENTION is not recognized by the optimizing compiler.

IEL0370I S DATA LIST INVALID [AFTER T_1]. T_2 IGNORED.

INVALID USE OF REPETITIVE SPECIFICATION IN DATA LIST FOR 'GET' STATEMENT [AFTER T_1]. T_2 IGNORED.

Example:

```
     GET DATA (A,B,(C DO I = 1 TO 3),D);
```

Explanation: A repetitive specification is not permitted in a GET DATA statement.

IEL0371I S FORMAT LIST INVALID [AFTER T].

FORMAT LIST MISSING OR INVALID AFTER T IN 'FORMAT' STATEMENT. 'A' FORMAT ASSUMED.

IEL0372I W INVALID CARRIAGE CONTROL CHARACTER T.

CARRIAGE CONTROL CHARACTER T IS INVALID. BLANK ASSUMED FOR CHARACTER.

Explanation: An invalid ANS print control character has been specified in a source record associated with the given statement. The permissible characters are: blank, 0, -, +, and 1.

IEL0373I S PICTURE T EXCEEDS MAXIMUM LENGTH.

COMPILER RESTRICTION. 'PICTURE' SPECIFICATION EXCEEDS MAXIMUM LENGTH. 'PICTURE' SPECIFICATION T IGNORED.

Example:

```
     DCL PICTURE PIC'(600)X';
         PICTURE PIC'(255)9V(2)9';
```

Explanation: The maximum length of a PICTURE variable is 256 characters including insertion characters.

IEL0374I I   TOO MANY STATEMENTS IN THIS RECORD FOR CORRECT NUMBERING.

LINE CONTAINS MORE THAN 30 STATEMENTS. NUMBER OF ALL FOLLOWING STATEMENTS IN LINE SET TO CONSTANT VALUE.

Explanation: The constant value set for statements that cannot be individually numbered is N+1.

IEL0375I E   FACTOR NESTING LEVEL EXCEEDS MAXIMUM AFTER T. T IGNORED.

COMPILER RESTRICTION. MAXIMUM FACTOR DEPTH EXCEEDED AFTER T. T IGNORED.

Explanation: The depth of factorization used in this statement has exceeded the maximum permitted by the compiler.

IEL0376I I   'TASK' SPECIFIED. PROCEDURE ASSUMED REENTRANT.

'TASK' OPTION SPECIFIED. PROCEDURE ASSUMED TO BE REENTRANT.

Explanation: The optimizing compiler does not generate special code for the TASK option, but as tasking procedures must normally be reenterable, the REENTRANT option is assumed.

IEL0377I W   BLANK ASSUMED [AFTER T].

NO BLANK BETWEEN KEYWORD AND FOLLOWING STRING. BLANK ASSUMED [AFTER T].

Example:

```
     DO I = 1 TO'3';
```

IEL0378I W   NON-INCREASING RECORD SEQUENCE NUMBER FOLLOWS.

NON-INCREASING RECORD SEQUENCE NUMBER FOLLOWS THIS STATEMENT. LINE NUMBERS MODIFIED.

Example:

```
Number                    Input Sequence Number
     1  P:PROC;                    0001
     2   A=B;                      0002
                                   0003
                                   0005
100004 END P;                      0004
```

Explanation: The compiler checks the sequence number given in the sequence number field of each source statement record. If the number is equal to or less than the preceding number, the number in the sequence number field is increased by 100000 for the purposes of the number used for the GONUMBER option. The sequence number quoted in the message refers to the record in which the latest PL/I statement began. Thus, in the example above, although the message would refer to record number 4, the record number actually quoted in the message would be '2'.

IEL0379I W   CONDITION T NOT SUPPORTED.

CONDITION T NOT SUPPORTED. STATEMENT IGNORED.

Example:

```
     SIGNAL ATTENTION;
```

IEL0380I S   'LIKE' IGNORED

'LIKE' IS INVALID IN ENTRY PARAMETER DESCRIPTOR LIST AND IS IGNORED.

Example:

```
     DCL TEST1 ENTRY (LIKE TEST) EXTERNAL;
```

IEL0381I E   INVALID 'INITIAL' ATTRIBUTE [AFTER T].

INVALID SPECIFICATION OF 'INITIAL' ATTRIBUTE [AFTER T]. ATTRIBUTE IGNORED.

Example:

```
     DCL A(4) FIXED INIT A(+1,+2,+3,+X);
         -------------------------->|
                        T
```

Explanation: Invalid syntax has been detected in the specification of a constant, expression, or function reference in the INITIAL attribute. Thus, in the above example, an invalid arithmetic constant X would be diagnosed.

|IEL0382I E   INVALID OPTION AFTER T_1.

INVALID OPTION AFTER T_1 IN RECORD I/O STATEMENT. OPTION T_2 ASSUMED.

Example:

```
          WRITE FILE (F) FROM (CARD) KEY (NUM);
                             ----------↑
                                       T1
               T2 = KEYFROM
```

Explanation: An inappropriate KEY, KEYTO, or KEYFROM, option has been specified for this RECORD I/O statement.

IEL0399I E   SEMICOLON ASSUMED [AFTER T].

SEMICOLON ASSUMED [AFTER T].

Example:

```
     IF X THEN GOTO Y ELSE;
               ------↑
                     T
```

IEL0400I E RIGHT PARENTHESIS ASSUMED [AFTER T].

RIGHT PARENTHESIS ASSUMED [AFTER T].

Example:

```
 1.   A = B + (C*D;
          -----↑
               T

 2.   DO WHILE (A¬B;
          -----↑
               T
```

IEL0401I S MORE THAN N QUALIFICATIONS IN NAME BEGINNING T.

COMPILER RESTRICTION. MORE THAN N QUALIFICATIONS IN NAME BEGINNING T. EXCESS QUALIFICATIONS IGNORED.

Example:

```
     A=B1.B2.B3....B64;
```

Explanation: The optimizing compiler permits up to 15 levels of structuring.

IEL0402I E LABEL VALUE LIST IGNORED.

LABEL VALUE LIST INVALID FOR 'DEFAULT' STATEMENT. LIST IGNORED.

Example:

```
     DEFAULT RANGE (L) LABEL (LAB1,LAB2);
```

This becomes:

```
     DEFAULT RANGE (L) LABEL;
```

Explanation: A label value list cannot be used with the DEFAULT statement.

IEL0403I S QUALIFICATION OR SUBSCRIPT ON ENTRY PREFIX IGNORED.

COMPILER RESTRICTION. QUALIFIED OR SUBSCRIPTED ENTRY PREFIX ON 'PROCEDURE' OR 'ENTRY' STATEMENT. QUALIFICATION OR SUBSCRIPT IGNORED.

Example:

```
     P: EV(3): ENTRY;
```

Explanation: The optimizing compiler does not permit initialization of aggregates of entry variables by the appearance of the subscripted or qualified entry variable name as a prefix to an ENTRY statement.

IEL0404I E ADJUSTABLE EXTENT INVALID IN 'RETURNS'.

ADJUSTABLE EXTENT INVALID IN 'RETURNS' SPECIFICATION. EXTENT IGNORED.

Example:

```
DCL X RETURNS(CHAR(Y));
```

IEL0405I E ARGUMENT SPECIFICATION T IGNORED.

INVALID ARGUMENT SPECIFICATION IN INTERLANGUAGE OPTION. T IGNORED.

Example:

```
DCL E ENTRY OPTIONS (COBOL NOMAP(FRED))
```

Explanation: The argument should be specified as ARGn where "n" is the number indicating the position in the argument list of the argument to which the interlanguage option is to apply.

IEL0406I E PARAMETER SPECIFICATION T IGNORED.

INVALID PARAMETER SPECIFICATION IN INTERLANGUAGE OPTION. T IGNORED.

Example:

```
E: ENTRY(X) OPTIONS(FORTRAN NOMAP(Y));
```

Explanation: The argument to the NOMAP, NOMAPIN, or NOMAPOUT options must be a parameter specified in the same PROCEDURE or ENTRY statement.

IEL0407I E INVALID OPTION T IGNORED.

INVALID OPTION T IGNORED.

Example:

```
DCL E ENTRY OPTIONS(MAIN);
```

|IEL0408I E CONFLICTING 'OPTIONS' SPECIFICATION. T ASSUMED.

CONFLICTING SPECIFICATION OF INTERLANGUAGE OPTIONS. T ASSUMED.

Example:

```
DCL SUB ENTRY OPTIONS(FORTRAN, COBOL);
```

Explanation: Conflicting interlanguage options have been found in an options list. The COBOL, FORTRAN, and ASSEMBLER options conflict with each other. The last of these to be specified is assumed.

IEL0409I E LENGTH OR PRECISION NOT IN 'VALUE' CLAUSE.

STRING OR AREA LENGTH OR PRECISION SPECIFICATION IN 'DEFAULT' STATEMENT IS NOT IN 'VALUE' CLAUSE. RESULTS OF EXECUTION UNDEFINED.

Example:

```
DEFAULT RANGE(S) CHAR(3);
```

Explanation: String lengths and area sizes should be specified inside a VALUE clause.

IEL0410I E ATTRIBUTE T INVALID FOR 'DEFAULT'.

ATTRIBUTE T INVALID FOR 'DEFAULT' STATEMENT. ATTRIBUTE IGNORED.

Example:

```
DEFAULT RANGE(A) ENTRY
   (ENTRY ignored)

DEFAULT RANGE(B) BUFFERED
   (BUFFERED ignored)
```

Explanation: Neither the RETURN, ENTRY, and LIKE attributes, nor file description attributes are permitted in a DEFAULT statement.

IEL0411I U ATTRIBUTE FACTORING LEVEL EXCEEDS N.

COMPILER RESTRICTION. ATTRIBUTE FACTORING LEVEL GREATER THAN N. PROCESSING TERMINATED.

Explanation: More than 15 levels of attribute factorization have been used.

Programmer Response: Expand the declaration containing the error into separate declarations.

IEL0412I S MORE THAN N PARAMETERS.

COMPILER RESTRICTION. MORE THAN N PARAMETERS. LIST TRUNCATED.

Explanation: More than 64 parameters have been declared in a PROCEDURE or ENTRY statement or in an ENTRY attribute.

IEL0413I E DECLARATION OF D IGNORED.

DECLARATION OF INTERNAL ENTRY NOT ALLOWED. DECLARATION OF D IGNORED.

Example:

```
          A:  PROC;
            DCL B ENTRY RETURNS (FIXED);
            B:  PROC ENTRY RETURNS(FIXED);
                  .
                  .
               END B;
               .
               .
               .
            END A;
```

Explanation: An internal entry point is declared according to its PROCEDURE or ENTRY statement. It cannot be declared in the invoking block in a DECLARE statement.

IEL0414I S NESTED 'LIKE' ATTRIBUTE IN DECLARATION OF D.

'LIKE' ATTRIBUTE IN DECLARATION OF D REFERENCES STRUCTURE WHICH CONTAINS 'LIKE'. EXPANSION TRUNCATED AT LATTER 'LIKE'.

Example:

```
     DCL 1 A, 2 B, 2 C LIKE A, 2 D, 3 B;
```

This becomes:

```
     DCL 1 A, 2 B, 2 C, 3 B, 3 C,

     (expansion truncated here)

     2 D, 3 B;
```

IEL0415I S 'LIKE' REFERENCE FOR D IS NOT A STRUCTURE.

'LIKE' REFERENCE IN DECLARATION OF D NOT A STRUCTURE. 'LIKE' ATTRIBUTE IGNORED.

Example:

```
     DCL A, 1 B LIKE A;
```

IEL0416I S 'LIKE' REFERENCE FOR D IS AMBIGUOUS.

AMBIGUOUS 'LIKE' REFERENCE IN DECLARATION OF D. UNDEFINED SELECTION OF POSSIBILITIES MADE.

Example:

```
     DCL 1 A, 2 B, 3 C, 4 D, 2 E, 3 C;
     DCL 1 X LIKE (A.C);
```

Explanation: An ambiguity has arisen through an incomplete qualification, and an undefined selection of one of the possible resolutions is made.

IEL0417I S 'LIKE' ATTRIBUTE FOR D REFERS TO INVALID STRUCTURE.

'LIKE' ATTRIBUTE IN DECLARATION OF D REFERENCES STRUCTURE WHICH IS UNDECLARED OR CONTAINS 'LIKE' ATTRIBUTE. FORMER 'LIKE' ATTRIBUTE IGNORED.

Example:

```
1. X: PROC; DCL 1 A LIKE B; END;

2. DCL 1 A, 2 B LIKE C; DCL 1 C, 2 D LIKE E;
```

IEL0418I U TOO MANY 'DEFAULT' SPECIFICATIONS AND 'LIKE' ATTRIBUTES.

COMPILER RESTRICTION. TOO MANY DEFAULT SPECIFICATIONS AND 'LIKE' ATTRIBUTES IN ONE BLOCK. PROCESSING TERMINATED.

Explanation: Details of default specification within the current scope, and LIKE attributes not yet resolved for the current blocks, are held in a directory. The total number of these specifications and attributes should be kept below 300 by expanding LIKE declarations and merging defaults.

IEL0419I S INVALID ATTRIBUTE SPECIFICATION IN 'VALUE' CLAUSE.

CONFLICTING OR REPEATED OR INVALID ATTRIBUTE SPECIFICATION IN 'VALUE' CLAUSE. RESULTS OF EXECUTION UNDEFINED.

Example:

```
DEFAULT RANGE(*) VALUE (FIXED
 CHAR(1), (BIN(17), FLOAT(3))DEC);
```

Explanation: When an illegal combination of attributes appears (after any defactoring of attributes has been performed) the combination has no effect. Individual attributes may still appear, however, and have effect in other combinations. In the above example, the attribute combinations FIXED CHAR(1) and DEC BIN(17) will be ignored, whereas the combination DEC FLOAT(3) will be accepted.

IEL0420I E PRECISION OR EXTENT MISSING IN 'VALUE' CLAUSE.

PRECISION OR EXTENT SPECIFICATION MISSING FOR ATTRIBUTE IN 'VALUE' CLAUSE. ATTRIBUTE IGNORED.

Example:

```
DEFAULT RANGE(*) VALUE (CHAR,FIXED BIN);
```

Explanation: The precision or extent specification must be included in an attribute specification in a VALUE clause.

IEL0421I S MULTIPLE DECLARATION OF D.

MULTIPLE DECLARATION OF D IN SAME STRUCTURE.

Example:

```
DCL 1 A, 2 B, 2 C, 2 B;
```

Explanation: For fully qualified references to the multiply-defined structure number, the last declaration will be taken. Incompletely qualified references will be further diagnosed as being ambiguous.

IEL0422I S MULTIPLE DECLARATION OF D IGNORED.

MULTIPLE DECLARATION OF D. DECLARATION IGNORED.

Explanation: For a multiply-declared item, all declarations but one are ignored.

Example:

```
1. DCL A, A;

2. DCL A;
   DCL A;
```

IEL0423I S MAJOR STRUCTURE LEVEL NUMBER ASSUMED TO BE 1.

MAJOR STRUCTURE LEVEL NUMBER NOT ONE. NUMBER REPLACED BY ONE.

Example:

```
DCL 2 G, 3 H;
```

IEL0424I S LOGICAL LEVEL NUMBER OF MEMBER REDUCED TO N.

COMPILER RESTRICTION. LOGICAL LEVEL NUMBER OF STRUCTURE MEMBER TOO LARGE. REDUCED TO N.

Example:

```
DCL 1 A, 370 B...;
```

IEL0425I S DECLARED LEVEL NUMBER OF MEMBER REDUCED TO N.

COMPILER RESTRICTION. DECLARED LEVEL NUMBER OF STRUCTURE MEMBER TOO LARGE. REDUCED TO N.

Example:

```
DCL 1 A, 300 B;
```

IEL0426I E INVALID REPETITION OF T.

INVALID REPETITION OF ATTRIBUTE T. SECOND SPECIFICATION IGNORED.

Example:

```
DCL (X,Y) CHAR(1) CHAR(2);
                  |       |
                  L-------J
                      T
```

IEL0427I E ATTRIBUTE T FOR D IGNORED.

ATTRIBUTE T IN DECLARATION OF D IGNORED.

Example:

```
     DCL X FIXED FLOAT;
                ↑
                T
```

Explanation: A conflicting, invalid, or repeated attribute in a declaration will be ignored. The particular attribute that is ignored is the one that also conflicts with the declaration of the identifier after default attributes have been applied, or that is invalid or repeated.

IEL0428I E AMBIGUOUS 'DEFAULT' FOR D. T IGNORED.

AMBIGUOUS DEFAULT SPECIFICATION IN DECLARATION OF D. ATTRIBUTE T IGNORED.

Example:

```
     DEFAULT RANGE(X) FLOAT;
     DEFAULT RANGE(V:Z) FIXED;
     DCL X;
```

IEL0429I E 'DEFAULT' AMBIGUOUS FOR RANGE T. T IGNORED.

'DEFAULT' SELECTION IS AMBIGUOUS FOR ANY CONTEXTUAL OR IMPLICIT DECLARATION IN RANGE T. ATTRIBUTE T IGNORED.

Example:

```
     DEFAULT RANGE (J:R) FIXED;
     DEFAULT RANGE (H:N) FLOAT;
```

An ambiguity exists for the range (J:N).

Explanation: The default ranges given in two or more range specifications should not overlap. The range given in the message is the extent of the ambiguous range. This message is produced even when there are no implicit declarations within the ambiguous range.

IEL0430I I NO 'MAIN' OPTION ON PROCEDURE.

NO 'MAIN' OPTION ON EXTERNAL PROCEDURE.

Example:

```
     P:PROC;
```

Explanation: An external procedure without the MAIN option cannot be executed unless link-edited with another external procedure with the MAIN option.

IEL0431I S PICTURE CHARACTER AFTER T REPLACED BY T.

INVALID 'PICTURE' SPECIFICATION. CHARACTER [AFTER T] REPLACED BY T.

Example:

```
DCL P PIC'59+';

is assumed to be:

DCL P PIC'599';
```

Explanation: Depending on circumstances, an invalid picture specification character is replaced either by '9' when valid or by '.'.

IEL0432I S SUBFIELD OF T HAS NO DIGIT POSITIONS.

SUBFIELD OF 'PICTURE' T HAS NO DIGIT POSITIONS. RESULTS OF EXECUTION ARE UNDEFINED.

Example:

```
DCL P PIC'$CR';
```

IEL0433I S PRECISION OF SUBFIELD OF T EXCEEDS N.

COMPILER RESTRICTION. PRECISION OF SUBFIELD OF PICTURE T EXCEEDS N. PICTURE SPECIFICATION IGNORED.

Example:

1. PIC '(16)9'
2. PIC '9E999'

Explanation: The maximum precision of a numeric picture is 15 for the mantissa and 2 for the exponent.

IEL0434I S T TRUNCATED AT INVALID 'F'.

PICTURE T TRUNCATED AT INVALID 'F' SPECIFICATION.

Example:

```
DCL P PIC'9E9F(3)';

is assumed to be:

DCL P PIC '9E9';
```

IEL0435I S INVALID PICTURE T.

INVALID PICTURE T. PICTURE TEXT FOLLOWING 'F' SPECIFICATION IGNORED.

Example:

```
DCL P PIC '99V9F(-3)9';

is assumed to be:

DCL P PIC '99V9 F(-3)';
```

IEL0436I E INVALID PICTURE. T REPLACED BY 'X'.

INVALID CHARACTER PICTURE SPECIFICATION. T REPLACED BY 'X'.

Example:

```
PIC '9XR'

is assumed to be:

PIC '9XX'
```

IEL0437I E PRECISION OF D REDUCED TO N.

COMPILER RESTRICTION. PRECISION OF D TOO LARGE. N ASSUMED FOR PRECISION.

Example:

```
DCL B BINARY (32,0),
    D DEC(17);
```

Explanation: The maximum precisions for arithmetic data types are given in the language reference manual for this compiler.

IEL0438I E INVALID 'RANGE' T IGNORED.

INVALID 'RANGE' SPECIFICATION T. SPECIFICATION IGNORED.

Example:

```
DEFAULT RANGE (C:B) BIN;
```

IEL0439I E ZERO VALUE ASSUMED FOR SCALE FACTOR.

COMPILER RESTRICTION. SCALE FACTOR IS OUTSIDE VALID RANGE. ZERO VALUE ASSUMED.

Example:

```
DCL F FIXED (6,-200);
```

IEL0440I E T WITHIN 'RETURNS' IGNORED.

COMPILER RESTRICTION. ATTRIBUTE T INVALID IN 'RETURNS' SPECIFICATION. ATTRIBUTE IGNORED.

Example:

```
1.  DCL  X RETURNS(RETURNS(FIXED));

2.  DCL Y RETURNS(ENTRY);
```

Explanation: A function procedure cannot return a value that is an entry name.

IEL0441I S   D HAS BOUND GREATER THAN N.

COMPILER RESTRICTION. D DECLARED WITH ARRAY BOUND GREATER THAN N. N ASSUMED FOR BOUND.

Example:

```
DCL A(10:32768);
```

Explanation: An array bound cannot be greater than 32767.

IEL0442I S   D HAS BOUND LESS THAN N.

COMPILER RESTRICTION. D DECLARED WITH ARRAY BOUND LESS THAN N. N ASSUMED FOR BOUND.

Example:

```
DCL A(-32769:10);
```

Explanation: An array bound cannot be less than -32768.

IEL0443I S   LOWER BOUND N OF D GREATER THAN HIGHER BOUND.

LOWER BOUND N GREATER THAN HIGHER BOUND IN DECLARATION OF D. BOUNDS INTERCHANGED.

Example:

```
DCL A(5:2);
```

Explanation: The lower bound of an array dimension must be declared to be numerically lower than the higher bound.

IEL0444I S   D HAS MORE THAN N DIMENSIONS.

COMPILER RESTRICTION. D DECLARED WITH NUMBER OF DIMENSIONS GREATER THAN N. NUMBER OF DIMENSIONS REDUCED.

Example:

```
DCL A(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16);
```

Explanation: An array cannot be declared with more than 15 dimensions.

IEL0445I S T NOT AN ENTRY NAME. IGNORED.

T IN 'GENERIC' SPECIFICATION IS NOT AN ENTRY NAME AND IS IGNORED.

Example:

```
DCL E1 ENTRY;
DCL E2 FILE VARIABLE;
DCL F GENERIC
(E1 WHEN (FIXED), E2 WHEN(FLOAT));
```

Explanation: Only names of entry points can begin in the declaration of a GENERIC entry name.

IEL0446I S REFERENCE TO T IS AMBIGUOUS.

REFERENCE TO T IN 'GENERIC' SPECIFICATION IS AMBIGUOUS. UNDEFINED SELECTION MADE.

Example:

```
DCL F GENERIC
     (E1 WHEN(FIXED), E2 WHEN(FLOAT)),
E2 ENTRY,
1 X, 2 E1 ENTRY,
Y LIKE X;
```

Explanation: An entry expression in the declaration of a GENERIC entry name must be an unambiguous reference to an entry constant or variable.

IEL0447I E QUALIFICATION OF ATTRIBUTE T FOR D INVALID.

QUALIFICATION OF ATTRIBUTE T SPECIFIED FOR MEMBER D IN 'GENERIC' SPECIFICATION IS INVALID. QUALIFICATION IGNORED.

Example:

```
DCL G GENERIC (E1 WHEN (BIT),
    E2 WHEN(CHAR(3)));
```

Explanation: Details of the attributes permitted in a generic descriptor list are given in the language reference manual for this compiler.

IEL0448I S ATTRIBUTE T FOR D INVALID.

INVALID ATTRIBUTE T FOR MEMBER D IN 'GENERIC' SPECIFICATION. ATTRIBUTE IGNORED.

Example:

```
DCL F GENERIC
      (E1 WHEN (FIXED),
      E2 WHEN (FLOAT,BASED));
```

Explanation: Only the following attributes can be used in a generic descriptor: ALIGNED, AREA, base, BIT, CHARACTER, ENTRY, EVENT, FILE, LABEL, mode, OFFSET, PICTURE 'picture specifications', POINTER, precision, scale, UNALIGNED, and VARYING. String lengths, area sizes, and label lists are not permitted.

IEL0449I S T CONFLICTS WITH PREVIOUS ATTRIBUTES FOR D.

ATTRIBUTE T CONFLICTS WITH PREVIOUS ATTRIBUTES OF MEMBER D IN 'GENERIC' SPECIFICATION. ATTRIBUTE IGNORED.

Example:

```
     DCL F GENERIC
           (E1 WHEN(FIXED),E2 WHEN(FLOAT,FIXED));
```

Explanation: When the attributes in a generic descriptor conflict, the second of the conflicting attributes is ignored.

|IEL0450I E T IN VALUE LIST OF D NOT A LABEL CONSTANT.

T IN LABEL VALUE LIST OF LABEL VARIABLE D IS NOT A LABEL CONSTANT AND IS IGNORED.

Example:

```
     P:  PROC;
         DCL L LABEL (L1,L2,L3);
     L1: ;
     L2: ;
        END L3 (is not a label)
```

Explanation: A label constant given in a label list should appear in the block within the scope of the label list.

IEL0451I S ADJUSTABLE EXTENTS FOR D INVALID WITH 'STATIC'.

ADJUSTABLE EXTENTS INVALID WITH 'STATIC' STORAGE CLASS IN DECLARATION OF D. N ASSUMED FOR EXTENT.

Example:

```
     DCL A (4:N) STATIC;
     DCL C CHAR(N) STATIC;
```

Explanation: Static variables cannot have an adjustable bound, extent, or length.

IEL0452I S ADJUSTABLE EXTENT INVALID FOR PARAMETER D.

ADJUSTABLE EXTENTS INVALID WITH 'PARAMETER' STORAGE CLASS IN DECLARATION OF D. '*' ASSUMED FOR EXTENT.

Example:

```
X:  PROC (P);
    DCL P(Y); (becomes P(*))
    END;
```

Explanation: A parameter cannot have an adjustable bound, extent, or length, but it can assume that of its argument if specified as '*'.

IEL0453I S ADJUSTABLE EXTENT INVALID FOR BASED D.

ADJUSTABLE EXTENT INVALID WITH 'BASED' STORAGE CLASS IN DECLARATION OF D. N ASSUMED FOR EXTENT.

Example:

1. DCL A(I:8) BASED;

   in this case I is assumed to be 1.

2. DCL B(4:J) BASED;

   in this case J is assumed to be 10.

Explanation: A based array cannot have adjustable bounds, a based area cannot have an adjustable extent, and a based string cannot have an adjustable length. If specified, an adjustable lower bound is assumed to be 1, an adjustable upper bound is assumed to be 10, an adjustable string length is assumed to be 1, and an adjustable area extent is assumed to be 1000.

IEL0454I S '*' EXTENT INVALID FOR D NOT 'CONTROLLED' OR 'PARAMETER'.

'*' EXTENT SPECIFIED IN DECLARATION OF D BUT NOT 'CONTROLLED' OR 'PARAMETER'. N ASSUMED FOR EXTENT.

Example:

```
DCL A (*) STATIC;
```

Explanation: An '*' bound, extent, or length can only be used to declare an adjustable bound, extent, or length for a controlled variable or a parameter.

IEL0455I S 'REFER' EXTENT INVALID FOR NON BASED D.

'REFER' EXTENT SPECIFIED IN DECLARATION OF D BUT NOT IN 'BASED' STRUCTURE. N ASSUMED FOR EXTENT.

Example:

1. DCL 1 A, 2 B, 2 C(X REFER(B):8);

2. DCL 1 A, 2 B, 2 C(4:X REFER (B));

Explanation: The REFER option can only be used in the declaration of a based structure that contains an adjustable array dimension. If REFER is used in this way for the lower bound, 1 is assumed; if it is used for the upper bound, 10 is assumed.

IEL0456I S AMBIGUOUS 'REFER' ITEM T FOR D.

'REFER' ITEM T FOR EXTENT IN DECLARATION OF D IS AMBIGUOUS. UNDEFINED SELECTION MADE.

Example:

```
     DCL 1 A BASED,
           2 B, 2 C, 3 B,
           3 D (X REFER B:10);
```

The reference B is ambiguous.

IEL0457 W 'REFER' T FOR D MAY BE INVALID.

IF THE STRUCTURE CONTAINS PADDING USE OF 'REFER' T FOR EXTENT OF D WILL BE INVALID AND RESULTS OF EXECUTION UNDEFINED.

Example:

```
     DCL 1 A BASED,
          2 B FIXED BIN,
          2 C,
            3 D FIXED DEC,
            3 E (X REFER(D)) FLOAT DEC,
            3 F FIXED DEC;
```

Explanation: Although structure A contains padding, this message is pictured for structures that, when mapped, do not contain padding.

IEL0458I S 'REFER' T FOR D NOT PREVIOUS BASE ELEMENT.

'REFER' ITEM T FOR EXTENT IN DECLARATION OF D IS NOT A PREVIOUS SCALAR BASE ELEMENT IN THE SAME STRUCTURE. N ASSUMED FOR EXTENT.

Example:

1. `DCL 1 A BASED, 2 B(X REFER(C):10),2 C;`

   -base element C follows the REFER item.

2. `DCL 1 A BASED, 2 B, 3 C(X REFER (B):10);`

   -B is not a base element in structure A.

3.
   ```
   DCL 1 A, 2 B, 3 C;
   DCL 1 D BASED, 2 E(X REFER(C):10);
   ```

   -C is not a base element in structure D.

IEL0459I I D TREATED AS NOT 'CONNECTED'.

ARRAY PARAMETER D TREATED AS NOT 'CONNECTED'. OPTIMIZATION MAY BE INHIBITED.

Example:

```
     P: PROC(X,Y);
        DCL (X,Y CONNECTED, Z) (10,10);

     1.  Z = X; (compiled as a do-group)
         Z = Y; (compiled as a single move instruction)

     2.  V = X(6,3); (compiled as subscript calculation
                          to obtain offset)
         V = Y(6,3); (offset is calculated at
                          compile-time, no further
                          calculation required)
```

Explanation: If the attribute CONNECTED is added to the declaration of the array, the subscript calculations will be optimized as shown.

IEL0460I E DEFAULT 'BUILTIN' OR 'GENERIC' FOR D IGNORED.

DEFAULT ATTRIBUTE 'BUILTIN' OR 'GENERIC' SPECIFIED FOR D CONFLICTS WITH USE OF IDENTIFIER IN IMPLICIT DECLARATION. ATTRIBUTE IGNORED.

Example:

```
     DEFAULT RANGE (P) BUILTIN CHAR STATIC;
     DECLARE A CHAR DEFINED (P);
```

IEL0461I S AGGREGATES INVALID IN GENERIC DESCRIPTOR LIST FOR T.

COMPILER RESTRICTION. AGGREGATES INVALID IN DESCRIPTOR LIST FOR MEMBER T IN 'GENERIC' SPECIFICATION. MEMBER IGNORED.

Example:

```
     DCL G GENERIC (E1 WHEN(,(*)),
         E2 WHEN (FIXED,FLOAT),
         E3 WHEN(,1,2,2);

     (E2 is a valid member, E1 and E3 are ignored)
```

IEL0462I S INITIALIZATION INVALID FOR STATIC LABEL D.

INITIALIZATION INVALID FOR 'STATIC' ENTRY VARIABLE D. INITIALIZATION IGNORED.

Example:

```
          DCL EV ENTRY VARIABLE STATIC INIT(EV1);
```

IEL0463I S ENTRY NAME T INVALID IN 'GENERIC' SPECIFICATION.

COMPILER RESTRICTION. 'BASED' 'DEFINED' OR SUBSCRIPTED ENTRY NAME T INVALID IN 'GENERIC' SPECIFICATION. ENTRY NAME IGNORED.

Example:

```
          DCL E1 ENTRY, E2 BASED;
```

```
DCL G GENERIC (E1 WHEN (FIXED) E2 WHEN (FLOAT));

(E2 ignored)
```

IEL0464I S D IS NOT 'BASED'.

D IN 'LOCATE' STATEMENT NOT 'BASED'. STATEMENT IGNORED.

Example:
```
P:  PROC;
    LOCATE FRED FILE(F);
    END P;

    FRED (is not declared based)
```

IEL0465I S D IS NOT LEVEL ONE.

D IN 'ALLOCATE' STATEMENT NOT LEVEL ONE. THIS AND ANY FOLLOWING ITEMS IGNORED.

Example:

```
DCL (A,1 X) CTL, 2(Y,Z)
ALLOCATE A,Y,X;

(Items Y and X are ignored)
```

Explanation: A minor structure cannot be allocated independently of its containing level 1 structure.

IEL0466I S D IS NOT 'BASED' OR 'CONTROLLED'.

D IN 'ALLOCATE' STATEMENT NOT 'BASED' OR 'CONTROLLED'. THIS AND ANY FOLLOWING ITEMS IGNORED.

Example:

```
DCL X, (Y,Z) CTL;
ALLOCATE Y,X,Z;

(X and Z will not be allocated)
```

Explanation: Only based or controlled variables can be allocated storage by means of the ALLOCATE statement.

IEL0467I E FINAL MEMBERS MISSING FROM STRUCTURE.

FINAL MEMBERS MISSING FROM STRUCTURE SPECIFICATION IN 'ALLOCATE' STATEMENT. DECLARATION USED FOR MISSING MEMBERS.

Example:

```
DCL 1 X CTL, 2 (Y,Z) CHAR(3);
ALLOCATE 1 X, 2 Y CHAR(4);
```

Explanation: The member 2 Z is assumed to be included in the ALLOCATE statement with the declared attributes CHAR(3).

IEL0468I E LEVEL NUMBER PRECEDING D IGNORED.

LEVEL NUMBER PRECEDING D IGNORED.

Example:

```
DCL X CTL
ALLOCATE 1 X;

(the level '1' is ignored)
```

Explanation: A level number is only required in an ALLOCATE statement for a structure where members of that structure are specified explicitly in the statement.

IEL0469I E DIMENSIONS ATTRIBUTE MISSING FOR D.

DIMENSIONS ATTRIBUTE MISSING FOR STRUCTURE MEMBER D IN 'ALLOCATE' STATEMENT. DECLARED DIMENSIONS ASSUMED.

Example:

```
DCL 1 X CTL, 2 Y(10), 2 Z;
ALLOCATE 1 X,2 Y, 2 Z;
```

Explanation: Except for level-1 identifiers, those identifiers declared with dimensions must, when given in an ALLOCATE statement, be specified with dimensions.

IEL0470I S WRONG NUMBER OF DIMENSIONS FOR D.

WRONG NUMBER OF DIMENSIONS FOR D IN 'ALLOCATE' STATEMENT. RESULTS OF EXECUTION UNDEFINED.

Example:

```
DCL X(10) CTL;
ALLOCATE X(5,2);
```

Explanation: An identifier declared with dimensions must, when given in an ALLCCATE statement, be specified with the same number of dimensions, although the bounds of a particular dimension may differ from those given in the declaration.

IEL0471I S CONFLICTING ATTRIBUTE T FOR D IGNORED.

CONFLICTING ATTRIBUTE T FOR D IN 'ALLOCATE' STATEMENT. ATTRIBUTE IGNORED.

Example:

```
DCL X CHAR(6) CTL;
ALLOCATE X BIT(6);
```

Explanation: The attribute of an identifier given in an ALLOCATE statement should not conflict with the attribute given in the declaration of the identifier. Note that string lengths and the upper and lower bounds of dimensions can differ between the declaration and the ALLOCATE statement.

IEL0472I E INVALID ATTRIBUTE T FOR D IGNORED.

INVALID ATTRIBUTE T FOR D IN 'ALLOCATE' STATEMENT. ATTRIBUTE IGNORED.

Example:

```
     ALLOCATE C ALIGNED;
```

Explanation: Only the following attributes can be used in an ALLOCATE statement: BIT, CHARACTER, AREA, and INITIAL.

IEL0473I E LEVEL NUMBER FOR T REPLACED BY ONE.

INVALID LEVEL NUMBER SPECIFIED FOR T. LEVEL ONE ASSUMED.

Example:

```
     ALLOCATE 2 X;
```

Explanation: The first identifier in an ALLOCATE statement must be a level 1 identifier.

IEL0474I E STRUCTURING ERROR FOLLOWING D,

ERROR IN SPECIFICATION OF STRUCTURING FOLLOWING D. DECLARED STRUCTURING ASSUMED FOR FINAL MEMBERS OF STRUCTURE.

Example:

```
     DCL (1 X, 2 Y, 2 Z, A) CTL;
     ALLOCATE 1 X, 2 Y, A;
```

(structure member Z is assumed to be included in the ALLOCATE statement)

Explanation: If any members of a structure appear in an ALLOCATE statement, all the members of that structure must appear.

IEL0475I E ATTRIBUTES FOR 'BASED' VARIABLE D IGNORED.

ATTRIBUTES FOR BASED VARIABLE D INVALID ON 'ALLOCATE' STATEMENT. ATTRIBUTES IGNORED.

Example:

```
     DCL X BASED (P);
     ALLOCATE X INIT(3);
     INIT(3) ignored
```

Explanation: Based variables cannot be given attributes when allocated.

IEL0476I E 'SET' OR 'IN' INVALID FOR 'CONTROLLED' D.

'SET' OR 'IN' OPTION INVALID IN 'ALLOCATE' STATEMENT FOR 'CONTROLLED' VARIABLE D. OPTION IGNORED.

Example:

```
     DCL X CTL, Y BASED;
     ALLOCATE X IN (A); (invalid)
     ALLOCATE Y IN (A); (valid)
```

Explanation: The object of the SET or IN options must be a based variable.

IEL0477I E 'CHAR' 'BIT' OR 'AREA' WITHOUT EXTENT.

'CHARACTER' OR 'BIT' OR 'AREA' SPECIFIED WITHOUT EXTENT IN 'ALLOCATE' STATEMENT. ATTRIBUTE IGNORED.

Example:

```
     DCL X CHAR(3) CTL;
     ALLOCATE X CHAR;
     'ALLOCATE X;' (assumed)
```

|IEL0478I W D HAS STRING OVERLAY DEFINING.

D HAS STRING OVERLAY DEFINING AND IS INCOMPATIBLE WITH THE PL/I F COMPILER.

Example:

```
          DCL X(10) PICTURE '9999',
              A(10) PICTURE '9' DEFINED X;
```

Explanation: In the above example the F compiler would have given correspondence defining but the optimizing compiler will give string overlay defining.

IEL0479I S STRING OR AREA SIZE REDUCED TO N.

COMPILER RESTRICTION. CHAR OR BIT OR AREA SIZE REDUCED TO COMPILER MAXIMUM.

Example:

```
     DCL A AREA(20000000),
         B BIT (40000),
         C CHAR(40000);
```

Explanation: The maximum sizes permitted by this compiler are 16777216 for an area, and 32767 for character and bit strings. Even so, these sizes may exceed the available main storage when the program is executed.

IEL0480I S D DEFINED ON 'DEFINED' OR 'BASED'.

D IS DECLARED AS 'DEFINED' ON A BASE WHICH ALSO HAS THE 'DEFINED' OR 'BASED' ATTRIBUTE. 'DEFINED' ATTRIBUTE IGNORED.

Example:

```
DCL A DEFINED B, B DEFINED C;
```

IEL0481I S D 'ISUB' 'DEFINED' ON CROSS-SECTION.

D IS DECLARED AS 'DEFINED' WITH AN 'ISUB' VARIABLE ON THE CROSS-SECTION OF A BASE. 'DEFINED' ATTRIBUTE IGNORED.

Example:

```
DCL D(G) DEFINED B (*,1SUB);
```

IEL0482I S D 'DEFINED' WITH WRONG NUMBER OF SUBSCRIPTS.

D IS DECLARED AS 'DEFINED' WITH AN 'ISUB' VARIABLE ON A BASE WITH THE WRONG NUMBER OF SUBSCRIPTS. 'DEFINED' ATTRIBUTE IGNORED.

Example:

```
DCL B (10,10),
    D(6) DEFINED B(1SUB);
```

IEL0483I S D 'DEFINED' WITH 'ISUB' AND 'POSITION' ATTRIBUTE.

D IS DECLARED AS 'DEFINED' WITH AN 'ISUB' VARIABLE AND HAS 'POSITION' ATTRIBUTE. 'DEFINED' ATTRIBUTE IGNORED.

Example:

```
DCL B (10,10),
D (6) DEFINED B(1SUB, 6) POS(3);
```

IEL0484I S MAPPING OF DEFINED ITEM D CONFLICTS WITH BASE.

MAPPING OF ELEMENT D OF ISUB-DEFINED ARRAY CONFLICTS WITH THAT OF BASE. 'DEFINED' ATTRIBUTE IGNORED.

Example:

```
DCL 1 B(10), 2 C, 3 D;
DCL 1 X(5,2) DEFINED B(1SUB+2SUB),
       2 Y,
       2 Z;
```

IEL0485I E CONFLICT BETWEEN DEFINED ITEM D AND BASE ATTRIBUTES IGNORED.

ATTRIBUTES OF ITEM D 'DEFINED' WITH AN 'ISUB' VARIABLE CONFLICT WITH THOSE OF BASE. CONFLICT IGNORED.

Example:

```
DCL 1 B (10), 2 C FIXED(31,0),
         2 D FLOAT,
    1 D(5,2) DEFINED B(1SUB+2SUB),
         2 E POINTER,
         2 F OFFSET;
```

IEL0486I E SIMPLE DEFINING ASSUMED AS ATTRIBUTES OF D CONFLICT WITH BASE.

ATTRIBUTES OF 'DEFINED' ITEM D CONFLICT WITH THOSE OF BASE. SIMPLE DEFINING ASSUMED.

Example:

```
DCL B POINTER,
A FIXED BINARY(31,0) DEFINED B;
```

Explanation: Simple defining is assumed only if the two items have matching size, alignment, and dimensionality. String lengths or bounds are ignored.

IEL0487I S D 'DEFINED' ON UNCONNECTED AGGREGATE.

D IS STRING OVERLAY 'DEFINED' ON AN AGGREGATE WHICH IS NOT 'CONNECTED'. 'DEFINED' ATTRIBUTE IGNORED.

Example:

```
DCL 1 B(10),
           2 C CHAR(2),
           2 F,
           A CHAR(20) DEFINED C;
```

Explanation: An aggregate used as the base in string overlay defining must occupy a contiguous area of storage.

IEL0488I S ATTRIBUTES OF 'DEFINED' ITEM D CONFLICT WITH BASE.

ATTRIBUTES OF 'DEFINED' ITEM D CONFLICT WITH THOSE OF BASE. 'DEFINED' ATTRIBUTE IGNORED.

Example:

```
DCL O OFFSET DEFINED B,
    1 B, 2 (C,D) CHAR;
```

Explanation: The mapping of the defined and base items differ and the defined item is a level 1 offset.

IEL0489I S 'POSITION' VALUE FOR D LESS THAN ONE OR EXCEEDS N.

COMPILER RESTRICTION. D IS DECLARED WITH 'POSITION' VALUE

LESS THAN ONE OR GREATER THAN N. 'POSITION' ATTRIBUTE IGNORED.

Example:

```
     DCL B CHAR,
        A CHAR DEFINED B POSITION(32768);
```

IEL0490I E INVALID 'DEFINED' FOR D.

INVALID USE OF 'DEFINED' IN DECLARATION OF D. COMPILER WILL ATTEMPT TO ASSUME STRING OVERLAY DEFINING.

Example:

```
          DCL B CHAR(5),
             D BIT(80) DEF(B);
```

Explanation: If the defined and base items do not match, both must be non-varying, unaligned and either picture or character or both bit strings. If these rules are infringed, the defining will be accepted provided that the base item occupies contiguous storage.

IEL0491I S 'DEFINED' BASE FOR D IS AMBIGUOUS.

BASE REFERENCE OF 'DEFINED' ATTRIBUTE IN DECLARATION OF D IS AMBIGUOUS. UNDEFINED SELECTION MADE.

Example:

```
     DCL 1 A, 2 B, 3 B,
           D DEFINED B;
```

(the identifier B is ambiguous)

IEL0492I S 'DEFINED' BASE FOR D IS NOT A VARIABLE.

D IS 'DEFINED' ON A BASE WHICH IS NOT A VARIABLE. 'DEFINED' ATTRIBUTE IGNORED.

Example:

```
     P: PROC;
         DCL X DEF P;
         END P;
```

IEL0493I W SIMPLE DEFINING APPLIES FOR D.

SIMPLE DEFINING APPLIES FOR D. IF OVERLAY DEFINING REQUIRED THEN ADD T TO DECLARATION.

Explanation: The purpose of this message is to indicate a difference between this implementation and that of the PL/I D and F compilers which may result in the different mapping for the structure.

Programmer Response: The above action should be carried out if the program was originally written for the D or F compilers or if the program is to exchange records to and from D or F programs, when the records are derived from such structures and therefore require identical mapping.

Example:

```
     DCL 1 A,
             2 B(10) CHAR(3),
             2 C(10) CHAR(2),
         1 D DEF A,
             2 E(5) CHAR(3),
             2 F(5) CHAR(2);
```

(simple defining will be used for structure D)

IEL0494I E   STRING OVERLAY DEFINING ASSUMED FOR D.

STRING LENGTH IN DEFINED ITEM D IS TOO LONG FOR SIMPLE DEFINING. STRING OVERLAY DEFINING ASSUMED.

Example:

```
     DCL 1 A,
             2 B CHAR(1),
             2 C CHAR(79),
         1 D DEF A,
             2 E CHAR(40),
             2 F CHAR(40);
```

Explanation: Simple defining cannot be used where the length of the defined string is greater than the length of the base string. In the above example, string D.E is longer than its corresponding base string A.B.

IEL0495I E   MAXIMUM LENGTHS OF DEFINED ITEM D AND BASE DIFFER.

AREA SIZE OR MAXIMUM LENGTH OF VARYING STRING IN SIMPLE DEFINED ITEM D DIFFERS FROM THAT OF THE CORRESPONDING BASE. RESULTS OF EXECUTION UNDEFINED.

Example:

```
     DCL 1 A, 2 B CHAR(3) VAR,
              2 C CHAR(4) VAR,
         1 D DEF A,
              2 E CHAR(2) VAR,
              2 F CHAR(3) VAR;
```

Explanation: If a defined item, for which simple defining is used, is a varying string that is shorter than the corresponding base string which is also varying, an error may occur during execution. A reference to the defined varying string may result in a string that is longer than its declared maximum length.

IEL0496I S   T INVALID IN 'CALL' STATEMENT.

BUILTIN FUNCTION T INVALID IN 'CALL' STATEMENT. STATEMENT IGNORED.

Example:

```
P: PROC;
   CALL SIN(X);
   END;
```

IEL0497I S D INVALID IN 'FETCH' OR 'RELEASE'.

D IN 'FETCH' OR 'RELEASE' STATEMENT IS INVALID. STATEMENT IGNORED.

Example:

```
P:  PROC;
    DCL (E1,E2) ENTRY OPTIONS(COBOL),
        E3 ENTRY OPTIONS(FORTRAN),
        E4 ENTRY;
    FETCH P;  (invalid)
    FETCH E1; (invalid)
    FETCH E2; (invalid)
    FETCH E3; (invalid)
    FETCH E4; (valid)
```

Explanation: The identifier in a FETCH statement must be the name of an external PL/I procedure, and must be declared as such in the procedure containing the FETCH statement. Non-PL/I routines or internal PL/I procedures cannot be obtained by a FETCH statement.

IEL0498I E INVALID SUBSCRIPTED PREFIX T.

SUBSCRIPTED STATEMENT PREFIX T IS NOT A NON-STATIC LABEL ARRAY. PREFIX IGNORED.

Example:

```
DCL LS(2) LABEL STATIC;
    LS(1):; (ignored)
DCL LA(3) LABEL AUTOMATIC;
    LA(2):; (accepted)
DCL L LABEL;
    L(3):; (ignored)
```

IEL0499I S D INITIALIZED BY PREFIX AND DECLARATION.

LABEL VARIABLE D IS INITIALIZED BY STATEMENT PREFIX AND BY DECLARATION. DECLARED 'INITIAL' IGNORED.

Example:

```
DCL LV(3) LABEL INIT(L1,L2,L3);
   .
   .
   LV(1):L1:  X=Y/Z;
```

IEL0500I S CONFLICT IN USE OF D AS T.

CONFLICT BETWEEN USE OF D AS T AND ITS DECLARED ATTRIBUTES. STATEMENT IGNORED.

Example:

```
     DCL P EVENT;
        .
        .
        .
     CALL P;
```

Explanation: This message is produced when an identifier has an explicit declaration that conflicts with its use when the use would constitute a contextual declaration in the absence of the explicit declaration.

IEL0501I E D HAS INVALID ATTRIBUTES. OPTION IGNORED.

ATTRIBUTES FOR D INVALID IN 'ENVIRONMENT' OPTION. OPTION IGNORED.

Example:

```
     DCL F FILE ENV(RECSIZE(X) PASSWORD(Y));
     DCL (X,Y) FLOAT;
```

Explanation: The attributes for arguments in the ENVIRONMENT option are restricted. In the example, the arguments, X and Y, should be declared as follows:

```
   DCL X FIXED BIN(31,0) STATIC;
   DCL Y CHAR STATIC;
```

IEL0502I S USE OF D CONFLICTS WITH PREVIOUS DECLARATION.

USE OF D AS A STATEMENT LABEL PREFIX IS A CONFLICTING OR MULTIPLE DECLARATION. PREFIX IGNORED.

Example:

```
     L1: X=1;
         .
         .
         .
     L1: A=B;
```

IEL0503I E T ASSUMED TO BE EXTERNAL ENTRY.

IDENTIFIER T IS NOT DECLARED. EXTERNAL ENTRY ASSUMED.

Example:

```
     1.  P1:  PROC;
              CALL FRED;
              END;

     2.  P2:  PROC;
              BERT = FRED(6);
              END;
```

Explanation: In the first example above, FRED is contextually declared BUILTIN. It is not however a recognized built-in function. In the second example, FRED is contextually declared BUILTIN in the absence of an explicit or default declaration as an array.

IEL0504I S T ASSUMED TO BE AN ARRAY.

IDENTIFIER T IN 'BUILTIN' CONTEXT IS INVALID. ASSUMED TO BE AN ARRAY.

Example:

```
     P:  PROC;
         DEFAULT RANGE (F) (10);
         BERT = FRED(6);
         END;
```

Explanation: When a contextual declaration for an identifier as BUILTIN conflicts with a default declaration for the same identifier as an array, the contextual declaration is superseded by the default declaration.

IEL0505I S CONFLICT BETWEEN ATTRIBUTES OF D AND USE AS T.

CONFLICT BETWEEN DECLARED ATTRIBUTES OF D AND ITS USE AS T IN BOUNDS SPECIFICATION. BOUNDS OF N TO 10 ASSUMED.

Example:

```
     DCL P, (P is float dec)
         X BASED,
         A (P-> X);
```

IEL0506I E CONFLICT BETWEEN ATTRIBUTES OF D AND USE AS T.

CONFLICT BETWEEN DECLARED ATTRIBUTES OF D AND ITS USE AS T IN LOCATOR QUALIFICATION. QUALIFICATION IGNORED.

Example:

```
     DCL P FLOAT,
         A BASED (P);
```

IEL0507I S CONFLICT BETWEEN ATTRIBUTES OF D AND USE AS T.

CONFLICT BETWEEN DECLARED ATTRIBUTES OF D AND ITS USE AS T IN ADJUSTABLE STRING OR AREA SPECIFICATION. DEFAULT EXTENT ASSUMED.

Example:

```
     DCL P DECIMAL,
         X BASED,
         A AREA (P-> X),
         B BIT (P-> X),
         C CHAR (P->X);
```

Explanation: The attributes assumed by default are AREA(1000), BIT (1), and CHAR(1).

IEL0508I S CONFLICT BETWEEN ATTRIBUTES OF D AND USE AS T.

CONFLICT BETWEEN DECLARED ATTRIBUTES OF D AND ITS USE AS T IN 'DEFINED' 'POSITION' OR 'INITIAL' ATTRIBUTE. ATTRIBUTE IGNORED.

Example:

```
DCL P DECIMAL,
    Q(10) DECIMAL,
    X BASED,
    A DEFINED (Q(P->X));
```

Explanation: Invalid INITIAL and POSITION attributes are ignored. The storage class AUTOMATIC is assumed for an invalid DEFINED attribute.

IEL0509I E   CONFLICT BETWEEN ATTRIBUTES OF D AND USE AS T.

CONFLICT BETWEEN DECLARED ATTRIBUTES OF D AND ITS USE AS T. CONTEXTUAL ATTRIBUTES ASSUMED.

Example:

```
P: PROC (F);
   READ FILE (F) INTO (A);

   (parameter F requires further explicit
   declaration as a file)
```

Explanation: Where the explicit declaration of an identifier conflicts with the use of the identifier, the contextual attributes derived from the usage are assumed.

IEL0510I E   CONFLICT BETWEEN ATTRIBUTES OF D AND USE AS T.

CONFLICT BETWEEN DECLARED ATTRIBUTES OF D AND ITS USE AS T IN 'SET' OR 'IN' OPTION. OPTION IGNORED.

Example:

```
DCL X BASED,
   (A,P) DECIMAL;
ALLOCATE X IN (A) SET (P);

('ALLOCATE X;' assumed)
```

IEL0511I S   D INVALID IN TARGET POSITION.

D IS NOT A VARIABLE AND IS IN A TARGET POSITION. STATEMENT IGNORED.

Example:

```
P:  PROC;
    P = 1;
```

Explanation: A target position can be one of the following:

1. The left-hand side of an assignment statement
2. A do-loop control variable
3. Data list in a GET statement
4. INTO option in a READ statement
5. SET option

6. KEYTO option in a READ statement
7. REPLY option

IEL0512I S T IS NOT DECLARED.

QUALIFIED NAME BEGINNING T IS NOT DECLARED. STATEMENT IGNORED.

Example:

```
     P:  PROC;
         A.B = 1;
         END;
```

Explanation: Structures must be explicitly declared.

IEL0513I S INVALID USE OF D AS 'BUILTIN'.

D IS DECLARED BUILTIN BUT IS EITHER NOT A BUILTIN FUNCTION NAME OR IS INVALIDLY USED WITHOUT ARGUMENTS. STATEMENT IGNORED.

Example:

```
     DCL E ENTRY VARIABLE,
         XYZ BUILTIN,
         SIN BUILTIN;
         .
         .
         .
         E = XYZ;
         .
         .
         .
         E = SIN;
```

Explanation: The identifier XYZ is not a built-in function. The built-in function SIN is used without an argument.

IEL0514I S D NOT LABEL KNOWN IN CURRENT BLOCK.

IDENTIFIER D AFTER 'GOTO' IS NOT A LABEL KNOWN IN THE CURRENT BLOCK. STATEMENT IGNORED.

Example:

```
     P:  PROC;
          BEGIN;
           L: X=1;
          END;
         GO TO P; (P is not known at this point)
         GOTO L; (L is not known at this point)
         END;
```

IEL0515I S INVALID USE OF D AS PSEUDO-VARIABLE.

INVALID USE OF D AS PSEUDO-VARIABLE. STATEMENT IGNORED.

Example:

```
1    DCL C CHAR(16);
     SUBSTR(SUBSTR(C,3),3)='1';

2.   DCL ONCHAR BUILTIN;
     READ FILE (X) INTO (ONCHAR);
```

IEL0516I S   D INVALID IN 'FROM' OPTION.

INVALID ITEM D IN 'FROM' OPTION. STATEMENT IGNORED.

Example:

```
READ FILE (FRED) FROM (FRED);
```

IEL0517I S   D INVALID AS 'DO' CONTROL VARIABLE.

INVALID USE OF D AS CONTROL VARIABLE IN ITERATIVE SPECIFICATION. NON-ITERATIVE 'DO' ASSUMED.

Example:

```
DO SIN(X) = 1 TO 10;
```

IEL0518I W   T IS NOT IMPLICITLY 'BUILTIN'.

T IS THE NAME OF A BUILTIN FUNCTION BUT ITS IMPLICIT DECLARATION DOES NOT IMPLY 'BUILTIN'.

Example:

```
P: PROC;
 X=DATE;
  END;
```

Explanation: A built-in function that does not require an argument must be declared BUILTIN. The declaration can be explicit, contextual, or implicit. (A contextual declaration is obtained by including a non-executing CALL statement for the built-in function name, and an implicit declaration is obtained by using the built-in function name with a null argument list.)

IEL0519I S   IDENTIFIER BEGINNING T IS AMBIGUOUS.

IDENTIFIER BEGINNING T IS AN AMBIGUOUS REFERENCE TO A STRUCTURE MEMBER. UNDEFINED SELECTION MADE.

Example:

```
DCL 1 A, 2 B, 3 C, 2 D, 3 C;
.
.
.
A.C = 1;
```

Explanation: If a name is an incomplete qualification of more than one identifier, but does not completely qualify any identifier, it is in error.

IEL0520I S TOO MANY SUBSCRIPTS FOR D.

'ENTRY' VARIABLE D HAS TOO MANY SUBSCRIPTS. STATEMENT IGNORED.

Example:

```
DCL 1 A(10), 2 B(3), 3 C ENTRY(FIXED,FLOAT);
.
.
.
X= B(9,2). C(5)(P);
```

Explanation: Subscripts in a qualified entry name must agree in number with the subscripts given in the declaration of the containing aggregate so that the argument list can be correctly distinguished.

IEL0521I S WRONG NUMBER OF ARGUMENTS FOR ENTRY.

WRONG NUMBER OF ARGUMENTS SPECIFIED IN REFERENCE TO ENTRY NAME. STATEMENT IGNORED.

Example:

```
P:  PROC(X);
.
.
.
CALL P(Y,Z);
```

IEL0522I S INVALID 'GOTO' INTO ITERATIVE 'DO' GROUP.

'GOTO' STATEMENT SPECIFIES INVALID BRANCH INTO AN ITERATIVE 'DO' GROUP. STATEMENT IGNORED.

Example:

```
P: PROC
   DO I=1 TO 10;
   L: A=A+1;
      END;
   GOTO L;
END P;
```

IEL0523I S INVALID 'GOTO' TO 'FORMAT' STATEMENT.

'GOTO' STATEMENT SPECIFIES INVALID BRANCH TO A FORMAT STATEMENT. STATEMENT IGNORED.

Example:

```
R: FORMAT (SKIP,COLUMN(2),A);
   GOTO R;
```

IEL0524I S AREA EXPRESSION SPECIFIED FOR RETURNED OFFSET.

COMPILER RESTRICTION. AREA SPECIFIED FOR OFFSET IN 'RETURNS' SPECIFICATION IS NOT A SIMPLE AREA NAME. AREA EXPRESSION IGNORED.

Example:
```
          X:  ENTRY RETURNS(OFFSET(P->A));
                CALL X;
```

Explanation: An area expression in a RETURNS option must be a single identifier that is an area name.

IEL0525I S INVALID 'INITIAL' ATTRIBUTE IGNORED.

INVALID INITIAL SPECIFICATION FOR SCALAR. 'INITIAL' ATTRIBUTE IGNORED.

Example:
```
         DCL A INIT((10)0);
```

IEL0526I S PSEUDO-VARIABLE INVALID AS CONTROL VARIABLE.

SPECIFIED PSEUDO-VARIABLE NOT ALLOWED AS CONTROL VARIABLE IN ITERATIVE SPECIFICATION. NON-ITERATIVE 'DO' ASSUMED.

Example:
```
          DO COMPLEX(A,B) = M TO N;
```

IEL0527I U STATEMENT TOO LARGE. COMPILATION TERMINATED IN PHASE P.

COMPILER RESTRICTION. STATEMENT TOO LARGE. COMPILATION TERMINATED IN PHASE P.

Explanation: The amount of main storage available for the compiler determines the maximum length of a source statement. If the storage exceeds 55K bytes, the maximum possible statement length can be used. This message may be produced also by a statement containing many non-static arrays with the INITIAL attribute, particularly if these arrays are controlled or are arrays of structures.

Programmer Response: Either increase the amount of main storage for the compiler (SIZE option) to exceed 55K bytes, or divide the statement into smaller statements. If neither of the above apply, check that the statement does not contain an unmatched quote character or comment delimiter. If due to array initialization, attempt to separate some of the initialization code by means of dummy BEGIN blocks or by using separate ALLOCATE statements. If this fails initialize the arrays by assignment.
If the TOTAL option is in use and the program contains many record I/O statements close together, break up the sequence of these statements by inserting BEGIN... END around half of them.

IEL0528I S D INVALID AS REMOTE FORMAT ITEM.

D NOT VALID AS REMOTE FORMAT ITEM. STATEMENT IGNORED.

Example:

```
          DCL L(10) LABEL, X;
          PUT FILE(F) EDIT(X) (R(L(1))); (valid)
          PUT FILE(F) EDIT(X) (R(L1)); (valid)
          PUT FILE(F) EDIT(X) (R(X)); (invalid)
      L1: FORMAT (F(5,2));
```

Explanation: This message is produced if the remote format item is neither a label on a FORMAT statement, nor a label variable, nor a function reference that returns a label.

IEL0529I S   D IS NOT 'BASED' OR 'CONTROLLED'.

D IN 'FREE' STATEMENT NOT 'BASED' OR 'CONTROLLED'. STATEMENT IGNORED.

Example:

```
      P:  PROC;
          DCL A;
          FREE A;
          END;
```

IEL0530I S   INVALID USE OF 'STRING' PSEUDO-VARIABLE.

COMPILER RESTRICTION. INVALID USE OF 'STRING' PSEUDO-VARIABLE. STATEMENT IGNORED.

Example:

```
      GET STRING(STRING(A)); (invalid)
      PUT STRING(STRING(A)); (invalid)
      DISPLAY(B) REPLY(STRING(A)); (invalid)
      READ FILE(F) INTO(X) KEYTO(STRING(A)); (invalid)
      STRING(A) = C; (valid)
```

Explanation: The STRING pseudovariable can only be used in an assignment statement.

IEL0531I S   STRING LENGTH EXCEEDS N.

COMPILER RESTRICTION. STRING LENGTH EXCEEDS N. REPETITION FACTOR OF ONE ASSUMED.

Example:

```
      P: PROC;
        A= (32768)'A';  /* BECOMES 'A' */
        A= (16384)'AA'; /* BECOMES 'AA' */
        END;
```

Explanation: An attempt has been made to produce a string with a length exceeding 32767 using a repetition factor. A repetition factor of one is assumed.

IEL0540I W   EXTENDED FLOAT ARITHMETIC WILL BE USED.

EXTENDED FLOAT ARITHMETIC WILL BE USED IN THIS PROGRAM BECAUSE IT CONTAINS ITEMS WITH EXTENDED PRECISION.

Explanation: The message is given as a warning that expressions may be evaluated using extended precision even though they do not contain variables declared with extended precision. The same expressions would be evaluated using long float precision if no variables in the source program were declared using extended precision. Although the use of long float may mean loss of precision, it avoids the performance degradation of using extended float.

IEL0541I I   'ORDER' MAY INHIBIT OPTIMIZATION.

'ORDER' OPTION APPLIES TO THIS BLOCK. OPTIMIZATION MAY BE INHIBITED.

Example:

```
P: PROC;
   A: PROC REORDER;
      B: PROC;
         END;
      END;
         C: PROC;
            D: PROC ORDER;
               E: PROC;
                  END;
               END;
            END;
   END;
```

Explanation: The message is produced for procedures P, C, D, and E. Procedure P has the ORDER option by default; procedure C inherits the ORDER option from procedure P; procedure D has the ORDER option declared explicitly; and procedure E inherits the ORDER option from procedure D. Procedure A has the REORDER option declared explicitly and procedure B inherits the REORDER option from procedure A. This message is produced only when the OPT(TIME) option is specified for the compilation of blocks to which the ORDER option applies.

IEL0542I S   AREA SPECIFIED FOR OFFSET IN ENTRY DECLARATION.

COMPILER RESTRICTION. AREA SPECIFIED FOR OFFSET IN 'ENTRY' DECLARATION IS IGNORED.

Example:

```
DCL E ENTRY (OFFSET(A));
```

is assumed to be:

```
DCL E ENTRY (OFFSET);
```

IEL0543I S   STRUCTURE TERMINATED AFTER N MEMBERS.

COMPILER RESTRICTION. STRUCTURE TERMINATED AFTER N ITEMS.

Explanation: The structure has too many separately identifiable items. (Items include all minor structures and elements.)

IEL0544I W 'BUILTIN' SUBROUTINE WILL NOT BE USED FOR D.

D DECLARED AS EXTERNAL ENTRY REQUIRES PROVISION OF SUBROUTINE BY USER PROGRAM. 'BUILTIN' SUBROUTINE WILL NOT BE USED.

Example:

```
P: PROC;
   DCL PLIDUMP ENTRY;
   CALL PLIDUMP ('HB','P');
   END;
```

Explanation: Built-in subroutines such as PLIDUMP are contextually declared to be built-in by their appearance in a CALL statement.

IEL0545I W 'ASSEMBLER' OPTION INVALID.

USE OF 'ASSEMBLER' OPTION INVALID ON 'PROCEDURE' OR 'ENTRY' STATEMENT. OPTION IGNORED.

Example:

```
P: PROC OPTIONS(ASSEMBLER);
```

Explanation: The ASSEMBLER option is valid only in an ENTRY declaration.

IEL0547I W 'INTER' OPTION ASSUMED.

'ASSEMBLER' OPTION SPECIFIED WITHOUT 'INTER' OPTION. 'INTER' OPTION ASSUMED.

Example:

```
DCL E ENTRY OPTIONS(ASSEMBELR);
```

Explanation: The compiler does not disable PL/I interrupt handling when processing an ENTRY statement declared with the ASSEMBLER option.

IEL0548I W PARAMETER TO MAIN PROCEDURE NOT VARYING CHARACTER STRING.

PARAMETER TO PRIMARY ENTRY POINT OF MAIN PROCEDURE IS NOT VARYING CHARACTER STRING.

Example:

```
P: PROC(X) OPTIONS(MAIN);
      DCL X FLOAT;
```

Explanation: OS passes arguments in the form of PL/I varying character strings, which comprise a 2-byte length field followed by the string data.

IEL0560I W EXTERNAL ENTRY NAME BEGINS 'IHE'.

EXTERNAL ENTRY NAME BEGINS 'IHE'. POSSIBLE PL/I F COMPILER BUILTIN SUBROUTINE.

Example:

```
     CALL IHESRTA(A,B,C,D,E); /*SORT ROUTINE*/
```

Explanation: F compiler subroutines commence with the characters 'IHE', and therefore it is likely that the program has not been correctly converted for use with the optimizing compiler.

IEL0580I E INVALID INITIALIZATION FOR 'STATIC' LABEL D.

INITIALIZATION INVALID FOR 'STATIC' LABEL VARIABLE D. INITIALIZATION ACCEPTED.

Example:

```
     1.  DCL LV LABEL STATIC INIT(LAB);
         LAB: ;

     2.  DCL L(10) LABEL STATIC;
         L(1): ;
```

Explanation: The compiler allows the illegal language shown above, but for the program to execute successfully, the OPT(TIME) compiler option must be specified.

IEL0581I S INVALID BIT AGGREGATE DEFINING IGNORED.

COMPILER RESTRICTION. INVALID USE OF 'DEFINED' FOR BIT AGGREGATE D. 'DEFINED' ATTRIBUTE IGNORED.

Explanation: String overlay defining on a bit aggregate is not permitted by this compiler when either the defined item is subscripted or the expression in the POSITION attribute is not an integer constant.

Example:

```
     DCL 1 B1(10),
             2 B2 BIT(1),
             2 B3 BIT(1),
             2 B4 BIT(2);

     1.  DCL 1 D11(10) DEF B1,
               2 D2 BIT(2),
               2 D3 BIT(3);
         (this declaration is valid)

     2.  DCL 1 D12 DEF B1(2),
               2 D2 BIT(2),
               2 D3 BIT(2);
```

```
        (this declaration is invalid)

    3.  DCL 1 D13 (10) DEF B1 POS(X),
                2 D2 BIT(1),
                2 D3 BIT(1);
        (this declaration is also invalid)
```

IEL0601I S INVALIDLY DECLARED VARIABLE. STATEMENT IGNORED.

INVALID DECLARATION OF A VARIABLE USED IN THIS STATEMENT. STATEMENT IGNORED.

Example:

```
DCL X BASED (A.B);
.
.
.
X = 1;
```

Explanation: A variable which has been incorrectly declared and for which a message will have been issued has been used elsewhere. The message is issued because the compiler was unable to complete the declaration of the variable.

IEL0602I S [PROLOGUE CODE.] LOCATOR QUALIFICATION OF NON-BASED D.

[PROLOGUE CODE.] LOCATOR QUALIFICATION OF NON-BASED VARIABLE D. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL P POINTER, B FIXED;
A = P -> B;
```

Explanation: Locators (pointers and offsets) can only qualify based variables.

IEL0604I S [PROLOGUE CODE.] AGGREGATE D INVALID AS LOCATOR QUALIFIER.

[PROLOGUE CODE.] USE OF AGGREGATE D FOR LOCATOR QUALIFICATION IS INVALID. [STATEMENT IGNORED.][RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL P(10) POINTER;
    P -> X = Y;
```

Explanation: A locator qualifier must be an element and cannot be an unsubscripted or unqualified reference to an aggregate containing locators.

IEL0605I S [PROLOGUE CODE.] LEVEL OF LOCATOR QUALIFICATION EXCEEDS N.

COMPILER RESTRICTION. [PROLOGUE CODE.] LOCATOR QUALIFICATION IS RECURSIVE OR NUMBER OF LEVELS EXCEEDS N.

[STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
     1.  DCL P BASED(Q),
             Q BASED(P);
         P->Q->A=B

     2.  P1->P2->P3...->P99->A=B;
```

IEL0606I S [PROLOGUE CODE.] NO LOCATOR QUALIFICATION FOR BASED VARIABLE D.

[PROLOGUE CODE.] BASED VARIABLE D IS REFERENCED WITHOUT LOCATOR QUALIFICATION. [STATEMENT IGNORED.][RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
     DCL B BASED;
         A=B;
```

Explanation: A based variable declared without an implicit pointer qualifier must be referred to with an explicit pointer qualifier.

IEL0607I S [PROLOGUE CODE.] T INVALID AS LOCATOR QUALIFIER.

[PROLOGUE CODE.] INVALID USE OF BUILTIN FUNCTION T AS LOCATOR QUALIFIER. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
     DCL A BASED;
     DATAFIELD -> A=B;
```

Explanation: A built-in function cannot be used as a locator qualifier.

IEL0608I S [PROLOGUE CODE.] ENTRY D INVALID AS LOCATOR QUALIFIER.

[PROLOGUE CODE.] INVALID USE OF ENTRY D AS A LOCATOR QUALIFIER. [STATEMENT IGNORED.][RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
     DCL X ENTRY RETURNS (FLOAT);
     DCL A BASED;
         X -> A=B;
```

Explanation: An entry name cannot be used as a locator qualifier.

IEL0609I S [PROLOGUE CODE.] EXPRESSION INVALID AS ARGUMENT TO 'STRING'.

[PROLOGUE CODE.] INVALID USE OF EXPRESSION AS ARGUMENT TO 'STRING' BUILTIN FUNCTION. [STATEMENT IGNORED.][RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
     A=STRING(B+C);
```

Explanation: The argument to the STRING built-in function must be an expression representing string data.

IEL0610I S [PROLOGUE CODE.] INVALID ARGUMENT TO 'STRING'.

[PROLOGUE CODE.] ELEMENTS OF ARGUMENT TO 'STRING' BUILTIN FUNCTION MUST BE EITHER ALL CHARACTER OR ALL BIT. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
     DCL 1 S,
            2 B BIT(1),
            2 C CHAR(6);
              A = STRING(S);
```

Explanation: The argument to the STRING built-in function must consist of string data that is either all BIT or all CHARACTER.

IEL0611I S [PROLOGUE CODE.] NO ARGUMENTS PASSED TO T.

[PROLOGUE CODE.] NO ARGUMENTS PASSED TO BUILTIN FUNCTION OR PSEUDO-VARIABLE T. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
         X=SUBSTR;
```

IEL0612I S INVALID ARGUMENT TO T.

EXPRESSION OR CONSTANT INVALID AS ARGUMENT TO PSEUDO-VARIABLE T. STATEMENT IGNORED.

Example:

```
     SUBSTR (A+B,I,J) = C;
```

Explanation: The argument to the pseudovariable must be an element variable.

IEL0613I S DATA TYPE OF ARGUMENT D INVALID FOR T.

[PROLOGUE CODE.] DATA TYPE OF ARGUMENT D INVALID FOR BUILTIN FUNCTION T. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL E FIXED BINARY;
   I = STATUS (E);
```

(E should be an event variable)

IEL0614I S [PROLOGUE CODE.] NO 'AREA' SPECIFIED FOR OFFSET D.

[PROLOGUE CODE.] NO 'AREA' SPECIFIED OR DECLARED FOR OFFSET D. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL O OFFSET,
    X BASED;
O -> X = Y;
```

Explanation: The use of an offset variable does not constitute the contextual declaration of an area variable.

IEL0616I W VARIABLE IN 'INITIAL' FOR D MAY BE UNINITIALIZED.

INITIAL SPECIFICATION FOR VARIABLE D MAY CONTAIN AN UNINITIALIZED VARIABLE. RESULTS OF EXECUTION UNDEFINED.

Example:

```
DCL M, N INIT(M);
```

Explanation: This is a possible error detected in compiling the prologue routine to the program block which contains the erroneous initial specification. Consequently, the statement number given in this message is that of the PROCEDURE or BEGIN statement for the block.

Programmer Response: The program may contain a preceding declaration which uses the INITIAL CALL form of the INITIAL attribute to invoke a procedure that assigns a value to the identifier used in the subsequent INITIAL specification. If so, this message may be ignored. Otherwise, the program should be modified to ensure that the identifier will be initialized before it is itself used in the INITIAL attribute.

IEL0617I S T NOT LEVEL ONE.

D IN 'FREE' STATEMENT NOT LEVEL ONE. STATEMENT IGNORED.

Example:

```
DCL 1 A BASED,
    2 B, 2 C;
    .
    .
    .
FREE B;
```

Explanation: A FREE statement cannot be used to free storage occupied by a part of a based or controlled item.

IEL0618I S [PROLOGUE CODE.] 'DCL' OR 'DFT' STATEMENT CONTAINS INVALID EXPRESSION.

[PROLOGUE CODE.] 'DECLARE' OR 'DEFAULT' STATEMENT CONTAINS AN INVALID EXPRESSION. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
    DCL A DEF B(1);
    DCL B;
    A=0;
```

Explanation: In the above example, B is undimensioned.

IEL0619I S [PROLOGUE CODE.] CONSTANT ARGUMENT TO 'ADDR'.

[PROLOGUE CODE.] CONSTANT IS INVALID ARGUMENT TO BUILTIN FUNCTION 'ADDR'. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
        DCL P POINTER;
        .
        .
        .
        P = ADDR(27);
        .
        .
        .
    L:  P = ADDR(L);
```

Explanation: A constant in PL/I is not considered to be associated with a particular location in storage. It cannot, therefore, have a storage address.

IEL0620I S [PROLOGUE CODE.] ARGUMENT N TO D IS NOT AN ARRAY.

[PROLOGUE CODE.] ARGUMENT NUMBER N TO ENTRY D IS NOT AN ARRAY BUT THE CORRESPONDING PARAMETER HAS A '*' BOUND. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
        DCL J;
            CALL E(J);
    E:  PROC(P);
        DCL P(*);
```

Explanation: A parameter with an adjustable (*) bound is assumed to be an array that obtains the value for the bound from the associated argument. Consequently, the argument must also be an array.

IEL0621I S [PROLOGUE CODE.] AGGREGATE ARGUMENT D INVALID FOR ELEMENT PARAMETER.

[PROLOGUE CODE.] PARAMETER CORRESPONDING TO AGGREGATE ARGUMENT D IS AN ELEMENT. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
        1.  DCL E ENTRY(FLOAT),
                 ARR(8) FLOAT;
            CALL E(ARR);

        2.  DCL ARR(8) FLOAT;
                 CALL E(ARR);
                 E:  PROC (PARAM);
                    DCL PARAM FLOAT;
```

Explanation: An aggregate argument cannot be passed to a parameter that is not an aggregate.

IEL0622I S RECORD VARIABLE D NOT 'CONNECTED'.

| RECORD VARIABLE D IS NOT 'CONNECTED'. STATEMENT IGNORED.

Example:

```
     DCL 1 A (4),
            2 B CHAR(3),
            2 C CHAR(7);
     READ FILE(F) INTO (B);
```

Explanation: The INTO or FROM option of a record-oriented input/output statement must refer to an identifier that represents a contiguous area of storage.

IEL0623I S [PROLOGUE CODE.] ARGUMENT N TO D INVALID FOR 'CONTROLLED' PARAMETER.

[PROLOGUE CODE.] ARGUMENT NUMBER N TO ENTRY D INVALID FOR 'CONTROLLED' PARAMETER. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
         DCL X(10);
         CALL E(X),
     E:  PROC(C)
         DCL C(10) CTL;
```

Explanation: An argument corresponding to a controlled parameter must be a level 1 unsubscripted variable with the CONTROLLED attribute. Other attributes must also match those of the parameter so that the argument need not be converted and assigned to a temporary argument.

IEL0624I S [PROLOGUE CODE.] ARGUMENT N TO D HAS TOO MANY DIMENSIONS.

COMPILER RESTRICTION. [PROLOGUE CODE.] RESULT OF EXPRESSION IN ARGUMENT NUMBER N TO ENTRY D HAS TOO MANY DIMENSIONS. [STATEMENT IGNORED.] [RESULTS OF PROLGOUE UNDEFINED.]

Example:

```
     DCL 1 A,
           2 B,
           2 C(2,2,2,2,2,2,2,2,2,2);
     CALL X(A+C);
```

Explanation: The expression (A+C) results in a temporary argument that is an array of structures, the first structure element having 10 dimensions, and the second having 20 dimensions. The maximum permitted number of dimensions is 15. If an argument contains both an array and a structure and there is no parameter descriptor, the temporary argument is created in the form of an array of structures.

IEL0625I S [PROLOGUE CODE.] '*' USED AS ARGUMENT TO D.

[PROLOGUE CODE.] '*' USED AS ARGUMENT TO D. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
     1.  DCL X ENTRY;
         CALL X(*);

     2.  A=HBOUND(*,1);
```

Explanation: An asterisk, which can be used in a subscript list to indicate a cross-section of an array, is meaningless in an argument list. The error may have occurred because an array declaration has been omitted.

IEL0626I S [PROLOGUE CODE.] STRUCTURING OF D DOES NOT MATCH PARAMETER.

[PROLOGUE CODE.] STRUCTURING OF ARGUMENT D DOES NOT MATCH THAT OF PARAMETER. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
     1.  DCL 1 S,2 S1, 2 S2;
         CALL P(S);
         P: PROC(F);
         DCL F;

     2.  DCL 1 H, 2 H1, 3 H2;
         CALL Q(H);
         Q:PROC(G);
         DCL 1 G, 2 G1, 2 G2;
```

Explanation: A structure passed as an argument should match the corresponding parameter exactly. (However, a parameter that is a structure may correspond to an argument that is not a structure.)

IEL0627I S [PROLOGUE CODE.] DIMENSIONS OF D DO NOT MATCH PARAMETER.

[PROLOGUE CODE.] NUMBER OF DIMENSIONS IN ARGUMENT D DOES

NOT MATCH THAT OF PARAMETER. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
1.  DCL S(10);
    CALL P(S);
    P:PROC(F);
    DCL F;

2.  DCL H(10);
    CALL Q(H);
    Q:PROC(G);
    DCL G(10,10);
```

Explanation: An array passed as an argument must match the corresponding array parameter for dimensions. (However, a parameter that is an array may correspond to an argument that is not an array.)

IEL0628I S [PROLOGUE CODE.] BOUNDS OF D DO NOT MATCH PARAMETERS.

[PROLOGUE CODE.] BOUNDS OF ARGUMENT D DO NOT MATCH THOSE OF PARAMETER. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
1.  DCL S(4,12);
    CALL P(S);
    P:PROC(F);
    DCL F(4,10);

2.  DCL 1 H(6), 2 H1, 2 H2(5);
    CALL Q(H);
    Q: PROC(G);
    DCL 1 G(6), 2 G1, 2 G2(4);
```

Explanation: An argument with fixed bounds must match the corresponding parameter at all levels.

IEL0629I S [PROLOGUE CODE.] USE OF CROSS-SECTION OF STRUCTURE D INVALID.

COMPILER RESTRICTION. [PROLOGUE CODE.] USE OF CROSS-SECTION OF STRUCTURE D IS INVALID. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL 1 S(4,4), 2 S1, 2 S2;
CALL X(S(2,*));
```

Explanation: A cross-section of an array of structures cannot be given as an argument. The reference must be either fully subscripted with an asterisk for each dimension or unsubscripted.

IEL0630I S [PROLOGUE CODE.] SUBSCRIPT CONTAINING D IS NOT AN ELEMENT.

[PROLOGUE CODE.] SUBSCRIPT CONTAINING D IS NOT AN ELEMENT EXPRESSION. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
1.  DCL A(10,10);
       A(2,A)=1;

2.  DCL A(10,10);
        A(2, SUBSTR(A,1))=1;
```

Explanation: An array subscript must be an expression that represents the value of a single integer.

IEL0631I S [PROLOGUE CODE.] WRONG NUMBER OF SUBSCRIPTS FOR D.

[PROLOGUE CODE.] WRONG NUMBER OF SUBSCRIPTS FOR D. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
1.  DCL A(5,5);
        X = A(2);

2.  DCL A;
        X = A(2);
```

Explanation: A reference to an array must contain the same number of subscripts as given in its declaration.

IEL0632I S [PROLOGUE CODE.] STRUCTURE IS INVALID ARGUMENT TO T.

COMPILER RESTRICTION. [PROLOGUE CODE.] STRUCTURE IS INVALID ARGUMENT TO BUILTIN FUNCTION T. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL 1 S,
      2 S1 CHAR,
      2 S2 CHAR(4);
S= SUBSTR(S,1,3);
```

Explanation: The only built-in functions that accept structures as arguments are ALLOCATION, ADDR, and STRING. All other operations on structures by built-in functions must be specified individually for each element.

IEL0633I S [PROLOGUE CODE.] EXPRESSION OR 'ISUB' ARRAY INVALID ARGUMENT TO T.

[PROLOGUE CODE.] EXPRESSION OR ISUB-DEFINED ARRAY IS INVALID ARGUMENT TO BUILTIN FUNCTION T. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
I=ALLOCATION(A+B);
```

Explanation: Operational expressions are not permitted as arguments to the built-in functions ALLOCATION, ADDR, and STRING.

IEL0634I S [PROLOGUE CODE.] ELEMENT IS INVALID ARGUMENT TO T.

[PROLOGUE CODE.] ELEMENT IS INVALID ARGUMENT TO BUILTIN FUNCTION T. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL X;
    I= HBOUND(X,1);
```

Explanation: Array built-in functions cannot have element arguments.

IEL0635I E [PROLOGUE CODE.] NON-CONNECTED ARGUMENT TO 'ADDR' INVALID.

[PROLOGUE CODE.] NON-CONNECTED ARGUMENT TO BUILTIN FUNCTION 'ADDR' INVALID. ARGUMENT ACCEPTED.

Example:

```
DCL A(10,10), P POINTER;
P=ADDR(A(*,1));
```

Explanation: The argument to the built-in function ADDR occupies non-connected storage. The value returned by the function is the address of the first byte of the argument. Care must be exercised when using this pointer to refer to a based variable, since it is probable that the based variable will be mapped over storage occupied not only by the argument, but by some other variable as well.

IEL0636I S EXPRESSION OR 'ISUB' ARRAY IN GET/PUT DATA.

EXPRESSION OR ISUB-DEFINED ARRAY USED IN GET/PUT DATA. STATEMENT IGNORED.

Example:

```
1.  DCL A(10), B(5) DEF A(2*1SUB);
    GET DATA(B);

2.  DCL C(6) CHAR(8);
    PUT DATA(SUBSTR(C,3));
```

Explanation: PL/I does not permit expressions in GET DATA or PUT DATA statements, and the optimizing compiler does not implement the transmission of iSUB defined arrays by these statements.

IEL0637I S [PROLOGUE CODE.] SECOND ARGUMENT TO T IS AGGREGATE.

[PROLOGUE CODE.] SECOND ARGUMENT TO BUILTIN FUNCTION T IS AN AGGREGATE. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL ARR(10), T;
T = HBOUND(ARR,ARR);
```

Explanation: With the exception of the POLY built-in function, the array built-in functions that have two arguments must have an element expression as the second argument.

IEL0638I S [PROLOGUE CODE.] ARGUMENT N TO 'POLY' HAS MORE THAN ONE DIMENSION.

[PROLOGUE CODE.] ARGUMENT NUMBER N TO BUILTIN FUNCTION 'POLY' HAS MORE THAN ONE DIMENSION. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL ARR(6,6);
X = POLY(ARR,X);
```

IEL0639I E [PROLOGUE CODE.] ARGUMENT TO 'ADDR' MAY HAVE WRONG ALIGNMENT.

[PROLOGUE CODE.] ARGUMENT TO BUILTIN FUNCTION 'ADDR' MAY HAVE WRONG ALIGNMENT. ARGUMENT ACCEPTED.

Example:

```
DCL 1 S UNALIGNED,
      2 T BIT(3),
      2 U BIT(8),
      P PTR;
P = ADDR(U);
```

Explanation: This implementation uses byte addresses for locator values and does not provide bit addressing mechanisms for them. Consequently, if the argument to the ADDR built-in function does not lie on a byte boundary, the address returned will be that of the containing byte.

IEL0640I W ARGUMENT N TO GENERIC ASSUMED TO MATCH AGGREGATE PARAMETER.

ARGUMENT NUMBER N TO GENERIC FUNCTION IS ASSUMED TO MATCH ITS CORRESPONDING AGGREGATE PARAMETER.

Example:

```
DCL G GENERIC,
      (G1 WHEN(FIXED),
       G2 WHEN(FLOAT)),
    (G1,G2) ENTRY,
    ARR(10) FLOAT;
```

```
CALL G(ARR);
```

Explanation: Matching of arguments and parameters is not performed on aggregate arguments to generic functions. Consequently, a mis-match will not be detected and an execution-time error could result.

IEL0641I S NESTING OF FUNCTIONS EXCEEDS MAXIMUM.

COMPILER RESTRICTION. LEVEL OF NESTING OF FUNCTIONS EXCEEDS MAXIMUM. STATEMENT IGNORED.

Explanation: The nominal limit on the number of nested functions in a source module is 50. However, this limit may vary according to the length of the labels prefixed to the PROCEDURE statements. If the average length of the labels exceeds eight characters, the maximum number of nesting levels will be less than 50.

IEL0642I S ARRAY D IN ELEMENT ASSIGNMENT.

INVALID USE OF ARRAY D IN ELEMENT ASSIGNMENT. STATEMENT IGNORED.

Example:

```
DCL A(8,8);
I = A + J;
```

Explanation: An unsubscripted array reference cannot appear on the right-hand side of an assignment to an element variable.

IEL0643I S STRUCTURE D IN ARRAY OR ELEMENT ASSIGNMENT.

INVALID USE OF STRUCTURE D IN ARRAY OR ELEMENT ASSIGNMENT. STATEMENT IGNORED.

Example:

```
DCL 1 A, 2 B, 2 C;
DCL (X,Y)(5);
I=A+J; (invalid)
X=Y+A; (also invalid)
```

Explanation: A structure cannot be used in an assignment to an array or to an element variable.

IEL0644I S AGGREGATE D USED WHERE ELEMENT REQUIRED.

AGGREGATE D USED WHERE ELEMENT EXPRESSION IS REQUIRED. STATEMENT IGNORED.

Example:

```
DCL 1 F, 2 B, 2 C;
READ FILE (F) INTO(X);
```

Explanation: A structure has been used where the language requires an element expression.

IEL0645I S DIMENSIONS OF D DO NOT MATCH FIRST AGGREGATE.

NUMBER OF DIMENSIONS IN AGGREGATE D DOES NOT MATCH THE FIRST AGGREGATE IN EXPRESSION. STATEMENT IGNORED.

Example:

```
DCL A(6,6), B(6), C(6);
PUT EDIT (A+B+C) (A(5));
```

Explanation: In an expression involving more than one aggregate, all the aggregates involved must have identical dimensions.

IEL0646I S BOUNDS OF D DO NOT MATCH FIRST AGGREGATE.

BOUNDS OF AGGREGATE D DO NOT MATCH THE FIRST AGGREGATE IN EXPRESSION. STATEMENT IGNORED.

Example:

```
DCL A(3,3), B(2:5,-3:-1);
DCL C(4,4);
PUT LIST (A+C); (incorrect)
PUT LIST (A+B); (also incorrect)
```

Explanation: In an expression involving more than one aggregate, all the aggregates involved must have identical dimensions.

IEL0647I S STRUCTURING OF D DOES NOT MATCH FIRST STRUCTURE.

STRUCTURING OF D DOES NOT MATCH THE FIRST STRUCTURE IN EXPRESSION. STATEMENT IGNORED.

Example:

```
DCL 1 A, 2 B, 2 C,
    1 X, 2 Y, 2 Z, 3 U;
    PUT LIST(A+X);
```

Explanation: In an expression involving more than one structure, all the structures involved must have identical structuring.

IEL0648I S AGGREGATE D USED IN EXTENT SPECIFICATION IN BLOCK.

AGGREGATE D USED FOR EXTENT SPECIFICATION IN 'DECLARE' OR 'DEFAULT' STATEMENT FOR BLOCK BEGINNING AT THIS STATEMENT. RESULTS OF EXECUTION UNDEFINED.

Example:

```
          DCL 1 S,
                2 P,
                2 Q;
          DCL 1 A,
                2 B CHAR(5),
                2 C CHAR(S);
```

Explanation: The implementation will assume that the entire content of the aggregate is to be used as the length specification. This may result in an execution-time error.

IEL0649I E TARGET OF 'BYNAME' ASSIGNMENT NOT A STRUCTURE.

TARGET OF ASSIGNMENT CONTAINING 'BYNAME' OPTION IS NOT A STRUCTURE. OPTION IGNORED.

Example:

```
          DCL (A,B) FIXED;
          A = B, BY NAME;
```

Explanation: The BY NAME option can only be used in a structure assignment.

IEL0650I S NO STRUCTURE IN SOURCE OF 'BYNAME' ASSIGNMENT.

NO STRUCTURE IN SOURCE OF 'BYNAME' ASSIGNMENT. STATEMENT IGNORED.

Example:

```
          DCL 1 A, 2 OR, 3 RE, 3 GR,
              1 B, 2 OR, 3 RE, 3 BL;
              A=5,BY NAME;
```

Explanation: The BY NAME option has been used to qualify the assignment of a value that is not a structure.

IEL0652I S INVALID USE OF D IN ARRAY 'INITIAL' IN THIS BLOCK.

INVALID USE OF AGGREGATE D IN ARRAY 'INITIAL' IN THIS BLOCK. 'INITIAL' ATTRIBUTE IGNORED.

Example:

```
          DCL ARRAY1 (8,9),
              ARRAY2 (8,9) INIT(ARRAY1),
              ARRAY3 (8) INIT(ARRAY1(*,1));
```

Explanation: The INITIAL attribute for an array can specify initial values for the array elements on an individual basis only. The type of initialization attempted above can be achieved by an assignment statement.

IEL0653I W RESULTS MAY BE UNDEFINED IN ASSIGNMENT TO 'REFER' STRUCTURE D.

ASSIGNMENT TO STRUCTURE D DECLARED WITH 'REFER' OPTION. RESULTS UNDEFINED IF 'REFER' EXTENTS CHANGED BY ASSIGNMENT.

Example:

```
DCL 1 A BASED(P1),2 B,2 C(X REFER B);
    1 S BASED(P2),2 P,2 C(Y REFER P);
A=S;
This becomes
A.B = S.P; (ignored by the compiler
A.C = S.C;  for mapping of C in
           this assignment)
```

Explanation: The values of the bounds or extents of the REFER items in both source and target structures are taken from the target before assignment. If these values do not match in source and target, the values of these extents or bounds in the target will be altered by the assignment, and will not correspond to the REFER items assigned to the target. Therefore, in any subsequent references, the target is undefined.

Programmer Response: If the bounds or extents differ, they should be made to match prior to the assignment of the REFER items. The use of the BY NAME option may be convenient in the structure assignment once the REFER bounds or extents have been correctly set up.

IEL0654I S   DIMENSIONS OF D DO NOT MATCH TARGET.

NUMBER OF DIMENSIONS OF AGGREGATE D DOES NOT MATCH THE TARGET OF THE ASSIGNMENT OR DUMMY ARGUMENT. STATEMENT IGNORED.

Example:

```
DCL A(5,6), B(5,6), C(5);
A = B + C;
```

Explanation: The number of dimensions of an aggregate to be assigned must match the number of dimensions of the target aggregate.

IEL0655I S   BOUNDS OF D DO NOT MATCH TARGET.

BOUNDS OF AGGREGATE D DO NOT MATCH THE TARGET OF THE ASSIGNMENT OR DUMMY ARGUMENT. STATEMENT IGNORED.

Example:

```
DCL A(3,3), B(4,4), C(2:5,-3:-1);
A = A + B; (incorrect)
A = B + C; (also incorrect)
```

Explanation: The bounds for each dimension of an aggregate to be assigned must match the bounds for each dimension of the target aggregate.

IEL0656I S STRUCTURING OF D DOES NOT MATCH TARGET.

STRUCTURING OF D DOES NOT MATCH THE TARGET OF THE ASSIGNMENT OR DUMMY ARGUMENT. STATEMENT IGNORED.

Example:

```
     DCL 1 A, 2 B, 2 C;
         1 P, 2 Q, 2 R, 2 S;
         A = P;
```

Explanation: Structures in a structure assignment must have identical structuring.

IEL0657I S AGGREGATE D USED IN EXTENT SPECIFICATION.

AGGREGATE D USED FOR EXTENT SPECIFICATION IN 'ALLOCATE' STATEMENT. STATEMENT IGNORED.

Example:

```
     DCL X(*) CTL,
         1 A, 2 B, 2 C;
     ALLOCATE X(A);
```

IEL0658I S NO MATCHING IDENTIFIERS FOR 'BYNAME' ASSIGNMENT.

NO MATCHING IDENTIFIERS AT CORRESPONDING LEVELS IN THE STRUCTURES IN 'BYNAME' ASSIGNMENT. STATEMENT IGNORED.

Example:

```
     DCL 1 A, 2 B, 2 C,
         1,X, 2 Y, 2 Z,
         1 P, 2 Q, 2 R;
     A = X, BY NAME; (incorrect)
     A = P + X, BY NAME; (also incorrect)
```

Explanation: In order to use the BY NAME option in a structure assignment, the structure should have matching names at corresponding levels, otherwise no assignment can take place.

IEL0659I U TOO MANY ACTIVE QUALIFIED REFERENCES.

COMPILER RESTRICTION. TOO MANY QUALIFIED REFERENCES ACTIVE IN THIS STATEMENT. PHASE P.

Example:

```
          DCL (A,B,C,...Z) (10);
          A,B,C,...Z=A;
```

Explanation: A qualified reference can result from the use of any of the following:

1. An item declared BASED.
2. An item declared DEFINED.

3. The first argument of the SUBSTR built-in function or pseudovariable.

4. A subscripted item or array expression.

5. A multiple concatenation operation.

A qualified reference is active only for the statement that contains it unless it is the control variable of a do-loop when it is active throughout the scope of the loop. The limit to the number of active qualified references is 32; this limit will only be exceeded if the statement with a qualified reference appears in a nest of do-loops with qualified control variables, or if the statement is a multiple assignment with many qualified references as targets.

Programmer Response: Either simplify a multiple assignment or change do-loop control variables that are qualified references to non-qualified references.

|IEL0671I W [PROLOGUE CODE.] DUMMY CREATED FOR ARGUMENT N TO ENTRY D.

[PROLOGUE CODE.] ARGUMENT NUMBER N TO ENTRY D DOES NOT MATCH ITS CORRESPONDING PARAMETER. A DUMMY ARGUMENT HAS BEEN CREATED.

Example:

```
DCL X ENTRY (FIXED);
CALL X(A);
```

Explanation: Wherever an argument is known by the compiler not to match its parameter, the compiler generates a dummy argument that does match the parameter. On invocation of the entry point, the value of the argument is converted and assigned to the dummy argument.

IEL0672I W [PROLOGUE CODE.] ARGUMENT N TO T IGNORED.

[PROLOGUE CODE.] ARGUMENT NUMBER N TO BUILTIN FUNCTION T IS NOT REQUIRED FOR FLOATING POINT RESULT. ARGUMENT IGNORED.

Example:

```
A = DIV(B,C,5,2);
```

Explanation: A superfluous argument has been given in a reference to a built-in function.

IEL0673I I [PROLOGUE CODE.] ARGUMENT N TO T IS NOT 'COMPLEX'.

[PROLOGUE CODE.] ARGUMENT NUMBER N TO BUILTIN FUNCTION T NOT COMPLEX. ZERO IMAGINARY PART ASSUMED.

Example:

```
DCL A REAL;
```

```
A = REAL(A);
```

IEL0674I S INVALID ELEMENT EXPRESSION IN 'DO' OR 'IF'.

INVALID SPECIFICATION OF ELEMENT EXPRESSION IN 'DO' OR 'IF' STATEMENT. VALUE ONE ASSUMED FOR EXPRESSION.

Example:

```
1.  LABEL: DO I = 1 TO LABEL;

2.         IF FILE1 = FILE2 THEN...;
```

Explanation: The expression in a DO or IF statement must be a valid element expression which can be evaluated by the compiler.

IEL0675I S INVALID ITERATIVE SPECIFICATION.

INVALID ITERATIVE SPECIFICATION. NON-ITERATIVE 'DO' ASSUMED.

Example:

```
DO FILE = 1 TO 4;
```

IEL0676I I [PROLOGUE CODE.] ARGUMENT N TO D ASSUMED TO BE ALIGNED.

[PROLOGUE CODE.] ARGUMENT NUMBER N TO ENTRY D IS OF TYPE 'ENTRY' AND IS ASSUMED TO BE ALIGNED.

IEL0681I S [PROLOGUE CODE.] ATTRIBUTES OF ARGUMENT N TO ENTRY D CONFLICT WITH PARAMETER.

[PROLOGUE CODE.] ATTRIBUTES OF ARGUMENT NUMBER N TO ENTRY D CONFLICT WITH THE CORRESPONDING PARAMETER. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL A FILE,
    X ENTRY(FLOAT);
    CALL X(A);
```

Explanation: The compiler has detected a conflict between the attributes of an argument and its parameter which cannot be resolved by creating a dummy argument and performing a conversion.

IEL0682I S [PROLOGUE CODE.] ARGUMENT N TO ENTRY D IS NOT 'CONTROLLED'.

[PROLOGUE CODE.] ARGUMENT NUMBER N TO ENTRY D IS NOT 'CONTROLLED' BUT THE CORRESPONDING PARAMETER IS. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL A,
    E ENTRY(CONTROLLED);
CALL E(A);
```

Explanation: A parameter with the CONTROLLED attribute must correspond to an argument with the CONTROLLED attribute.

IEL0683I S [PROLOGUE CODE.] WRONG NUMBER OF ARGUMENTS TO T.

[PROLOGUE CODE.] WRONG NUMBER OF ARGUMENTS TO BUILTIN FUNCTION T. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
SUBSTR(A,B,C,D);
```

Explanation: A built-in function with either too few or too many arguments has been detected.

IEL0684I S [PROLOGUE CODE.] INVALID DATA TYPE FOR ARGUMENT N TO T.

[PROLOGUE CODE.] ARGUMENT NUMBER N HAS INCORRECT DATA TYPE FOR BUILTIN FUNCTION T. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL F FILE;
A = SIN(F);
```

IEL0685I S [PROLOGUE CODE.] MODE OF ARGUMENT N TO T IS INCORRECT.

[PROLOGUE CODE.] THE MODE OF ARGUMENT NUMBER N TO BUILTIN FUNCTION T IS INCORRECT. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL A COMPLEX;
B = CEIL(A);
```

IEL0686I S [PROLOGUE CODE.] ARGUMENT N TO T IS NOT INTEGER CONSTANT.

[PROLOGUE CODE.] ARGUMENT NUMBER N TO BUILTIN FUNCTION T IS NOT AN INTEGER CONSTANT. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
A = DECIMAL(B,C,D)
```

IEL0687I S [PROLOGUE CODE.] CONSTANT OR FUNCTION OR TEMPORARY RESULT HAS INVALID ATTRIBUTES FOR EXPRESSION.

[PROLOGUE CODE.] CONSTANT OR FUNCTION OR TEMPORARY RESULT

HAS INVALID ATTRIBUTES FOR EXPRESSION. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
L:  A = 1;
    B = 2 + L;

EC: PROC(Z) RETURNS(OFFSET);
    B = 2 + EC(1);
```

IEL0688I S [PROLOGUE CODE.] ASSIGNMENT TO CONSTANT.

[PROLOGUE CODE.] TARGET OF ASSIGNMENT IS A CONSTANT. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
1 = A + B;
```

Explanation: The target of an assignment can never be a constant; it must always be a variable.

IEL0689I S [PROLOGUE CODE.] SOURCE OF ASSIGNMENT DOES NOT MATCH TARGET.

[PROLOGUE CODE.] ATTRIBUTES OF SOURCE OF ASSIGNMENT STATEMENT CONFLICT WITH THE TARGET OR DUMMY ARGUMENT. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL LV LABEL VARIABLE,
    FV FILE VARIABLE;
    LV = FV;
```

IEL0690I S [PROLOGUE CODE.] OPERAND D INVALID IN ELEMENT EXPRESSION.

[PROLOGUE CODE.] INVALID USE OF OPERAND D IN AN ELEMENT EXPRESSION. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL A(10), F FILE;

a.  READ FILE(F) SET(P) KEY(A + B);

b.  B = F + C;
```

Explanation: An element expression cannot refer to a structure or unsubscripted array. Arithmetic operations can never involve non-arithmetic data such as files, events, or locators.

IEL0691I U 'DO' OR 'IF' NESTING LEVEL EXCEEDS N.

COMPILER RESTRICTION. LEVEL OF NESTING FOR 'DO' OR 'IF'

STATEMENTS EXCEEDS N.

IEL0692I S [PROLOGUE CODE.] ARGUMENT N TO ENTRY D INVALID.

[PROLOGUE CODE.] ARGUMENT NUMBER N TO ENTRY D INVALID. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
ON CONDITION(OLD)
   CALL RENEW(OLD);
```

IEL0694I S [PROLOGUE CODE.] NO SELECTION POSSIBLE FOR 'GENERIC' NAME.

[PROLOGUE CODE.] NO SELECTION POSSIBLE FOR 'GENERIC' NAME. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
DCL A CHAR(2),
    B FLOAT DEC,
    C FIXED BIN,
    GEN GENERIC
(GN1 WHEN (CHAR,*,CPLX),
GN2 WHEN (*,*,FIXED BIN(31,0)),
GN3 WHEN (CHAR,FLOAT DEC(16),FIXED BIN(15,0)),
GN4 WHEN (*,*,*,*));
```

Explanation: A reference to a generic name should contain arguments with attributes that match the generic descriptor list for one of the generic entry constants.

IEL0695I S [PROLOGUE CODE.] OPERANDS OF COMPARE CONFLICT.

[PROLOGUE CODE.] ATTRIBUTES OF OPERANDS IN AN EQUAL OR NOT-EQUAL OPERATION CONFLICT. [STATEMENT IGNORED.] [RESULTS OF PROLOGUE UNDEFINED.]

Example:

```
    DCL F FILE;
L:  IF L = F THEN...;
```

IEL0701I S 'FORTRAN' FUNCTION D NOT ALLOWED IN ARGUMENT.

COMPILER RESTRICTION. 'FORTRAN' FUNCTION D NOT ALLOWED IN ARGUMENT. 'FORTRAN' OPTION IGNORED.

Example:

```
DCL E ENTRY OPTIONS(FORTRAN);
CALL X(E(A));
```

Explanation: A function reference to a FORTRAN program is not permitted as an argument other than to a built-in function.

IEL0702I W 'NOMAP' SPECIFIED. MAPPING OF PARAMETER N TO D MAY DIFFER IN T.

MAPPING OF PARAMETER N TO ENTRY D MAY DIFFER IN PL/I AND T BUT DUMMY PARAMETER NOT CREATED BECAUSE OF 'NOMAP' OPTION.

Programmer Response: Ensure either that the parameter and its corresponding argument are mapped identically in the two language implementations or that differences in mapping are allowed for in the descriptions (or declarations) used in the two languages.

IEL0703I W 'NOMAP' SPECIFIED. MAPPING OF ARGUMENT N TO D MAY DIFFER IN T.

MAPPING OF ARGUMENT NUMBER N TO ENTRY D MAY DIFFER IN PL/I AND T BUT DUMMY ARGUMENT NOT CREATED BECAUSE OF 'NOMAP' OPTION.

Programmer Response: Ensure either that the argument and its corresponding parameter are mapped identically in the two language implementations or that differences in mapping are allowed for in the descriptions (or declarations) used in the two languages.

IEL0704I S MORE THAN N ARGUMENTS TO T ENTRY D.

COMPILER RESTRICTION. NUMBER OF ARGUMENTS TO T ENTRY D EXCEEDS N. EXCESS ARGUMENTS IGNORED.

Explanation: The maximum number of arguments that can be passed to a FORTRAN or COBOL routine in a single invocation is 64.

Programmer Response: Eliminate the excess number of arguments. If necessary and feasible, make these arguments known in both the invoking and invoked routines by declaring them STATIC EXTERNAL in PL/I and the equivalent to this in the invoked routine.

IEL0705I S EXTENTS OF PARAMETER N TO T ENTRY D NOT FIXED.

EXTENTS OF PARAMETER N TO T ENTRY D ARE NOT FIXED. RESULTS OF EXECUTION UNDEFINED.

Example:

```
          E:  ENTRY(P) OPTIONS(FORTRAN);
              DCL P(*,*);
```

Explanation: All bounds and extents of parameters to entry points invoked from COBOL or FORTRAN must be specified as decimal integer constants.

IEL0706I I T MAPPING USED FOR DUMMY ARGUMENT N TO D.

T MAPPING USED FOR DUMMY ARGUMENT NUMBER N TO ENTRY D.

Example:

```
          DCL 1 A,
               2 B CHAR(1),
               2 C FIXED BIN(31,0),
             CC ENTRY OPTIONS(COBOL);
         CALL CC(A+I);
```

Explanation: COBOL or FORTRAN mapping has been used for a dummy argument that has been created for an argument that is to be passed to a COBOL or FORTRAN routine.

IEL0707I I   PL/I MAPPING USED FOR DUMMY ARGUMENT N TO D.

PL/I MAPPING USED FOR DUMMY ARGUMENT NUMBER N TO ENTRY D.

Example:

```
DCL FF ENTRY OPTIONS(FORTRAN,

             NOMAPIN(ARG1)),
             A(10,10);
             CALL FF(A+I);
```

Explanation: A dummy argument is created for this argument according to normal PL/I rules with the NOMAPOUT option. The explicit use of the NOMAPIN option will combine with NOMAPOUT to produce the effective specification of the NOMAP option.

IEL0708I I   DUMMY CREATED FOR ARGUMENT N TO T ENTRY D.

MAPPING OF ARGUMENT NUMBER N TO ENTRY D MAY DIFFER IN PL/I AND T. DUMMY ARGUMENT CREATED.

Example:

```
          DCL 1 A,
               2 B CHAR(1),
               2 C FIXED BIN(31,0),
              X(10,10)
              COB ENTRY OPTIONS(COBOL),
              FORT ENTRY OPTIONS(FORTRAN);
          CALL COB(A);
          (message produced for A)
          CALL FORT(X);
          (message produced for X)
```

IEL0709I I   DUMMY CREATED FOR PARAMETER N TO T ENTRY D.

MAPPING OF PARAMETER N TO ENTRY D MAY DIFFER IN PL/I AND T. DUMMY PARAMETER CREATED.

Example:

```
         F:  ENTRY(X) OPTIONS(FORTRAN);
         C:  ENTRY(A) OPTIONS(COBOL);
            DCL 1 A,
                  2 B CHAR(1),
                  2 C FIXED BIN(31,0),
               X(10,10);
```

IEL0710I E D CONTAINS DATA INVALID FOR 'COBOL'.

RECORD VARIABLE D FOR 'COBOL' FILE CONTAINS 'AREA' OR 'BIT' DATA WITH NO EQUIVALENT IN 'COBOL'. PL/I MAPPING ASSUMED FOR VARIABLE.

Example:

```
DCL A BIT(8), F FILE
   ENV (COBOL....)  RECORD;
   READ FILE (F) INTO (A);
```

Explanation: A PL/I data type specified for a COBOL File has no equivalent in COBOL.

IEL0711I S T IGNORED FOR 'CALL' WITH TASKING OPTION.

T OPTION IGNORED FOR 'CALL' WITH TASKING OPTION.

Example:

```
DCL A ENTRY OPTIONS(COBOL);
CALL A TASK;
```

Explanation: Interlanguage subroutines cannot be tasks.

IEL0712I W PL/I MAPPING ASSUMED FOR ARRAY RECORD VARIABLE.

RECORD VARIABLE IS AN ARRAY. PL/I MAPPING ASSUMED FOR VARIABLE.

Example:

```
DCL A(8), F FILE
    ENV (COBOL....)  RECORD;
READ FILE (F) INTO (A);
```

Explanation: A PL/I data type specified for a COBOL file has no equivalent in COBOL.

IEL0713I S 'COBOL' FILE D INVALID IN ASSIGNMENT OR AS ARGUMENT.

USE OF COBOL FILE D IN ASSIGNMENT OR AS AN ARGUMENT IS INVALID. 'COBOL' OPTION WILL NOT APPLY TO TARGET.

Example:

```
DCL PROC ENTRY (FILE),
    COBFIL FILE ENV(COBOL...);
    CALL PROC(COBFILE);
```

IEL0714I W D CONTAINS DATA INVALID FOR 'COBOL'.

RECORD VARIABLE FOR 'COBOL' FILE CONTAINS ELEMENT WITH NO

DIRECT EQUIVALENT IN 'COBOL'. 'COBOL' MAPPING ASSUMED FOR VARIABLE.

Example:

```
     DCL A FIXED BIN (15,6),
     F FILE ENV (COBOL) RECORD;
     READ FILE (F) INTO (A);
```

Explanation: "A" has fractional precision which is not available for fixed binary (COMPUTATIONAL) variables in COBOL.

IEL0715I E   STATEMENT INVALID FOR 'COBOL' FILE D.

STATEMENT INVALID FOR COBOL FILE D. PL/I MAPPING ASSUMED FOR RECORD.

Example:

```
     DCL F FILE ENV(COBOL);
     DELETE FILE(F);
```

IEL0716I E   'SET' OPTION INVALID FOR 'COBOL' FILE D.

'SET' OPTION ON 'READ' STATEMENT INVALID FOR COBOL FILE D. PL/I MAPPING ASSUMED FOR RECORD.

Explanation: Locate mode input/output is not permitted for a 'COBOL' file. Move mode must be used.

IEL0717I E   'EVENT' OPTION INVALID FOR 'COBOL' FILE D.

'EVENT' OPTION INVALID FOR COBOL FILE D WHEN PL/I AND COBOL MAPPING MAY DIFFER. PL/I MAPPING ASSUMED FOR RECORD.

Example:

```
     DCL F FILE ENV(COBOL),
         1 R,
         2 S CHAR(1)
         2 T FIXED BIN(31,0);
      READ FILE (F) INTO (R) EVENT (EV);
```

Explanation: The EVENT option is permitted only if it can be deduced at compile-time that the mapping of the record will be the same in PL/I and COBOL.

IEL0720I E   ARGUMENT N TO D CONTAINS DATA INVALID FOR T.

ARGUMENT N TO ENTRY D CONTAINS 'AREA' OR 'BIT' DATA WITH NO EQUIVALENT IN T. PL/I MAPPING ASSUMED FOR ARGUMENT IF AGGREGATE.

Example:

```
DCL I BIT(10), E ENTRY
EXTERNAL OPTIONS (FORTRAN);
CALL E(I);
```

Explanation: An argument which has no direct equivalent in COBOL or FORTRAN has been encountered in a CALL statement or function reference to invoke a COBOL or FORTRAN routine. Note that arguments with the attributes BIT(8) and BIT(32) are acceptable to FORTRAN.

IEL0721I E PARAMETER N TO D CONTAINS DATA INVALID FOR T.

PARAMETER N TO ENTRY D CONTAINS 'AREA' OR 'BIT' DATA WHICH HAS NO EQUIVALENT IN T. PL/I MAPPING ASSUMED FOR PARAMETER IF AGGREGATE.

Example:

```
E: ENTRY (X,Y,Z) OPTIONS (COBOL);
DCL Y BIT(8);
```

Explanation: A parameter which has no direct equivalent in COBOL or FORTRAN has been encountered in a PROCEDURE or ENTRY statement invoked from a COBOL or FORTRAN routine.

IEL0722I E ARGUMENT N TO 'COBOL' ENTRY D IS AN ARRAY.

ARGUMENT N TO ENTRY D IS AN ARRAY WHICH IS INVALID FOR 'COBOL'. PL/I MAPPING ASSUMED FOR ARGUMENT.

Example:

```
DCL E ENTRY EXTERNAL OPTIONS (COBOL),
   I(8) FIXED BIN;
   CALL E(I);
```

Explanation: COBOL data types do not include the equivalent of PL/I arrays.

IEL0723I E PARAMETER N TO 'COBOL' ENTRY D IS AN ARRAY.

PARAMETER N TO ENTRY D IS AN ARRAY WHICH IS INVALID FOR 'COBOL'. PL/I MAPPING ASSUMED FOR PARAMETER.

Example:

```
E:  ENTRY (A,B,C) OPTIONS (COBOL);
    DCL A(8) FIXED BIN;
```

Explanation: COBOL data types do not include the equivalent of PL/I arrays.

IEL0724I W DATA IN ARGUMENT N TO D INVALID FOR T.

ARGUMENT NUMBER N TO ENTRY D CONTAINS ELEMENT WITH NO DIRECT EQUIVALENT IN T.

Example:

```
DCL E ENTRY EXTERNAL (FORTRAN);
DCL I FIXED BIN (10,6);
CALL E(I);
```

Explanation: 'I' should have the precision (n,0).

IEL0725I W DATA IN PARAMETER N TO D INVALID FOR T.

PARAMETER N TO ENTRY D CONTAINS ELEMENT WHICH HAS NO DIRECT EQUIVALENT IN T.

Example:

```
E:  ENTRY (I,J,K) OPTIONS (COBOL);
    DCL I FLOAT DEC (20);
```

Explanation: COBOL does not implement extended precision floating-point variables, but only variables with short precision (COMPUTATIONAL-1) or long precision (COMPUTATIONAL-2).

IEL0726I I EXTENDED PRECISION ITEM FOR D VALID ONLY FOR 'FORTRAN' H PROGRAMS.

AN ARGUMENT OR PARAMETER OR RETURNED VALUE FOR ENTRY D HAS EXTENDED PRECISION. VALID ONLY FOR 'FORTRAN' PROGRAMS COMPILED BY THE 'FORTRAN' H COMPILER.

Example:

```
DCL E ENTRY EXTERNAL OPTIONS(FORTRAN);
DCL I FLOAT BIN(60);
CALL E(I);
```

IEL0727I S PARAMETER N TO T ENTRY D MUST NOT BE 'CONTROLLED'.

PARAMETER N TO ENTRY D HAS 'CONTROLLED' STORAGE CLASS WITH NO EQUIVALENT IN T. RESULTS OF EXECUTION UNDEFINED.

Example:

```
E:  ENTRY (I,J) OPTIONS (COBOL);
    DCL J CONTROLLED;
```

IEL0728I I DATA TYPE RETURNED BY D INVALID IN 'FORTRAN'.

DATA TYPE OF RETURNED VALUE FROM PL/I ENTRY D HAS NO DIRECT EQUIVALENT IN 'FORTRAN' BUT ENTRY CAN BE INVOKED AS A FUNCTION.

Explanation: This message is printed if a value returned by a function is arithmetic with a precision other than (n,0) or is a fixed-length character string with a length greater than 255.

IEL0729I S DATA TYPE RETURNED BY D INVALID IN 'FORTRAN'.

DATA TYPE OF RETURNED VALUE FROM 'FORTRAN' FUNCTION D HAS NO DIRECT EQUIVALENT IN 'FORTRAN'.

Explanation: This message is printed if the returned value is arithmetic with a precision other than (n,0) or is a fixed-length character string with a length greater than 255.

IEL0730I S ARGUMENT N TO T ENTRY D IS NOT 'CONNECTED'.

ARGUMENT N TO T ENTRY D IS NOT 'CONNECTED' AND THE 'NOMAP' OPTION IS SPECIFIED. RESULTS OF EXECUTION UNDEFINED.

Example:

```
DCL 1 A(3), 2 B, 3 C, 2 D,
    3 A ENTRY EXTERNAL OPTIONS
      (COBOL, NOMAP);
   CALL E (B);
```

Explanation: An argument must occupy a contiguous area of storage when passed as a parameter to an invoked routine.

IEL0731I I ARGUMENT N TO T ENTRY D ASSUMED TO BE 'CONNECTED'.

ARGUMENT PASSED TO UNCONNECTED PARAMETER N OF T ENTRY D IS ASSUMED TO BE CONNECTED.

Example:

```
E:  ENTRY (A,B,D) OPTIONS
   (COBOL, NOMAP);
   DCL 1 A(3), 2 B, 3 C, 2 D,
       3 E;
```

Explanation: A parameter must be a variable that occupies a contiguous area of storage.

IEL0732I I DATA TYPE RETURNED BY D INVALID IN 'FORTRAN'.

DATA TYPE OF RETURNED VALUE FROM PL/I ENTRY D IS INVALID FOR 'FORTRAN'. ENTRY CANNOT BE INVOKED AS A FUNCTION.

Explanation: If the returned value is a character string, it should be fixed-length; if it is a bit string, it should be BIT(8) or BIT(32).

IEL0733I W DATA TYPE RETURNED BY D INVALID IN 'FORTRAN'.

DATA TYPE OF RETURNED VALUE FROM 'FORTRAN' FUNCTION D IS INVALID FOR 'FORTRAN'.

Explanation: If the returned value is a character string, it should be fixed-length; if it is a bit string, it should be BIT(8) or BIT(32).

IEL0734I E ARGUMENT N TO D NOT CORRECTLY ALIGNED FOR 'FORTRAN'.

ARGUMENT N TO ENTRY D IS NOT CORRECTLY ALIGNED FOR 'FORTRAN' AND THE 'NOMAP' OPTION IS SPECIFIED. RESULTS OF EXECUTION UNDEFINED.

Example:

```
     DCL E ENTRY EXTERNAL
          OPTIONS (FORTRAN, NOMAP),
          I UNALIGNED;
          CALL E(I);
```

IEL0735I W ARGUMENT N TO D NOT CORRECTLY ALIGNED FOR T.

ARGUMENT N TO ENTRY D IS AN ELEMENT WHICH MAY NOT BE CORRECTLY ALIGNED FOR T. NO DUMMY ARGUMENT CREATED.

Example:

```
                DCL 1 A UNALIGNED,
                    2 B BIT(5),
                    2 C BIT(27),
                FF ENTRY OPTIONS(FORTRAN);
                CALL FF(C);
             /* C IS AN UNALIGNED ELEMENT */
```

Explanation: Although, according to PL/I rules, a dummy argument is not created for an element argument, the alignment of the argument may not be acceptable as a parameter to a COBOL or FORTRAN routine, and an addressing interrupt may occur when the routine is invoked.

IEL0736I E STRUCTURE ARGUMENT N TO D INVALID FOR 'FORTRAN'.

ARGUMENT N TO ENTRY D IS A STRUCTURE WHICH IS INVALID FOR 'FORTRAN'. PL/I MAPPING ASSUMED FOR ARGUMENT.

Example:

```
     DCL E ENTRY EXTERNAL
          OPTIONS (FORTRAN),
       1 I,2 J,2 K;
       CALL E (I);
```

Explanation: FORTRAN data types do not include the equivalent of PL/I structures.

IEL0737I E STRUCTURE PARAMETER N TO D INVALID FOR 'FORTRAN'.

PARAMETER N TO ENTRY D IS A STRUCTURE WHICH IS INVALID FOR 'FORTRAN'. PL/I MAPPING ASSUMED FOR PARAMETER.

Explanation: FORTRAN data types do not include the equivalent of PL/I structures.

IEL0738I I D CANNOT BE INVOKED AS FUNCTION FROM 'FORTRAN'.

ENTRY D HAS NO PARAMETERS. THE ENTRY CANNOT BE INVOKED AS

A FUNCTION FROM 'FORTRAN'.

Explanation: An entry point in PL/I without parameters can only be invoked from FORTRAN by a CALL statement.

IEL0739I S D HAS NO ARGUMENTS. 'FORTRAN' OPTION IGNORED.

FUNCTION D HAS NO ARGUMENTS. 'FORTRAN' OPTION IGNORED.

Explanation: A FORTRAN function cannot be invoked from a PL/I function reference that does not include an argument.

|IEL0740I S D CANNOT BE A FUNCTION.

'COBOL' OR 'ASSEMBLER' ENTRY D CANNOT BE INVOKED AS A FUNCTION. INTERLANGUAGE OPTION IGNORED.

Example:

```
    DCL SUB ENTRY OPTIONS(COBOL);
    DCL (A,B);
    A=  SUB(B);
```

Explanation: A COBOL or ASSEMBLER procedure cannot be invoked as a function by a PL/I program.

IEL0741I S D CANNOT BE MAPPED CORRECTLY.

COMPILER RESTRICTION. AGGREGATE D CANNOT BE MAPPED CORRECTLY. RESULTS OF EXECUTION UNDEFINED.

Example:

```
     DCL A(32000:32010,32000:32010)
         CHAR(1000);
```

Explanation: The size of the offset byte address from the virtual origin of an aggregate cannot exceed (2^{24})-1.

IEL0742I S AGGREGATE D EXCEEDS MAXIMUM LENGTH.

COMPILER RESTRICTION. AGGREGATE D EXCEEDS MAXIMUM LENGTH. RESULTS OF EXECUTION UNDEFINED.

Example:

```
     DCL A (256,256,256) FIXED BINARY;
```

Explanation: The size of the offset byte address from the virtual origin of an aggregate cannot exceed (2^{24})-1.

IEL0761I E BIT VALUE ONE ASSUMED IN 'IF' EXPRESSION.

VARIABLE IN 'IF' OR 'WHILE' EXPRESSION CANNOT BE CONVERTED TO BIT STRING. BIT CONSTANT OF LENGTH AND VALUE ONE ASSUMED.

Example:

```
     DCL F FILE;
     IF F THEN X = Y;
```

Explanation: Only string and arithmetic element variables and constants are permitted in the conditional clause of an IF statement or WHILE expression.

IEL0762I W

TOO FEW ARGUMENTS IN CALL TO D.

FEWER ARGUMENTS THAN PARAMETERS FOR CALL TO 'ASSEMBLER' PROCEDURE D.

Example:

```
    DCL P ENTRY(FIXED,FIXED) OPTIONS(ASSEMBLER);
    CALL P(A);
```

Explanation: An assembly language external procedure has been invoked with fewer arguments than the number of parameters to the corresponding DECLARE statement.

IEL0763I E

NEGATIVE SECOND ARGUMENT TO 'BIT' OR 'CHAR'.

NEGATIVE SECOND ARGUMENT TO 'BIT' OR 'CHAR' BUILTIN FUNCTION. ZERO ASSUMED.

Example:

```
   PUT LIST(BIT(I,-3));
```

Explanation: The second argument to the 'BIT' or 'CHAR' built-in function specifies the string length for the converted first argument. This length cannot be negative.

IEL0764I E

LENGTH OF STRING OPERATION RESULT EXCEEDS N.

COMPILER RESTRICTION. LENGTH OF RESULT OF STRING OPERATION EXCEEDS N. LENGTH OF N ASSUMED.

Example:

```
     DCL (A,B) CHAR(32767);
     A=A||B;
```

Explanation: A character string cannot exceed 32767 characters in length.

IEL0765I S

NON-CONSTANT VALUE IN STATIC INITIAL FOR D IN THIS BLOCK.

NON-CONSTANT VALUE IN A STATIC 'INITIAL' SPECIFICATION FOR VARIABLE D IN THE BLOCK BEGINNING WITH THIS STATEMENT. 'INITIAL' SPECIFICATION IGNORED.

Example:

```
     DCL X(2) STATIC INITIAL((X)1);
```

Explanation: Only constants may appear in the INITIAL attribute for STATIC variables.

IEL0766I E 'REPEAT' STRING RESULT EXCEEDS MAXIMUM LENGTH.

COMPILER RESTRICTION. STRING RESULT FROM 'REPEAT' BUILTIN FUNCTION GREATER THAN ALLOWED MAXIMUM LENGTH. ZERO REPETITION FACTOR ASSUMED.

Example:

```
REPEAT('XXX',40000);
```

Explanation: The result should not produce a string greater than 32,767 characters (or bits) in length.

IEL0767I E NEGATIVE REPETITION FACTOR FOR 'REPEAT'.

NEGATIVE REPETITION FACTOR SPECIFIED FOR 'REPEAT' BUILTIN FUNCTION. ZERO REPETITION FACTOR ASSUMED.

Example:

```
REPEAT('XXX',-2);
```

IEL0768I W CONSTANT IN 'IF' OR 'WHILE' CLAUSE.

CONSTANT SPECIFIED IN 'IF' OR 'WHILE' CLAUSE. FLOW WILL BE UNCONDITIONAL.

Example:

```
IF 1 THEN ...;
     ELSE ...;
```

(The THEN clause will always be executed.)

```
IF 0 THEN ...;
     ELSE ...;
```

(The ELSE clause will always be executed.)

```
DO WHILE(1);
```

(The loop will always be executed and may be permanent.)

```
DO WHILE(0);
```

(The loop will never be executed.)

Explanation: A constant has been supplied in an IF statement or WHILE expression. Execution of the statement can result in one flow of control only.

IEL0769I S AREA VARIABLE FOR OFFSET D INVALID.

COMPILER RESTRICTION. SPECIFICATION OF AREA VARIABLE ASSOCIATED WITH OFFSET D NOT VALID IN THIS STATEMENT.

Example:

```
DCL O OFFSET (A),
    A AREA BASED (P);
    .
    .
    .
    O -> A = X;
```

Explanation: See the relevant section of the language reference manual for this compiler for an explanation of the language restrictions concerning the use of qualified AREA variables.

IEL0776I E   CONSTANT SUBSCRIPT OF D OUT OF RANGE.

VALUE OF CONSTANT SUBSCRIPT FOR ARRAY D IS OUT OF RANGE BUT HAS NOT BEEN REPLACED.

Example:

```
(SUBSCRIPTRANGE):PIG:PROC;
                  DCL A(2,3);
                  A(6,3) = 1;
                  END;
```

IEL0777I W   TOO MANY ITEMS IN 'INITIAL' LIST FOR D.

TOO MANY ITEMS IN 'INITIAL' LIST FOR ARRAY D. REDUNDANT ITEMS IGNORED.

Example:

```
DCL A(2) INIT(1,2,3);
```

IEL0778I S   'ISUB' VARIABLE FOR D OUT OF RANGE.

'ISUB' VARIABLE FOR DEFINED ARRAY D OUT OF RANGE. RESULTS OF EXECUTION UNDEFINED.

Example:

```
DCL A(10,10),
   B(4,4)
   DEFINED A(1*1SUB,3SUB);
```

Explanation: The 3SUB variable is a reference to the third dimension of an array having only two dimensions.

IEL0779I S   INVALID REPETITION FACTOR IN 'INITIAL' FOR D.

ZERO OR NEGATIVE REPETITION FACTOR IN 'INITIAL' ATTRIBUTE FOR ARRAY D. REPETITION FACTOR IGNORED.

Example:

```
DCL A(10) INIT((0)1,(-2)1);
```

IEL0787I W   INITIAL VALUE OF ITERATIVE SPECIFICATION OUT OF RANGE.

INITIAL VALUE OF ITERATIVE SPECIFICATION IS OUTSIDE THE RANGE OF THE 'BY' AND 'TO' EXPRESSIONS. LOOP WILL NOT BE EXECUTED.

Example:

```
1.  DO I = 1 BY -2 TO 10;

2.  DO I = 1 BY 3 TO -10;
```

IEL0798I S INVALID IDENTIFIER IN CHECK LIST.

INVALID IDENTIFIER IN CHECK LIST. IDENTIFIER IGNORED.

Example:

```
(CHECK(SQRT)): P:PROC;
          DCL SQRT BUILTIN;
```

Explanation: The CHECK condition can be enabled only for variables, label variables, and label or entry constants.

Programmer Response: Remove the invalid identifier from the name list in the CHECK condition prefix.

IEL0799I W AREA ASSOCIATED WITH OFFSET D MAY BE INVALID FOR LOCATOR CONVERSION IN 'RETURN'.

COMPILER RESTRICTION. AREA ASSOCIATED WITH OFFSET D INVALID FOR LOCATOR CONVERSION FOR 'RETURN' STATEMENT. 'RETURN' EXPRESSION WILL BE IGNORED IF THE INVALID COMBINATION OF 'RETURN' AND 'ENTRY' IS USED.

Example:

```
P: PROC RETURNS (OFFSET(A));
Q: ENTRY RETURNS (POINTER);
   DCL A AREA BASED,
       PTR POINTER,
   RETURN (PTR);
```

Explanation: If locator conversion is required in a RETURN statement, the offset must have an associated area. The area must be unsubscripted, it cannot be defined; if it is based it must be based on an explicit non-based, non-defined unsubscripted pointer.

Programmer response: Ensure that the combination of return expression and entry type never requires locator conversion to be performed.

IEL0800I S INVALID SPECIFICATION IN 'WAIT'.

INVALID SPECIFICATION OF NUMBER OF EVENTS IN 'WAIT' STATEMENT. SPECIFICATION IGNORED.

Example:

```
DCL (E1,E2) EVENT, F FILE;
.
.
.
WAIT (E1,E2)(F);
```

Explanation: The number of events specification in the WAIT statement must be convertible to a FIXED BINARY integer.

IEL0801I S INVALID EXPRESSION IN 'DELAY'.

INVALID EXPRESSION IN 'DELAY' STATEMENT. ZERO ASSUMED.

Example:

```
     DCL F FILE;
     DELAY (F);
```

Explanation: The expression in the DELAY statement must be convertible to a FIXED BINARY integer.

IEL0802I S INVALID EXPRESSION IN 'RETURN'.

INVALID EXPRESSION IN 'RETURN' STATEMENT. EXPRESSION IGNORED.

Example:

```
     DCL C CONDITION;
     RETURN (C);
```

Explanation: The expression in a RETURN statement must be problem data or locator, area, label, event, file, or task program control data.

IEL0803I S INVALID 'DISPLAY' EXPRESSION.

'DISPLAY' EXPRESSION IS NOT A VALID DATA TYPE OR ELEMENT EXPRESSION. STATEMENT IGNORED.

Example:

```
     DCL F FILE, A(3) CHAR(1)
     DISPLAY (F);
     DISPLAY (A);
```

Explanation: The argument of a DISPLAY statement must be an element expression that can be converted to character form.

IEL0804I W 'DISPLAY' STRING LENGTH EXCEEDS 72.

STRING LENGTH FOR 'DISPLAY' EXCEEDS 72 CHARACTERS. TERMINAL MAY NOT SUPPORT THIS [FIRST N CHARACTERS USED].

Example:

```
     DCL A CHAR(150);
     DCL B CHAR(80);
     DISPLAY (A);
     DISPLAY (B);
```

Explanation: The first 126 characters of a DISPLAY expression will always be transmitted to the terminal, but if the terminal is restricted to a length of 72 bytes, only the first 72 characters will be printed. If the length of the DISPLAY expression is more than 126, only the first 126 characters will be printed, even if there is no restriction on the terminal used by the system.

The example above would result in the full text of the message being produced for DISPLAY(A), but the text of the message enclosed within square brackets will not be produced for DISPLAY(B).

IEL0805I S 'REPLY' CONTAINS NON-ELEMENT EXPRESSION.

'REPLY' EXPRESSION IS NOT A CHARACTER STRING EXPRESSION. 'REPLY' OPTION IGNORED.

Example:

```
DCL ABC(2,3) FIXED BINARY;
DISPLAY('MESSAGE') REPLY(ABC);
```

Explanation: The expression in the REPLY statement is not a character string variable. The REPLY option is ignored. If an EVENT option is present, this too is ignored.

IEL0806I E 'REPLY' STRING LENGTH EXCEEDS N.

COMPILER RESTRICTION. STRING LENGTH FOR 'REPLY' TOO LONG. FIRST N CHARACTERS USED.

Example:

```
DCL R CHAR(100);
DISPLAY('MESSAGE') REPLY(R);
```

Explanation: The length of the REPLY expression is more than 72 bytes. Only the first 72 bytes of the message are transmitted.

IEL0808I W T NOT ENABLED. 'SIGNAL' IGNORED.

T CONDITION NOT ENABLED. STATEMENT IGNORED.

Example:

```
(NOZERODIVIDE): PIG:PROC;
                SIGNAL ZERODIVIDE;
                END;
```

Explanation: The SIGNAL statement for a disabled condition acts as a null statement.

IEL0809I W 'CHECK' NOT ENABLED FOR D

'CHECK' CONDITION NOT ENABLED FOR VARIABLE D IN SIGNAL STATEMENT. VARIABLE IGNCRED.

Example:

```
(CHECK(A,B)):PIG:PROC;
```

```
SIGNAL CHECK(A,B,C);
END;
```

IEL0810I S INVALID EXPRESSION IN 'PRIORITY' OPTION.

INVALID EXPRESSION IN 'PRIORITY' OPTION. OPTION IGNORED.

Explanation: The expression in the PRIORITY option must represent an integer value with the precision (15,0).

IEL0811I S INVALID SPECIFICATION OF 'IGNORE' EXPRESSION.

INVALID SPECIFICATION OF 'IGNORE' EXPRESSION. VALUE ONE ASSUMED.

Example:

```
DCL F FILE;
READ FILE(F) IGNORE(F);
```

Explanation: The expression in the IGNORE option must represent an arithmetic integer value.

IEL0812I S 'KEY' SPECIFICATION INVALID.

'KEY' SPECIFICATION INVALID. RESULTS OF EXECUTION UNDEFINED.

Example:

```
DCL F FILE, A(5) FIXED;
READ FILE(F) INTO(A) KEY(A);
```

Explanation: The expression in the KEY option must represent a valid key derived from a character string or an arithmetic variable.

IEL0816I S ATTRIBUTES OF D CONFLICT WITH USE.

CONFLICT BETWEEN ATTRIBUTES OF FILE D AND ITS USE IN THIS STATEMENT. STATEMENT IGNORED.

Example:

```
DCL F FILE OUTPUT;
READ FILE(F) INTO(A);
```

IEL0817I S RECORD VARIABLE INVALID.

RECORD VARIABLE INVALID. STATEMENT IGNORED.

Example:

```
DCL 1 A, 2 B BIT(M) UNALIGNED,
         2 C BIT(N) UNALIGNED;
READ FILE(F) INTO(B);
```

Explanation: The variable named in the INTO or FROM option cannot be an unaligned and non-varying bit string that is also based, defined, a parameter, or contained in an aggregate. Neither can it be any minor structure that starts or ends with an unaligned, non-varying bit string.

IEL0818I S INVALID SET OF OPTIONS.

INVALID SET OF OPTIONS ON RECORD I/O STATEMENT. STATEMENT IGNORED.

Example:

```
     REWRITE FILE(F) EVENT(E);
```

Explanation: The input/output statement has an invalid or incomplete set of options. In the example above, the FROM option is missing.

IEL0819I I RECORD I/O FUNCTION OPTIMIZED.

RECORD I/O FUNCTION OPTIMIZED. NO LIBRARY SUBROUTINE CALL REQUIRED.

Explanation: Under certain circumstances, the optimizing compiler will generate inline code for record-oriented I/O statements.

IEL0820I S ATTRIBUTES OF D CONFLICT WITH THOSE ON 'OPEN'.

ATTRIBUTES ON 'OPEN' STATEMENT CONFLICT WITH THOSE DECLARED FOR FILE D. 'UNDEFINEDFILE' CONDITION MAY BE RAISED ON ATTEMPT TO OPEN THE FILE.

Example:

```
          DCL F FILE INPUT;
             OPEN FILE(F) OUTPUT;
```

IEL0827I E 'PAGESIZE' OR 'LINESIZE' CONFLICT WITH ATTRIBUTES OF D.

SPECIFICATION OF 'PAGESIZE' OR 'LINESIZE' OPTION CONFLICTS WITH ATTRIBUTES OF FILE D. OPTION IGNORED.

Explanation: The PAGESIZE option can only be specified for PRINT files. The LINESIZE option can only be specified for STREAM OUTPUT files (including PRINT files).

IEL0828I S INVALID 'TITLE' 'PAGESIZE' OR 'LINESIZE' FOR D.

INVALID SPECIFICATION OF 'TITLE' 'PAGESIZE' OR 'LINESIZE' OPTION FOR FILE D. OPTION IGNORED.

Explanation: Arguments to the options PAGESIZE and LINESIZE must be expressions that represent an arithmetic value. The argument to the TITLE option must be an expression that represents a character-string value.

| IEL0830I S NO ACCESS 'ENV' OPTION FOR 'DIRECT' FILE D.

'DIRECT' ATTRIBUTE BUT NO ACCESS TYPE 'ENVIRONMENT' OPTION FOR FILE D. 'UNDEFINEDFILE' MAY BE RAISED ON ATTEMPT TO OPEN THE FILE.

Example:

```
DCL D FILE DIRECT;
```

Explanation: INDEXED, REGIONAL, or VSAM must be specified in the ENVIRONMENT option for a file with the DIRECT attribute.

Programmer Response: Specify access type in ENVIRONMENT option.

IEL0834I S ATTRIBUTES AND ENVIRONMENT OPTIONS FOR D CONFLICT.

ATTRIBUTES AND 'ENVIRONMENT' OPTIONS FOR FILE D CONFLICT. 'UNDEFINEDFILE' MAY BE RAISED ON ATTEMPT TO OPEN THE FILE.

Example:

```
DCL A FILE DIRECT ENV(CONSECUTIVE...);
```

IEL0835I S INVALID ATTRIBUTES FOR D IGNORED.

INVALID ATTRIBUTE(S) FOR FILE D IGNORED.

Example:

```
DCL F FILE FLOAT;
```

IEL0836I S T FOR D CONFLICTS WITH PREVIOUSLY DECLARED ATTRIBUTES.

ATTRIBUTE T IN FILE D CONFLICTS WITH ONE PREVIOUSLY DECLARED AND IS IGNORED.

Example:

```
DCL F FILE INPUT STREAM OUTPUT...;
```

IEL0837I S INVALID 'ENVIRONMENT' OPTION(S) FOR D IGNORED.

INVALID 'ENVIRONMENT' OPTION(S) FOR FILE D IGNORED.

Example:

```
DCL F FILE ENV(REGIONAL...);
```

IEL0838I S T FOR D CONFLICTS WITH PREVIOUSLY DECLARED OPTIONS.

'ENVIRONMENT' OPTION T IN FILE D CONFLICTS WITH ONE PREVIOUSLY DECLARED AND IS IGNORED.

Example:

```
     DCL F FILE ENVIRONMENT
     (INDEXED F RECSIZE(80) CONSECUTIVE...);
```

IEL0840I S INVALID OPTION(S) ON 'CLOSE' FOR D.

INVALID 'ENVIRONMENT' OPTION(S) ON 'CLOSE' STATEMENT FOR FILE D. OPTION(S) IGNORED.

Example:

```
     CLOSE FILE(F) ENV(INDEXED);
```

Explanation: The only option allowed in a CLOSE statement is the LEAVE option.

IEL0848I S INVALID 'GET' OR 'PUT' STATEMENT IGNORED.

'GET' OR 'PUT' STATEMENT REFERENCES A 'RECORD' FILE. STATEMENT IGNORED.

Explanation: A GET or PUT statement may only reference a file with the STREAM attribute.

IEL0849I E INVALID 'E' OR 'F' FORMAT ITEM.

COMPILER RESTRICTION. THE NUMBER OF DIGITS AFTER THE DECIMAL POINT IN AN 'E' OR 'F' FORMAT ITEM IS GREATER THAN N. N IS ASSUMED.

IEL0850I E INVALID 'E' OR 'F' FORMAT ITEM.

THE NUMBER OF DIGITS AFTER THE DECIMAL POINT IN 'E' OR 'F' FORMAT ITEM IS NEGATIVE. ZERO IS ASSUMED.

IEL0851I E CONFLICTING OPTIONS. T IGNORED.

CONFLICTING OPTIONS IN 'PUT' STATEMENT. OPTION T IGNORED.

Example:

```
     PUT FILE (A) SKIP (3) LINE (7);
```

Explanation: The only legal combination of PAGE, SKIP, and LINE is PAGE and LINE.

IEL0852I E T VALID ONLY FOR PRINT FILES.

T OPTION VALID ONLY FOR PRINT FILES. OPTION IGNORED.

Example:

```
     DCL A FILE STREAM;
     OPEN FILE (A) OUTPUT;
     PUT FILE (A) LINE (3) LIST (B,C);
```

Explanation: The options LINE and PAGE are allowed only on statements referring to STREAM OUTPUT PRINT files.

IEL0853I W D NOT ARITHMETIC OR STRING.

DATA LIST ITEM D NOT ARITHMETIC OR STRING. ITEM IGNORED.

Example:

```
     LAB2: A = B + C;
     GET LIST (D,E,LAB2);
```

Explanation: Elements in a data list must have arithmetic or string data type, that is they must be problem data.

IEL0854I S CONSTANT IN 'GET' OR 'PUT' DATA LIST.

CONSTANT INVALID IN DATA LIST IN 'GET' OR 'PUT' 'DATA' STATEMENT. DATA ITEM DELETED.

Example:

```
     PUT DATA(3);
```

IEL0855I S INVALID STRING OPTION.

'STRING' OPTION DOES NOT CONTAIN A CHARACTER STRING VARIABLE. STATEMENT IGNORED.

Example:

```
     DCL A FIXED BIN;
     PUT STRING(A) LIST(B,C,D);
```

Explanation: The variable referred to by the STRING option must be a character string variable.

IEL0856I E NO DATA ITEM IN FORMAT LIST.

NO DATA FORMAT ITEM IN FORMAT LIST. FORMAT LIST WILL BE USED ONLY ONCE. DATA LIST IGNORED.

Explanation: Data items cannot be transmitted unless a data format item is given in the format list. No assumptions are made.

Example:

```
     PUT EDIT(A) (X(4));
```

IEL0857I S CONTROL FORMAT ITEM(S) INVALID WITH 'STRING' OPTION.

INVALID CONTROL FORMAT ITEM(S) IN 'GET' OR 'PUT' STATEMENT WITH 'STRING' OPTION. FORMAT ITEM(S) IGNORED.

Example:

```
          PUT STRING(NAME) EDIT(A,B)(F(5),SKIP,F(5));
```

Explanation: Control format items SKIP, LINE, PAGE, and COLUMN are not permitted in GET STRING or PUT STRING statements.

IEL0858I S INVALID 'A' OR 'B' FORMAT ITEM.

INVALID 'A' OR 'B' FORMAT ITEM IN 'GET' STATEMENT. 'A(1)' OR 'B(1)' ASSUMED.

Example:

```
          GET EDIT(CHAR1,CHAR2) (A);
```

Explanation: An A-format item must be specified with an explicit width when used in GET statements.

IEL0859I W 'A' OR 'B' FORMAT ITEM INVALID IF USED BY 'GET'.

'A(1)' OR 'B(1)' ASSUMED FOR 'A' OR 'B' FORMAT ITEM IF FORMAT LIST IS USED BY A 'GET' STATEMENT.

Example:

```
          F:  FORMAT (A);
              GET EDIT(CHAR) (R(F));
```

Explanation: An A-format or B-format item must be specified with an explicit width when used in GET statements.

IEL0860I E WIDTH IN FORMAT ITEM EXCEEDS N.

COMPILER RESTRICTION. WIDTH IN FORMAT ITEM IS GREATER THAN N. N ASSUMED.

Example:

```
          PUT EDIT(A,B) (F(5),X(43000),F(5));
```

Explanation: An A-format item cannot have a width that is greater than 32767.

IEL0861I W 'E' FORMAT ITEM WIDTH TOO SMALL FOR NEGATIVE VALUES.

'E' FORMAT ITEM HAS WIDTH TOO SMALL FOR MINUS SIGN TO BE PRINTED.

Example:

```
          A = -3.57;
          PUT EDIT(A) (E(8,2,3));
```

In this example, the resultant output should be '-3.57E+00' which is nine characters whereas the width allowed in the PUT statement is only eight characters. This will cause the minus sign to be lost.

IEL0862I S 'E' FORMAT ITEM WIDTH TOO SMALL FOR DATA.

'E' FORMAT ITEM HAS WIDTH TOO SMALL FOR COMPLETE OUTPUT OF THE ITEM. ITEM IGNORED.

Example:

```
     A = -3.57;
     PUT EDIT(A) (E(7,2,3));
```

In this example, the resultant output should be '-3.57E+00' which is nine characters whereas the width allowed in the PUT statement is only seven characters. This would cause the minus sign and the most significant digit to be lost; therefore, the complete item is ignored..

IEL0863I S INVALID ARGUMENT TO 'E' OR 'F' FORMAT ITEM.

INVALID ARGUMENTS TO 'E' OR 'F' FORMAT ITEM. FORMAT ITEM IGNORED.

Example:

```
     PUT EDIT(A) (E(8,4,3));
```

IEL0864I E 'PAGE' OR 'LINE' IN 'GET' OR 'PUT' IGNORED.

INVALID 'PAGE' OR 'LINE' FORMAT ITEM IN 'GET' OR 'PUT' STATEMENT. FORMAT ITEM IGNORED.

IEL0865I W 'PAGE' OR 'LINE' IGNORED FOR 'GET'.

'PAGE' OR 'LINE' FORMAT ITEM WILL BE IGNORED IF FORMAT LIST IS USED BY A 'GET' STATEMENT.

IEL0866I S SECOND ARGUMENT TO T INVALID.

SECOND ARGUMENT OF BUILTIN FUNCTION T TOO LARGE OR TOO SMALL. VALUE ONE ASSUMED.

Example:

```
     DCL A(3,4);
         I = HBOUND(A,3);
```

IEL0867I S INVALID ARGUMENT TO 'ALLOCATION' FUNCTION.

ARGUMENT TO 'ALLOCATION' BUILTIN FUNCTION NOT LEVEL ONE 'CONTROLLED'. FUNCTION RETURNS ZERO VALUE.

Example:

```
     a.  DCL A AUTOMATIC;
              I = ALLOCATION(A);
```

```
b. DCL 1 B, 2 C, 2 D;
        I = ALLOCATION(C);
```

Explanation: The argument to the ALLOCATION built-in function must be a level-1 controlled variable.

IEL0868I S INVALID ARGUMENT TO T.

INVALID ARGUMENT TO BUILTIN FUNCTION T. FUNCTION RETURNS NULL VALUE.

Example:

```
DCL A FIXED,
    C AREA,
    D PTR,
    E OFFSET;
D=POINTER(A,C);(1st argument invalid)
D= POINTER(E,A);(2nd argument invalid)
E= OFFSET(A,C); (1st argument invalid)
E= OFFSET(D,A); (2nd argument invalid)
```

IEL0869I S ARGUMENT TO T IS NOT A FILE.

ARGUMENT TO BUILTIN FUNCTION T IS NOT A FILE. FUNCTION RETURNS ZERO VALUE.

Example:

```
DCL X FLOAT,
    I FIXED;
    I = COUNT(X);
```

IEL0870I S T USED AS ARGUMENT DOES NOT MATCH PARAMETER DESCRIPTOR.

BUILTIN FUNCTION T USED AS ARGUMENT DOES NOT MATCH CORRESPONDING PARAMETER DESCRIPTOR. RESULTS OF EXECUTION UNDEFINED.

Example:

```
DCL SIN BUILTIN,
    X ENTRY(ENTRY(FIXED));
CALL X(SIN);
```

Explanation: In the above example, the declaration of X is incorrect. X should be declared ENTRY(FLOAT...) where "..." is the precision and/or the mode. This message also applies to the declaration of a parameter for an internal procedure.

IEL0871I I FIXED POINT ARITHMETIC USED FOR T RESULT.

RESULT OF BUILTIN FUNCTION T WILL BE EVALUATED USING FIXED POINT ARITHMETIC OPERATIONS.

Explanation: This message describes a difference between the optimizing compiler implementation and that of the

PL/I (F) Compiler. The F compiler converts the arguments of the SUM or PROD built-in functions to floating-point in all cases.

IEL0872I W 'ADDR' BUILTIN FUNCTION POINTS AT STRING LENGTH FIELD.

'ADDR' BUILTIN FUNCTION RETURNS A POINTER TO THE TWO-BYTE LENGTH FIELD PRECEDING THE VARYING STRING VALUE.

IEL0873I S INVALID FORMAT ITEM IGNORED.

INVALID DATA TYPE IN FORMAT ITEM. ITEM IGNORED.

Example:

```
     DCL F FILE;
     PUT EDIT (A) (A(F));
```

Explanation: Fields in format items are converted to fixed binary. Unless the field specification is arithmetic or string, this conversion cannot take place.

Programmer Response: Change the specification of the format item.

|IEL0874I E INVALID 'SKIP' OR 'LINE' OPTION.

INVALID DATA TYPE IN 'SKIP' OR 'LINE' OPTION. VALUE ONE ASSUMED.

Example:

```
   DCL F FILE;
   PUT SKIP(F);
```

Explanation: The expression in a SKIP or LINE option must be convertible to a fixed decimal integer.

IEL0886I E SECOND ARGUMENT TO 'SUBSTR' SET TO ONE.

SECOND ARGUMENT OF BUILTIN FUNCTION OR PSEUDO-VARIABLE 'SUBSTR' LESS THAN ONE. VALUE SET TO ONE.

Example:

```
     SUBSTRING = SUBSTR(STRING,0,J);
```

Explanation: The second argument of the SUBSTR built-in function must be greater than or equal to 1.

IEL0887I E SECOND ARGUMENT TO 'SUBSTR' TOO LARGE.

SECOND ARGUMENT OF BUILTIN FUNCTION OR PSEUDO-VARIABLE 'SUBSTR' GREATER THAN STRING LENGTH. NULL STRING RETURNED.

Example:

```
     DCL STRING CHAR(6);
     SUBSTRING = (SUBSTR(STRING,7,J);
```

Explanation: The value of the second argument of the SUBSTR built-in function must be less than or equal to the length of the string in the first argument.

IEL0888I E THIRD ARGUMENT TO 'SUBSTR' NEGATIVE.

THIRD ARGUMENT OF BUILTIN FUNCTION OR PSEUDO-VARIABLE 'SUBSTR' NEGATIVE. NULL STRING RETURNED.

Example:

```
     SUBSTRING = SUBSTR(STRING,I,-1);
```

Explanation: The third argument of the SUBSTR built-in function must be greater than or equal to zero.

IEL0889I E THIRD ARGUMENT TO 'SUBSTR' TOO LARGE.

THIRD ARGUMENT OF BUILTIN FUNCTION OR PSEUDO-VARIABLE 'SUBSTR' GREATER THAN STRING LENGTH. RETURNED VALUE TRUNCATED AT END OF SOURCE STRING.

Example:

```
     DCL STRING CHAR(6);
     SUBSTRING = SUBSTR(STRING,I,7);
```

Explanation: The third argument of the SUBSTR built-in function must be less than or equal to the length of the string in the first argument.

IEL0890I E ARGUMENTS TO 'SUBSTR' TOO LARGE.

THE SUM OF THE SECOND AND THIRD ARGUMENTS OF BUILTIN FUNCTION OR PSEUDO-VARIABLE 'SUBSTR' IS GREATER THAN THE STRING LENGTH PLUS ONE. RETURNED VALUE TRUNCATED AT END OF SOURCE STRING.

Example:

```
     DCL STRING CHAR(6);
     SUBSTRING = SUBSTR(STRING,6,2);
```

Explanation: The value of the first argument plus the value of the second argument, less one, must be less than or equal to the length of the string in the first argument.

IEL0891I W RESULT OF BIT STRING OPERATION WILL BE TRUNCATED.

RESULT OF BIT STRING OPERATION WILL BE TRUNCATED.

Example:

```
DCL A BIT(10),
   B BIT(20),
   C BIT(15);
A = B&C;
```

IEL0892I W RESULT OF STRING OPERATION TRUNCATED.

TARGET STRING SHORTER THAN SOURCE. RESULT TRUNCATED ON ASSIGNMENT.

Example:

```
DCL B1 BIT(5),
    (B2,B3) BIT(7)
    B1 = B2;
```

Explanation: This message warns of a possible error caused by the loss of truncated bits when the assignment takes place. If the STRINGSIZE condition is enabled, the condition will not be raised at execution time.

IEL0903I S INVALID ARGUMENT TO 'HIGH' OR 'LOW' REPLACED BY '(1)'.

INVALID ARGUMENT TO 'HIGH' OR 'LOW' BUILTIN FUNCTION. '(1)' ASSUMED.

IEL0904I S OPERATOR(S) INVALID FOR 'COMPLEX' DATA.

OPERATOR(S) INVALID FOR 'COMPLEX' DATA. '=' ASSUMED.

Example:

```
DCL (A,B) COMPLEX;
    IF A > B THEN GOTO...;
```

Explanation: Operators permitted for use with complex data are limited to '=' and '¬=' (equals and not-equals) operators.

IEL0905I S EXPRESSION IN 'INITIAL' FOR STATIC VARIABLE D.

SPECIFICATION OF 'INITIAL' ATTRIBUTE FOR STATIC VARIABLE D CONTAINS EXPRESSION. 'INITIAL' ATTRIBUTE IGNORED.

Example:

```
          DCL A STATIC COMPLEX INITIAL(3+4I+5I);
```

IEL0906I I CONVERSION WILL BE DONE BY SUBROUTINE CALL.

DATA CONVERSION WILL BE DONE BY SUBROUTINE CALL.

Explanation: This message informs that the program contains one or more conversions that will require a PL/I library subroutine. Its purpose is to indicate where the program might be made more efficient if it can be recoded to ensure that the conversion is performed more efficiently by compiler-generated instructions.

IEL0907I S WRONG NUMBER OF ARGUMENTS FOR ENTRY D.

WRONG NUMBER OF ARGUMENTS SPECIFIED FOR FUNCTION OR CALL D. RESULTS OF EXECUTION UNDEFINED.

Example:

```
        DCL P ENTRY(FLOAT,FLOAT) EXTERNAL;

    1.  CALL P(A);

    2.  A = P(A,A,A);
```

Explanation: A procedure has been referenced with a number of arguments different from the number in the parameter descriptor.

Programmer Response: Correct the source.

| IEL0908I W 'RETURN' EXPRESSION MAY CONFLICT WITH ENTRY SPECIFICATION.

DATA TYPE OF RETURNED EXPRESSION CONFLICTS WITH 'RETURNS' OPTION OF AN ENTRY SPECIFICATION IN THIS BLOCK. 'RETURN' EXPRESSION WILL BE IGNORED IF THE INVALID COMBINATION OF 'RETURN' AND ENTRY IS USED.

Example:

```
    P:  PROC RETURNS(FILE);
    E:  ENTRY RETURNS(DEC FLOAT);
        DCL F DEC FLOAT,
            G FILE;
        RETURN(F);
        RETURN(G);
```

In this example, the first RETURN statement conflicts with the PROC statement and will be treated as a RETURN without an expression if executed during an invocation of P. Similarly, the expression in the second RETURN statement will be ignored if executed during an invocation of E.

IEL0909I I DATA VARIABLE USED FOR PROGRAM CONTROL.

BASED REFERENCE TO PROGRAM CONTROL DATA REFERS TO STORAGE USED BY VARIABLE D BUT IS ACCEPTED AS VALID.

Example:

```
        DCL BLV LABEL BASED,
            P POINTER;
            P->BLV = LABCON
            .
            .

    1.  GOTO P->BLV (valid)
        .
        .
```

2. GOTO ADDR(FLOAT)->BLV
   (invalid but allowed)

Explanation: The global* optimization process must include analysis of all possible values of label variables, entry variables, and pointers in the program before it can attempt to perform move-out and strength-reduction. During the process, the second condition in the example above would be detected. This condition would restrict the global* optimization process, since this process cannot detect all the possible label constant values that might be assigned to FLOAT.

*Global optimization is defined in the explanation for IEL0910I.

IEL0910I W TOO MANY CALLS AND FUNCTION REFERENCES FOR OPTIMIZATION.

COMPILER RESTRICTION. TOO MANY 'CALL' STATEMENTS AND FUNCTION REFERENCES. OPTIMIZATION IS INHIBITED FOR THE PROGRAM.

Explanation: A program that is to be compiled with full optimization (with OPT(TIME) specified) has so many branches of control between blocks that the capacity of the compiler to analyze them has been exceeded. The compilation is completed without global optimization; some local optimization may have been performed. In this context, local optimization includes such things as the in-line simplification of calculations such as I*3 and A**4, and the matching of data items with format items in edit I/O. Conversely, global optimization is concerned with the commoning of expressions, the moving of invariant expressions from loops, and further simplification of expressions. If full global optimization is performed, then any or all of the types of optimization may be carried out, either within or between flow units (logical divisions of the PL/I source program). If certain compiler limitations are exceeded, then global optimization is restricted to expression commoning alone. Furthermore, this is performed solely within flow units.

The compiler permits up to 256 separate CALL statements and function references involving both entry constants and entry variables. This limit includes each entry constant or entry variable passed as an argument to an external procedure. A further limit of 2048 exists for all possible assignments of entry constants to entry variables.

Programmer Response: To obtain global optimization it is necessary to simplify the program's structure so that the number of branches between begin blocks and between internal procedures is kept within the limits described above.

IEL0911I W TOO MANY LOCATOR LABEL OR ENTRY ASSIGNMENTS FOR OPTIMIZATION.

COMPILER RESTRICTION. TOO MANY LOCATOR LABEL OR ENTRY VARIABLE ASSIGNMENTS. OPTIMIZATION IS INHIBITED FOR THE PROGRAM.

Explanation: A program that is to be compiled with global* optimization (with OPT(TIME) specified) has so many locator and entry variable assignments that the capacity of the compiler to analyze them has been exceeded. The compilation is completed without global optimization.

The compiler permits up to 1360 locator, label, or entry variable assignments without inhibiting optimization.

* Local and global optimization are defined in the explanation for IEL0910I.

Programmer Response: To obtain global optimization it is necessary to reduce the number of locator and entry variable assignments that appear in the source program.

IEL0912I W TOO MANY BASED LOCATOR LABEL OR ENTRY ASSIGNMENTS FOR OPTIMIZATION.

COMPILER RESTRICTION. TOO MANY ASSIGNMENTS WITH BASED LOCATORS LABEL OR ENTRY VARIABLES. OPTIMIZATION IS INHIBITED FOR THE PROGRAM.

Explanation: A program that is to be compiled with global* optimization (with OPT(TIME) specified) has so many based locator, based label, and based entry variable assignments that the capacity of the compiler to analyze them has been exceeded. The compilation is completed without global optimization.

The compiler permits up to 680 based locator, label or entry variable assignments without inhibiting optimization.

* Local and global optimization are defined in the explanation for IEL0910I.

Programmer Response: To obtain global optimization it is necessary to reduce the number of based locator, based label, and based entry variable assignments that appear in the source program.

IEL0913I W TOO MANY LOCATOR TEMPORARIES ACTIVE FOR OPTIMIZATION.

COMPILER RESTRICTION. TOO MANY LOCATOR TEMPORARIES ACTIVE. OPTIMIZATION IS INHIBITED FOR THE PROGRAM.

Explanation: The compiler creates a 'locator temporary' for functions that return locator values. A program that is to be compiled with global* optimization (with OPT(TIME) specified) has so many of these locator temporaries created that the capacity of the compiler to analyze them has been exceeded. The compilation is completed without global optimization.

The compiler permits up to 10 locator temporaries without inhibiting optimization.

* Local and global optimization are defined in the explanation for IEL0910I.

Programmer Response: To obtain global optimization it is necessary to reduce the number of locator values returned by functions that appear in the source program.

IEL0914I W STATEMENT MAY NEVER BE EXECUTED.

STATEMENT MAY NEVER BE EXECUTED.

Example:

```
          P:  PROCEDURE;
          DCL X, Y CHAR(1);
          IF X='2' THEN Y='';
                   ELSE GOTO L2;
          GOTO L2;
       L1:  A=5;
       L2:  B=6;
          END P;
```

In this example, the message will be produced for the statement labelled L1, since there is no possibility of control being transferred to it.

The statement "GOTO L2;" in this example can be executed. However, the optimization process has modified the THEN clause to branch directly to the label constant L2 rather than to the statement following the ELSE clause. The message is then produced and the redundant statement is eliminated.

Explanation: This message warns that the compiler has detected a statement that can never be executed as the flow of control must always pass by it.

IEL0915I W TOO MANY STATEMENT LABEL CONSTANTS FOR OPTIMIZATION.

COMPILER RESTRICTION. TOO MANY STATEMENT LABEL CONSTANTS. OPTIMIZATION IS INHIBITED FOR THE PROGRAM.

Explanation: A program that is to be compiled with global* optimization (with OPT(TIME) specified) has so many statement label constants that the capacity of the compiler to analyze them has been exceeded. The compilation is completed without global optimization.

The compiler permits up to 2048 statement label constants without inhibiting optimization.

* Local and global optimization are defined in the explanation for IEL0910I.

Programmer Response: To obtain global optimization it is necessary to reduce the number of statement label constants used in the source program.

IEL0916I W ITEM(S) D MAY BE UNINITIALIZED.

ITEM(S) D MAY BE UNINITIALIZED WHEN USED IN THIS BLOCK.

Example:

```
Y:  PROC;
    DCL X;
    Z=X;
    END Y;
```

Explanation: This message refers only to variables declared within the block. The flow-analysis stage of optimization checks all possible flow-paths through a program although many of the possible flow-paths may never be used. In doing so, the flow analysis determines flow-paths originating from statements prefixed by label constants that can be branched to from on-units, as well as those that originate from PROCEDURE and ENTRY statements.

It is possible, therefore, that this message is produced for items that are initialized correctly for the flowpaths that will actually be used owing to the presence of other flow paths that will never be used. This is aggravated by the necessity to consider label constants as external entry points. In the following example, an on-unit returns control to a block by means of a GOTO statement. The variable X is detected as uninitialized if the block is entered through the label constant Y, although it may have been initialized before the on-unit is entered.

```
P:  PROC
    X = 100;
    ON OFL GOTO Y;
    .
    .
    .
Y:  A = X;
    .
    .
```

The final value assigned to a static variable in one invocation of a procedure will be the 'initial' value of that variable in a subsequent invocation of that procedure.

IEL0917I W   N FLOW UNITS IN BLOCK. GLOBAL OPTIMIZATION RESTRICTED.

BLOCK CONTAINS N FLOW UNITS. GLOBAL OPTIMIZATION PERFORMED ONLY IN DO GROUPS.

Explanation: The block has been split into flow units for the purposes of global* optimization. However, the compiler limit of 255 flow units in a block has been exceeded, and consequently, global optimization is restricted. Before scanning to the next block, the compiler looks for do-groups in the current block, in the hope that flow analysis (and full global optimization) may be completed for these.

* Local and global optimization are defined in the explanation for IEL0910I.

Programmer Response: If full optimization is required for the block, either simplify the flow of control within the block, or divide the block into two or more simpler blocks.

IEL0918I W GO TO D MAY PASS CONTROL OUT OF BLOCK.

GO TO D MAY CAUSE CONTROL TO BE PASSED OUT OF THE CURRENT BLOCK.

Explanation: D is a label variable declared to be STATIC and INITIAL. Since the initialization is done at compile time, no environment information can be supplied to the label variable; since it has been detected that control may be passed out of the current block, the GOTO is executed by the library. This will cause an execution-time error. If this message appears, message IEL0580I (severity E) will have been produced for the specified label variable.

Programmer Response: Redeclare the LABEL variable as AUTOMATIC.

IEL0919I W N VARIABLES IN PROGRAM. GLOBAL OPTIMIZATION RESTRICTED.

N VARIABLES IN PROGRAM. GLOBAL OPTIMIZATION PERFORMED FOR 255 VARIABLES. LOCAL OPTIMIZATION PERFORMED ON REMAINDER.

Explanation: The compiler will consider 255 variables in the program for global* optimization. The remainder are considered solely for local* optimization.

Explicitly declared variables will be considered for global optimization in preference to contextually declared variables, and the latter will in turn be considered in preference to implicitly declared variables. Furthermore, the highest preference will be given to those variables declared in the final DECLARE statements in the outermost block.

If the program contains more than 255 variables, most benefit will be obtained from the global optimization of arithmetic variables, particularly DO loop control variables and subscripting variables. Little or no benefit will be gained from the optimization of string variables or program control data.

Arithmetic variables should not, therefore, be implicitly declared but should be declared in the final DECLARE statements in the outermost block. Further benefits may be obtained if declared but unreferenced variables are eliminated from the program.

* Local and global optimization are defined in the explanation for IEL0910I.

IEL0920I W N FLOW UNITS IN DO GROUP. GLOBAL OPTIMIZATION RESTRICTED.

DO GROUP CONTAINS N FLOW UNITS. GLOBAL OPTIMIZATION IS RESTRICTED.

Explanation: The compiler limit of 255 flow units in a do-group has been exceeded and full global* optimization is inhibited within the group. Partial global optimization will be performed for flow units within the group.

* Local and global optimization are defined in the explanation for IEL0910I.

IEL0921I E LESS THAN N CHARACTERS OF T IN D PRINTED.

QUALIFIED NAME OF ELEMENT T OF STRUCTURE D WILL BE TRUNCATED TO LESS THAN N CHARACTERS IN DATA DIRECTED I/O.

Example:

```
     PUT DATA (PAYROLL);
```

where PAYROLL is declared as a base element of a structure which when fully qualified exceeds 255 characters, including periods.

IEL0923I E D INVALID TYPE IN DATA LIST FOR DATA DIRECTED I/O OR CHECK.

COMPILER RESTRICTION. TYPE OF 'BASED' VARIABLE D IN DATA LIST NOT SUPPORTED FOR DATA DIRECTED I/O OR CHECK. ITEM IGNORED.

Example:

```
    DCL 1 STR BASED(P),
          2 LEN FIXED BIN,
          2 TITLE CHAR(N REFER(LEN));
    PUT DATA(TITLE);
```

Explanation: The optimizing compiler does not permit PUT DATA and GET DATA statements or the CHECK prefix option on certain types of based variables. See the chapter 'Stream-Oriented Transmission' in the language reference manual for details.

IEL0924I E D INVALID TYPE IN DATA LIST FOR DATA DIRECTED I/O OR CHECK.

COMPILER RESTRICTION. TYPE OF 'DEFINED' VARIABLE D IN DATA LIST NOT SUPPORTED FOR DATA DIRECTED I/O OR CHECK. ITEM IGNORED.

Example:

```
     DCL A CHAR(100),
         B CHAR(10) DEF A POS(N);
     PUT DATA(B);
     DCL C(100,100) CHAR(1),
         D(10,10) CHAR(1) DEF C(1SUB,2SUB);
     PUT DATA(D);
     DCL E CTL,
         F DEF E;
     PUT DATA(F);
```

Explanation: The optimizing compiler does not permit the transmission of the following types of defined variables by means of the PUT DATA statement or the CHECK prefix option:

1. A string-overlay defined item.

2. An iSUB-defined item.

3. An item defined on a controlled base variable.

IEL0925I W GLOBAL OPTIMIZATION RESTRICTED.

FLOW WITHIN BLOCK OR DO GROUP IS TOO COMPLEX. GLOBAL OPTIMIZATION IS RESTRICTED.

Explanation: The block or do-group has been split into flow units for the purposes of flow analysis and global* optimization. However, the compiler limit of 1024 connections between flow units has been exceeded, and consequently, global optimization has been restricted within the block or group. Partial global optimization will be performed for flow units within the block or group.

* Local and global optimization are defined in the explanation for IEL0910I.

IEL0926I S 'SIZE' RAISED WHEN CONVERTING CONSTANT [TO D].

'SIZE' CONDITION RAISED WHEN CONVERTING CONSTANT [TO D]. RESULT OF CONVERSION UNDEFINED.

Example:

```
DCL A FIXED DECIMAL(2,0);
    A = 999;
```

Explanation: A constant converted at compile-time has raised the 'SIZE' condition.

IEL0927I S 'CONVERSION' RAISED WHEN CONVERTING CONSTANT [TO D].

'CONVERSION' CONDITION RAISED WHEN CONVERTING CONSTANT [TO D]. RESULT OF CONVERSION UNDEFINED.

Example:

```
READ FILE(BERT) IGNORE('JACK AND JIM');
```

Explanation: The IGNORE option should refer to an arithmetic integer value.

IEL0931I E LENGTH OF D EXCEEDS LENGTH OF 'DEFINED' BASE.

LENGTH OF VARIABLE D EXCEEDS LENGTH OF VARIABLE ON WHICH IT IS DEFINED. THIS DEFINING HAS BEEN ACCEPTED.

Example:

```
DCL A CHAR(6);
DCL B CHAR(10) DEF A;
```

Explanation: The optimizing compiler will accept this invalid form of defining to permit execution of programs that require it. However, it is possible that an assignment to the defined item will cause storage to be overwritten and an unpredictable error to occur.

Programmer Response: If this defining is required, check that any conditional link-editing and execution steps will not be inhibited.

IEL0932I S AGGREGATE DESCRIPTOR FOR D TOO LARGE.

COMPILER RESTRICTION. AGGREGATE DESCRIPTOR FOR D TOO LARGE. RESULTS OF EXECUTION UNDEFINED.

Explanation: An aggregate descriptor is a control block created by the compiler to handle the addressing of the base elements in an aggregate. Its format is described in the execution logic manual for this compiler. A large number of base elements in a large aggregate has caused the aggregate descriptor to exceed 4095 bytes, the limit of its internal addressability.

IEL0933I W D INVALID TYPE FOR DATA DIRECTED I/O OR CHECK.

COMPILER RESTRICTION. TYPE OF 'BASED' VARIABLE D NOT SUPPORTED FOR DATA DIRECTED I/O OR CHECK. ITEM IGNORED.

Example:

```
     DCL 1 STR BASED(P),
              2 LEN FIXED BIN,
              2 TITLE CHAR(N REFER(LEN));
     PUT DATA;
```

Explanation: The optimizing compiler does not permit PUT DATA and GET DATA statements or the CHECK prefix option on certain types of based variables. See the chapter 'Stream-Oriented Transmission' in the language reference manual for details.

IEL0934I W D INVALID TYPE FOR DATA DIRECTED I/O OR CHECK.

COMPILER RESTRICTION. TYPE OF 'DEFINED' VARIABLE D NOT SUPPORTED FOR DATA DIRECTED I/O OR CHECK. ITEM IGNORED.

Example:

```
    DCL A CHAR(100), B CHAR(10) DEF A POS(N);
    DCL C(100,100)CHAR(1);
    DCL D(10,10) CHAR(1) DEF C(1SUB,2SUB);
    DCL E CTL,F DEF E;
    PUT DATA;
```

Explanation: The optimizing compiler does not permit the transmission of the following types of defined variables by means of the PUT DATA statement or the CHECK prefix option:

1. A string-overlay defined item.

2. An iSUB-defined item.

3. An item defined on a controlled base variable.

IEL0940I W   T MAY INCREASE EXECUTION TIME.

T CONFLICTS WITH THE OPTIMIZE OPTION. EXECUTION TIME MAY BE INCREASED.

Explanation: The FLOW option and the CHECK, STRINGRANGE, SUBSCRIPTRANGE, SIZE, and STRINGSIZE conditions are program debugging aids causing many extra machine instructions to be generated and executed; their use is inconsistent with the use of the OPT(TIME) option, which specifies that the compiler is to optimize the generated machine instructions in order that a very efficient program may be produced.

Programmer Response: Remove the FLOW option and/or conditions if the full benefit of optimization is to be obtained.

IEL0960I W   GENERATED EXTERNAL NAMES MAY BE AMBIGUOUS.

COMPILER GENERATED EXTERNAL NAMES MAY BE AMBIGUOUS IF THE PROGRAM IS LINK-EDITED WITH A PROCEDURE OF SIMILAR NAME.

Explanation: The optimizing compiler will generate names for internal controlled variables and internal files, if used. These names are processed by the linkage editor. If two external PL/I procedures with similar names, such as ATESTER and BTESTER are to be link-edited together, it is possible for both procedures to have the name /TESTER1 generated for them. However, this cannot occur unless both procedures have at least 36 generated names each.

IEL0961I S   STATEMENT TOO LARGE.

COMPILER RESTRICTION. STATEMENT TOO LARGE. RESULTS OF EXECUTION UNDEFINED.

Explanation: The size of the statement may force the compiler to generate a set of instructions that exceeds 4096 bytes of storage. The use of an RX branch instruction does not permit an offset that exceeds 4096. Consequently execution of the statement may produce unpredictable errors.

Programmer Response: Divide the statement into two or more smaller statements.

IEL0966I E   EXTERNAL NAME D EXCEEDS N CHARACTERS.

COMPILER RESTRICTION. EXTERNAL NAME D TOO LONG. NAME SHORTENED TO N CHARACTERS.

Example:

```
     DCL ABCDEFGHI FILE...;
```

Explanation: Since external identifiers in PL/I are resolved by the linkage editor, it follows that such names should not exceed the limit imposed by the linkage editor on the length of names. The method of truncation used by the compiler will, in many cases, create unique identifiers so that the compilation can continue, and linkage editing and execution may be successful.

|IEL0967I W D TRUNCATED TO N CHARACTERS.

| COMPILER RESTRICTION. EXTERNAL ENTRY NAME D WITH
| INTERLANGUAGE OPTION IS TOO LONG. NAME TRUNCATED
| TO FIRST N CHARACTERS.

Explanation: In PL/I the usual method of truncating external names is to concatenate the first four and last three characters to form a seven-character identifier.
| External names for COBOL, FORTRAN, and ASSEMBLER routines can be up to eight characters in length, and any truncation of names of greater length than this involves the removal of the excess characters. To permit inter-language communication, PL/I adopts this technique for identifiers that are associated with COBOL, FORTRAN, or
| ASSEMBLER routines.

IEL0969I E NO LABEL ON 'FORMAT' STATEMENT.

'FORMAT' STATEMENT HAS NO LABEL. STATEMENT IGNORED.

Example:

```
     F: ;
     FORMAT(A);
```

IEL0970I U COMPILER CANNOT PROCEED. ERROR N IN PHASE P. CORRECT SOURCE AND RE-COMPILE.

COMPILER ERROR NUMBER IN PHASE P. COMPILER UNABLE TO PROCEED. CORRECTION OF SOURCE ERRORS MAY LEAD TO SUCCESFUL COMPILATION.

Explanation: Errors have prevented succesful compilation.

Programmer Response: Correct the errors indicated by other messages (if any) and recompile the program. If there are no other messages, take the action suggested for message IEL0230I.

IEL0971I W FIRST USE OF OPTION T FOR FILE D IGNORED.

ENVIRONMENT OPTION T SPECIFIED MORE THAN ONCE IN DECLARATION OF FILE D. FIRST USE IGNORED.

Example:

```
     DCL A FILE ENV (RECSIZE(20)RECSIZE(20));
```

|IEL0990 E 'PASSWORD' ENVIRONMENT OPTION SPECIFIED WITHOUT 'VSAM'.
|
| 'PASSWORD' ENVIRONMENT OPTION SPECIFIED WITHOUT 'VSAM'

ENVIRONMENT OPTION IN FILE D. 'VSAM' ASSUMED.

Explanation: A password can be declared only for a VSAM file.

Part II: Execution-time(IBM) Messages

Execution-time messages are produced by the PL/I Transient Library and are printed on the device associated with the file SYSPRINT. If SYSPRINT is unusable the message will be printed on the operator's console. However, any message associated with the system action for the CHECK condition or the COPY or SNAP options will not be transmitted to the operator's console.

Messages are printed at execution time when:

1. An error occurs for which there is no specific on-condition in PL/I. (A message is printed and the ERROR condition is raised).

2. An on-condition is raised for which there is no on-unit in the program and for which the standard system action includes printing an execution-time message.

Format of Execution-time Messages

Execution-time messages have the following format:

```
 message-number - ONCODE=nnnn  'condition-name' CONDITION RAISED - text
- location message
```

Each message number is of the form IBMnnn, where "IBM" indicates that the message is a PL/I execution-time message, and "nnn" is the number of the message. The final character "I" or "A" indicates to the operator whether the message is informatory or whether he should take some action.

The on-code is that for the condition name that follows. The text "ONCODE=" and the actual on-code generated by the system only appear when the message is printed; they are not listed in the message texts in this publication.

The condition name is the PL/I on-condition raised in association with the exceptional or error condition that caused the message to be printed. The condition name and the text "CONDITION RAISED" are not normally listed in the message texts in this publication, although they are always included when the message is printed. However, certain messages have no following main text or a main text such as "BY 'SIGNAL' STATEMENT" or "IN I/O STATEMENT", and for these messages the condition name and the text "CONDITION RAISED" form part of the message text in this publication.

The nature of the location message will depend on whether the PL/I program was compiled with the GOSTMT option or the NOGOSTMT option. If GOSTMT was specified, the location message will be as follows:

```
IN STATEMENT nnnn AT OFFSET nnnnnn IN PROCEDURE WITH ENTRY entry-name
```

If NOGOSTMT was specified, the location message will be as follows:

```
AT OFFSET nnnnnn IN PROCEDURE WITH ENTRY entry-name
```

The location messages generated by the system only appear when the messages are printed; they are not listed in the message texts in this publication.

An example of an execution-time message in the form in which it will be printed is given below:

```
IBM037I  ONCODE=0612 'CONVERSION' CONDITION RAISED CONVERSION FROM
         CHARACTER TO BIT ON INPUT AFTER 'TRANSMIT' DETECTED IN
         STATEMENT 207 AT OFFSET 004A8C IN PROCEDURE WITH ENTRY PROG1
```

Many messages are not self-explanatory because they are in a necessarily concise form, and an explanation, an example, or a programmer response, or a combination of any of these, has been added where necessary.

Symbols in Messages

Many of the messages reproduced in this publication contain symbols indicating where the system will insert information when it prints the message. The symbols used are:

ccc - A condition name.

eee - An entry name.

fff - A file name.

Before Calling IBM . . .

Unless the programmer response for the message specifies otherwise, before calling IBM for programming support with regard to an execution-time error:

1. Ensure that the program includes an active ERROR on-unit that includes the statement:

   ```
   CALL PLIDUMP('HB');
   ```

2. Run the program again.

If the problem recurs, ensure that the following are available:

1. Listings of the source program, the object program, and the job control statements for the execution of the program.
2. Relevant data sets.
3. Job stream (job control statements and data) in machine readable form.

The requirements for problem determination and APAR submission are given in the programmer's guide for this compiler.

IBM002I INTERRUPT DURING PL/I PROGRAM-MANAGEMENT LIBRARY ROUTINE. PROGRAM TERMINATED.

Explanation: An interrupt has occurred during the handling of a PL/I on-condition or error condition in the program management routine or a routine invoked by it. It indicates that a disastrous error has occurred during execution of the program, such as the overwriting of control blocks or sections of code. The program is terminated, and a return code of 4000 is produced. A dump is produced if there is a SYSUDUMP or SYSABEND DD statement for the job step. The most common cause of this type of error is the overwriting of control information by the PL/I program. Usually this is caused by arrays exceeding their bounds or by the SUBSTR function having invalid arguments. However, this message can occur if execution of a GOTO statement is attempted using a label variable which is uninitialized or has an invalid value.

Programmer Response: Refer to the chapter "Program Checkout" in the programmer's guide for this compiler for advice on how to tackle this type of error. If arrays are used in the program, recompile the program enabling SUBSCRIPTRANGE, and execute it again. Similarly, if SUBSTR is used, recompile the program enabling STRINGRANGE. (If both are used, enable both conditions.)

If a GOTO statement containing a label variable is used, enable CHECK for the label variable(s), and execute the program again.

IBM003I ERRONEOUS 'PARM' OPTION HAS BEEN IGNORED.

Example:

```
//A EXEC PGM=B,
//       PARM='USERNOSLASH'
```

Explanation: Either the PARM field passed at execution contained data which was not recognizable as supported keyword options, or user information was not separated from the PL/I information by the character '/'. The whole PARM field is passed to the main procedure unchanged.

Programmer Response: Correct the misspelt word, or add a slash between the PL/I and user data.

IBM005I TOO MANY FILES AND CONTROLLED VARIABLES.

Explanation: The total length of the pseudo-register vector for the program is more than 4096 bytes. Four bytes are used for each file constant, four bytes for each controlled variable, and four bytes for each fetched procedure.

IBM006I NO MAIN PROCEDURE, PROGRAM NOT EXECUTED.

Explanation: An attempt has been made to execute a program containing one or more external PL/I procedures, none of which has the MAIN option in its PROCEDURE statement.

Programmer Response: Ensure that the first external PL/I procedure to be invoked has the MAIN option in its PROCEDURE statement.

IBM008I NO MAIN STORAGE AVAILABLE.

Explanation: There is insufficient main storage for program execution to begin.

Programmer Response: If possible, arrange for the program to be executed in a larger partition or region of main storage. Otherwise, convert the program into a number of smaller phases with a root phase that remains in storage throughout execution and overlay phases that are loaded when required for execution and are overwritten by subsequent phases when no longer required. For further information, see the programmer's guide.

IBM009I A PL/I TASK HAS TERMINATED ABNORMALLY WITH (SYSTEM CODE=sss|USER CODE=nnnn). PROGRAM TERMINATED.

Explanation: A task has terminated abnormally without entering the usual termination routines.

IBM010I PROGRAM TERMINATED, SYSTEM CODE=sss|USER CODE=nnnn.

Explanation: A non-multitasking program has been terminated by the issue of an ABEND macro. If the code is sss (where s is a hexadecimal digit), the macro was issued by the control program. If the code is nnnn (where n is a decimal digit), the macro was issued by the problem program.

A PL/I library error-handling routine will attempt to put out this message on SYSPRINT if open; if not it will appear at the console. If a PLIDUMP DD statement is present, an attempt will be made to produce a PL/I dump. If insufficient main storage is available the message will not be put out and a system dump will be produced with the appropriate system completion code. A system completion code of 80A may indicate that there is not enough main storage available to produce the message, load the dump routines, and initialize program execution. The user-code form of this message can occur only if a user-written routine is called from PL/I and if this routine issues an ABEND macro.

IBM011I TASK taskname TERMINATED, SYSTEM CODE=sss|USER CODE=nnnn.

Explanation: The task has been terminated abnormally by the issue of an ABEND macro. If the code is sss (where s is a hexadecimal digit), the macro was issued by the control program. If the code is nnn (where n is a decimal digit), the macro was issued by the problem program. See also the explanation for message IBM010I.

IBM012I ISASIZE INSUFFICIENT. PROGRAM TERMINATED.

Explanation: The initial storage area (ISA) and 251 further segments of storage have been allocated. No further segments can be allocated. The program is terminated with a return code of 1000.

Programmer Response: Check that the program was not looping recursively. If the program is running correctly the ISASIZE parameter should be increased. The REPORT option can be used to determine the storage required.

IBM013I NO SUITABLE PLIDUMP DD CARD.

Explanation: The PLIDUMP routines could not open the PLIDUMP file because the PLIDUMP DD card for this job step was omitted or specified unsuitable options. No output is given by PLIDUMP under these conditions, but the options specified (or defaulted) are used to determine whether the program continues or terminates.

Programmer Response: Supply a suitable PLIDUMP DD card that specifies a sequential access method for output.

IBM014I TRANSIENT LIBRARY LEVEL LOWER THAN RESIDENT.

Explanation: One of the initialization modules has detected that the transient library in use comes from a release prior to that of the resident library. Compatibility is not guaranteed in these circumstances, and the program may fail when it is executed.

Programmer Response: Even if failure does not occur, it is advisable to install the latest version of the transient library and rerun the program; this message should not then appear.

IBM015I INVALID SPECIFICATION OF 'COUNT' OR 'FLOW' OPTION. OPTION IGNORED.

Example: If no compilation in the job has the 'COUNT' or 'FLOW' option, the following statement is in error:

```
DCL PLIXOPT STATIC CHAR(4) VAR EXT INIT('FLOW');
```

Explanation: COUNT and FLOW are only valid at execution time if one or the other was specified at compile time, and COUNT requires the resident library to be at a release the same as or greater than the current release of the compiler.

IBM016I PLIXOPT IS NOT A VALID EXECUTION-TIME OPTIONS STRING.

Example:

```
DCL PLIXOPT EXT INIT(12345);
```

Explanation: A static external variable named PLIXOPT has been found, but either it was not CHARACTER VARYING with current length less than or equal to 250, or it contained no valid initialization options.

Programmer Response: If the variable PLIXOPT in the source program was intended to set execution-time options, correct its attributes or contents. If not, change its name.

IBM017I INVALID OPTION IN PLIXOPT HAS BEEN IGNORED.

Example:

```
    DCL PLIXOPT CHAR(20) VAR INIT('FLOW(10,5),TEPORT') EXT;
```

Explanation: The options specified in the variable PLIXOPT contain one or more errors.

IBM018I ERROR IN IMPLICIT CLOSE.

Explanation: An error condition was detected during the implicit closing of a record I/O file. The possible error conditions are:

1. TRANSMIT (if last operation was a LOCATE, or if the file has the attributes REGIONAL, SEQUENTIAL, and OUTPUT).
2. I/O sequence error.
3. KEY(VSAM).

Programmer Response: Close the file explicitly to obtain normal condition handling.

IBM020I 'CONVERSION' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the 'CONVERSION' condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the CONVERSION condition in the program.

IBM021I UNKNOWN SOURCE ATTRIBUTES ON INPUT.

Explanation: The CONVERSION condition has been raised within a GET LIST or GET DATA statement with the FILE option. The attributes of the source data cannot be determined. For example:

```
   DCL (A,B) CHAR(14);
   GET LIST (A,B)
```

Where the input stream contains 'PIG'C, 'DOG',--- the condition will be raised when the first item is encountered. The value for ONSOURCE will be: "'PIG'C", and that for ONCHAR will be: "C".

IBM022I UNKNOWN SOURCE ATTRIBUTES ON INPUT AFTER 'TRANSMIT' DETECTED.

Explanation: The CONVERSION condition has been raised after an error has caused the TRANSMIT condition to be raised. For an explanation of the conversion error, see the explanation given for message IBM021I.

IBM023I UNKNOWN SOURCE ATTRIBUTES.

Explanation: The CONVERSION condition has been raised within a GET LIST STRING or GET DATA STRING statement. For an explanation of the conversion error, see the explanation given for message IBM021I.

IBM024I CONVERSION FROM F-FORMAT ON INPUT

Explanation: An invalid character has been detected in an F-format input field.

Programmer Response: Include a suitable on-unit in the program to monitor errors in the input data that are revealed by the CONVERSION condition. Use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to assign a valid numeric value so that the program can continue to be executed normally. Otherwise check that all input is in the correct format before executing the program.

IBM025I CONVERSION FROM F-FORMAT ON INPUT AFTER 'TRANSMIT' DETECTED

Explanation: An invalid character has been detected in an F-format input field. A transmission error has also occurred; the conversion error may be directly attributable to the transmission error.

Programmer Response: If the conversion error recurs after the transmission error has been eliminated, take the steps given for the preceding message.

IBM027I CONVERSION FROM E-FORMAT ON INPUT

Explanation: An invalid character has been detected in an E-format input field.

Programmer Response: Take the steps advised for conversion errors in message IBM024I.

IBM028I CONVERSION FROM E-FORMAT ON INPUT AFTER 'TRANSMIT' DETECTED

Explanation: An invalid character has been detected in an E-format input field. A transmission error has also occurred; the conversion error may be directly attributable to the transmission error.

Programmer Response: If the conversion error recurs after the transmission error has been eliminated, take the steps advised for message IBM024I.

IBM029I CONVERSION FROM B-FORMAT ON INPUT

Explanation: An invalid character has been detected in a B-format input field.

Programmer Response: Include a suitable on-unit in the program to monitor errors in the input data that are revealed by the CONVERSION condition. Use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to assign a valid bit character so that the program can continue to be executed normally. Otherwise check that all input is in the correct format before executing the program.

IBM031I CONVERSION FROM B-FORMAT ITEM ON INPUT AFTER 'TRANSMIT' DETECTED.

Explanation: An invalid character has been detected in a B-format input field. A transmission error has also occurred; the conversion error may be directly attributable to the transmission error.

Programmer Response: If the conversion error recurs after the transmission error has been eliminated, take the steps given for message IBM029I.

IBM032I CONVERSION FROM 'CHARACTER' TO ARITHMETIC

Explanation: An invalid character has been detected in a character string that is being converted to an arithmetic data type.

Programmer Response: If the error is in the conversion of a PL/I source program constant or in the conversion of a character string created during the execution of the program, correct the source program, recompile it, and rerun it.

IBM033I CONVERSION FROM CHARACTER TO ARITHMETIC ON INPUT OR OUTPUT

Explanation: A character which is invalid for conversion to an arithmetic form has been detected in one of the following:

1. An arithmetic constant in a list-directed or data-directed item.
2. A character constant being converted to an arithmetic form in a list-directed or data-directed item.
3. An A-format input field being converted to an arithmetic form.

Programmer Response: Take the steps advised for message IBM024I.

IBM034I CONVERSION FROM CHARACTER ON INPUT AFTER TRANSMIT DETECTED

Explanation: A character which is invalid for conversion to an arithmetic form has been detected in one of the following:

1. An arithmetic constant in a list-directed or data-directed input item.

2. A character constant being converted to an arithmetic form in a list-directed or data-directed input item.

3. An A-format input field being converted to an arithmetic form.

A transmission error has also occurred; the conversion error may be directly attributable to the transmission error.

Programmer Response: If the conversion error recurs after the transmission error has been eliminated, take the steps advised for message IBM024I.

IBM035I CONVERSION FROM CHARACTER TO BIT

Explanation: An invalid character has occurred in a character string that is being converted to a bit string.

Programmer Response: If the error is in the conversion of a PL/I source program constant or in the conversion of a character string created during the execution of the program, correct the source program, recompile it, and rerun it.

IBM036I CONVERSION FROM CHARACTER TO BIT ON INPUT OR OUTPUT

Explanation: A character other than 0 or 1 appears in one of the following:

1. A bit constant in a list-directed or data-directed item.

2. A character constant being converted to bit form in a list-directed or data-directed item.

3. An A format input field being converted to bit form.

4. A B-format input field (excluding any leading or trailing blanks).

IBM037I CONVERSION FROM CHARACTER TO BIT ON INPUT AFTER 'TRANSMIT' DETECTED

Explanation: A character other than 1 or 0 appears in one of the following:

1. A bit constant in a list-directed or data-directed input item.

2. A character constant being converted to bit form in a list-directed or data-directed input item.

3. An A-format input field being converted to bit form.

4. A B-format input field (excluding any leading or trailing blanks).

A transmission error has also occurred; the conversion error may be directly attributable to the transmission error.

Programmer Response: If the conversion error recurs after the transmission error has been eliminated, take the steps advised for message IBM024I.

IBM038I CONVERSION TO PICTURE CHARACTER STRING

Explanation: A character that does not match the picture specification has occurred in a conversion to a PICTURE character string.

Programmer Response: Ensure that the character string that is to be converted to a PICTURE character string is suitable for the conversion. If necessary, use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to replace an erroneous character with a character that is valid for conversion.

IBM039I CONVERSION TO PICTURE CHARACTER STRING ON INPUT OR OUTPUT

Explanation: A character that does not match the picture specification has occurred in a stream-oriented item that requires conversion to a PICTURE character string.

Programmer Response: Either ensure that all input data to the program is in the correct format or take the steps given for the preceding message to ensure that the program has adequate error recovery facilities allowing it to comment on any invalid data found in its input and to continue processing.

IBM040I CONVERSION TO PICTURE CHAR STRING ON INPUT AFTER 'TRANSMIT' DETECTED

Explanation: A character that does not match the picture specification has occurred in a stream-oriented input item that requires conversion to a PICTURE character string. A transmission error has also occurred; the conversion error may be directly attributable to the transmission error.

Programmer Response: If the conversion error recurs after the transmission error has been eliminated, take the steps advised for message IBM039I.

IBM042I CONVERSION FROM P-FORMAT (ARITH) ON INPUT

Explanation: An edit-directed P-format input item contains a character that does not match the picture specification.

Programmer Response: Either ensure that all input data to the program is in the correct format before executing the program or use the program to check the data. If necessary, use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to replace an erroneous character with a character that is valid for conversion.

IBM043I CONVERSION FROM P-FORMAT (ARITH) ON INPUT AFTER 'TRANSMIT' DETECTED

Explanation: An invalid character has been detected in a P-format (arithmetic) input field. A transmission error has also occurred; the conversion error may be directly attributable to the transmission error.

Programmer Response: If the conversion error recurs after the transmission error has been eliminated, take the steps advised for message IBM042I.

IBM045I CONVERSION FROM P-FORMAT (CHAR) ON INPUT

Explanation: An invalid character has been detected in a P-format input item.

Programmer Response: Either ensure that all input data to the program is in the correct format before executing the program or use the program to check the data. If necessary, use the ONSOURCE and ONCHAR built-in functions to identify the error and the ONSOURCE and ONCHAR pseudovariables to replace an erroneous character with a character that is valid for conversion.

IBM046I CONVERSION FROM P-FORMAT (CHAR) ON INPUT AFTER 'TRANSMIT' DETECTED

Explanation: An invalid character has been detected in a P-format (character) input item. A transmission error has also occurred; the conversion error may be directly attributable to the transmission error.

Programmer Response: If the conversion error recurs after the transmission error has been eliminated, take the steps advised for message IBM045I.

IBM100I 'NAME' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the NAME condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the NAME condition in the program.

IBM101I INVALID ELEMENT-VARIABLE IN STREAM FOR 'GET FILE DATA'

Explanation: The NAME condition is raised immediately if any of the following errors is detected:

1. An identifier in the input stream has no counterpart in the data list of the GET statement, or the GET statement has no data list and an identifier that is not known in the block is encountered in the stream.
2. Invalid blank characters are found within an identifier in

the input stream.

3. The name field or part of a qualified name is omitted.

4. There are more than 256 characters in a fully-qualified name.

5. Blanks are found within an array subscript other than between the optional sign and the decimal digits.

6. An array subscript is missing or indicates too many dimensions.

7. A value in a subscript is not a decimal digit.

8. The subscript is beyond the declared range of subscripts for a particular array.

9. The left-parenthesis is missing after the name of an array.

10. A character, other than "=" or a blank, is found after a right-parenthesis that delimits an array subscript in the input stream.

11. The end-of-file or a non-blank delimiter is found before "=" in an item in the input stream.

Programmer Response: Use the DATAFIELD condition built-in function in a NAME on-unit to obtain the invalid data item.

IBM120I 'RECORD' CONDITION RAISED BY 'SIGNAL' STATEMENT

Explanation: A SIGNAL statement to raise the RECORD condition has been executed. There was no on-unit for this condition.

Programmer Response: Supply an on-unit for the RECORD condition or remove the SIGNAL statement.

IBM121I LENGTH OF RECORD VARIABLE LESS THAN RECORD LENGTH

Explanation: This message is produced for records that are longer than the associated PL/I variable. For a READ statement, the record is truncated to the length of the variable in the INTO option. For a LOCATE statement (F-format records only), a buffer is not allocated. For a WRITE statement (F-format records only), the record is transmitted with additional bytes to make up the length. The contents of the padding bytes are undefined. For a REWRITE statement, the record is replaced by the shorter record made up to the correct length with the appropriate number of padding bytes, the contents of which are undefined.

Programmer Response: Either supply an on-unit for the RECORD condition so that the program can continue to be executed, or modify the program to make the length of the record variable the same as the length of the records on the data set. The language reference manual for this compiler gives details of how such records are handled when the RECORD condition is raised.

IBM122I LENGTH OF RECORD VARIABLE GREATER THAN RECORD LENGTH

Explanation: This message is produced for records that are shorter than the associated PL/I variable. For a READ statement using F-format records and a fixed-length variable in the INTO option, the excess bytes in the variable are undefined. For a LOCATE statement, where the maximum length of the records is less than the length of the PL/I variable, the buffer is not allocated. For a WRITE statement, the variable in the FROM option is longer than the maximum length of the records, and is truncated to the maximum record length. For a REWRITE statement, the variable in the FROM option is longer than the record it is to replace, and is truncated to the length of this record.

Programmer Response: Either supply an on-unit for the RECORD condition so that the program can continue to be executed, or modify the program to make the length of the record variable the same as the length of the records on the data set. The language reference manual for this compiler gives details of how such records are handled when the RECORD condition is raised.

IBM123I 'WRITE' OR 'LOCATE' VARIABLE HAS ZERO LENGTH

Explanation: A WRITE or REWRITE statement has attempted to transmit a record variable of zero length, or a LOCATE statement has attempted to obtain buffer space for a zero length record variable.

Programmer Response: Modify the program to ensure that the varying-length string used as a record variable is not a null string when the WRITE, REWRITE, or LOCATE statement is executed.

IBM124I ZERO LENGTH RECORD READ FROM REGIONAL DATA SET

Explanation: A record of zero length has been read from a REGIONAL data set associated with a DIRECT file. A zero-length record on a direct-access device indicates the end of the data set. However, the message above will only be produced if the data set has been created incorrectly.

Programmer Response: Check that the data set was created correctly as a regional data set. Recreate the data set if necessary and if it is possible to do so. Check also that the record has been accessed with a key that is valid for the data set.

IBM125I 'RECORD' VARIABLE TOO SHORT TO CONTAIN EMBEDDED KEY.

Explanation: A WRITE or REWRITE statement has attempted to transmit, or a LOCATE statement has attempted to allocate buffer space for, a record variable too short to contain the data set embedded key. For a WRITE or REWRITE statement, no transmission takes place; for a LOCATE statement, a buffer is not allocated.

IBM140I 'TRANSMIT' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the TRANSMIT condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the TRANSMIT condition in the program.

IBM141I UNCORRECTABLE ERROR IN OUTPUT

Explanation: Data management routines have detected an uncorrectable error while transmitting output data between main storage and an external storage device. The condition is raised on the completion of a WRITE, REWRITE, or LOCATE statement. For BUFFERED files, this condition may be raised only after the execution of several I/O statements following the I/O statement which transmitted the record. No further processing of an OUTPUT file, other than a file associated with a unit record device, can occur. Processing of an UPDATE file may continue. For INDEXED data sets, the condition can also occur while searching through the indexes or tracing an overflow record.

Programmer Response: If the error recurs, obtain a dump of the input/output buffer areas by using PLIDUMP in a TRANSMIT on-unit. See the programmer's guide for details of PLIDUMP. The resultant output, together with all relevant listings and data sets, should be preserved for examination by IBM.

IBM142I UNCORRECTABLE ERROR IN INPUT

Explanation: Data management routines have detected an uncorrectable error while transmitting input data between main storage and an external storage device. If the block contains VS-format records, the error is raised once only for the block. Otherwise, the condition is raised on the completion of a READ or REWRITE statement for each record in the block that contains the error and for every item transmitted by GET statements from a block that contains the error. The contents of the record or data item are undefined. However, processing of subsequent records in the input file can be continued. For INDEXED data sets, the condition can be raised while searching the indexes or tracing an overflow record.

Programmer Response: If the error recurs, obtain a dump of the input/output buffers by using PLIDUMP in a TRANSMIT on-unit. See the programmer's guide for details of PLIDUMP. The resultant output, together with all relevant listings and data sets, should be preserved for examination by IBM.

| IBM143I 'TRANSMIT' CONDITION RAISED. UNREADABLE OMR DATA.

Explanation: One or more OMR columns contain a marginal mark, weak mark, or poor erasure that cannot be read. The condition is raised on completion of the READ operation for the card. An X'3F' character is substituted for unreadable characters, and also put in the last byte of the record. The card is stacker selected to the alternate stacker.

IBM144I WRITE ERROR IN INDEX SET.

Explanation: Data management has detected a physical error whilst attempting to write on the index set of a VSAM KSDS. The condition is raised on the completion of a WRITE, REWRITE, LOCATE, or DELETE statement. No further processing of an OUTPUT file can occur. Processing of an UPDATE file may continue.

IBM145I READ ERROR IN INDEX SET.

Explanation: Data management has detected a physical error whilst attempting to read from the index set of a VSAM KSDS. The condition is raised on the completion of a READ, WRITE, REWRITE, LOCATE, or DELETE statement. No further processing of an OUTPUT file can occur. Processing of an UPDATE file may continue. If the error occurs on a READ statement, no data is transferred to the record variable. For sequential access, data set positioning may be lost, causing a subsequent READ without KEY to raise ERROR (see message IBM831I)

IBM146I WRITE ERROR IN SEQUENCE SET.

Explanation: Data management has detected a physical error whilst attempting to write on the sequence set of a VSAM KSDS. The condition is raised on the completion of a WRITE, REWRITE, LOCATE, or DELETE statement. No further processing of an OUTPUT file can occur. Processing of an UPDATE file may continue.

IBM147I READ ERROR IN SEQUENCE SET.

Explanation: Data management has detected a physical error whilst attempting to read from the sequence set of a VSAM KSDS. The condition is raised on the completion of a READ, WRITE, REWRITE, LOCATE, or DELETE statement. No further processing of an OUTPUT file can occur. Processing of an UPDATE file may continue. If the error occurs on a READ statement, no data is transferred to the record variable. For sequential access, data set positioning may be lost, causing a subsequent READ without KEY to raise ERROR (see message IBM831I)

IBM160I 'KEY' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the KEY condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the KEY condition in the program.

IBM161I KEY SPECIFIED CANNOT BE FOUND

Explanation: A READ, REWRITE, or DELETE statement specified a recorded key which could not be found on the data set. In the case of an INDEXED data set, the key in error is either higher than the highest level index or the record is not in the prime area or the overflow areas of the data set. In the case of a DIRECT file associated with a data set with REGIONAL organization, the key in error is not in the specified region or within the search limit defined by the LIMCT subparameter of the DCB parameter.

Programmer Response: Determine why the key was incorrect and modify the program or the data set as necessary. Use of the ONKEY built-in function in a KEY on-unit will aid in determining the value of the erroneous key.

IBM162I KEY SPECIFIED ALREADY IN USE ON DATA SET

Explanation: In the case of data set with INDEXED organization, an attempt has been made to transmit a keyed record to a data set which already holds a record with the same key. In the case of a data set with REGIONAL(1) or REGIONAL(2) organization that is being created sequentially, an attempt has been made to transmit a record to a region that already contains a record.

Programmer Response: Either check the validity of the data that is being processed before executing the program or use the program to check the data. Use of the ONKEY built-in function in a KEY on-unit will aid in identifying an erroneous key, in correcting it, and in permitting processing to continue normally.

IBM163I KEY VALUE IS LESS THAN VALUE OF PREVIOUS KEY

Explanation: A key with a value that is less than the value of the preceding key has been detected during the creation or extension of an INDEXED or REGIONAL SEQUENTIAL data set.

Programmer Response: Ensure that the records that are to be written onto an INDEXED or REGIONAL data set that is being created or extended are in the correct ascending key sequence order. Otherwise use a KEY on-unit to comment on the error and, where possible, to permit processing to continue normally.

IBM164I KEY SPECIFIED CANNOT BE CONVERTED TO VALID DATA

Explanation: A WRITE, READ, REWRITE, DELETE, or LOCATE statement for a REGIONAL data set specified a key with a character-string value consisting entirely of blanks or containing characters other than 0-9 or blank as part of the region number.

Programmer Response: Ensure that the key is in the correct format. If necessary, use the ONKEY built-in function in a KEY on-unit to identify the erroneous key. The on-unit can be used to report any such errors and allow processing to continue. Records associated with the erroneous keys can be transmitted in a subsequent run for which the keys have been corrected.

IBM165I KEY SPECIFIED IS INVALID

Explanation: For an INDEXED data set, either the KEY or the KEYFROM expression is a null string or an attempt has been made to rewrite a record where the embedded key of the replacement record is not equal to that of the record that is to be overwritten. For a REGIONAL data set, the key specified is a null string or is a string commencing with '11111111'B.

Programmer Response: Follow the steps advised for the previous message.

IBM166I KEY SPECIFIES POSITION OUTSIDE REGIONAL DATA SET

Explanation: A WRITE, READ, REWRITE, or DELETE statement specifies a key whose relative record or track value exceeds the number of records or tracks respectively for the REGIONAL data set.

Programmer Response: Follow the steps advised for message IBM164I.

IBM167I NO SPACE AVAILABLE TO ADD KEYED RECORD

Explanation: For a SEQUENTIAL file associated with an INDEXED data set, an attempt has been made to write or locate a record during the creation or extension of such a data set when the space allocated to the data set is full. For a DIRECT file associated with an INDEXED data set, there is no space in the available overflow areas to accept the overflow record caused by the insertion of a new record by a WRITE statement.

For a DIRECT file associated with a REGIONAL data set, there is no space to add the record in the specified limit of search as specified in the LIMCT subparameter of the DCB parameter. Note that the data set is not necessarily full.

Programmer Response: Use the ONKEY built-in function to identify the key value that caused the error. If the key is in error, correct it and recommence the job from the point reached when the error occurred. If the key is correct, organize the data set so that the rejected record can be accommodated.

| IBM168I 'KEYFROM' VALUE LIES OUTSIDE KEY RANGE(S) DEFINED FOR DATA SET.

| Explanation: A WRITE or LOCATE statement specified a key with a value outside the key ranges specified for the data set when it was defined. (VSAM KSDS).

IBM180I 'ENDFILE' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the ENDFILE condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the ENDFILE condition in the program.

IBM181I 'ENDFILE' CONDITION RAISED.

Explanation: The end of an input file has been detected.

Programmer Response: Include an on-unit for the ENDFILE condition for each input file in the program to handle the end-of-file processing.

IBM182I END OF FILE PREVIOUSLY ENCOUNTERED ON STREAM INPUT

Explanation: The ENDFILE condition was raised when the file mark was encountered but an attempt is being made to read beyond the end of the file. Either an ENDFILE on-unit has been executed and a further attempt to read the file is being made or the end-of-file mark was encountered between items in the data list of the current GET statement.

Programmer Response: If the program contains an ENDFILE on-unit, make sure that it does not attempt to read the file after the ENDFILE condition has been raised for it. If the error occurred during execution of a GET statement with two or more items in the data list, make sure that the GET statement can be completed by providing sufficient data items before the end-of-file mark is encountered.

IBM200I 'UNDEFINEDFILE' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the UNDEFINEDFILE condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the UNDEFINEDFILE condition in the program.

IBM201I CONFLICTING DECLARE AND OPEN ATTRIBUTES.

Explanation: An attribute in an OPEN statement conflicts with one in a DECLARE statement. The attributes may have been written explicitly or implied by other attributes, for example, DIRECT implies KEYED. Some RECORD input/output statements imply file attributes in an implicit OPEN statement, for example, LOCATE implies RECORD OUTPUT BUFFERED SEQUENTIAL. A list of conflicting attributes follows:

BACKWARDS	STREAM, OUTPUT/UPDATE, DIRECT, KEYED, EXCLUSIVE, PRINT, TRANSIENT
BUFFERED	STREAM, UNBUFFERED, PRINT,
DIRECT	STREAM, SEQUENTIAL, BACKWARDS, PRINT, TRANSIENT
EXCLUSIVE	STREAM, INPUT/OUTPUT, SEQUENTIAL, BACKWARDS, PRINT, TRANSIENT
INPUT	OUTPUT/UPDATE, EXCLUSIVE, PRINT
KEYED	STREAM, BACKWARDS, PRINT
OUTPUT	INPUT/UPDATE, EXCLUSIVE, BACKWARDS

PRINT	RECORD, INPUT/UPDATE, DIRECT/SEQUENTIAL, BUFFERED/UNBUFFERED, KEYED, EXCLUSIVE, BACKWARDS, TRANSIENT
RECORD	STREAM, PRINT
SEQUENTIAL	STREAM, DIRECT, EXCLUSIVE, PRINT, TRANSIENT
STREAM	RECORD, UPDATE, DIRECT/SEQUENTIAL, BUFFERED/UNBUFFERED, KEYED, EXCLUSIVE, BACKWARDS, TRANSIENT
TRANSIENT	STREAM, UPDATE, DIRECT/SEQUENTIAL, EXCLUSIVE, BACKWARDS, PRINT
UNBUFFERED	STREAM, BUFFERED, PRINT
UPDATE	STREAM, INPUT/OUTPUT, BACKWARDS, PRINT, TRANSIENT

IBM202I DEVICE TYPE CONFLICTS WITH FILE ATTRIBUTE.

Explanation: Certain types of file cannot be associated with particular input/output devices. For example, a file with the UPDATE attribute cannot be associated with a paper tape reader, a printer, or a magnetic-tape device.

IBM203I BLOCK SIZE NOT SPECIFIED.

Explanation: For an output file, the block size must be specified, either in the ENVIRONMENT attribute or in the DCB parameter of the DD statement.

IBM204I NO DD STATEMENT.

Explanation: The job stream must contain a DD statement with a ddname that is either a file name (if the TITLE option is not specified) or the name provided by the TITLE option.

IBM205I I/O ERROR. 'REGIONAL' DATA SET CANNOT BE FORMATTED.

Example:

```
TF: PROC;
OPEN FILE(F) DIRECT OUTPUT;
END;
```

Explanation: When a REGIONAL data set is opened for direct output, data management routines format the data set into specified regions by writing dummy or control records into the data set. In this case there was an I/O error that prevented the data set from being formatted correctly.

Programmer Response: If the problem recurs, have the direct access device or storage medium checked by a Customer Engineer.

IBM206I 'LINESIZE' OR 'PAGESIZE' OUTSIDE IMPLEMENTATION-DEFINED LIMITS

Explanation: The implementation-defined maximum or minimum for the LINESIZE option of the ENVIRONMENT attribute has been exceeded. For F-format and U-format records, the maximum is 32,759; for V-format records, the maximum is 32,751. The minimum in all cases is 1.

Programmer Response: Check that the argument to the LINESIZE option is within the prescribed limits. If the argument is a variable, check that it is a FIXED BINARY (31,0) STATIC variable that was correctly initialized before the file was opened.

IBM207I KEY LENGTH NOT SPECIFIED.

Explanation: No key length has been specified, either in the ENVIRONMENT attribute or in the DCB parameter of the associated DD statement.

IBM208I WRONG BLOCKSIZE OR RECORD LENGTH SPECIFIED

Explanation: One of the following errors may have occurred:

1. Block size is less than record length.
2. For FB-format records, block size is not a multiple of record length.
3. For VS-format and VBS-format consecutive files:
    a. LRECL=X was specified but RECSIZE was not specified or was invalid in the ENVIRONMENT attribute.
    b. The file was opened for update with a specified logical record size exceeding 32,756.
4. For VS-format REGIONAL(3) files, logical record size was greater than block size minus four.
5. FUNC=EO is specified with a record length not equal to 80 or FUNC=CO is specified with a record size not equal to 160.
6. Column binary is specified with a record length not equal to 160 on an output file.
7. FUNC=I (punch interpret) is specified with a record length not equal to 80 (or 81 if control characters are in use).

Programmer Response: The four numbered responses below apply to the correspondingly numbered explanations above:

1. Check the block size and record length specified in the BLKSIZE and RECSIZE options of the ENVIRONMENT attribute. If LINESIZE was specified, check that it is compatible with BLKSIZE.
2. If the argument of either option is a variable, check that it is FIXED BINARY(31,0) STATIC and that it has been initialized.
3. (a) Specify a record size in the ENVIRONMENT attribute, or

correct its value.
(b) Specify a record size less than 32,757.

4. Specify a record size less than or equal to the block size minus four.

IBM209I CONFLICTING ATTRIBUTES AND FILE ORGANIZATION.

Explanation: The file organization conflicts with one or more explicit or implicit file attributes. The following is a list of all possible conflicts:

Organization	*Attributes*
CONSECUTIVE	DIRECT, EXCLUSIVE, KEYED, TRANSIENT
INDEXED	STREAM, TRANSIENT, DIRECT OUTPUT, OUTPUT without KEYED
REGIONAL	STREAM, TRANSIENT, OUTPUT without KEYED
TP	Non-TRANSIENT
VSAM	STREAM, TRANSIENT, BACKWARDS, DIRECT OUTPUT, OUTPUT without KEYED(KSDS), KEYED(ESDS), DIRECT(ESDS)
None	KEYED, TRANSIENT

IBM210I RECORD FORMAT INVALID FOR THIS FILE ORGANIZATION.

Explanation: The following combinations of file organization and record format are valid:

CONSECUTIVE	All
INDEXED	F, FB, V, VB
REGIONAL(1)	F
REGIONAL(2)	F
REGIONAL(3)	F, V, VS, U
TP(M), TP(R)	None

Programmer Reponse: Amend the file declaration so that the record format is compatible with the file organization.

IBM211I RECORD FORMAT NOT SPECIFIED.

Explanation: A record format must be supplied for a file with the RECORD attribute either in the ENVIRONMENT attribute or in a DD statement.

IBM212I KEYLENGTH NEGATIVE OR GREATER THAN 255

Explanation: The KEYLENGTH option of the ENVIRONMENT attribute for this file has an invalid key length that is greater than

255 or is negative.

Programmer Response: The argument of the KEYLENGTH option should be checked to ensure that it is either a constant or a variable with the attributes FIXED BINARY (31,0) STATIC whose value neither exceeds 255 nor is negative when the file is opened. If the argument is a variable, check that it has been correctly initialized.

IBM215I INVALID BUFOFF VALUE

Explanation: The values that can be specified in the BUFOFF option for an ASCII input data set are in the range 0 thru 99.

Programmer Response: Ensure that the value specified in the BUFOFF option is within the range of values given above. If the argument is a variable, ensure that it is correctly initialized.

IBM219I 'MODE' OR 'FUNC' CONFLICTS WITH FILE ATTRIBUTE.

Explanation: The MODE or FUNC DCB subparameter conflicts with a file attribute. Refer to the Programmer's Guide for details of all possible conflicts.

IBM220I 'MODE' OR 'FUNC' CONFLICTS WITH RECORD FORMAT.

Explanation: OMR or RCE files, 3525 print files and 3525 associated files can only be F-format.

IBM221I DEVICE TYPE CONFLICTS WITH 'MODE'.

Explanation: OMR can only be used on a 3505 and RCE on a 3525 device.

IBM222I TOTAL OPTION INVALID ON AN 'OMR' OR ASSOCIATED FILE.

Explanation: Either OMR (MODE=EO or MODE=CO) is specified on a file with the TOTAL option, or device association is specified on a file with the TOTAL option.

IBM223I CONFLICT BETWEEN 'MODE' AND 'FUNC'.

Explanation: Refer to the Programmer's Guide for details of all possible conflicts.

IBM225I VALUE OF ENVIRONMENT OPTION DOES NOT MATCH ACTUAL DATA SET VALUE.

Explanation: For VSAM data sets the values of KEYLOC, KEYLENGTH, and RECSIZE are specified when the data set is defined. If values are specified on any file declarations they must match the defined values.

IBM226I 'NCP' OR 'STRNO' VALUE NOT 1.

Explanation: For VSAM files only one outstanding operation is permitted, that is, an operation with the EVENT option must be waited for before another operation is started. Either an NCP value greater than one has been specified in the ENV attribute or a STRNO value greater than one has been specified in the AMP parameter in the DD statement.

IBM227I 'TOTAL' OPTION SPECIFIED FOR ESDS.

Explanation: The specification of TOTAL may cause the compiler to generate in-line code for I/O statements for CONSECUTIVE files. If the data set to be accessed is a VSAM Entry Sequenced Data set (ESDS) this code is invalid.

IBM228I PASSWORD INVALID OR NOT SPECIFIED.

Explanation: For VSAM data sets defined with a password, ENV (PASSWORD) must be specified. If this password is invalid or is not specified the system operator is allowed a number of attempts to specify the correct password (the number of retries is specified when the data set is defined). If these attempts fail UNDEFINEDFILE is raised.

IBM229I NO ENTRY IN VSAM CATALOG.

Explanation: Before using a VSAM data set a catalog entry must be made and space allocated for the data set, using the AMS DEFINE utility. The catalog containing the data set must be specified in a JOBCAT or STEPCAT DD statement (unless it is the master catalog). This message may mean that ENV(VSAM) has been specified for a file, but the data set has not been converted from ISAM to VSAM.

IBM230I I/O ERROR READING CATALOG OR VOLUME LABEL.

Explanation: An I/O error prevented the reading of a VSAM catalog or a volume label.

IBM231I TIMESTAMP MISMATCH.

Explanation: For VSAM data sets the index and data can be updated separately and the time of the latest update of each is recorded. If these times do not match, the integrity of the data is uncertain. Similarly the timestamp in the data set catalog record may not match the timestamp on the volume containing the data set; this indicates the extent information in the catalog record may not agree with the extents indicated in the VTOC for the volume.

IBM232I DATA SET NOT AVAILABLE.

Explanation: The data set to be accessed is already being used by another program and is not shareable. Refer to the Programmer's Guide for further information.

IBM233I DATA SET NOT PROPERLY CLOSED.

Explanation: The last time the data set was opened the close operation failed, leaving the data set in an unusable state. Use of the access method services VERIFY utility program may restore the data set to a usable state.

IBM234I DATA SET NEVER LOADED.

Explanation: A file cannot be opened for INPUT or UPDATE to access a VSAM data set until one or more records have been loaded into the data set using a SEQUENTIAL OUTPUT file. Having once loaded records into the data set records can be added using a DIRECT UPDATE file even after all records have been deleted from the data set.

IBM235I UNIDENTIFIED ERROR DURING VSAM OPEN.

Explanation: The VSAM routines have detected an error during the open process, the cause of which cannot be determined explicitly. Message IEC250 on the operator's console will give more detail.

Programmer Response: If the error recurs after resubmitting the job, use PLIDUMP to obtain a storage dump and retain all the relevant documentation for study by IBM.

IBM236I OPERATING SYSTEM UNABLE TO OPEN FILE ccc.

Explanation: The file cannot be opened because:

1. TRANSIENT files and INDEXED data sets are not supported under CMS.

2. When a VSAM data set is opened through the ISAM compatibility interface, this condition will occur if VSAM detects errors during the open process. Message IEC250 on the operator's console will give more detail.

IBM300I 'ZERODIVIDE' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the ZERODIVIDE condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the ZERODIVIDE condition in the program.

IBM301I 'ZERODIVIDE' CONDITION RAISED

Explanation: The program has attempted to execute a statement in which a value of zero has been used as the divisor in a division operation.

Programmer Response: Either check the data that could produce a zero divisor before running the program or insert a ZERODIVIDE on-unit to handle the condition whenever it arises.

IBM320I 'UNDERFLOW' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the UNDERFLOW condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the UNDERFLOW condition in the program.

IBM321I 'UNDERFLOW' CONDITION RAISED

Explanation: The magnitude of a floating-point number is smaller than the permitted minimum.

IBM340I 'SIZE' CONDITION RAISED BY 'SIGNAL' STATEMENT

Explanation: The program contains a SIGNAL statement to raise the SIZE condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the SIZE condition in the program.

IBM341I 'SIZE' CONDITION RAISED IN I/O STATEMENT.

Explanation: The high-order (leftmost) significant binary or decimal digits are lost in an input/output operation where the size of the value being transmitted exceeds the declared (or default) size of the data item.

IBM342I 'SIZE' CONDITION RAISED

Explanation: The high-order (leftmost) significant binary or decimal digits are lost in an assignment to a variable or temporary variable where the size of the value being assigned exceeds the declared (or default) size of the data item.

Programmer Response: Either modify the program so that the data item is large enough for the value being assigned to it or use a SIZE on-unit to permit processing to continue when the SIZE condition is raised.

IBM360I 'STRINGRANGE' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the STRINGRANGE condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include on on-unit for the STRINGRANGE condition in the program.

IBM361I 'STRINGRANGE' CONDITION RAISED

Explanation: In the expression SUBSTR(S,I,J), I and J are such that the substring does not lie wholly within the string S.

Programmer Response: It should be possible to modify the source program so that this condition cannot occur.

IBM380I 'AREA' CONDITION RAISED BY THE 'SIGNAL' STATEMENT

Explanation: The program contains a SIGNAL statement to raise the AREA condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the AREA condition in the program.

IBM381I 'AREA' ASSIGNMENT NOT EXECUTED, TARGET AREA TOO SMALL

Explanation: In an assignment of an area variable, the current extent of the area on the right-hand side of the assignment statement is greater than the size of the area to which it is to be assigned.

Programmer Response: Correct the program.

IBM382I NOT ENOUGH CONTIGUOUS SPACE IN THE AREA FOR ALLOCATION

Explanation: In the execution of an ALLOCATE statement, insufficient space is available in the specified area for the allocation.

Programmer Response: Provide an on-unit to permit the allocation to be retried. If necessary, change the value of the pointer qualifying the reference to the inadequate area so that it points to another area in which the allocation can be retried.

IBM400I 'CONDITION' CONDITION RAISED BY 'SIGNAL' STATEMENT, CONDITION ccc.

Explanation: The program contains a SIGNAL statement to raise the CONDITION condition for which there is no on-unit. (ccc denotes the condition-name associated with CONDITION in the SIGNAL statement.)

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the CONDITION condition in the program.

IBM420I 'SUBSCRIPTRANGE' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the SUBSCRIPTRANGE condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the SUBSCRIPTRANGE condition in the program.

IBM421I 'SUBSCRIPTRANGE' CONDITION RAISED.

Explanation: An array subscript has been found to have a value exceeding the declared bound for the array.

Programmer Response: In order to ensure that the program can continue processing after encountering a subscript range error, include an on-unit for this condition which executes a GOTO statement to the appropriate place in the program, and recompile it. Normal return from a subscriptrange on-unit will produce this message and raise the error condition. Note that array handling operations are made slower when SUBSCRIPTRANGE is enabled.

IBM440I 'STRINGSIZE' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the STRINGSIZE condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the STRINGSIZE condition in the program.

IBM441I 'STRINGSIZE' CONDITION RAISED

Explanation: The 'STRINGSIZE' condition is raised when a string is assigned to a shorter string, causing right-hand characters or bits in the source string to be truncated.

Programmer Response: Determine whether or not truncation of the right-hand characters or bits in the source string is correct. Use an on-unit to record the relevant data or modify the program as required. Note that string-handling operations are made slower when STRINGSIZE is enabled.

IBM460I 'OVERFLOW' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the OVERFLOW condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the OVERFLOW condition in the program.

IBM461I 'OVERFLOW' CONDITION RAISED

Explanation: The OVERFLOW condition occurs when the magnitude of a floating-point number exceeds the permitted maximum.

Programmer Response: Modify the program to ensure that the condition does not recur, or provide an on-unit to handle the condition.

IBM480I 'FIXEDOVERFLOW' CONDITION RAISED BY 'SIGNAL' STATEMENT.

Explanation: The program contains a SIGNAL statement to raise the FIXEDOVERFLOW condition for which there is no on-unit.

Programmer Response: Either remove the SIGNAL statement or include an on-unit for the FIXEDOVERFLOW condition in the program.

IBM482I 'FIXEDOVERFLOW' CONDITION RAISED

Explanation: The FIXEDOVERLOW condition occurs when the length of the result of a fixed-point arithmetic operation exceeds the permitted maximum (15 for decimal values, and 31 for binary values).

Programmer Response: Modify the program to ensure that the condition does not recur, or provide an on-unit to handle the condition.

IBM500I TASK TERMINATION FOLLOWING MULTIPLE EXCEPTION INTERRUPT

(Plus one or more of the following:)

IMPRECISE INTERRUPT. 'ADDRESSING' UNPROCESSED.

IMPRECISE INTERRUPT. 'DATA INTERRUPT' UNPROCESSED.

IMPRECISE INTERRUPT. 'OVERFLOW' UNPROCESSED.

IMPRECISE INTERRUPT. 'PROTECTION' UNPROCESSED.

IMPRECISE INTERRUPT. 'SPECIFICATION' UNPROCESSED.

IMPRECISE INTERRUPT. 'ZERODIVIDE' UNPROCESSED.

Explanation: A multiple-exception imprecise interrupt has occurred and the task is about to be terminated as a result of one of these exceptions. The remaining exceptions will not be processed.

IBM531I OPERATION EXCEPTION

Explanation: An attempt has been made to execute an instruction with an invalid System/360 or 370 operation code.

Programmer Response: It is possible that an error in the program has caused part of the executable instructions to be overwritten by data. Refer to the section on program checkout in the programmer's guide for suggestions for deleting and correcting such errors. Other possible causes of an operation exception might be an attempt to invoke an external procedure or other routine that was not incorporated into the executable program by the linkage editor, or the execution of a branch instruction that has been made incorrect because a control block had previously been overwritten. Consequently, it is advisable to check the linkage editor diagnostics to ensure that all requested external procedures and subroutines have in fact been incorporated into the executable program, and that any overlay phases do not overwrite any phases that are still active.

IBM532I PRIVILEGED OPERATION EXCEPTION

Explanation: An attempt has been made to execute certain System/360 or 370 instructions which can only be executed by the supervisor program. This condition can only be raised for a PL/I program which includes a non-PL/I routine that contains such an instruction or in which an error has occurred causing an executable instruction in the program to be overwritten with data that is identical to one of the privileged instructions.

Programmer Response: If the error is not in a non-PL/I routine included in the executable program, the PL/I program should be checked for an error that could cause the executable instructions to be overwritten by data that matches a privileged operation. The section on program checkout in the programmer's guide contains suggestions for detecting and correcting such errors.

IBM533I EXECUTE EXCEPTION

Explanation: An attempt has been made to use an IBM System/360 or 370 EXECUTE instruction to execute another EXECUTE instruction. This can occur if a routine that contains this error has been included in the PL/I program, or if an executable instruction that is the subject of a compiler-generated EXECUTE instruction has been overwritten by data that matches the operation code for the EXECUTE instruction.

Programmer Response: If the error is not in a non-PL/I routine included in the executable program, the PL/I program should be checked for an error that could cause the executable instruction to be overwritten by data that matches the operation code for the EXECUTE instruction on. The section on program checkout in the programmer's guide contains suggestions for detecting and correcting such errors.

IBM534I PROTECTION EXCEPTION

Explanation: An attempt has been made to store data in main storage that is outside the partition allocated to the program.

Programmer Response: If the error is not in a non-PL/I routine included in the executable program, the PL/I program should be checked for an error that could cause the address used by the store instruction to be corrupted. The section on program checkout in the programmer's guide contains suggestions for detecting and correcting such errors.

IBM535I ADDRESSING EXCEPTION

Explanation: An invalid address has been supplied as an operand to an IBM System/360 or 370 instruction.

Programmer Response: If the error is not in a non-PL/I routine included in the executable program, the PL/I program should be checked for an error that could cause the address to be corrupted. The section on program checkout in the programmer's guide contains suggestions for detecting and correcting such errors.

IBM536I SPECIFICATION EXCEPTION

Explanation: An alignment error in the operands of an IBM System/360 or 370 instruction, or an error in the specification of the operands, has occurred.

Programmer Response: If the error is not in a non-PL/I routine included in the executable program, the PL/I program should be checked for an error that could cause the operand to be corrupted by overwriting control blocks or sections of executable code. The section on program checkout in the programmer's guide contains suggestions for detecting and correcting such errors.

IBM537I DATA EXCEPTION

Explanation: An attempt has been made to process FIXED DECIMAL data that is not in the correct format.

Programmer Response: The PL/I program should be checked for an error such as an operation on a FIXED DECIMAL data item before it has been initialized, or an error which could cause the data item to be overwritten. Refer to the chapter on program checkout in the programmer's guide for hints on how to trace such errors.

IBM538I OPERATION EXCEPTION, FLOATING POINT INSTRUCTIONS NOT SUPPORTED.

Explanation: An attempt has been made to execute a floating-point instruction on a machine that does not have hardware facilities for floating-point arithmetic. The floating-point instruction is contained either in instructions generated by the compiler, or in a non-PL/I routine in this program.

IBM560I EVENT VARIABLE AS ARG TO CPLN P-V, ALREADY IN USE WITH FILE fff

Explanation: The event variable used in this statement is already active and is associated with an input/output operation on the named file.

Programmer Response: Modify the program so that the COMPLETION pseudovariable refers to the event variable when it is inactive.

IBM561I TASK VARIABLE ALREADY USED WITH ENTRY eee

Explanation: The task variable specified in a CALL statement is already associated with an active task. eee denotes the entry point of the task with which the variable is associated.

IBM562I EVENT VARIABLE AS ARG TO CPLN P-V ALREADY IN USE WITH 'DISPLAY' STMNT.

Explanation: The event variable used in this statement is already active and is associated with a DISPLAY statement.

Programmer Action: Modify the program so that the COMPLETION pseudovariable refers to the event variable when it is inactive.

IBM563I EVENT VARIABLE ALREADY IN USE WITH FILE fff

Explanation: The event variable used in this statement is already active and is associated with another input/output operation on the named file.

Programmer Response: Modify the program so that the input/output operation refers to another event variable, or include a WAIT statement to prevent execution of the statement until the active event is complete.

IBM564I EVENT VARIABLE ASSIGNED TO, ALREADY IN USE WITH FILE fff

Explanation: An attempt has been made to assign a value to an event variable while it is still associated with an input/output operation. For example:

```
READ FILE(X) INTO(Y) EVENT(Z);
Z=Z1;
```

Programmer Response: Modify the program so that the event variable used as the target in the assignment, or as the argument of the COMPLETION pseudovariable, is not the same event variable associated with an input/output operation. Alternatively, include a WAIT statement to prevent execution of this statement until the active event is complete.

IBM565I NOT ENOUGH MAIN STORAGE AVAILABLE FOR SUBTASK.

Explanation: On execution of a CALL statement with multitasking options, it has been found that there is insufficient main storage available to create a new task. Enough main storage must be available before execution of the procedure called as a subtask can begin.

IBM566I 'TASK' NOT CREATED, SINCE NUMBER OF ACTIVE TASKS WOULD EXCEED LIMIT.

Explanation: The number of tasks allowed to be active at the same time is determined by an option that can be specified when the PL/I program is invoked. If no value is specified, a standard default of 20 tasks is assumed. For example:

```
DO I=1 TO N;
CALL A TASK;
END;
CALL A TASK;
A PROC;
WAIT(E);
END;
```

The second CALL statement, if executed, would take the number of concurrently active tasks over the implicit or explicit limit.

Programmer Response: Increase the number of tasks allowed to be active concurrently or modify the program so that the existing number of active tasks is not exceeded.

IBM567I WAIT IN ON UNIT FOR I/O EVENT BEING WAITED FOR BY THIS TASK.

Explanation: This error is caused when a WAIT statement specifies an event variable, and the completion of the event causes entry to an on-unit for an I/O condition which contains another WAIT statement for the same event variable.

For example:

```
ON RECORD(F) BEGIN;
          .
          .
          .
          WAIT(E);
          .
          .
          .
          END;

WRITE FILE(F)...EVENT(E);
WAIT(E); (this statement raises
             the record condition)
```

Programmer Response: Remove the WAIT statement from the on-unit for the input/output condition.

IBM568I EVENT VARIABLE ASSIGNED TO, ALREADY IN USE WITH 'DISPLAY' STMNT

Explanation: The event variable specified as the argument of the COMPLETION built-in function, or used as the target in an assignment, is still associated with a DISPLAY statement. For example:

```
DISPLAY('MESSAGE TO OPERATOR')
REPLY(A) EVENT(E);
COMPLETION(E)='1'B;
```

Programmer Response: Modify the program so that the event variable used as the target in the assignment or as the argument of the COMPLETION pseudovariable is not the same event variable associated with the DISPLAY statement. Alternatively, include a WAIT statement to prevent execution of this statement until the active event is complete.

IBM569I EVENT VARIABLE ASSIGNED TO IS ALREADY ACTIVE, USED WITH ENTRY eee

Explanation: An event variable that is still active has been specified as the target of an event variable assignment. For example:

```
CALL P EVENT(E);
E=E1;
```

Programmer Response: Modify the program either to ensure that the event variable in question is inactive when the assignment is executed, by inserting a WAIT statement, or use another, inactive, event variable.

IBM570I EVENT VARIABLE ALREADY ACTIVE, USED WITH ENTRY eee

Explanation: An event variable that is still active has been specified in the EVENT option of an input/output statement.

Programmer Response: Modify the program either to ensure that the event in question is inactive when this statement is executed, by inserting a WAIT statement, or use another, inactive, event variable if the statement can be executed correctly before the currently active event is complete.

IBM571I EVENT VARIABLE ALREADY IN USE WITH 'DISPLAY' STMNT

Explanation: The event variable specified in the statement is already associated with a DISPLAY statement.

Programmer Response: Either use a different event variable or insert a WAIT statement so that the DISPLAY statement is complete before this statement is executed.

IBM572I CALL WITH TASK OPTION INVALID IN 'PUT FILE(SYSPRINT)' STATEMENT.

Explanation: A CALL TASK statement is not allowed, because during execution of a PUT statement no other task in the program can produce output on SYSPRINT, and task interlock (the situation where two or more tasks are waiting for each other) is likely to occur. For example:

```
    PUT LIST(F(X));
E:  PROC(X);
    CALL F TASK;
```

IBM573I EVENT VARIABLE AS ARG TO CPLN P-V, ALREADY IN USE WITH ENTRY eee

Explanation: An event variable used as the argument to the COMPLETION pseudovariable must be inactive.

IBM574I 'CALL' STATEMENT WITH 'TASK' OPTION SPECIFIES UNRESOLVED ENTRY POINT.

Explanation: The address of the procedure to be invoked as a task has been found to be zero. This is probably caused by the CALL statement specifying an external procedure which was not link-edited into the load module.

Programmer Response: Ensure that the procedure invoked as a task is present in the load module.

IBM575I ATTEMPT TO USE FORTRAN OR COBOL IN TWO TASKS CONCURRENTLY.

Explanation: An attempt has been made to invoke a COBOL or FORTRAN subroutine when either:

1. A COBOL or FORTRAN subroutine was active as part of another task, or,

2. There was an active procedure in another task which had called the FORTRAN subroutine.

Programmer Response: Change the PL/I source program by inserting WAIT statements so that a call or function reference to the other language subroutine cannot be attempted in either of the above situations.

IBM576I ATTEMPT TO CALL TASK IN NON-TASKING ENVIRONMENT.

Explanation: An attempt was made to call a task when the multitasking library was not selected in the link-edit step.

Programmer Response: When concatenating libraries in the SYSLIB specification of the link-edit step, ensure that the multitasking library SYS1.PLITASK is placed before the base library SYS1.PLIBASE (See the programmer's guide).

IBM590I UNABLE TO FIND FETCHABLE PROCEDURE WITH ENTRY eee

Explanation: The job and/or link libraries available when the program was executed did not contain a member with a name or alias matching that used in the FETCH statement.

Programmer Response: Ensure that the job or link library contains the load module that is to be fetched, and that it is stored with the same name or alias as that used in the FETCH statement.

IBM591I PERMANENT I/O ERROR WHILE FETCHING PROCEDURE WITH ENTRY eee

Explanation: A permanent I/O error has occurred during the search through the job and/or link libraries for the load module named in the FETCH statement.

Programmer Response: Ensure that the required load module has been correctly incorporated into the appropriate library, and then rerun the job. If the problem recurs, inform your installation system programmer, who will take the appropriate action.

| IBM592I FETCH/RELEASE IS NOT SUPPORTED IN CMS.

Explanation: An attempt has been made to FETCH or RELEASE another program from PL/I. The FETCH/RELEASE facility is not available under CMS. The ERROR condition has been raised.

IBM600I INCORRECT VALUES OF W,D,S FIELDS IN E-FORMAT SPECIFICATION

Explanation: An edit-directed input/output operation for an E-format item has been specified incorrectly.

Programmer Response: Correct the E-format item according to the language rules.

IBM601I VALUE OF W FIELD TOO SMALL IN F-FORMAT SPECIFICATION

Explanation: An edit-directed input/output operation for an F-format item has been specified incorrectly with a W-specification that is too small to allow room for the decimal-point when the number of fractional digits is specified as zero.

Programmer Response: Correct the F-format item according to the language rules.

IBM604I INVALID ASSIGNMENT TO PICTURED CHARACTER STRING

Explanation: A data item which is not a character string cannot be assigned to a pictured character string because it does not match the declared characteristics of the pictured target variable.

Programmer Response: Modify the program so that the assignment is possible by altering the characteristics either of the source variable or of the target variable.

IBM605I ITERATION FACTOR IN FORMAT LIST / DEPTH OF R-FORMAT NESTING TOO LARGE.

IBM606I INVALID REMOTE FORMAT ITEM IN FORMAT LIST.

IBM607I REMOTE FORMAT STATEMENT IS OUTSIDE THE CURRENT BLOCK.

IBM608I LABEL VARIABLE IN R-FORMAT ITEM DOES NOT REFER TO R-FORMAT STMT LABEL.

IBM650I SOURCE NOT MODIFIED IN CONVERSION ON-UNIT, RE-TRY NOT ATTEMPTED

Explanation: The CONVERSION condition has been raised by the presence of an invalid character in the string to be converted. The character has not been corrected in an on-unit using the ONCHAR or ONSOURCE pseudovariable.

Programmer Response: Modify the CONVERSION on-unit to use either the ONCHAR or the ONSOURCE pseudovariable to assign a valid character to replace the invalid character in the source string.

IBM670I X LT 0 IN SQRT(X)

Explanation: The built-in function SQRT has been invoked with an argument that is less than zero. On-codes associated with this message are:

1500	Short floating-point SQRT error
1501	Long floating-point SQRT error
1502	Extended floating-point SQRT error

Programmer Response: Modify the program so that the argument of the SQRT built-in function can never be less than zero.

IBM671I X LE 0 IN LOG(X), LOG2(X), OR LOG10(X)

Explanation: One of the built-in functions LOG, LOG2, or LOG10 has been invoked with an argument that is less than or equal to zero. The invocation may have been direct or as part of the evaluation of an exponentiation calculation. On-codes associated with the message are:

1503	Extended floating-point LOG, LOG2, or LOG10 error
1504	Short floating-point LOG, LOG2, or LOG10 error
1505	Long floating-point LOG, LOG2, or LOG10 error

Programmer Response: Modify the program so that the argument of the LOG, LOG2, or LOG10 built-in function is greater than zero.

IBM672I ABS(X) TOO LARGE IN SIN(X),COS(X),SIND(X),COSD(X),TAN(X) OR TAND(X)

Explanation: The argument passed to TAN, TAND, SIN, SIND, COS, or COSD exceeds the limit specified below:

Floating-Point Precision	_Limit_	
Binary $p \le 21$ Decimal $p \le 6$	x<(2**18)*K	where K = pi for x in radians (SIN, COS, or TAN)
Binary $21 < p \le 53$ Decimal $6 < p \le 16$	x<(2**50)*K	where K = 180 for x in degrees (SIND, COSD, TAND)
Binary $53 < p \le 109$ Decimal $16 < p \le 33$	x<(2**100)*K/pi	

The error has arisen during one of the following:

1. The evaluation of SIN, SIND, COS, COSD, TAN, or TAND when invoked implicitly.

2. The evaluation of TAN, when invoked during the evaluation of TAN or TANH with a complex argument.

3. The evaluation of SIN or COS, when invoked during the evaluation of EXP, SIN, SINH, COS, COSH, TAN or TANH with a complex argument.

4. The evaluation of a general exponentiation function with

complex arguments. On-codes associated with this message are:

1506	Short floating-point SIN, SIND, COS, or COSD error
1507	Long floating-point SIN, SIND, COS, or COSD error
1501	Extended floating-point SIN, SIND, COS, or COSD error
1508	Short floating-point TAN or TAND error
1509	Long floating-point TAN or TAND error
1517	Extended floating-point TAN or TAND error

IBM674I X=Y=O IN ATAN(Y,X),OR ATAND(Y,X)

Explanation: Two arguments, both zero, have been given for the ATAN or ATAND built-in function. ATAN or ATAND has been invoked either directly with a real argument or indirectly in the evaluation of the LOG built-in function with a complex argument. On-codes associated with this message are:

1510	Short floating-point ATAN or ATAND error
1511	Long floating-point ATAN or ATAND error
1522	Extended floating-point ATAN or ATAND error

Programmer Response: Modify the program so that the arguments of ATAN or ATAND are not both zero.

IBM675I ABS(X) GE 1 IN ATANH(X)

Explanation: The ATANH built-in function has been used with a floating-point argument with an absolute value that equals or exceeds 1. On-codes associated with this message are:

1514	Short floating-point ATANH error
1515	Long floating-point ATANH error
1516	Extended floating-point ATANH error

Programmer Response: Modify the program so that the absolute value of a floating-point assignment to the ATANH built-in function does not equal or exceed 1.

IBM676I ABS(X) GT 1 IN ASIN(X) OR ACOS(X)

Explanation: The absolute value of the floating-point argument of the ASIN or ACOS built-in function exceeds 1. On-codes associated with this message are:

1518	Short floating-point ASIN or ACOS error
1519	Long floating-point ASIN or ACOS error
1520	Extended floating-point ASIN or ACOS error

Programmer Response: Modify the program so that the ASIN or ACOS built-in function is never invoked with a floating-point argument whose absolute value exceeds 1.

IBM700I ATTEMPT TO ASSIGN TO UNALLOCATED CONTROLLED VARIABLE IN GET DATA FOR FILE fff

Example:

```
                DCL X CONTROLLED FIXED BIN;
                GET DATA(X);

                        (Input stream contains X=5,.....)
```

Explanation: A variable in the stream accessed by a GET FILE DATA statement is CONTROLLED but there is no current allocation for the variable.

Programmer Response: Either remove the data item from the input stream or insert an ALLOCATE statement for the variable before the GET FILE DATA statement.

IBM701I ATTEMPT TO ASSIGN TO UNALLOCATED CONTROLLED VARIABLE IN GET DATA.

Example:

```
                DCL STR CHAR(4) INIT('X=5;')
                X CONTROLLED FIXED BIN;
                GET STRING(STR) DATA(X);
```

Explanation: A variable in a string accessed by a GET STRING DATA statement is CONTROLLED but there is no current allocation for the variable.

Programmer Response: Either remove the data item from the string or insert an ALLOCATE statement for the variable before the GET STRING DATA statement.

IBM722I X=O AND Y IS NOT REAL AND +VE IN X ** Y

Explanation: In an exponentiation operation the floating-point base is zero and the exponent is not positive and real. On-codes associated with this message are:

1550	Short floating-point real base with integer exponent
1551	Long floating-point real base with integer exponent
1560	Extended floating-point real base with integer exponent
1552	Short floating-point real base with floating-point exponent
1553	Long floating-point real base with floating-point exponent
1561	Extended floating-point real base with floating-point exponent
1554	Short floating-point complex base with integer exponent
1555	Long floating-point complex base with integer exponent
1562	Extended floating-point real base with integer exponent
1556	Short floating-point complex base with complex exponent
1557	Long floating-point complex base with complex

exponent
1563 Extended floating-point complex base with complex exponent

Programmer Response: Modify the program so that the exponentiation operation involves a non-zero floating-point base or a positive real exponent.

IBM724I Z=+1I OR -1I IN ATAN(Z) OR Z=+1 OR -1 IN ATANH(Z)

Explanation: The complex floating-point argument of the ATAN built-in function has the value of +1I or -1I. Alternatively, the complex floating-point argument of the ATANH built-in function has the value +1 or -1. On-codes associated with this message are:

1558	Short floating-point complex ATAN or ATANH error
1559	Long floating-point complex ATAN or ATANH error
1564	Extended floating-point complex ATAN or ATANH error

Programmer Response: Modify the program so that the complex floating-point argument of ATAN can never be +1I or -1I, or the complex floating-point argument of the ATANH built-in function never has the value +1 or -1.

IBM750I 'GOTO' TO AN INVALID BLOCK ATTEMPTED.

Example:

```
DCL L LABEL;
    BEGIN;
 A: L = A;
    END;
    GOTO L;
```

Explanation: A GOTO statement that transfers control to a label variable is invalid because:

1. The generation of the block that was active when the label variable was assigned was no longer active when the GOTO statement was executed.

2. The label variable was uninitialized.

3. The element of the label array, to which control is to be transferred, does not exist in the program.

4. An attempt has been made to transfer control to a block that is not within the scope of this task.

Programmer Response: Modify the program so that the GOTO statement transfers control to a label variable that was assigned in a block that is still active.

IBM772I 'WAIT' WITH MULTIPLE 'EVENTS' NOT IN THIS SYSTEM

Explanation: A WAIT statement with more than one event variable has been encountered. The PL/I Transient Library for this system was generated to handle WAIT statements for single events only.

Programmer Response: Modify the program so that the WAIT statement specifies one event only.

IBM802I GET/PUT STRING EXCEEDS STRING SIZE

Explanation: For input, a GET statement has attempted to access data that exceeds the length of the source string. For output, a PUT statement has attempted to assign data that is longer than the target string.

IBM803I FURTHER OUTPUT PREVENTED BY PRIOR CONDITION FOR FILE fff.

Explanation: A PL/I WRITE, LOCATE, or PUT statement has been issued for a file to which a previous attempt to transmit a record caused the TRANSMIT condition to be raised immediately or, if the EVENT option has been specified, to be stacked until the event was waited on. The data set is not a unit-record device and no further processing of the file is possible.

IBM804I 'PRINT' OPTION/FORMAT ITEM USED WITH NON-'PRINT' FILE fff.

Explanation: An attempt has been made to use one of the options PAGE or LINE for a file that is not a print file.

IBM805I 'DISPLAY' WITH 'REPLY' OPTION HAS ZERO LENGTH STRING.

Explanation: The current length of the character string to be displayed, or the maximum length of the character string to which the reply is to be assigned, is zero.

IBM807I NO PRECEDING 'READ SET' OR 'READ INTO' FOR 'REWRITE' OR 'DELETE' ON FILE fff

Explanation: A REWRITE or DELETE statement without the KEY option has been executed when the last input/output operation on the file was not a READ statement with the SET or INTO option or was a READ statement with the IGNORE option.

Programmer Response: Modify the program so that the REWRITE or DELETE statement is either preceded by a READ statement or, in the case of a REWRITE statement, replaced by a WRITE statement, according to the requirements of the program. A preceding READ statement with the IGNORE option will also cause this message to be issued.

IBM808I INVALID ELEMENT VARIABLE IN STRING FOR 'GET STRING DATA'

Explanation: The identifier in the string named in the STRING option of a GET STRING DATA statement does not match the identifier in the data specification. Note that the DATAFIELD built-in function will not return a value in this case.

Programmer Response: Modify the program so that the string contains the identifier in the data specification.

IBM809I INVALID FILE OPERATION.

Explanation: An attempt has been made to carry out an operation on a file that is invalid with the file as declared. For example, it is not possible to execute a REWRITE statement on a STREAM file, read an output file, or write an input file. A list of other possible conflicts follows:

Statement and Option	Conflicting File Attribute or Organization
Any record I/O statement	STREAM
Any stream I/O statement	RECORD
READ	OUTPUT
READ SET	UNBUFFERED
READ EVENT	BUFFERED
READ KEY	REGIONAL SEQUENTIAL or CONSECUTIVE
READ IGNORE	DIRECT
READ NOLOCK	SEQUENTIAL or INPUT
WRITE	INPUT SEQUENTIAL UPDATE
	INDEXED DIRECT NOWRITE
	REGIONAL (not KEYED)
WRITE EVENT	BUFFERED
REWRITE	INPUT or OUTPUT
REWRITE (without FROM)	UNBUFFERED or DIRECT
REWRITE KEY	SEQUENTIAL
REWRITE EVENT	BUFFERED
LOCATE	INPUT or UPDATE
	UNBUFFERED
	DIRECT
LOCATE KEYFROM	INDEXED or REGIONAL (without KEYED)
DELETE	INPUT or OUTPUT
	CONSECUTIVE
	REGIONAL SEQUENTIAL
	RKP=0 (blocked records)
	OPTCD=L not specified
DELETE KEY	SEQUENTIAL
UNLOCK	INPUT or OUTPUT
	SEQUENTIAL
GET	OUTPUT
PUT	INPUT

Programmer Response: Ensure that the file declaration and the input/output statements for the named file are compatible.

IBM811I I/O ERROR. CAUSE NOT KNOWN.

Explanation: The data management routines have detected an error during an input/output operation, the cause of which could not be determined explicitly.

Programmer Response: If the error recurs after resubmitting the job, use PLIDUMP to obtain a storage dump and retain all the relevant documentation for study by IBM.

IBM812I NO PRECEDING 'READ SET' OR 'READ INTO' FOR 'REWRITE'.

Explanation: A REWRITE statement has been executed for which no preceding READ statement, either with the INTO option or with the SET option, has been executed.

Programmer Response: Modify the program so that the REWRITE statement is either preceded by a READ statement or replaced by a WRITE statement, according to the requirements of the program.

IBM813I LAST 'READ' BEFORE THIS 'REWRITE' OR 'DELETE' IS INCOMPLETE.

Explanation: An attempt has been made to execute a REWRITE or DELETE statement before a preceding READ statement (with the EVENT option) for the same file has been completed.

Programmer Response: Modify the program so that the REWRITE or DELETE statement is executed after completion of the READ statement by inserting a WAIT statement for the given event variable into the flow of control between the two statements.

IBM814I TOO MANY INCOMPLETE I/O OPERATIONS.

Explanation: An attempt has been made to initiate an input/output operation beyond the limit imposed by the NCP (number of channel programs) subparameter of the DCB parameter or option of the ENVIRONMENT attribute. For a file with the attributes SEQUENTIAL and UNBUFFERED, the default for NCP is one. The limit, for VSAM files, is specified either by the NCP option of the ENVIRONMENT attribute or by the STRNO subparameter of the AMP parameter in the DD statement. Except when using the ISAM compatibility interface the limit is one for both SEQUENTIAL and DIRECT UNBUFFERED files.

Programmer Response: Modify the program so that the input/output operation is not initiated until an incomplete input/output operation has been completed.

IBM816I IMPLICIT 'OPEN' UNSUCCESSFUL.

Explanation: An error has occurred during the implicit opening of a file. The UNDEFINEDFILE condition was raised and a normal return was made from the associated on-unit, but the file was still unopened.

Programmer Response: Ensure that the file has been completely and correctly declared, and that the input/output statement that implicitly opens the file is not in conflict with the file declaration.

IBM818I UNEXPECTED END OF FILE/STRING DETECTED IN STREAM INPUT.

Explanation: The end of the file has been detected before the completion of a GET FILE statement.

Programmer Response: For edit-directed input, ensure that the last item of data in the stream has the same number of characters as specified in the associated format item. If the error occurs during execution of an X-format item, ensure that the same number of characters to be skipped are present before the last data item in the stream.

For list-directed and data-directed input, ensure that the last item of data in the data set, if a string, is terminated by a quote character (and if a bit-string, by a 'B') that precedes the end-of-file marker.

IBM819I ATTEMPT TO CLOSE FILE IN THE WRONG TASK.

Explanation: A file can only be closed by the task that opened it.

IBM820I INVALID ATTEMPT TO ACCESS A LOCKED RECORD.

Explanation: In an exclusive environment an attempt has been made to read, rewrite, or delete a record when either the record or the data set is locked by another file in this task.

IBM821I I/O STATEMENT OCCURRED BEFORE PREVIOUS 'READ' COMPLETED BY 'WAIT'.

Explanation: While an indexed sequential file was open for direct updating, an input/output statement was attempted before the completion of a previous READ statement with the EVENT option.

Programmer Response: Include a WAIT statement so that the erroneous input/output statement cannot be executed until the completion of the previous READ statement with the EVENT option.

IBM823I INVALID CONTROL FORMAT ITEM FOR 'GET'/'PUT' STRING.

Explanation: An invalid control format item (PAGE, LINE, SKIP, or COL) has been detected in a remote format list for a GET or PUT STRING statement. For example:

```
          DCL(A,B) CHAR(10),
                C CHAR(80);
          F:  FORMAT(A(10), SKIP,A(10));
              A='FRED'; B = 'HARRY';
              PUT STRING(C) EDIT(A,B) (R(F));
```

Programmer Response: Modify the source program so that GET or PUT STRING statements do not attempt to use invalid control format items in remote format lists.

IBM824I RECORDS STILL LOCKED IN SUBTASK WHILE ATTEMPTING TO CLOSE EXCLUSIVE FILE fff

Example:

```
DCL F FILE EXCLUSIVE;
OPEN FILE(F);
CALL P TASK PRIORITY(1);
CLOSE FILE(F); /* ERROR RAISED HERE */

P:  PROC;
   READ FILE (F) EVENT (E);
   DELAY (1000);
   WAIT (E);
   END P;
```

Explanation: When an exclusive file is closed, any records still locked must be unlocked or a system ABEND will occur. A record can only be unlocked in the task in which it was locked. Consequently, a CLOSE statement cannot imply an UNLOCK statement for any files locked in a subtask of the closing task.

IBM825I EVENT VARIABLE ALREADY IN USE.

Explanation: An input/output statement with an EVENT option has been attempted while a previous input/output statement with an EVENT option that uses the same event variable is still incomplete.

Programmer Response: Either change the event variable used in the second EVENT option or insert a WAIT statement for the event variable between the two input/output statements.

IBM826I EVENT VARIABLE ALREADY IN USE WITH DISPLAY STATEMENT.

Explanation: An input/output statement with an EVENT option has been attempted while a previous DISPLAY statement with an EVENT option that uses the same event variable is still incomplete.

Programmer Response: Either change the event variable used in the second EVENT option or insert a WAIT statement for the event variable between the DISPLAY statement and the input/output statement.

IBM827I EVENT VARIABLE ALREADY ACTIVE, USED WITH ENTRY eee

Explanation: An event variable that has already been used in the EVENT option in a CALL statement is still active when again used in the EVENT option of an input/output statement.

Programmer Response: Either use a different event variable or insert a WAIT statement so that the input/output statement is not executed until the event variable becomes inactive.

IBM828I INCORRECT SEQUENCE OF I/O OPERATIONS ON AN ASSOCIATED FILE.

Explanation: Operations on a set of associated files were not carried out in the correct sequence, as follows:

1. Appropriate I/O operations were not carried out in the

sequence Read-Punch-Print. Only the Print operation may be omitted or repeated.

2. An attempt was made to print more than the maximum number of lines on a card, using a print file that was associated with a read or punch file.

IBM829I INSUFFICIENT VIRTUAL STORAGE AVAILABLE TO VSAM.

<u>Explanation</u>: During an OPEN/CLOSE or any other operation on a VSAM data set insufficient storage was available for workspace and control blocks.

IBM830I I/O ERROR DURING 'CLOSE'.

<u>Explanation</u>: An I/O error occurred while a VSAM close routine was either reading or writing a catalog record, or completing an outstanding I/O request.

IBM831I NO POSITIONING ESTABLISHED FOR SEQUENTIAL READ.

<u>Explanation</u>: A READ statement without the KEY option has been attempted on a VSAM data set, after sequential positioning has been lost as the result of a previous error during sequential processing (for example, read error on index set or failure to position to next highest key after a 'key not found' condition).

IBM832I INSUFFICIENT SPACE FOR VSAM DATA SET.

<u>Explanation</u>: VSAM has been unable to allocate additional DASD space for the data set (ESDS or KSDS). The condition is raised on attempting to write or locate a record during the sequential creation or extension of such a data set when the space allocated to the data set is full. For a KSDS, the condition may also occur when the associated PL/I file is opened for update and attempts are made to write new records or to increase the size of existing records by the WRITE and REWRITE statements respectively.

IBM833I RECORD ALREADY HELD IN EXCLUSIVE CONTROL.

<u>Explanation</u>: The VSAM data set control interval containing the requested record is in the process of being updated by another file which used the same DD statement.

IBM834I REQUESTED RECORD LIES ON NON-MOUNTED VOLUME.

<u>Explanation</u>: The requested record lies on a non-mounted volume of a VSAM data set spanning several volumes.

IBM835I ATTEMPT TO REPOSITION FOR SEQUENTIAL READ FAILED.

<u>Explanation</u>: The attempt to reposition to the next highest key for subsequent sequential retrieval on a VSAM KSDS, after the 'key not found' condition, has failed. If processing of the file is continued, the next I/O statement should have a positioning KEY option. (See message IBM831I).

IBM836I NUMBER OF CONCURRENT OPERATIONS ON DATA SET EXCEEDS 'STRNO'.

Explanation: When several files are accessing a VSAM data set by means of the same DD statement (that is, using the same title) the STRNO subparameter of the DD statement specifies the total number of operations on all files that can be active at the same time. A read-rewrite pair of operations on a sequential file counts as one operation. For example, if three sequential files are to update the same data set at the same time, STRNO=3 should be specified in the DD statement.

IBM850I AGGREGATE LENGTH EXCEEDS 2**24 BYTES

Explanation: The length of the structure or array to be mapped is greater than 2^{24} thereby exceeding the limits of addressability.

Programmer Response: Reduce the size of the array or structure to a size that can be accommodated within the main storage available. If a variable is used to specify the dimension or length, check that it has been correctly initialized before the storage is allocated to the aggregate.

IBM851I UNABLE TO MAP ARRAY STRUCTURE ELEMENT

Explanation: The program contains a structure with an adjustable element and an array element with extents that cause the relative virtual origin to exceed $2^{32}-1$. For example:

```
DCL 1 A,
      I B CHAR(N),
      I C (32766:32767,32766:32767,32766:32767)
         CHAR(32767);
```

Programmer Response: Ensure that aggregates with array elements remain within the limit of addressability ($2^{32}-1$).

IBM852I AGGREGATE CANNOT BE MAPPED TO COBOL OR FORTRAN FORM

Explanation: An attempt has been made to either pass to or obtain from a FORTRAN routine an array of more than 7 dimensions, or to pass to or obtain from a COBOL routine a structure with more than three dimensions.

Programmer Response: Ensure that PL/I aggregates, passed to or from COBOL or FORTRAN routines are within the limits described above.

IBM853I INVALID DATA TYPE EXPECTED FROM FORTRAN FUNCTION.

Explanation: This message may be produced only when the program is executed in conjunction with a program compiled by the checkout compiler. It is caused by the use of a FORTRAN function that returns a value with attributes declared in the RETURNS option of the entry declaration that are other than FLOAT, FIXED BINARY, or BIT (non-VARYING). After the ERROR condition is raised, a normal return from the ERROR on-unit will cause the PL/I program to be resumed, although no value will have been returned to it.

IBM854I MAXIMUM DEPTH OF ITERATION EXCEEDED ON ARRAY INITIALIZATION.

Example:

```
DCL A(8192) CHAR(1) INIT((2)((2)((2)((2)((2)((2)((2)
    ((2)((2)((2)((2)((2)((2)(1)'A')))))))))))));
```

Explanation: The error is raised if the depth of iteration within the initial attribute on an automatic array exceeds 12.

IBM880I A PROGRAM CHECK HAS OCCURRED IN THE SORT/MERGE PROGRAM.

Explanation: An error has occurred during execution of the sort/merge program when invoked from a PL/I program by means of the PL/I sort interface facilities. The sort program was unable to continue and control has been passed to the PL/I error-handler.

Programmer Response: Since the problem has occurred during execution of the sort/merge program, refer to the appropriate sort/merge program manual for an explanation of any diagnostic messages produced by the sort program and for any other information that may be necessary to correct the fault.

|IBM881I SORT IS NOT SUPPORTED IN CMS.

Explanation: An attempt has been made to call the SORT facility from PL/I. SORT is not available under CMS. The ERROR condition has been raised.

IBM900I 'WAIT' STATEMENT WOULD CAUSE PERMANENT WAIT. PROGRAM TERMINATED.

Explanation: A WAIT statement that can never be completed has been encountered. For example:

```
COMPLETION (E1) = '0'B;
      WAIT(E1);
```

The event E1 is inactive and incomplete.

Programmer Response: Modify the program so that the WAIT statement can never wait for an event that is inactive and incomplete.

IBM910I AN ABEND HAS OCCURRED, USER CODE=nnnn

Explanation: An ABEND macro instruction has been issued by the problem program and an analysis has not detected any overwriting of PL/I control blocks. (The user code is presented in decimal digits.)

IBM911I AN ABEND HAS OCCURRED, SYSTEM CODE=sss

IBM920I 'CHECKPOINT/RESTART' IS NOT SUPPORTED IN A CHECKER ENVIRONMENT.

Explanation: A PL/I program that uses the checkpoint/restart facilities has been compiled by the PL/I optimizing compiler and executed in association with a PL/I program compiled by the PL/I checkout compiler. The CALL PLICKPT statement is ignored in these circumstances.

IBM921I GOTO OUT OF ON-UNIT MAY CAUSE FURTHER USE OF COBOL SUBROUTINE TO FAIL.

Explanation: If a COBOL subroutine is reinvoked after an interrupt in the previous invocation was handled by a PL/I on-unit that was terminated by a GOTO statement, the COBOL subroutine will fail.

IBM924I CLOSING FILE IN ON-UNIT MAY CAUSE FURTHER ERRORS IN THIS STATEMENT.

Explanation: An on-unit for an I/O condition has been entered and the file associated with the on-unit has been closed in the on-unit. A GOTO should have been used to exit from the on-unit, but was not. The result of a normal return is undefined.

| IBM925I PLIRETC VALUE REDUCED TO 999.

| Explanation: The value passed to the PLIRETC built-in procedure is greater than 999. The value has been reduced to 999 which is the maximum permitted user value.

| IBM926I CHECKPOINT/RESTART IS NOT SUPPORTED IN CMS.

| Explanation: An attempt has been made to call the CHECKPOINT/RESTART facility from PL/I. CHECKPOINT/RESTART is not available under CMS. The ERROR condition has been raised.

Part III: PL/I Prompter(IKJ) Messages

This part lists the messages that can be produced by the PL/I prompter, which is used to invoke the PL/I optimizing compiler in a TSO environment, and providés further explanation of the message texts.

Format of PL/I Prompter Messages

The format of the messages produced by the prompter is:

```
    [message-number] text
```

Each message number is of the form IKJ65nnn[I|A] where "IKJ65" indicates that the message is a PL/I prompter message, and "nnn" is the number of the message. The final character "I" or "A" indicates whether the message is informatory or whether some action from the programmer is necessary for the prompter to continue.

The message number is printed only if a request that messages are to include such identifiers has been made, either when your user-identification was added to the system or in a subsequent PROFILE command.

The text describes the error that has been detected. If the text ends with a plus sign, it is possible to obtain an additional message, containing more information, by entering a question mark. In some cases, the additional message can have several forms, depending on the error.

If a message ending in a plus sign is accompanied by message IKJ65045A ("REENTER..."), you need not fulfill the request to reenter before entering a question mark to obtain further information. But although message IKJ65045A will not be reissued, its request must nevertheless be fulfilled in due course before the prompter can continue.

To correct an error you may need to refer to the TSO manual for the optimizing compiler. If, at any time, you want to terminate the prompter, in order, for example, to enter another command, press the ATTN key (or its equivalent).

The messages are formatted in this publication as they will appear on the terminal except that page width restricts line length to 72 characters, whereas on a terminal, up to 120 characters can be used. The entry for each message number in the following list includes all texts associated with that number. That is, the original text (sometimes this has two forms) plus any additional texts that may be obtained by entering a question mark.

Symbols in Messages

Many of the messages in this publication contain symbols indicating where information will be inserted when the message is printed. The symbols used are:

ddd - a name taken from the PLI or RUN OPT command that was used to invoke the prompter.

xxx - a name or number generated by the prompter.

Before Calling IBM . . .

If you consider that a message has been produced erroneously, then before calling IBM for programming support, retain the listing produced at the terminal and inform the TSO operator, who is required to generate other diagnostic information for messages relating to data set allocation.

IKJ65001I DATA SET ddd NOT ALLOCATED, TOO MANY DATA SETS+

UTILITY DATA SET NOT ALLOCATED, TOO MANY DATA SETS+

USE FREE COMMAND TO FREE UNUSED DATA SETS

Explanation: A data set required by the optimizing compiler cannot be allocated because insufficient DD statements are included in the LOGON procedure. Allocation may be possible if you can free data sets used for any previous operations.

You can determine the number of data sets that need to be freed because the data sets are allocated, if required, for the files in the following order: SYSIN, SYSLIN, SYSPRINT, SYSPUNCH, SYSLIB, and SYSUT1. You may need to refer to the data set naming conventions to find out which file the specified data set name is associated with.

IKJ65002I DATA SET ddd NOT ALLOCATED, DATA SET NOT ON VOLUME+

CATALOG INFORMATION INCORRECT

Explanation: The data set cannot be found on the volume specified in the operating system's data set catalog. It is possible that the data set has been deleted or moved to another volume by a utility program without altering the catalog entry. Reenter another data set name. If you enter a null line, the default data set name will be assumed, except for the primary input data set and data sets specified in the LIB operand. If no alternative data set can be used, or if the error persists, consult your system operations personnel.

IKJ65003I DATA SET ddd NOT ALLOCATED, REQUIRED VOLUME NOT MOUNTED+

VOLUME OR CVOL NOT ON SYSTEM AND CANNOT BE ACCESSED

Explanation: The data set cannot be found because the volume on which it resides, or the volume containing index information (the control volume) is not mounted ready for use. Reenter the name of a data set that resides on a volume that is mounted. If you enter a null line, the default data set name will be assumed, except for the primary input data set and data sets specified in the LIB operand. If no alternative data set can be used, or if it is the control volume that is not mounted, request the system operator to mount the volume required.

IKJ65004I DATA SET ddd NOT ALLOCATED, SYSTEM OR INSTALLATION ERROR+

CATALOG ERROR CODE xxx

DYNAMIC ALLOCATION ERROR CODE xxx

CATALOG I/O ERROR

Explanation: The data set cannot be allocated because of an error in a routine handling the data set catalog or the dynamic allocation of data sets. Reenter another data set name. If you enter a null line, the default data set name will be assumed, except for the primary input data set or data sets specified in the LIB operand. If no alternative data set can be used, or if the error persists, consult your system maintenance personnel.

IKJ65005I DATA SET ddd NOT ALLOCATED+

UTILITY DATA SET NOT ALLOCATED+

INVALID UNIT IN USER ATTRIBUTE DATA SET

NO UNIT AVAILABLE

Explanation: The data set cannot be allocated because the attributes associated with your user-identification specify a unit type that is invalid or unavailable on your system. If you are authorized to use the ACCOUNT command, you can change the unit type associated with your user-identification. Otherwise, consult your system maintenance personnel.

IKJ65006I DATA SET ddd ALREADY IN USE, TRY LATER+

DATA SET IS ALREADY ALLOCATED TO ANOTHER JOB OR USER

Explanation: The data set cannot be allocated to you because it is already allocated to another TSO user, or to another job running in the system. If all uses of the data set have the SHR status, this message will not be produced. You can either enter a null line to cause the default data set name to be applied (except for the primary input data set or data sets specified in the LIB operand), or terminate the prompter. If you terminate the prompter, you can reinvoke it by specifying a different data set name, or carry on with another operation until the required data set is freed.

IKJ65007I DATA SET ddd NOT IN CATALOG

Explanation: The data set cannot be found in the operating system's data set catalog. Check the name of the data set, taking into account the data set naming conventions used by the prompter, or check that the data set is cataloged. Reenter the correct cataloged data set name. If you enter a null line, the default data set name will be assumed, except for the primary input data set and data sets specified in the LIB operand.

IKJ65008I MEMBER ddd NOT IN DATA SET ddd

Explanation: The member cannot be found in the partitioned data set. Check the name of the member and the data set, taking into account the data set naming conventions used by the prompter. Reenter the correct data set name plus member name. If you enter a null line, the default data set name will be assumed (without a member name), except for the primary input data set or data sets specified in the LIB operand.

IKJ65009I DATA SET ddd NOT USABLE+

I/O SYNAD ERROR

OPEN ERROR

Explanation: This message applies only to partitioned data sets. The data set cannot be used because of an error detected when opening the data set, or when reading information from the data set's directory. Reenter another data set name. If you enter a null line, the default data set name will be assumed, except for the primary input data set and data sets specified in the LIB operand. If no alternative data set can be used, or if the error persists, consult your system maintenance personnel.

IKJ65010I INVALID DATA SET NAME, ddd EXCEEDS 44 CHARACTERS

Explanation: A qualified data set name must not exceed 44 characters in length. The separating periods and any qualifiers added by the prompter are included in the count. Reenter a name which forms a valid qualified data set name. If you enter a null line, the default data set name will be assumed, except for the primary input data set and data sets specified in the LIB operand.

IKJ65011I ddd IS A DELETED OPTION AND HAS BEEN IGNORED+

THE OPTION WAS DELETED AT SYSGEN AND IS NOT AVAILABLE FOR USE

Explanation: The option cannot be used because when your system was generated it was deleted from the list of options supported by the optimizing compiler. You may be able to reinstate the option temporarily by using the CONTROL option. However, at the moment the option specified has been ignored.

IKJ65012I NOT ENOUGH MAIN STORAGE TO EXECUTE COMMAND

Explanation: There is insufficient space available for the prompter in the main storage region. The prompter requires much less main storage than the optimizing compiler, so you need a much larger region size to use the optimizing compiler successfully.

If possible, relog-on with a LOGON procedure that has a region size of at least 50K bytes for the compiler, plus the space required by TSO (usually about 30K bytes). Alternatively, if you are authorized to use the ACCOUNT command, you can increase the region size for the current LOGON procedure. Otherwise, consult your system operations personnel.

IKJ65013I COMMAND SYSTEM ERROR+

xxx ERROR CODE xxx

Explanation: An error has occurred in one of the TSO service routines. The routine is specified in the text inserted before the word ENTER. If this message is produced, consult your system maintenance personnel.

IKJ65014I DATA SET ddd NOT ALLOCATED, NOT ENOUGH SPACE ON VOLUMES+

USE DELETE COMMAND TO DELETE UNUSED DATA SETS

Explanation: One of the data sets required by the optimizing compiler cannot be allocated because insufficient space exists on each of the available volumes. The prompter is terminated. The space that the prompter requests for each data set is listed in the TSO manual for the optimizing compiler. Allocation may be possible if you can delete any data sets that are no longer required.

You can determine the amount of space that you may need to make available because the data sets are allocated, if required, for the files in the following order: SYSIN, SYSLIN, SYSPRINT, SYSPUNCH, SYSLIB, and SYSUT1. You may need to refer to the data set naming conventions to find out which file the specified data set name is associated with.

IKJ65015I DATA SET ddd NOT ALLOCATED, SHARED+

USE FREE COMMAND TO FREE THE DATA SET

Explanation: The data set cannot be allocated because it is already allocated for this session. If all uses of the data set have the SHR status, this message will not be produced. You can either enter a null line to cause the default data set name to be applied (except for the primary input data set or data sets specified in the LIB operand), or terminate the prompter. If you terminate the prompter, you can then free the data set and invoke the prompter again. The data set will be reallocated for the optimizing compiler.

IKJ65016I DATA SET ddd WILL CREATE INVALID CATALOG STRUCTURE+

A NAME CANNOT BE BOTH AN INDEX AND THE LAST QUALIFIER OF A QUALIFIED DATA SET NAME

Explanation: The qualified data set name uses the same name both as an index and as the last qualifier. For example, when specifying a simple name you have used the same name that the prompter will add as the descriptive qualifier. You should reenter a name that will form a valid cataloged data set name. If you enter a null line, the default data set name will be assumed, except for the primary input data set and data sets specified in the LIB operand.

IKJ65017I DATA SET ddd NOT ON DIRECT-ACCESS DEVICE, NOT SUPPORTED

Explanation: TSO does not support data sets that reside on devices which are not direct-access. The operating system's data set catalog indicates that this data set resides on another type of device. You should reenter the name of a data set on a direct-access device. If you enter a null-line the default data set name will be assumed, except for the primary input data set and data sets specified in the LIB operand.

IKJ65018I DATA SET ddd RESIDES ON MULTIPLE VOLUMES, NOT SUPPORTED

Explanation: TSO does not support data sets that reside on more than one volume. The operating system's data set catalog indicates that this data set resides on more than one volume. You should reenter the name of a data set that resides on only one volume. If you enter a null-line, the default data set name will be assumed, except for the primary input data set and data sets specified in the LIB operand.

IKJ65019I ddd IS AN UNSUPPORTED OPTION

Explanation: You have specified an option which is supported only by the PL/I Checkout Compiler. The option is ignored.

IKJ65020I TRANSFER COUNT IN FLOW OPTION INVALID+

VALID VALUES ARE 0 THROUGH 32767

Explanation: The first argument of the FLOW option, which specifies the number of transfers of control that are to be included in FLOW and SNAP lists, must specify a value within the range 0 through 32767. You should reenter a correct value. If you enter a null line, the default value will be assumed.

IKJ65021I VALUE MUST BE ALL NUMERIC

Explanation: The value in the argument of the SIZE option (except for the final K) should contain only digits. Reenter the value correctly. If you enter a null line, the default value will be assumed.

IKJ65022I TRANSFER COUNT DEFAULT ASSUMED.

Explanation: You have entered a null line in response to a request to reenter the transfer count for the FLOW option.

IKJ65023I VALUE IN LINECOUNT OPTION INVALID+

VALID VALUES ARE 10 THROUGH 32767

Explanation: The argument of the LINECOUNT option, which specifies the number of lines for each page of the SYSPRINT file, must specify a value within the range 10 through 32767. You should reenter a correct value. If you enter a null line, the default value will be assumed.

IKJ65024I LINECOUNT DEFAULT ASSUMED

Explanation: You have entered a null line in response to a request to reenter the argument of the LINECOUNT option.

IKJ65025I VALUE IN SIZE OPTION INVALID+

VALID VALUES ARE MAX, OR 50K THROUGH 16384K

Explanation: The argument of the SIZE option, which specifies the amount of main storage to be used by the optimizing compiler, must be MAX (indicating that the compiler is to use all of the main storage available in the region) or be within the range 50K through 16384K (or the equivalent values in bytes). You should reenter a correct value. If you enter a null line, the default value will be assumed.

IKJ65026I SIZE DEFAULT ASSUMED

Explanation: You have entered a null line in response to a request to reenter the argument of the SIZE option.

IKJ65027I MEMBER SPECIFIED FOR DATA SET ddd IS IGNORED+

MEMBER NAME FOR LIB DATA SET IS TAKEN FROM INCLUDE STATEMENT

Explanation: The prompter has ignored the member name specified for a partitioned data set in the LIB operand. The LIB operand specifies one or more data sets to be used as secondary input to the preprocessor. The names of the members required from these data sets are taken from %INCLUDE statements in the PL/I source program.

IKJ65028I DATA SET ddd IS NOT A PARTITIONED DATA SET+

THE DATA SET FOR LIB MUST BE A PARTITIONED DATA SET

Explanation: Data sets specified in the LIB operand must have partitioned organization. The LIB operand specifies data sets to be used as secondary input to the preprocessor. You should reenter the name of a partitioned data set. If you enter a null line, the LIB operand is ignored.

IKJ65029I DECK OUTPUT IGNORED+

DECK AND MDECK PRODUCE OUTPUT ON THE SAME DATA SET

Explanation: The output generated by the DECK option and the MDECK option cannot be written on the same data set. NODECK is assumed instead of DECK. This error may have occurred because the prompter uses the same default data set name for both options.

IKJ65034I BLOCK COUNT IN FLOW OPTION INVALID+

VALID VALUES ARE 0 THROUGH 32767

Explanation: The second argument of the FLOW option, which specifies the number of blocks that are to be included in FLOW and SNAP lists, must be within the range 0 through 32767. You should reenter a correct value. If you enter a null line, the default value will be assumed.

IKJ65035I BLOCK COUNT DEFAULT ASSUMED

Explanation: You have entered a null line in response to a request to reenter the block count for the FLOW option.

IKJ65036I OPTIMIZING COMPILER INVOKED

Explanation: The prompter has invoked the PL/I optimizing compiler. Any subsequent messages issued are listed in the first part of this manual.

IKJ65037I INVALID CONTROL FIELD

Explanation: The argument of the CONTROL option is incorrect. The correct "password" established for your installation should have been specified. Processing is terminated.

IKJ65038I SECOND SUBFIELD ON OBJ OPTION IGNORED

Explanation: There must not be two arguments specified for the OBJECT option when using the optimizing compiler. The second argument is ignored.

IKJ65039I INVALID RIGHT HAND MARGIN+

VALUE MUST BE 1 THROUGH 100

Explanation: The second argument of the MARGINS option, which specifies the position of the right hand margin for the PL/I source program, must specify a position from 1 through 100. You should reenter a correct value. If you enter a null line, the default value will be assumed.

IKJ65040I MARGINS OPTION IGNORED.

Explanation: You have entered a null line in response to a request to enter the left or right margin position. The option is ignored.

IKJ65041I INVALID LEFT HAND MARGIN+

VALUE MUST BE LESS THAN RIGHT HAND MARGIN

Explanation: The first argument of the MARGINS option, which specifies the position of the left hand margin for the PL/I source program, must specify a position to the left of the right hand margin specified in the second argument. You should reenter a correct value. If you enter a null line, the default value will be assumed.

IKJ65043I INVALID PRINTER CONTROL CHARACTER POSITION+

PRINTER CONTROL CHARACTER MUST BE OUTSIDE THE LEFT AND RIGHT MARGINS

Explanation: The third argument of the MARGINS option, which specifies the position of a printer control character that is used to format the source listing, must specify a position outside the part of the line used by the PL/I source program. You should reenter a correct value. If you enter a null line, the default value will be assumed.

IKJ65044I PRINTER CONTROL CHARACTER POSITION DEFAULT ASSUMED

Explanation: You have entered a null line in response to a request to reenter the printer control character position.

IKJ65045A REENTER -

Explanation: The prompter is waiting for you to reenter, correctly, the erroneous item specified in the preceding message. If the preceding message ends with a plus-sign, you can still enter a question mark to get more information before reentering the item requested.

IKJ65046I INVALID ARGUMENT IN OPTIMIZE OPTION+

VALID ARGUMENTS ARE 0, 2, TIME OR NO.

Explanation: The argument for the OPTIMIZE option must be 0, 2, NO, or TIME. NO and 0 are equivalent to NOOPTIMIZE; they suppress optimization of the object program. TIME and 2 both cause the object program to be optimized to reduce its execution time. You should reenter a correct value. If you enter a null-line, the default value is assumed.

IKJ65047I OPTIMIZE DEFAULT ASSUMED

Explanation: You have entered a null-line in response to a request to reenter the argument for the OPTIMIZE option.

IKJ65048I INVALID ARGUMENT IN CHARSET OPTION+

VALID ARGUMENTS ARE EBCDIC, BCD, 60, AND 48

Explanation: You have entered the CHARSET option with an invalid argument. You should reenter the invalid argument again correctly. If you enter a null line, the default form of the option will be assumed.

IKJ65049I DEFAULT DATA SET NAME ASSUMED

Explanation: You have entered a null line in response to a request to reenter a data set name.

IKJ65050I CHARSET DEFAULT ASSUMED

Explanation: You have entered a null line in response to a request to reenter a CHARSET argument.

IKJ65051I LIB HAS BEEN SPECIFIED WITH NOMACRO. LIB IS IGNORED

Explanation: The MACRO option has been deleted for your installation, with a default of NOMACRO. Therefore, use of the LIB operand to specify secondary input data sets for the preprocessor has no meaning and is ignored.

IKJ65052I A DATA SET NAME MUST BE SPECIFIED FOR LIB

Explanation: You have entered a null line in response to a request to reenter a data set name for the LIB operand. This is invalid because no default data set name can be assumed by the prompter.

IKJ65053I INVALID RIGHT HAND MARGIN IN SEQUENCE OPTION+

THE RIGHT HAND MARGIN MUST BE 1 THROUGH 100 AND MUST NOT OVERLAP THE SOURCE PROGRAM SPECIFIED BY MARGINS

Explanation: The second argument of the SEQUENCE option, which specifies the position of the right-hand margin for the sequence number, must specify a position from 1 through 100, which does not cause the sequence number to overlap the source program. You should reenter a correct value. If you enter a null-line, the SEQUENCE option is ignored.

IKJ650541 INVALID LEFT HAND MARGIN IN SEQUENCE OPTION+

THE LEFT HAND MARGIN MUST BE LESS THAN THE RIGHT HAND MARGIN AND MUST NOT OVERLAP THE SOURCE PROGRAM SPECIFIED BY MARGINS

Explanation: The first argument of the SEQUENCE option, which specifies the position of the left-hand margin for the sequence number, must specify a position to the left of the right hand margin (specified in the second argument) which does not cause the sequence number to overlap the source program. You should reenter a correct value. If you enter a null-line, the SEQUENCE option is ignored.

IKJ65055I SEQUENCE OPTION IGNORED

Explanation: You have entered a null-line in response to a request to reenter the left or right hand margin of the sequence number.

IKJ65056I INVALID NUMBER OF SUBTASKS IN ISASIZE OPTION+

VALID VALUES ARE 1 THROUGH 255

Explanation: The third argument of the ISASIZE option, which specifies the maximum number of subtasks that will be active at any one time during execution, must specify a number from 0 through 255. You should reenter a correct value. If you enter a null-line, the value 0 is assumed.

IKJ65057I ZERO SUBTASKS ASSUMED

Explanation: You have entered a null-line in response to a request to reenter the number of subtasks for the ISASIZE option.

IKJ65058I LIB HAS BEEN SPECIFIED WITH NOMACRO. MACRO IS ASSUMED

Explanation: The LIB operand has no meaning if the NOMACRO option applies for the compilation. Therefore MACRO is assumed instead of NOMACRO.

IKJ65063I FIRST ARGUMENT IN LIST OPTION INVALID+

VALID VALUES ARE 1 THROUGH 99999999

Explanation: The first argument of the LIST option, which causes the compiler's generated code to be printed, must specify a value in the range 1 through 99999999. You should reenter a correct value. If you enter a null line, the LIST option is passed to the compiler with no arguments.

IKJ65064I SECOND ARGUMENT IN LIST OPTION INVALID+

THE VALUE MUST BE GREATER THAN THE FIRST ARGUMENT

Explanation: The second argument of the LIST option, which causes the compiler's generated code to be listed, must be greater than the first argument. You should reenter a correct value. If you enter a null line, the LIST option is passed to the compiler with no arguments.

IKJ65065I SECOND ARGUMENT IN LIST OPTION IGNORED

Explanation: You have entered a null line as a response to a request to reenter the second argument of the LIST OPTION. The second argument is ignored and the LIST option passed to the compiler with any one argument.

IKJ65066I ARGUMENTS TO LIST IGNORED

You have entered a null line as a response to a request to reenter the five arguments of the LIST option.

IKJ65067I MEMBER mmm SPECIFIED BUT ddd NOT A PARTITIONED DATA SET

Explanation: The specified data set is not a partitioned data set, but a member name has been specified. Check the name of the data set and reenter the correct name. If you enter a null line, the default data-set name will be assumed, except for the primary input data set and data sets specified in the LIB operand.

OS PL/I Optimizing Compiler:
Messages

Order No. SC33-0027-2

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity   Accuracy   Completeness   Organization   Index   Figures   Examples   Legibility

What is your occupation? ______________________________

Number of latest Technical Newsletter (if any) concerning this publication: ____________

Please indicate in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Cut or Fold Along Line

SC33-0027-2

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Cut or Fold Along Line

OS PL/I Optimizing Compiler: Messages (File No. S360/S370-29) Printed in U.S.A. SC33-0027-2

Fold Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department 813(HP)
1133 Westchester Avenue
White Plains, New York 10604

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)