File No. S360-21 (OS)
Order No. GC26-3756-6

OS

IBM

Systems Reference Library

OS Assembler (F) Programmer's Guide

Program Number 360S-AS-037

OS Release 21

This publication complements the IBM System/360 Operating System Assembler Language publication. It provides a guide to program assembling, linkage editing, executing, interpreting listings, assembler programming considerations, diagnostic messages, and object output cards.

Seventh Edition (January, 1972)

This is a major revision of, and obsoletes, GC26-3756-5 and Technical Newsletter GN33-8100. This edition reflects the changes in the assembler listed in the "Summary of Amendments".

This edition applies to release 21 of CS and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are periodically made to specifications herein; before using this publication in connection with the operation of IBM systems, consult the latest SRL Newsletter, Order No. GN20-0360, for the editions that are applicable and current. Changes to the text and to an illustration are indicated by a vertical line to the left of the change.

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

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Nordic Laboratory, Publications Development, Box 962, S-181 09 Lidingo 9, Sweden.

Preface

This publication is oriented to the F level assembler program (the assembler) functioning in the IBM System/360 Operating System (MFT and MVT).

This publication is divided into an introduction and four sections which describe the following:

1. Assembler options and data set requirements.

2. Use of IBM-provided cataloged procedures for assembling; assembling and linkage editing; assembling, linkage editing, and executing assembler language source programs.

3. Use and interpretation of the assembler listing.

4. Programming considerations.

In addition, the appendixes provide a procedure for dynamic invocation of the assembler, a list and explanation of object output cards, and a sample program listing.

Other System Reference Library publications in the IBM System/360 Operating System series provide fuller, more detailed discussions of the topics introduced in this publication: a careful reading of the publication OS Introduction, Order No. GC28-6534, is recommended. Knowledge of the assembler language is assumed. Where appropriate, the reader is directed to the following publications.

OS Job Control Language Reference, Order No. GC28-6704

OS Storage Estimates, Order No. GC28-6551

OS Loader and Linkage Editor, Order No. GC28-6538

OS Supervisor Services and Macro Instructions, Order No. GC28-6646

OS Data Management Macro Instructions, Order No. GC26-3794

OS TESTRAN, Order No. GC28-6648

OS Messages and Codes, Order No. GC28-6631

OS Assembler Language, Order No. GC28-6514

OS Utilities, Order No. GC28-6586

OS FORTRAN IV Library, Order No. GC28-6596

OS MVT Guide, Order No. GC28-6720

OS MFT Guide, Order No. GC27-6939

OS Data Management for System Programmers, Order No. GC28-6550

OS Data Management Services Guide, Order No. GC26-3746

OS FORTRAN IV (E) Programmer's Guide, Order No. GC28-6603

OS FORTRAN IV (G and H) Programmer's Guide, Order No. GC28-6817

OS COBOL (E) Programmer's Guide, Order No. GC24-5029

OS USA Standard COBOL Programmer's Guide, Order No. GC28-6399

Contents

Figures

RELEASE 21 LISTINGS

Maintenance
The sample program listings have been replaced by Release 21 Assembler F listings.

NUM AND STMT OPTIONS

New Programming Feature
The NUM and STMT options are now the default values assumed when the TERM option is specified.

COMPATIBILITY WITH IBM SYSTEM/370 MODELS

Maintenance
The assembler can operate on IBM System/370 Models 135 and up.

MESSAGES

Maintenance
Certain explanations of messages have been rewritten for clarity.

TITLE CHANGES

Maintenance
References to OS publications have been altered to reflect their current titles.

Introduction

Through the medium of job control statements, the programmer specifies job requirements directly to the operating system, thus eliminating many of the functions previously performed by the operating personnel. The job consists of one or more job steps. For example, the job of assembling, linkage-editing, and executing a source program involves three job steps:

1. Translating the source program. i.e., executing the assembler component of the operating system to produce an object module.

2. Processing the output of the assembler, i.e., executing the linkage-editor component of the operating system to produce a load module.

3. Executing the assembled and linkage-edited program, i.e., executing the load module.

A procedure is a sequence of job control language statements specifying a job. Procedures may enter the system via the input stream or from a library of procedures, which are previously defined and contained in a procedure library. The input stream is the flow of job control statements and, optionally, input data entering the system from one input device. At the sequential scheduling system level of the operating system, only one input stream may exist at a time.

The job definition (JOB), execute (EXEC), data definition (DD), and delimiter (/*) job control statements are shown in this publication as they are used to specify assembler processing. Detailed explanations of these statements are given in OS Job Control Language Reference.

Operating system factors influencing program preparation, such as terminating the program, saving and restoring general registers, and linking of independently produced object modules, are discussed in "Programming Considerations", as are guides to determine whether assembler dictionary sizes and complexity limitations of source statements will be exceeded.

Assembler Options and Data Set Requirements

Assembler Options

The programmer may specify the assembler options listed in Figure 1 in the PARM field of the EXEC statement. The options can be coded in any order. They must be separated by commas with no embedded blanks. The entire field must be contained between apostrophes or parentheses. Parentheses allow the PARM field to be continued onto another card, when necessary. If an entry is omitted, a standard setting is assumed by the assembler. The standard default values are underlined in Figure 1.

The options in Figure 1 are defined as follows:

DECK -- The object module is placed on the device specified in the SYSPUNCH DD statement.
LOAD -- The object module is placed on the device specified in the SYSGO DD statement.

Note: Specification of the parameter LOAD causes object output to be written on a data set with ddname SYSGO. This action occurs independently of the output on SYSPUNCH caused by the parameter DECK. The output on SYSGO and SYSPUNCH is identical except that SYSPUNCH is closed with a disposition of LEAVE, and SYSGO is closed with a disposition of REREAD.

LIST -- An assembler listing is produced.
TEST -- The object module contains the special source symbol table required by the test translator (TESTRAN) routine and the TSO Test command processor.
XREF -- The assembler produces a cross-reference table of symbols as part of the listing.
RENT -- The assembler checks for a possible coding violation of program reenterability.

The prefix NO is used with the above options to indicate which options are not wanted.

LINECNT=nn This parameter specifies the number of lines to be printed between headings in the listing. The permissible range is 01 to 99 lines.
NOALGN -- The assembler suppresses the diagnostic message IEU033 ALIGNMENT ERROR if fixed point, floating point, or logical data referenced by an instruction operand is not aligned on the proper boundary. The message will be produced, however, for references to instructions (e.g., by a branch) which are not aligned on the proper (halfword) boundary. See the "Model 85 Programming Considerations" section for information on alignment requirements.
ALGN -- The assembler does not suppress the alignment error diagnostic message.
OS -- The assembler will have complete Operating System Assembler F Capability.
DOS -- The assembler will behave like Disk Operating System (DOS) Assemblers D and F. Anything defined in either of these assemblers with the exception of &SYSPARM will be accepted. CXD, DXD, and OPSYN will be treated as undefined Q-type DC and DS statements and RLDs will appear in the Relocation Dictionary in order of their occurrence (unsorted). The DOS option is incompatible with the LOAD, TEST, RENT, NOALGN, or TERM options. If any of these options are specified along with DOS, the assembler generates a diagnostic message (IEU078).
TERM -- The assembler writes diagnostic information on the SYSTERM data set. Refer to Appendix E for a description of SYSTERM output. Options NUM and STMT can be specified only if TERM is used.
NUM -- The line number field (columns 73-80) or TSO, through the EDIT command, supplied numbers are written on SYSTERM in the beginning of each statement line for which diagnostic information is given. This option is valid only in connection with TERM.
STMT -- Statement number will be written on SYSTERM for statements for which diagnostic information is given. This option is valid only in connection with TERM.

Note 1: It is recommended to use the NUM option when using the TERM option, to avoid unnecessary spacing on a terminal listing. When the TERM option is specified, the NUM and STMT options are taken as the default values.

Note 2: If option NOTERM is used for an assembly, NONUM and NOSTMT will not be listed after *OPTIONS IN EFFECT* in the diagnostic section of the SYSPRINT listing.

If contradictory options are entered, e.g., LIST, NOLIST, the rightmost option, NOLIST, is used.

The following is an example of specifying assembler options:

```
    EXEC PGM=IEUASM,PARM='LOAD,NODECK,TEST'
```

```
PARM='{ DECK   LOAD   LIST   TEST   XREF          nn, ALGN   OS  RENT   TERM   NUM   STMT}'
      {NODECK,NOLOAD,NOLIST,NOTEST,NOXREF,LINECNT=55,NOALGN,DOS,NORENT,NOTERM,NONUM,NOSTMT}
```

Figure 1. Assembler Options

Assembler Data Set Requirements

The assembler requires the following four data sets:

- SYSUT1, SYSUT2, SYSUT3 -- utility data sets used as intermediate external storage.

- SYSIN -- an input data set containing the source statements to be processed.

In addition to the above, four additional data sets may be required:

- SYSLIB -- a data set containing macro definitions (for macro definitions not defined in the source program) and/or source coding to be called for through COPY assembler instructions.

- SYSPRINT -- a data set containing output text for printing (unless NOLIST option is specified).

- SYSPUNCH -- a data set containing object module output usually for punching (unless NODECK option is specified).

- SYSGO -- a data set containing object module output usually for the linkage editor (only if LOAD option is specified).

- SYSTERM -- data set containing diagnostic information (if the TERM option is specified).

The above data sets are described in the following text. The ddname that must be used in the DD statement describing the data set appears as the heading for each description.

DDnames SYSUT1, SYSUT2, SYSUT3

These utility data sets are used by the assembler as intermediate external storage devices when processing the source program. The input/output device(s) assigned to these data sets must be capable of sequential access to records. The assembler does not support multi-volume utility data sets. Refer to the OS Storage Estimates manual for the space required.

DDname SYSIN

This data set contains the input to the assembler -- the source statements to be processed. The input/output device assigned to this data set may be either the device transmitting the input stream, or another sequential input device designated by the programmer. The DD statement describing this data set appears in the input stream. The IBM-supplied procedures do not contain this statement.

DDnames SYSLIB

From this data set, the assembler obtains macro definitions and assembler language statements to be called by the COPY assembler instruction. It is a partitioned data set and each macro definition or sequence of assembler statements is a separate member, with the member name being the macro instruction mnemonic or COPY code name. The data set may be defined as SYS1.MACLIB or a user's private macro definition or COPY library. SYS1.MACLIB contains macro definitions for the system macro instruction provided by IBM. A user's private library may be concatenated with SYS1.MACLIB. The two libraries should have the same attributes, i.e., the same blocking factors, block sizes, and record formats. If different block sizes are used the data sets with the largest block size must be specified first. The OS Job Control Language Reference publication explains the concatenation of the data sets.

DDname SYSPRINT

This data set is used by the assembler to produce a listing. Output may be directed to a printer, magnetic tape, DASD, or a remote terminal (TSO). The assembler uses the machine code carriage-control characters for this data set.

DDname SYSPUNCH

The assembler uses this data set to produce the object module. The input/output unit assigned to this data set may be either a card punch or an intermediate storage device (capable of sequential access).

DDname SYSGO

This is a DASD, magnetic tape, or card punch data set used by the assembler. It contains the same output text as SYSPUNCH. It is used as input for the linkage editor and may also be used as a punch device (see Note under "Assembler Options").

DDname SYSTERM

This data set is used by the assembler to write diagnostic information. The output unit assigned to this data set must be a remote terminal (TSO).

Defining Data Set Characteristics

Before a data set can be made available to a problem program, descriptive information defining the data set must be placed into a data control block for the access routines. Sources of information for the data control block are keyword operands in the DCB macro instruction or, in some cases, the DD statement, data set label, or user's problem program. General information concerning data set definition is contained | in the OS Data Management Services Guide manual. Characteristics of data sets supplied by the DCB macro instruction are | described in the OS Data Management Macro Instructions manual.

The specific information that must be supplied depends upon the data set organization and access method. The following access methods are used to process the assembler data sets:

Access Method	Data Sets
QSAM (Queued Sequential)	SYSPRINT, SYSPUNCH, SYSGO, SYSIN, SYSTERM
BSAM (Basic Sequential)	SYSUT1, SYSUT2, SYSUT3
BPAM (Basic Partitioned)	SYSLIB

Figure 3 summarizes the assembler capabilities and restrictions on record length and format, as well as the blocksize buffering facilities available to the user. The values shown in Figure 3 are based upon | the minimum OS MFT main storage requirements of Assembler F (44K), which will allow a symbol table length of approximately 7000 bytes. If more than the | minimum main storage is available, the block sizes and buffer numbers can be increased. However, if the user specifies a combination of blocking and buffering which does not leave room for the symbol table, either message IEU996 will be issued or abnormal termination of the task will occur (ABEND 804).

In addition to the data set characteristics shown in Figure 3, the following options are available to the user | (refer to the OS Data Management Macro Instructions publication). Options not shown below are fixed by the assembler and cannot be specified.

Data Sets	Options
SYSIN, SYSPUNCH, SYSPRINT, SYSGO	DEVD (device type) BFALN (buffer boundary alignment) BUFL (buffer length) EROPT (error option)
SYSUT1, 2, 3	DEVD (device type) OPTCD (optional service for validity checking and chained scheduling) TRTCH (if 7-track tapes are used, TRTCH=C must be specified)

Return Codes

Figure 2 shows the return codes issued by the assembler for use with the COND=parameter of JOB or EXEC statements. | The COND= parameter is explained in OS Job Control Language Reference.

The return code issued by the assembler is the highest severity code that is:

1. Associated with any error detected by the assembler (see Appendix A for diagnostic messages and severity codes).

2. Associated with MNOTE messages produced by macro instructions.

3. Associated with an unrecoverable I/O error occurring during the assembly.

If a permanent I/O error occurs on any of the assembler files or a DD card for a required data set is missing, or there is insufficient main storage available, a message is printed on SYSPRINT (or on the operator's console if the SYSPRINT DD card is missing or if the I/O error is on SYSPRINT) and a return with a user return code of 20 is given by the assembler. This terminates the assembly.

Return Code	Explanation
0	No errors detected
4	Minor errors detected; successful program execution is probable
8	Errors detected; unsuccessful program execution is possible
12	Serious errors detected: unsuccessful program execution is probable
16	Critical errors detected; normal execution is impossible
20	Unrecoverable I/O error occurred during assembly or missing data sets; assembly terminated

Figure 2. Return Codes

	SYSIN	SYSLIB	SYSTERM SYSPRINT	SYSPUNCH	SYSGO	SYSUT1 SYSUT2 SYSUT3
LRECL	Fixed at 80	Fixed at 80	Fixed at 121	Fixed at 80	Fixed at 80	N/A
RECFM [1]	User must specify in LABEL or DD card F, FS, FBS, FB, FBST, FBT, FT, FST	User must specify in LABEL or DD card F, FB, FBT, FT	F and M set by assembler, user may specify B and/or T in label or DD card FM, FMB, FMT, FMBT	F set by assembler, user may specify B and/or T in label or DD card F, FB, FT, FBT	F set by assembler, user may specify B and/or T in label or DD card F, FB, FT, FBT	Fixed for U
BLKSIZE [2]	User must specify in LABEL or DD card, must be a multiple of LRECL	User must specify in LABEL or DD card, must be a multiple of LRECL	Optional, if omitted BLKSIZE=LRECL	Optional, but must be a multiple of LRECL; if omitted BLKSIZE=LRECL	Optional, but must be a multiple of LRECL; if omitted BLKSIZE=LRECL	Optional, but must be in the range of 550-3624; the value specified on the SYSUT1 DD card is chosen for all three work files; if omitted an adequate value is chosen by the assembler. [4]
BUFNO	Optional; if omitted 2 is used	Set by assembler to 1	Optional; if omitted 2 is used	Optional; if omitted 3 is used for unit record and 1 for other devices	Optional; if omitted 3 is used for unit record and 1 for other devices	User can not specify; either 1 or 2
For 44K availability	BLKSIZE times BUFNO can not be greater than 3600	BLKSIZE can not be greater than 3600 [4]	BLKSIZE times BUFNO can not be greater than 1210	BLKSIZE times BUFNO can not be greater than 400	BLKSIZE times BUFNO can not be greater than 400	BLKSIZE should be the value calculated by the assembler algorithm. [4]
For calculating main storage requirements	L1 = BLKSIZE times BUFNO	L2 = BLKSIZE	L3 = BLKSIZE times BUFNO	L4 = BLKSIZE times BUFNO	L5 = BLKSIZE times BUFNO	

[3] Minimum amount of main storage required for the assembler is the largest of the following:

(1) 45056

(2) $L_1 + L_2 + 41000$

(3) $L_3 + L_4 + L_5 + 41000$

Maximum amount of main storage that the assembler can effectively use is approximately 500,000 bytes

[1] U = undefined, F = fixed length records, B = blocked records, S = standard blocks, T = track overflow, M = machine code carriage control

[2] Blocking is not allowed on unit records devices. Blocking on other direct access can not be greater than the track size unless T is specified on RECFM. If BLKSIZE is not a multiple of LRECL, BLKSIZE is truncated.

[3] For MVT environment add 5,000 for core required

[4] A smaller blocksize may have to be specified for SYSLIB and/or SYSUT 1,2, and 3 if global or local dictionaries overflow. See item 4 under "Correction of Dictionary Overflow."

Figure 3. Data Set Characteristics

Cataloged Procedures

This section describes four IBM-provided cataloged procedures: a procedure fcr assembling (ASMFC), a procedure for assembling and linkage editing (ASMFCL), and a procedure for assembling, linkage editing, and executing (ASMFCLG), and a procedure for assembling and loader-executing (ASMFCG). The procedures rely on conventions regarding the naming of device classes. These conventions, shown in Figure 4, must be incorporated into the system at system generation time.

Device Classname	Devices Assigned
SYSSQ	Any devices allowing sequential access to records for reading and writing
SYSDA	Direct-access devices
SYSCP	Card punches

Figure 4. Device Naming Conventions

To use cataloged procedures, EXEC statements naming the desired procedures are placed in the input stream following the JOB statement. Subsequently, the specified cataloged procedure is brought from a procedure library and merged into the input stream.

Cataloged Procedure for Assembly (ASMFC)

This procedure requests the operating system to load and execute the assembler. The name ASMFC must be used to call this procedure. The result of execution is an object module, in punched card form, and an assembler listing.

In the following example, input enters via the input stream. The statements entered in the input stream to use this procedure are:

```
//jobname      JOB
//stepname     EXEC PROC=ASMFC
//ASM.SYSIN    DD   *
                    |
                    |
          source program statements
                    |
                    |
/*  (delimiter statement)
```

The statements of the ASMFC procedure are brought from the procedure library and merged into the input stream.

Figure 5 shows the statements that make up the ASMFC procedure.

```
1 //ASM        EXEC  PGM=IEUASM,REGION=50K

2 //SYSLIB     DD    DSNAME=SYS1.MACLIB,DISP=SHR

3 //SYSUT1     DD    DSNAME=&SYSUT1,UNIT=SYSSQ,SPACE=(1700,(400,50)),       X
  //                 SEP=(SYSLIB)

4 //SYSUT2     DD    DSNAME=&SYSUT2,UNIT=SYSSQ,SPACE=(1700,(400,50))

5 //SYSUT3     DD    DSNAME=&SYSUT3,SPACE=(1700,(400,50)),                  X
  //                 UNIT=(SYSSQ,SEP=(SYSUT2,SYSUT1,SYSLIB))

6 //SYSPRINT   DD    SYSOUT=A

7 //SYSPUNCH   DD    SYSOUT=B
     --------
     --------
```

1 PARM= or COND=parameters may be added to this statement by the EXEC statement that calls the procedure (see Overriding Statements in Cataloged Procedures). The system name IEUASM identifies Assembler F.

2 This statement identifies the macro library data set. The data set name SYS1.MACLIB is an IBM designation.

3 4 5 These statements specify the assembler utility data sets. The device classname used here, SYSSQ, may represent a collection of tape drives, or direct-access units, or both. The I/O units assigned to this name are specified by the installation when the system is generated. A unit name, e.g., 2311 may be substituted for SYSSQ. The DSNAME parameters guarantee use of Dedicated Workfiles if this feature is part of the Scheduler.

The SEP=subparameter in statement 5 and the SPACE=parameter in statements 3, 4, and 5 are effective only if the device assigned is a direct-access device: otherwise they are ignored. The space required is dependent on the make-up of the source program. The OS Job Control Language Reference publication explains space allocation.

6 This statement defines the standard system output class, SYSOUT=A, as the destination for the assembler listing.

7 This statement describes the data set that will contain the object module produced by the assembler.

Figure 5. Cataloged Procedure for Assembly (ASMFC)

Cataloged Procedure for Assembly and Linkage Editing (ASMFCL)

This procedure consists of two job steps: assembling and linkage editing. The name ASMFCL must be used to call this procedure. Execution of this procedure results in the production of an assembler listing, a linkage editor listing, and a load module.

The following example assumes input to the assembler via the input job stream. It also makes provision in the //LKED job step for concatenating the input to the linkage editor from the //ASM job step with any additional linkage editor input in the input job stream. This additional input can be a previously produced object module which is to be linked to the object module produced by job step //ASM.

An example of the statements entered in the input stream to use this procedure is:

```
//jobname       JOB
//stepname      EXEC PROC=ASMFCL
//ASM.SYSIN     DD    *
                 .
                 .
    source program statements
                 .
                 .
/*               .
//LKED.SYSIN    DD    *  }
                 .       }
                 .       }  necessary only if linkage
    object module or     }  editor is to combine modules
    linkage editor       }  or read linkage editor control
    control statements   }  information from the job stream
/*                       }
```

The procedure is brought from the procedure library and merged into the input stream.

Figure 6 shows the statements that make up the ASMFCL procedure. Only those statements not previously discussed are explained.

```
  //ASM        EXEC  PGM=IEUASM,PARM=LOAD,REGION=50K
  //SYSLIB     DD    DSNAME=SYS1.MACLIB,DISP=SHR
  //SYSUT1     DD    DSNAME=&SYSUT1,UNIT=SYSSQ,SPACE=(1700,(400,50)),          X
  //                 SEP=(SYSLIB)
  //SYSUT2     DD    DSNAME=&SYSUT2,UNIT=SYSSQ,SPACE=(1700,(400,50))
  //SYSUT3     DD    DSNAME=&SYSUT3,SPACE=(1700,(400,50)),                     X
  //                 UNIT=(SYSSQ,SEP=(SYSUT2,SYSUT1,SYSLIB))
  //SYSPRINT   DD    SYSOUT=A
  //SYSPUNCH   DD    SYSOUT=B
1 //SYSGO      DD    DSNAME=&LOADSET,UNIT=SYSSQ,SPACE=(80,(200,50)),           X
  //                 DISP=(MOD,PASS)
2 //LKED       EXEC  PGM=IEWL,PARM=(XREF,LIST,NCAL),REGION=96K,                X
  //                 COND=(8,LT,ASM)
3 //SYSLIN     DD    DSNAME=&LOADSET,DISP=(OLD,DELETE)
4 //           DD    DDNAME=SYSIN
5 //SYSLMOD    DD    DSNAME=&GOSET(GO),UNIT=SYSDA,SPACE=(1024,(50,20,1)),      X
  //                 DISP=(MOD,PASS)
6 //SYSUT1     DD    DSNAME=&SYSUT1,UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),         X
  //                 SPACE=(1024,(50,20))
7 //SYSPRINT   DD    SYSOUT=A
    --------
```

1 In this procedure the SYSGO DD statement describes a temporary data set -- the object module -- which is to be passed to the linkage editor.

2 This statement initiates linkage editor execution. The linkage editor options in the PARM=field cause the linkage editor to produce a cross-reference table, module map, and a list of all control statements processed by the linkage editor. The NCAL option suppresses the automatic library call function of the linkage editor.

3 This statement identifies the linkage editor input data set on the same one produced as output by the assembler.

4 This statement is used to concatenate any input to the linkage editor from the input stream with the input from the assembler.

5 This statement specifies the linkage-editor output data set (the load module). As specified, the data set will be deleted at the end of the job. If it is desired to retain the load module, the DSNAME parameter must be respecified and a DISP parameter added. See "Overriding Statements in Cataloged Procedures". If the output of the linkage editor is to be retained, the DSNAME parameter must specify a library name and member name where the load module is to be placed. The DISP parameter must specify either KEEP or CATLG.

6 This statement specifies the utility data set for the linkage editor.

7 This statement identifies the standard output class as the destination for the linkage editor listing.

Figure 6. Cataloged Procedure fcr Assembling and Linkage Editing (ASMFCL)

Cataloged Procedure for Assembly, Linkage Editing, and Execution (ASMFCLG)

This procedure consists of three job steps: assembling, linkage editing, and executing.

Figure 7 shows the statements that make up the ASMFCLG procedure. Only those statements not previously discussed are explained in the figure.

The name ASMFCLG must be used to call this procedure. Assembler and linkage editor listings are produced.

The statements entered in the input stream to use this procedure are:

```
//jobname                JOB
//stepname               EXEC PROC=ASMFCLG
//ASM.SYSIN              DD    *
                          |
      source program statements
                          |
/*                        |           }
//LKED.SYSIN             DD    *      }  necessary only if linkage editor
                          |           }  is to combine modules or read
            object module or          }  linkage editor control information
            linkage editor            }  from job stream
            control statements        }
                          |           }
/*                        |
//GO.ddname              DD  (parameters) }
//GO.ddname              DD  (parameters) }
//GO.ddname              DD    *          }  only if necessary
                          |               }
          problem program input           }
                          |
/*
```

```
  //ASM         EXEC  PGM=IEUASM,PARM=LOAD,REGION=50K
  //SYSLIB      DD    DSNAME=SYS1.MACLIB,DISP=SHR
  //SYSUT1      DD    DSNAME=&SYSUT1,UNIT=SYSSQ,SPACE=(1700,(400,50)),        X
  //                  SEP=(SYSLIB)
  //SYSUT2      DD    DSNAME=&SYSUT2,UNIT=SYSSQ,SPACE=(1700,(400,50))
  //SYSUT3      DD    DSNAME=&SYSUT3,SPACE=(1700,(400,50)),                   X
  //                  UNIT=(SYSSQ,SEP=(SYSUT2,SYSUT1,SYSLIB))
  //SYSPRINT    DD    SYSOUT=A
  //SYSPUNCH    DD    SYSOUT=B
  //SYSGO       DD    DSNAME=&LOADSET,UNIT=SYSSQ,SPACE=(80,(200,50)),         X
  //                  DISP=(MOD,PASS)
1 //LKED        EXEC  PGM=IEWL,PARM=(XREF,LET,LIST,NCAL),REGION=96K,          X
  //                  COND=(8,LT,ASM)
  //SYSLIN      DD    DSNAME=&LOADSET,DISP=(OLD,DELETE)
  //            DD    DDNAME=SYSIN
2 //SYSLMOD     DD    DSNAME=&GOSET(GO),UNIT=SYSDA,SPACE=(1024,(50,20,1)),    X
  //                  DISP=(MOD,PASS)
  //SYSUT1      DD    DSNAME=&SYSUT1,UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),      X
  //                  SPACE=(1024,(50,20))
  //SYSPRINT    DD    SYSOUT=A
3 //GO          EXEC  PGM=*.LKED.SYSLMOD,COND=((8,LT,ASM),(4,LT,LKED))
          --------
          --------
```

1 The LET linkage-editor option specified in this statement causes the linkage editor to mark the load module as executable even though errors were encountered during processing.

2 The output of the linkage editor is specified as a member of a temporary data set, residing on a direct-access device, and is to be passed to a succeeding job step.

3 This statement initiates execution of the assembled and linkage edited program. The notation *.LKED.SYSLMOD identifies the program to be executed as being in the data set described in job step LKED by the DD statement named SYSLMOD. When running with MVT the REGION parameter can be calculated with the help of the OS Storage Estimates publication.

Figure 7. Cataloged Procedure for Assembly, Linkage Editing and Execution (ASMFCLG)

Cataloged Procedure for Assembly and Loader Execution (ASMFCG)

This procedure consists of two job steps assembling and loader-executing. The result of loader-execution is a combination of linkage-editing and loading the program for execution. Load modules for program libraries are not produced.

Figure 8 shows the statements that make up the ASMFCG procedure. Only those statements not previously discussed are explained in the figure.

The name ASMFCG must be used to call this procedure. Assembler and loader listings are produced.

The statements entered in the input stream to use this procedure are:

```
//jobname       JOB
//stepname      EXEC          PROC=ASMFCG
//ASM.SYSIN     DD            *
                 |
                 |
                 |
           source program
                 |
/*               |
//GO.ddname      DD           (parameters) ┐
//GO.ddname      DD           (parameters) │ only
//GO.ddname      DD           *            │ if
                 |                         │ necessary
                 |                         │
        problem program input              ┘
                 |
/*               |
```

Overriding Statements in Cataloged Procedures

Any parameter in a cataloged procedure can be overridden except the PGM= parameter in the EXEC statement. Such overriding of statements or fields is effective only for the duration of the job step in which the statements appear. The statements, as stored in the procedure library of the system, remain unchanged.

Overriding for the purposes of respecification, addition, or nullification is accomplished by including in the input stream statements containing the desired changes and identifying the statements to be overridden.

EXEC Statements

The PARM= and COND= parameters can be added or, if present, re-specified by including in the EXEC statement calling the procedure the notation PARM.stepname=, or COND.stepname=, followed by the desired parameters. "Stepname" identifies the EXEC statement within the procedure to which the modification applies. Overriding the PGM= parameter is not possible.

If the procedure consists of more than one job step, a PARM.stepname= or COND.stepname= parameter may be entered for each step. The entries must be in order, i.e., PARM.step1=, PARM.step2=, etc.

DD Statements

All parameters in the operand field of DD statements may be overridden by including in the input stream (following the EXEC card calling the procedure) a DD statement with the notation //stepname.ddname in the name field. "Stepname" refers to the job step in which the statement identified by "ddname" appears.

Examples

In the assembly procedure ASMFC (Figure 5), the production of a punched object deck could be suppressed and the UNIT= and SPACE= parameters of data set SYSUT1 re-specified, by including the following statements in the input stream:

```
//stepname     EXEC    PROC=ASMFC,                  X
//                     PARM.ASM=NODECK
//ASM.SYSUT1   DD      UNIT=2311,                   X
//                     SPACE=(200,(300,40))
//ASM.SYSIN    DD      *
```

In procedure ASMFCLG (Figure 7), suppressing production of an assembler listing and adding the COND= parameter to the EXEC statement, which specifies execution of the linkage editor, may be desired. In this case, the EXEC statement in the input stream would appear as follows:

```
//stepname   EXEC   PROC=ASMFCLG,                         X
//                  PARM.ASM=(NOLIST,LOAD),               X
//                  COND.LKED=(8,LT,stepname.ASM)
```

Note: Overriding the LIST parameter effectively deletes the PARM=LOAD so this must be repeated in the override statement.

For current execution of procedure ASMFCLG, no assembler listing would be produced, and execution of the linkage editor job step //LKED would be suppressed if the return code issued by the assembler (step ASM) was greater than 8.

Using the procedure ASMFCL (Figure 6) to:

1. Read input from a non-labeled 9-track tape on unit 282 that has a standard blocking factor of 10.
2. Put the output listing on a labeled tape TAPE10, with a data set name of PROG1 and a blocking factor of 5.
3. Block the SYSGO output of the assembler and use it as input to the linkage editor with a blocking factor of 5.

```
    //ASM        EXEC   PGM=IEUASM,PARM='LOAD',REGION=50K
    //SYSLIB     DD     DSNAME=SYS1.MACLIB,DISP=SHR
    //SYSUT1     DD     DSNAME=&SYSUT1,UNIT=SYSSQ,SPACE=(1700,(400,50)),        X
    //                  SEP=(SYSLIB)
    //SYSUT2     DD     DSNAME=&SYSUT2,UNIT=SYSSQ,SPACE=(1700,(400,50))
    //SYSUT3     DD     DSNAME=&SYSUT3,SPACE=(1700,(400,50)),                   X
    //                  UNIT=(SYSSQ,SEP=(SYSUT2,SYSUT1,SYSLIB))
    //SYSPRINT   DD     SYSOUT=A
    //SYSPUNCH   DD     SYSOUT=B
    //SYSGO      DD     DSNAME=&LOADSET,UNIT=SYSSQ,SPACE=(80,(200,50)),         X
    //                  DISP=(MOD,PASS)
1   //GO         EXEC   PGM=LOADER,PARM='MAP,PRINT,NOCALL,LET'
2   //SYSLIN     DD     DSNAME=&LOADSET,DISP=(OLD,DELETE)
3   //SYSLOUT    DD     SYSOUT=A
          --------
          --------
```

1 This statement initiates loader-execution. The loader options in the PARM=field cause the loader to produce a map, print the map and diagnostics. The NOCALL option is the same as NCAL for linkage editor and the LET option is the same as for linkage editor.

2 This statement defines the loader input data set as the same one produced as output by the assembler.

3 This statement identifies the standard output class as the destination for the loader listing.

Figure 8. Cataloged Procedure for Assembly and Loader-Execution (ASMFCG)

4. Linkage edit the module only if there are no errors in the assembly, i.e., COND=0.

5. Linkage edit on to a previously allocated and cataloged data set USER.LIBRARY with a member name of PROG.

The input stream appears as follows:

```
//jobname          JOB
//stepname         EXEC   PROC=ASMFCL,                            X
//                        COND.LKED=(0,NE,stepname.ASM)
//ASM.SYSPRINT     DD     DSNAME=PROG1,UNIT=TAPE,                 X
//                        VOLUME=SER=TAPE10,DCB=(BLKSIZE=605)
//ASM.SYSGO        DD     DCB=(BLKSIZE=400)
//ASM.SYSIN        DD     UNIT=282,LABEL=(,NL),                   X
//                        DCB=(RECFM=FSB,BLKSIZE=800)
//LKED.SYSIN       DD     DCB=stepname.ASM.SYSGO
//LKED.SYSLMOD     DD     DSNAME=USER.LIBRARY(PROG),DISP=OLD
/*
```

Note: The order of appearance of ddnames within job steps ASM and LKED has been preserved. Thus, SYSPRINT precedes SYSGO within step ASM. The ddname ASM.SYSIN was placed last since SYSIN does not occur at all within step ASM. These points are covered in "Appendix A. Using Cataloged and In-stream Procedures" in the OS Job Control Language Reference manual.

To assemble two programs, linkage edit the two assemblies into one load module and execute the load module, using the cataloged procedures described above, the input stream appears as follows:

```
//stepname1    EXEC    PROC=ASMFC,PARM.ASM='LOAD'
//ASM.SYSGO    DD      DSNAME=&LOADSET,UNIT=SYSSQ,            X
//                     SPACE=(80,(200,50)),                   X
//                     DISP=(MOD,PASS),DCB=(BLKSIZE=400)
//ASM.SYSIN    DD      *
                       '
                       '
                       '
                       source program 1 statements
                       '
                       '
/*                     '
//stepname2    EXEC    PROC=ASMFCLG
//ASM.SYSGO    DD      DCB=(BLKSIZE=400),DISP=(MOD,PASS)
//ASM.SYSIN    DD      *
                       '
                       '
                       '
                       source program 2 statements
                       '
                       '
                       '
/*                     '
//LKED.SYSLIN  DD      DCB=BLKSIZE=400
//LKED.SYSIN   DD      *
               ENTRY   PROG
/*
//GO.ddname            dd cards for GO step
```

The OS Job Control Language Reference publication provides an additional description of overriding techniques.

Assembler Listing

The assembler listing (Figure 10) consists of five sections, ordered as follows: external symbol dictionary items, the source and object program statements, relocation dictionary items, symbol cross reference table, and diagnostic messages. In addition, three statistical messages may appear in the listing:

1. After the diagnostics, a statements-flagged message indicates the total number of statements in error. It appears as follows: nnn STATEMENTS FLAGGED IN THIS ASSEMBLY.

2. After the statements-flagged message, the assembler prints the highest severity code encountered (if non-zero). This is equal to the assembler return code. The message appears as follows: nn WAS HIGHEST SEVERITY CODE.

3. After the severity code, the assembler prints a count of the number of records read from SYSIN and from SYSLIB. It also prints the options for the assembly. (See the section "Assembler Options".) These messages appear as follows:

   STATISTICS SOURCE RECORDS (SYSIN) = nnnnn SOURCE RECORDS (SYSLIB) = nnnnn
   OPTIONS IN EFFECT xxxx,xxxxxx, etc.

4. After the options in effect, the assembler prints a count of lines printed, which appears as follows: nnn PRINTED LINES. This is a count of the actual number of 121-byte records generated by the assembler; it may be less than the total number of printed and blank lines appearing on the listing if the SPACE n assembler instruction is used. For a SPACE n that does not cause an eject, the assembler inserts n blank lines in the listing by generating n/3 blank 121-byte records -- rounded to the next lower integer if a fraction results; e.g., for a SPACE 2, no blank records are generated. The assembler does not generate a blank record to force a page eject.

In addition to the above items, the assembler prints the deck identification (as specified in the TITLE statement) and current date on every page of the listing. If the timer is available, the assembler prints the time of day to the left of the date on page 1 of the ESD listing. This is the time when printing starts, rather than the start of the assembly, and is intended only to provide unique identification for assemblies made on the same day. The time is printed as hh.mm, where hh is the hour of the day (midnight beginning at 00), and mm is the number of minutes past the hour.

External Symbol Dictionary (ESD)

This section of the listing contains the external symbol dictionary information passed to the linkage-editor or loader in the object module. The entries describe the control sections, external references, and entry points in the assembled program. There are six types of entries, shown in Figure 9, along with their associated fields. The circled numbers refer to the corresponding heading in the sample listing (Figure 10). The X's indicate entries accompanying each type designation.

❶ SYMBOL	❷ TYPE	❸ ID	❹ ADDR	❺ LENGTH	❻ LD ID
X	SD	X	X	X	-
X	LD	-	X	-	X
X	ER	X	-	-	-
-	PC	X	X	X	-
-	CM	X	X	X	-
X	XD	X	X	X	-
X	WX	X	-	-	-

Figure 9. Types of ESD Entries

❶ This column contains the name of every external dummy section, control section, entry point, and external symbol.

❷ This column contains the type designator for the entry, as shown in the figure. The type designators are defined as:

SD--Names section definition. The symbol appeared in the name field of a CSECT or START statement.
LD--The symbol appeared as the operand of the ENTRY statement.
ER--External reference. The symbol appeared as the operand of an EXTRN

statement, or was defined as a V-type address constant.

PC--Unnamed control section (private code) definition.

CM--Common control section definition.

XD--External dummy section (same as PR, Pseudo Register in the Linkage Editor manual).

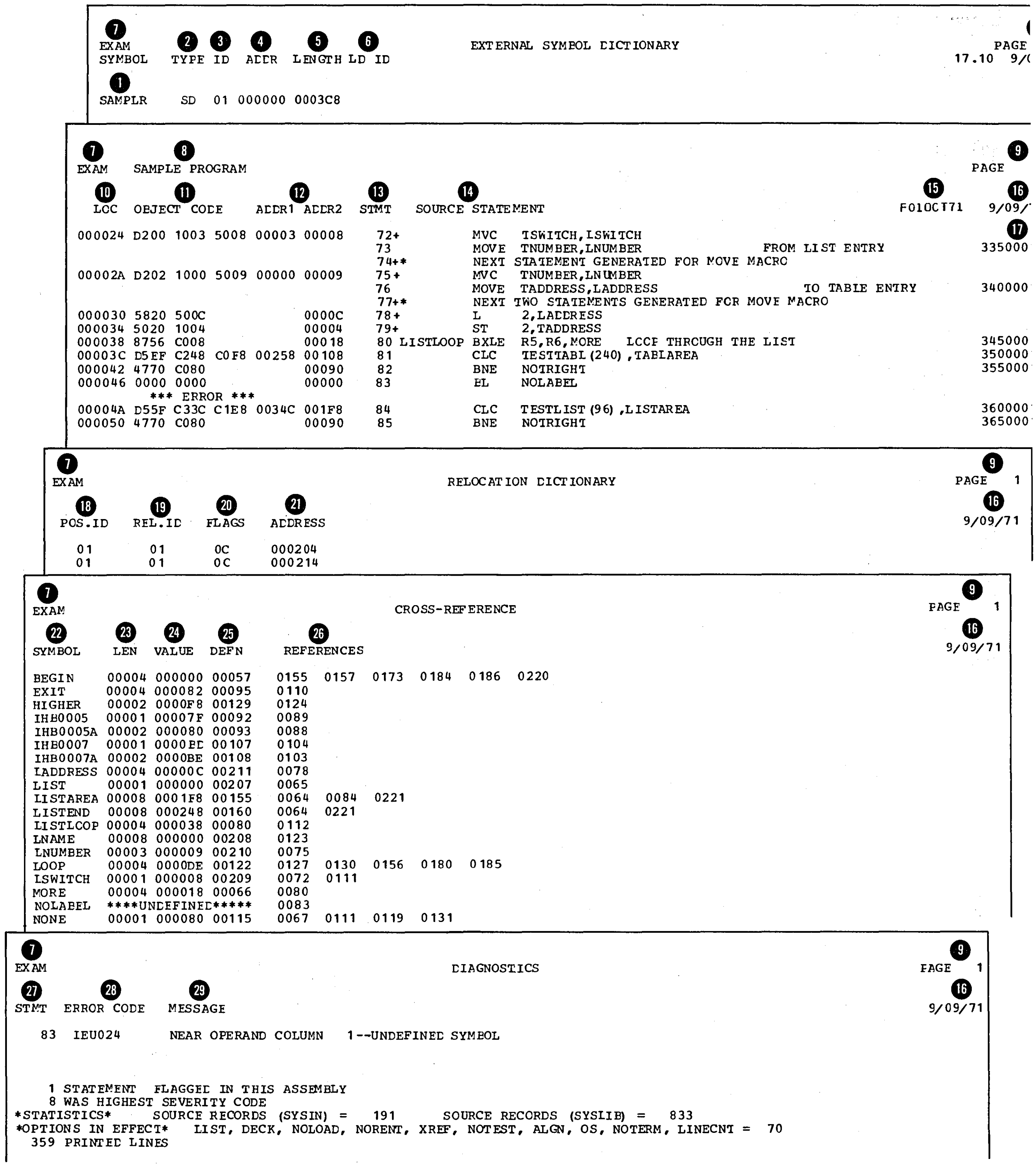
WX--Weak external reference. The symbol appeared as the operand of a WXTRN statement.

3 This column contains the external symbol dictionary identification number (ESDID). The number is a unique two-digit hexadecimal number identifying the entry. It is used by the LD entry of the ESD and by the relocation dictionary for cross-referencing the ESD.

4 This column contains the address of the symbol (hexadecimal notation) for SD- and LD-type entries, and zeros for ER- and WX-type entries. For PC- and CM-type entries, it indicates the beginning address of the control section. For XD-type entries, it indicates the alignment by printing a number one less than the number of bytes in the unit of alignment, e.g., 7 indicates double word alignment.

5 This column contains the assembled length, in bytes, of the control section (hexadecimal notation).

6 This column contains, for LD-type entries, the identification (ID) number assigned to the ESD entry that identifies the control section in which the symbol was defined.


7
2
3
4
5
6
1
7
8
9
10
11
12
13
14
15
16
17
7
9
18
19
20
21
16
7
9
22
23
24
25
26
16
7
9
27
28
29
16

```
EXAM                                                EXTERNAL SYMBOL DICTIONARY                                       PAGE
SYMBOL   TYPE ID  ADDR   LENGTH LD ID                                                                              17.10  9/(

SAMPLR    SD   01 000000 0003C8
```

```
EXAM   SAMPLE PROGRAM                                                                                             PAGE

  LOC  OBJECT CODE       ADDR1 ADDR2  STMT   SOURCE STATEMENT                                              F01OCT71    9/09/

000024 D200 1003 5008 00003 00008     72+           MVC   ISWITCH,LSWITCH
                                      73            MOVE  TNUMBER,LNUMBER             FROM LIST ENTRY            335000
                                      74++          NEXT STATEMENT GENERATED FOR MOVE MACRO
00002A D202 1000 5009 00000 00009     75+           MVC   TNUMBER,LNUMBER
                                      76            MOVE  TADDRESS,LADDRESS                 TO TABLE ENTRY      340000
                                      77++          NEXT TWO STATEMENTS GENERATED FOR MOVE MACRO
000030 5820 500C              0000C   78+           L     2,LADDRESS
000034 5020 1004              00004   79+           ST    2,TADDRESS
000038 8756 C008              00018   80 LISTLOOP BXLE  R5,R6,MORE   LOOP THROUGH THE LIST                      345000
00003C D5EF C248 C0F8 00258 00108     81            CLC   TESTTABL(240),TABLAREA                                350000
000042 4770 C080              00090   82            BNE   NOTRIGHT                                              355000
000046 0000 0000              00000   83            BL    NOLABEL
        *** ERROR ***
00004A D55F C33C C1E8 0034C 001F8     84            CLC   TESTLIST(96),LISTAREA                                 360000
000050 4770 C080              00090   85            BNE   NOTRIGHT                                              365000
```

```
EXAM                                          RELOCATION DICTIONARY                                        PAGE   1

 POS.ID    REL.ID    FLAGS    ADDRESS                                                                      9/09/71

   01        01       0C      000204
   01        01       0C      000214
```

```
EXAM                                     CROSS-REFERENCE                                          PAGE   1

SYMBOL     LEN   VALUE  DEFN      REFERENCES                                                      9/09/71

BEGIN     00004 000000 00057    0155  0157  0173  0184  0186  0220
EXIT      00004 000082 00095    0110
HIGHER    00002 0000F8 00129    0124
IHB0005   00001 00007F 00092    0089
IHB0005A  00002 000080 00093    0088
IHB0007   00001 0000BD 00107    0104
IHB0007A  00002 0000BE 00108    0103
LADDRESS  00004 00000C 00211    0078
LIST      00001 000000 00207    0065
LISTAREA  00008 0001F8 00155    0064  0084  0221
LISTEND   00008 000248 00160    0064  0221
LISTLOOP  00004 000038 00080    0112
LNAME     00008 000000 00208    0123
LNUMBER   00003 000009 00210    0075
LOOP      00004 0000DE 00122    0127  0130  0156  0180  0185
LSWITCH   00001 000008 00209    0072  0111
MORE      00004 000018 00066    0080
NOLABEL   ****UNDEFINED*****    0083
NONE      00001 000080 00115    0067  0111  0119  0131
```

```
EXAM                                          DIAGNOSTICS                                          PAGE   1

STMT  ERROR CODE    MESSAGE                                                                       9/09/71

   83  IEU024        NEAR OPERAND COLUMN   1--UNDEFINED SYMBOL


    1 STATEMENT  FLAGGED IN THIS ASSEMBLY
    8 WAS HIGHEST SEVERITY CODE
*STATISTICS*     SOURCE RECORDS (SYSIN) =    191      SOURCE RECORDS (SYSLIB) =    833
*OPTIONS IN EFFECT*   LIST, DECK, NOLOAD, NORENT, XREF, NOTEST, ALGN, OS, NOTERM, LINECNT =   70
  359 PRINTED LINES
```

Figure 10. Assembler Listing

Source and Object Program

This section of the listing documents the source statements and the resulting object program.

7 This is the four-character deck identification. It is the symbol that appears in the name field of the first TITLE statement. The assembler prints the deck identification and date (item 16) on every page of the listing.

8 This is the information taken from the operand field of a TITLE statement.

Note: TITLE, SPACE and EJECT statements will not appear in the source listing unless the statement is continued onto another card. Then the first card of the statement is printed. However, any of these three types of statements, if generated as macro instruction expansion, will never be listed regardless of continuation.

9 Listing page number. Each section of the listing starts with page 1.

10 This column contains the assembled address (hexadecimal notation) of the object code.

11 This column contains the object code produced by the source statement. The entries are always left-justified. The notation is hexadecimal. Entries are machine instructions or assembled constants. Machine instructions are printed in full with a blank inserted after every four digits (two bytes). Constants may be only partially printed (see the PRINT assembler instruction in the OS Assembler Language publication).

12 These two columns contain effective addresses (the result of adding together a base register value and displacement value):

a. The column headed ADDR1 contains the effective address for the first operand of an SS instruction.
b. The column headed ADDR2 contains the effective address of the second operand of any instruction referencing storage.

Both address fields contain six digits; however, if the high-order digit is a zero, it is not printed.

13 This column contains the statement number. A plus sign (+) to the right of the number indicates that the statement was generated as the result of macro instruction processing.

14 This column contains the source program statement. The following items apply to this section of the listing:

a. Source statements are listed, including those brought into the program by the COPY assembler instruction, and including macro definitions submitted with the main program for assembly. Listing control instructions are not printed, except for the following case: PRINT is listed when PRINT ON is in effect and a PRINT statement is encountered.
b. Macro definitions obtained from SYSLIB are not listed.
c. The statements generated as the result of a macro instruction follow the macro instruction in the listing.
d. Assembler or machine instructions in the source program that contain variable symbols are listed twice: as they appear in the source input, and with values substituted for the variable symbols.
e. Diagnostic messages are not listed inline in the source and object program section. An error indicator, ***ERROR***, follows the statement in error. The message appears in the diagnostic section of the listing.
f. MNOTE messages are listed inline in the source object program section. An MNOTE indicator appears in the diagnostic section of the listing for MNOTE statements other than MNOTE*. The MNOTE message format is serverity code, message text.
g. The MNOTE* form of the MNOTE statements results in an inline message only. An MNOTE indicator does not appear in the diagnostic section of the listing.
h. When an error is found in a programmer macro definition, it is treated the same as any other assembly error: the error indication appears after the statement in error, and a diagnostic is placed in the list of diagnostics. However, when an error is encountered during the expansion of a macro instruction (system- or programmer-defined) the error indication appears in place of the erroneous statement which is not listed. The error indication follows the last statement listed before the erroneous statement was encountered, and the associated diagnostic message is placed in the

list of diagnostics.

i. Literals that have not been assigned locations by an LTORG statement appear in the listing following the END statement. Literals are identified by the equal (=) sign preceding them.

j. If the END statement contains an operand, the transfer address appears in the location column (LOC).

k. In the case of COM, CSECT, and DSECT statements, the location field contains the beginning address of these control sections, i.e., the first occurrence.

l. In the case of EXTRN, WXTRN, ENTRY, and DXD instructions, the location field and object code field are blank.

m. For a USING statement, the location field contains the value of the first operand.

n. For LTORG and ORG statements, the location field contains the location assigned to the literal pool or the value of the ORG operand.

o. For an EQU statement, the location field contains the value assigned.

p. Generated statements always print in normal statement format. Because of this, it is possible for a generated statement to occupy three or more continuation lines on the listing. This is unlike source statements, which are restricted to two continuation lines.

Note: When the listing is directed to a terminal under TSO, the following items apply to ICTL, EJECT, and SPACE:

ICTL -	the end column, operand e, must be within 41-71.
EJECT-	only one blank line is created on the terminal listing.
SPACE-	the decimal value specified in the operand is divided by three, and the integer result indicates the number of blank lines created.

15 This column contains the identifier of the assembler (F) and the date when this version was released by System Development Division to DPD Program Information Department.

16 Current date (date run is made).

17 Identification-sequence field from the source statement.

Relocation Dictionary

This section of the listing contains the relocation dictionary information passed to the linkage editor in the object module. The entries describe the address constants in the assembled program that are affected by relocation.

18 This column contains the external symbol dictionary ID number assigned to the ESD entry that describes the control section in which the address constant is used as an operand.

19 This column contains the external symbol dictionary ID number assigned to the ESD entry that describes the control section in which the referenced symbol is defined.

20 The two-digit hexadecimal number in this column is interpreted as follow:

First Digit. A zero indicates that the entry describes an A-type or Y-type address constant. A one indicates that the entry describes a V-type address constant . A two indicates that the entry describes a Q-type address constant. A three describes a CXD entry.
Second Digit. The first three bits of this digit indicate the length of the constant and whether the base should be added or subtracted:

Bits 0 and 1	Bit 2
00 = 1 byte	0 = +
01 = 2 bytes	1 = -
10 = 3 bytes	
11 = 4 bytes	

21 This column contains the assembled address of the field where the address constant is stored.

Cross Reference

This section of the listing information concerns symbols which are defined and used in the program.

22 This column contains the symbols.

23 This column states the length (decimal notation), in bytes, of the field occupied by the symbol value.

24 This column contains either the address the symbol represents, or a value to which the symbol is equated.

25 This column contains the statement number of the statement in which the symbol was defined.

26 This column contains the statement numbers of statements in which the symbol appears as an operand. In the

case of a duplicate symbol, the assembler fills this column with the message:

****DUPLICATE****

The following notes apply to the cross-reference section:

- Symbols appearing in V-type address constants do not appear in the cross-reference listing.
- A PRINT OFF listing control instruction does not affect the production of the cross-reference section of the listing.
- In the case of an undefined symbol, the assembler fills columns 23, 24, and 25 with the message:

  ****UNDEFINED****.

Diagnostics

This section contains the diagnostic messages issued as a result of error conditions encountered in the program. The text, severity code, and explanatory notes for each message are contained in "Appendix A".

27 This column contains the number of the statement in error.

28 This column contains the message identifier.

29 This column contains the message, and, in most cases, an operand column pointer that indicates the vicinity of the error. In the following example, the approximate location of the addressability error occurred in the 9th column of the operand field:

Example:

```
STMT   ERROR CODE   MESSAGE

 21    IEU035       NEAR OPERAND COLUMN 9 -- ADDRESSABILITY ERROR
```

The following notes apply to the diagnostic section:

- An MNOTE indicator of the form NMOTE STATEMENT appears in the diagnostic section if an MNOTE statement other than MNOTE * is issued by a macro instruction. The MNOTE statement itself is inline in the source and object program section of the listing. The operand field of an MNOTE * is printed as a comment, but does not appear in the diagnostic section.
- A message identifier consists of six characters and is of the form: IEUxxx

  IEU identifies the issuing agent as Assembler F, and xxx is a unique number assigned to the message.

Note: Editing errors in system macro definitions (macro definitions included in a macro library) are discovered when the macro definitions are read from the macro library. This occurs after the END statement has been read. They will therefore be flagged after the END statement. If the programmer does not know which of his system macros caused an error it is necessary to punch all system macro definitions used in the program, including inner macro definitions, and insert them in the program as programmer macro definitions, since the programmer macro definitions are flagged inline. To aid in debugging it is advisable to test all macro definitions as programmer macro definitions before incorporating them in a library as system macro definitions.

Programming Considerations

This section consists of a number of discrete subjects about assembler language programming.

Saving and Restoring General Register Contents

A problem program should save the values contained in the general register upon commencing execution and, upon completion, restore to the general registers these same values. Thus, as control is passed from the operating system to a problem program and, in turn, to a subprogram, the status of the registers used by each program is preserved. This is done through use of the SAVE and RETURN system macro instructions.

The SAVE macro instruction shculd be the first statement in the program. It stores the contents of register 14, 15, and 0 through 12 in an area provided by the program that passes control. When a problem program is given control, register 13 points to an area in which the general register contents should be saved.

If the program calls any subprcgrams, or uses any operating system services other than GETMAIN, FREEMAIN, ATTACH, and XCTL, it must first save the contents of register 13 and then load the address of an 18 fullword save area into register 13. This save area is in the problem program and is used by any subprograms or operating system services called by the problem program.

At completion, the problem program restores the contents of general registers 14, 15 and 0-12 by use of the RETURN system macro instruction (which also indicates program completion). The contents cf register 13 must be restored before execution of the RETURN macro instruction. The coding sequence that follows illustrates the basic process of saving and restoring the register. A complete discussion of the SAVE and RETURN macro instructions and the saving and restoring of registers is contained in the <u>OS Data Management Services Guide</u> and <u>OS Data Management Macro Instructions</u> publications.

Name	Operation	Operand
BEGIN	SAVE	(14,12)
	.	
	.	set up base register
	.	
	ST	13,SAVEBLK+4
	LA	13,SAVEBLK
	.	
	.	
	L	13,SAVEBLK+4
	RETURN	(14,12)
SAVEBLK	DC	18F'0'

Program Termination

Completion of an assembler source program is indicated by using the RETURN system macro instruction to pass control from the terminating program to the program that initiated it. The initiating prcgram may be the operating system or, if a subprogram issued the RETURN, the program that called it.

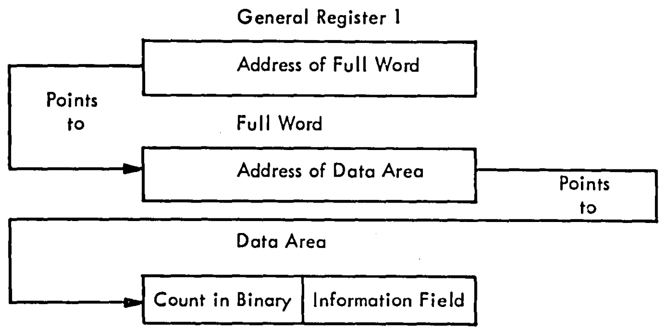
In addition to indicating program completion and restoring registers, the RETURN macro instruction may also pass a return code -- a condition indicator that may be used by the program receiving control. If the return is to the operating system, the return code is compared against the condition stated in the COND= parameter of the JCB or EXEC statements. If return is to another problem program, the return code is available in general register 15, and may be used as desired. Register 13 should be restored before issuing the RETURN macro instruction.

The RETURN system macro instruction is discussed in detail in the <u>OS Supervisor Services and Macro Instructions</u> publication.

PARM Field Access

Access to information in the PARM field of an EXEC statement is gained through general register 1. When control is given to the problem program, general register 1 contains the address of a full word which, in turn, contains the address of the data area containing the information.

The data area consists of a halfword containing the count (in binary) of the number of information characters, followed by the information field. The information field is aligned to a half-word boundary. The following diagram illustrates this process.



Macro Definition Library Additions

Source statement coding, to be retrieved by the COPY assembler instruction, and macro definitions may be added to the macro library. The IEBUPDTE utility program is used for this purpose. Details of this program and its control statements are contained in the OS Utilities publication. The following sequence of job control statements can be used to call the utility program and identify the needed data sets. It is assumed that the job control statements, IEBUPDTE program control statements, and data are to enter the system via the input stream.

```
//jobname    JOB
//stepname   EXEC  PGM=IEBUPDTE,PARM=MOD
//SYSUT1     DD    DSNAME=SYS1.MACLIB,DISP=OLD
//SYSUT2     DD    DSNAME=SYS1.MACLIB,DISP=OLD
//SYSPRINT   DD    SYSOUT=A
//SYSIN      DD    *
                .
                .
                .
IEBUPDTE control statements and source statements or
macro-definitions to be added to the macro-library
(SYS1.MACLIB)
                .
                .
                .
/* (delimiter statement)
```

Load Module Modification - Entry Point Restatement

If the editing functions of the linkage editor are to be used to modify a load module, the entry point to the load module must be restated when the load module is reprocessed by the linkage editor. Otherwise, the first byte of the first control section processed by the linkage editor will become the entry point. To enable restatement of the original entry point, or designation of a new entry point, the entry point must have been identified originally as an external symbol, i.e., appeared as an entry in the external symbol dictionary. External symbol identification is done automatically by the assembler if the entry point is the name of a control section or START statement; otherwise, an assembler ENTRY statement must be used to identify the entry point name as an external symbol.

When a new object module is added to or replaces part of the load module, the entry point is restated in one of three ways:

- By placing the entry point symbol in the operand field of an EXTRN statement and an END statement in the new object module.
- By using an END statement in the new object module to designate a new object module.
- By using a linkage editor ENTRY statement to designate either the original entry point or a new entry point for the load module.

Further discussion of load module entry points is contained in the OS Loader and Linkage Editor publication.

Object Module Linkage

Object modules, whether Assembler-, FORTRAN-, or COBOL-generated, may be combined by the linkage editor to produce a composite load module, provided each object module conforms to the data formats and linkage conventions required. This topic discusses the use of the CALL system macro instruction to link an assembler language "main" program to subprograms produced by FORTRAN and COBOL. The OS Supervisor Services and Macro Instructions publication contains additional details concerning linkage conventions and the CALL system macro instruction.

Figure 11 shows the statements used to establish the assembler program linkage to the called subprograms.

If any input/output operations are performed by called subprograms, appropriate DD statements for the data sets used by the subprograms must be supplied. See the appropriate FORTRAN IV Programmer's Guide publications for explanation of the DD statements used to describe data sets for FORTRAN programs and a description of the special FORTRAN data set record formats. The COBOL Programmer's Guide publications provide DD statement information for COBOL programs.

Dictionary Size and Source Statement Complexity

This section describes the compositicn of the assembler dictionaries and their entry sizes, and describes methods for determining if the limits on source statement complexity will be exceeded.

Dictionary entries, e.g., sequence symbol names, prototype symbolic parameters, vary in length. Therefore, the number of entries a dictionary can hold is determined by the types of entries.

Source statement complexity -- the number of symbols, characters, operators, delimiters, references to length attributes, self-defining terms, literals, and expressions appearing in a source statement -- determines whether or not the source statement can be successfully processed.

```
                 SAVE     (14,12)
                 .
                 .        set up base register
                 .
1                ST       13,SVAREA+4
                 LA       15,SVAREA
                 ST       15,8(13)
                 LR       13,15
                 .
                 .
                 .
2                CALL     name,(V1,V2,V3),VL
                 .
                 .
                 .
                 L        13,SVAREA+4
                 RETURN   (14,12)
3  SVAREA        DC       18F'0'
4  V1            DC       (data)
5  V2            DC       (data)
6  V3            DC       (data)
                 END
```

1 This is an example of OS linkage convention. See the publication OS Supervisor Services and Macro Instructions for details.

2 The symbol used for "name" in this statement is:

a. The name of a subroutine or function, when the linkage is to a FORTRAN-written subprogram.

b. The name defined by the following COBOL statements in the procedure division:

ENTER LINKAGE. ENTRY'name'.

c. The name of a CSECT or START statement, or a name used in the operand field of an ENTRY statement in an assembler subprogram.

The order in which the parameter list is written must reflect the order in which the called subprogram expects the argument. If the called routine is a FORTRAN-written function, the returned argument is not in the parameter list: a real or double precision function returns the value in floating point register zero; an integer function returns the value in general purpose register zero.

CAUTION: When linking to FORTRAN-written subprograms, consideration must be given to the storage requirements of IBCOM (FORTRAN execution-time I/O and interrupt handling routines) which accompanies the compiled FORTRAN subprogram. In some instances the call for IBCOM is not automatically generated during the FORTRAN compilation. The OS FORTRAN IV Library publication provides information about IBCOM requirements and assembler statements used to call IBCOM.

FORTRAN - written subprograms and FORTRAN library subprograms allow variable-length parameter lists in linkages which call them; therefore all linkages to FORTRAN subprograms are required to have the high-order bit in the last parameter in the linkage set to 1. COBOL-written subprograms have fixed-length calling linkages; therefore, for COBOL the high-order bit in the last parameter need not be set to 1.

3 This statement reserves the save area needed by the called subprogram. When control is passed to the subprogram, register 13 contains the address of this area.

4 5 6 When linking to a FORTRAN or COBOL subprogram, the data formats declared in these statements are determined by the data formats required by the FORTRAN or COBOL subprograms.

Figure 11. Linkage Statements

Dictionaries Used in Conditional Assembly and Macro Instruction Expansion

To accomplish macro instruction expansion and conditional assembly, the assembler constructs a general dictionary consisting of two parts: one global dictionary for the entire program, and an area for all of the local dictionaries.

The global dictionary contains one entry for each machine operation code, extended mnemonic operation code, assembler operation code, macro instruction, and global SET variable symbol.

The local dictionary area consists of one local dictionary for each different macro definition in the program, and one local dictionary for the main portion of the program (those statements not within a macro definition, also called "open code"). The contents of the local dictionaries are described in subsequent paragraphs.

The capacity of the general dictionary (global dictionary and all local dictionaries) is up to 64 blocks of 1024 bytes each. The division of the dictionary into global and local sections is done dynamically: as the global dictionary becomes larger, it occupies blocks taken from the local dictionary area. Thus, the global dictionary is always core resident. As it expands into the logical dictionary area, the local dictionaries may overflow onto a utility file. The size of the dictionaries in core depends upon core availability. The minimum core allocation is three blocks for the global dictionary and two blocks for each local dictionary.

Each block in the global and local dictionaries contains complete entries. Any entry not fitting into a block is placed in the next block; the remaining bytes in the current block are not used.

The global and local dictionaries take two forms: one when the dictionary entries are collected, i.e., picked up during the initial scan of the source program, and one during the actual conditional assembly and macro generation, i.e., generation time. The following text describes the global and local dictionaries at both collection time amd generation time.

Global Dictionary at Collection Time

One global dictionary is built for the entire program. It contains machine operation codes, extended mnemonic operation codes, assembler operation codes, OPSYN defined operation codes, macro instruction mnemonics, and global SET variable symbols. One entry is made as shown in Figure 12.

Entry	Size
Each machine operation code **	5 bytes plus mnemonic*
Each extended mnemonic operation code or assembler operation **	6 bytes plus mnemonic*
Each macro mnemonic operation code	10 bytes plus mnemonic*
Each global SET variable symbol	7 bytes plus name*

*One byte is used for each character in the name or mnemonic.

**For the first two types of entries, a total of 0780_{16} (1920_{10}) bytes of core is required.

Figure 12. Global Dictionary Entries at Collection Time

Fixed overhead for this dictionary is:

8 bytes for the first block
4 bytes for each succeeding block
5 bytes for the last block

Local Dictionaries at Collection Time

For the main portion of the program (those statements not within a macro definition), one local dictionary is constructed in which ordinary symbols, sequence symbols, and local SET variable symbols are entered. In addition, one local dictionary is constructed for each different macro definition in the program. These local dictionaries contain one entry for each local SET variable symbol, sequence symbol, and prototype symbolic parameter declared within the macro definition. If a sequence symbol is defined before it is referenced, an extra entry for the symbol is made. Figure 13 shows the size of each type of entry.

Entry	Size
Each sequence symbol	10 bytes plus name*
Each local SET variable symbol	7 bytes plus name*
Each prototype symbolic parameter	5 bytes plus name*
Each ordinary symbol appearing in the main portion of the program.	10 bytes plus name*

*One byte is used for each character in the name or mnemonic.

Figure 13. Local Dictionary Entries at Collection Time

Fixed overhead for this dictionary is:

8 bytes for the first block (if in the main program)

32 bytes for the first block (if in a macro definition)
4 bytes for each succeeding block
5 bytes for the last block

Global Dictionary at Generation Time

The sizes of the global dictionary entries at generation time are shown in Figure 14.

Entry	Size
Each macro mnemonic operation code	3 bytes
Each global SETA symbol (dimensioned)	2 bytes plus 4N*
Each global SETA symbol (undimensioned)	4 bytes
Each global SETB symbol (dimensioned)	2 bytes plus (N/8)* (N/8 is rounded to the next highest integer)
Each global SETB symbol (undimensioned)	1 bit
Each global SETC symbol (dimensioned)	2 bytes plus 9N*
Each global SETC symbol (undimensioned)	9 bytes

*N = dimension

Figure 14. Global Dictionary Entries at Generation Time

Fixed overhead for this dictionary is:

4 bytes plus word alignment.

Local Dictionaries at Generation Time

Figure 15 shows the sizes of the various entries appearing in the local dictionaries at generation time.

Entry	Size
Each sequence symbol	5 bytes
Each local SETA symbol (dimensioned)	2 bytes plus 4N*
Each local SETA symbol (undimensioned)	4 bytes
Each local SETB symbol (dimensioned)	2 bytes plus (N/8)* (N/8 is rounded to the next highest integer)
Each local SETB symbol (undimensioned)	1 bit
Each local SETC symbol(dimensioned)	2 bytes plus 9N*
Each local SETC symbol (undimensioned)	9 bytes
Each ordinary symbol appearing in the main portion of the program.**	5 bytes

*N=dimension
**These entries appear only in the main program local dictionary.

Figure 15. Local Dictionary Entries at Generation Time

Fixed overhead for this dictionary is

20 bytes plus word alignment.

Additional Dictionary Requirements

The generation time global dictionary and the generation time local dictionary for the main portion of the program must be resident in main storage.

In addition, if the program contains any macro instructions, main storage is required for the largest local dictionary of the macro definitions being processed. Furthermore, during processing of macro definitions containing inner macro instructions, main storage is required for the generation time local dictionaries for the inner macro instructions contained within the macro definition.

In addition to those requirements specified for the local dictionary of the main portion of the program, each macro definition local dictionary requires space for entries shown in Figure 16.

Entry	Size
Each character string (1)	3 bytes plus L
Each hexadecimal, binary, decimal, and character self-defining term (2)	7 bytes plus L
Each symbol (3)	9 bytes plus L
Each sublist	9 bytes plus 3N bytes plus Y

L = Length of entry in bytes
N = Number of entries in sublist
Y = $E_1 + E_2 + E_3 + \ldots E_n$
where E = size of an entry (formats 1,2, and 3 above)

Figure 16. Macro Definition Local Dictionary Parameter Table

Fixed overhead for the macro definition local dictionary parameter table is 22 bytes. Each nested macro instruction also requires space in its local dictionary for the following:

Parameter pointer list	8 bytes plus 2N (N = the number of operands)
Pointers to parameter pointer list and parameter table	8 bytes plus word alignment

Correction of Dictionary Overflow

If an assembly is terminated at collection time with either a GLOBAL DICTIONARY FULL message (IEU053) or a LOCAL DICTIONARY FULL message (IEU054), the programmer can take one or more of the following steps:

1. Split the assembly into two or more parts and assemble each separately.

2. Allocate more main storage for the assembler (the global and local dictionaries together can occupy up to 64K).

3. Specify a smaller SYSLIB blocksize. Thus, if BLKSIZE=3600, try BLKSIZE=1800 or BLKSIZE=1200, reblock the library to the size chosen, and try the assembly again.

4. Specify a smaller blocksize for the utility files SYSUT1, 2, and 3. The minimum blocksize normally used by the assembler is 1700 bytes. Reduce this by specifying DCB=BLKSIZE=n on the SYSUT1 DD card. SYSUT2 and 3 use the same blocksize as SYSUT1.

If the assembly is terminated at generation time with a GENERATION TIME DICTIONARY AREA OVERFLOWED message (IEU068), the programmer should allocate more main storage to the assembler and re-assemble his program. If he cannot allocate more main storage to the assembler, the programmer should split the assembly into two or more parts and assemble each separately.

Symbol Table Overflow

Assembler performance can degrade when the source text plus macro-generated statements contains many ordinary symbols. If there are more ordinary symbols than will fit in the symbol table, the assembler will make one or more additional passes over the text. No symbols will be lost, but assembly time will increase.

In general, the assembler can handle 400 ordinary symbols without overflow in its minimum main storage (see Figure 3). Because of input and/or output blocking differences, the minimum amount of main storage varies. It is approximately 49,00 bytes for MFT, and 51,000 bytes for MVT. The assembler can process one additional symbol for each 18 bytes above the minimum amount of main storage.

Source Statement Complexity

The complexity of a source statement is limited both by the macro generator and the assembler portions of the assembler. The following topics provide the information necessary to determine if statement-complexity limitations for either portion of the assembler are being exceeded.

Macro Generation and Conditional Assembly Limitation

For any statement which

1. Is a conditional assembly statement,

2. Is a DC or DS statement,

3. Is an EXTRN or WXTRN statement,

4. Contains a sequence symbol or a variable symbol,

5. Is not a macro instruction or prototype statement,

the total number of explicit occurrences of

1. Ordinary symbols (includes machine mnemonics, assembler mnemonics, conditional assembly mnemonics, and macro instruction mnemonics),

2. Variable symbols,

3. Sequence symbols,

must not exceed 50 for the entire statement.

For macro instructions and prototype statements the number of occurrences of ordinary symbols, variable symbols, and sequence symbols must not exceed 50 in the name and operation fields combined; or in each operand unless the operand is a sublist, in which case the limit is applied to each sublist operand. In any operand if a character string has the same form as a symbol, it is counted as a symbol.

Examples of Counts:

```
&B2 SETB (T'NAME EQ 'W')'count=3 (&B2,SETB,NAME)
EXTRN A,B,C,&C          count=5 (EXTRN,A,B,C,&C)
```

Assembler Portion Limitations

1. Generated statements may not exceed 236 characters. Statement length includes name, operation, operand, and comments. If a comments field exists, the blank separating the operand and comments field is included in the statement length. The statement is truncated if it exceeds 236 characters.

2. DC, DS, DXD, and literal DCs cannot contain more than 32 operands per statement.

System/360 Model 91 Programming Considerations

The assembly language programmer should be aware of the operational differences between the Model 91 and other System/360 models. The Model 91 requires a simulation routine to execute most decimal instructions and it yields different floating-point instructions execution results. The Model 91 also decodes and executes instructions concurrently.

These and other coding and timing considerations are discussed in detail in IBM System/360 Model 91 Functional Characteristics, Order No. GA22-6907. Additional information on how to control sequential and nonsequential instruction execution is given below.

Controlling Instruction Execution Sequence

The CPU maintains a logical consistency with respect to its own operations, including the beginning and ending of I/O operations, but it does not assume responsibility for such consistency in the operations performed by asynchronous units. Consequently, for any asynchronous unit that depends upon a strict adherence to sequential (or serial) execution, a problem program must set up its own procedures to ensure the proper instructions sequence.

For a program section that requires the serial or sequential execution of instructions, the following 'no-operation' instruction:

```
BCR  M,0   where M ≠ 0
```

causes the instruction decoder to halt, and the instructions that have already been decoded to be executed. (This action is called a pipe-line drain.) On the Model 91, this instruction ensures that all the instructions preceding it are executed before the instruction succeeding it is decoded. Use of this instruction should be minimized since it may affect the performance of the Model 91.

Isolating an instruction by preceding it and succeeding it with a BCR instruction eliminates multiple imprecise interruptions from more than one instruction by virtue of the pipe-line drain effect. However, since multiple exceptions may occur in one instruction, this technique does not eliminate a multiple imprecise interruption nor does it change an imprecise interruption into a precise interruption. The use of the BCR instruction does not assure a programmer that he can fix up an error situation. In general, the only information available will be the address of the BCR instruction. The length of the instruction preceding the BCR instruction is not recorded, and generally there is no way to determine what that instruction is.

System/360 Model 85 Programming Considerations

The Model 85 has two special features available to the assembler language programmer. They are extended-precision (two doubleword) floating point instructions and byte-oriented (unaligned) operands. Detailed information on these features is in the IBM System/360 Principles of Operation manual, Order No. GA22-6821.

Assembler F supports these features with mnemonic operation codes for the extended-precision instructions, a two doubleword data constant (DC), an option for suppressing the alignment error message, and an assembler instruction for equating one operation code to another.

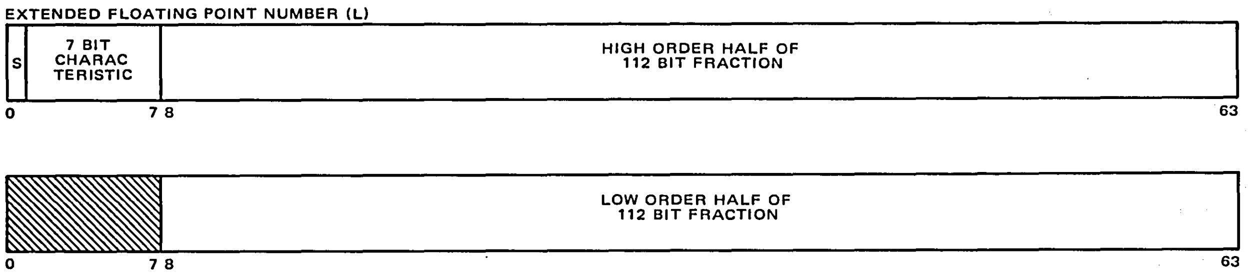

EXTENDED FLOATING POINT NUMBER (L)
S
7 BIT CHARAC TERISTIC
HIGH ORDER HALF OF 112 BIT FRACTION
0
7 8
63
LOW ORDER HALF OF 112 BIT FRACTION
0
7 8
63

Figure 17. Extended-Precision Floating Point Format

These assembler features are explained in the following paragraphs.

Extended-Precision Machine Instructions

The extended-precision arithmetic instructions and the rounding instructions of the Model 85 are shown in Figure 18. The data format for extended operands of the AXR, SXR, MXR, and LRDR instructions and for extended results of the AXR, SXR, MXR, MXDR, and MXD instructions is shown in Figure 17. A complete description of these instructions is in the Principles of Operation manual.

The Extended-Precision Floating-Point Simulator

A program containing extended-precision arithmetic and rounding instructions can be executed on a model that does not have these instructions using the extended-precision floating-point simulator routine of the supervisor. The routine is accessed through the user's program interrupt handler. The user must supply a SPIE macro instruction and a routine to transfer control to the simulator routine. This is explained in detail under "Extended Precision Floating-Point Simulation" in OS Supervisor Services and Macro Instructions.

There are two versions of the simulator. For machines that support the instructions listed in Figure 18, a simulation routine for an extended-precision divide operation is available. The other version is intended for other System/360 models. It simulates the instructions listed in Figure 18 as well as the divide operation.

Because the assembler does not recognize any operation code for an extended-precision divide instruction, a supervisor macro instruction has been provided to produce the proper machine language for the simulator. The format of that macro is described under "DXR" in OS Supervisor Services and Macro Instructions.

Name	Mnemonic	Type	Op Code
ADD NORMALIZED (extended operands, extended result)	AXR	RR	36
SUBTRACT NORMALIZED (extended operands, extended result)	SXR	RR	37
MULTIPLY (extended operands, extended result)	MXR	RR	26
MULTIPLY (long operands, extended result)	MXDR	RR	27
MULTIPLY (long operands, extended result)	MXD	RX	67
LOAD ROUNDED (extended to long)	LRDR	RR	25
LOAD ROUNDED (long to short)	LRER	RR	35

Figure 18. Extended-Precision and Rounding Instructions

Approximating Extended-Precision Floating Point Instructions

An easier way to debug a program containing extended-precision floating-point instructions on a machine that does not contain these instructions, is to approximate them to long floating-point instructions. This is done with the OPSYN assembler instruction.

For example, to "equate" MXR in a source program to MDR, the following instruction is placed at the beginning of the program:

```
MXR   OPSYN   MDR   REPLACE ALL MXR OPERATIONS
                    WITH MDR
```

The MDR instruction is then assembled for each occurrence of the MXR instruction in the source module. The program can be run and debugged on a model that does not have the MXR instruction. Later, the programmer can remove the OPSYN statement and run his program on a machine that supports MXR.

Support of Unaligned Data

The Module 85 will execute unprivileged RX- and RS- format instructions with fixed point, floating-point, or logical operands that are not on integral boundaries.

Assembly of such instructions normally produces the diagnostic message "IEU033 Alignment Error". A new PARM option in the EXEC statement for the Assembler F, ALGN or NOALGN, makes it possible to suppress the message and thereby obtain a "clean" assembly listing. The object code is not affected.

Note that an assembled program that requires use of the byte-oriented operand feature must be run on a Model 85 or 195 machine. Further, it cannot run successfully under the Operating System if it violates any alignment restrictions imposed by OS.

Type L Data Constant

A Define Constant (DC) operand type, L, has been added to provide extended-precision floating-point constants for the programmer. It can be used as a Define Storage (DS) operand or in a literal. Unless changed by a length modifier, the type L constant is 16 bytes long and is aligned on a doubleword boundary. Its format is that of two contiguous type D constants, as shown in Figure 17, except that it is assembled with the sign of the second doubleword equal to that of the first, and the characteristic of the second equal to that of the first minus 14, modulo 128.

Model 195 and System/ 370 Programming Considerations

The Model 195 and the System/370 machines have the following special features: extended-precision (two doubleword) floating-point instructions and byte-oriented (unaligned) operands. The previous descriptions of these features under "System/360 Model 91 Programming Considerations" and "System/360 Model 85 Programming Considerations" also apply to the Model 195 and to System/370 machines. Detailed information can be found in IBM System/360 Model 195 Functional Characteristics, Order No. GA22-6943 and in IBM System/370 Principles of Operation, Order No. GA22-7000.

Note: The Model 195 does not need the decimal simulator routine used by the Model 91.

Appendix A. Diagnostic Messages

This section explains the messages issued by the assembler. They are written on SYSPRINT (if option LIST is in effect) and on SYSTERM (if option TERM is in effect). Messages with serial numbers over 900 are also produced on the operator console.

Message Format

On SYSPRINT:	xx IEUnnn	text	(See Figure 10.)
On SYSTERM:	IEUnnn	text	(See Appendix E.)
On operator console:	IEUnnnI	text	

xx	Statement number for statement in error
nnn	Message serial number. For messages with serial number over 900, the number is followed by the character I.
text	Message text

Severity Codes

The severity code indicates the effect of an error on the execution of a program being assembled:

*	Informational message; no effect on execution
0	Informational message; normal execution is expected
4	Warning message; successful execution is probable
8	Error; execution may fail
12	Serious error; successful execution is improbable
16	Critical error; successful execution is impossible
20	Assembler program terminated abnormally

IEU001 DUPLICATION FACTOR ERROR

Explanation: A duplication factor is not an absolute expression, or is zero in a literal; * in duplication factor expression; invalid syntax in expression.

Severity Code: 12

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU002 RELOCATABLE DUPLICATION FACTOR

Explanation: A relocatable expression has been used to specify the duplication factor.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU003 LENGTH ERROR

Explanation: The length specification is out of permissible range or specified invalidly; * in length expression; invalid syntax in expression; no left-parenthesis delimiter for expression.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU004 RELOCATABLE LENGTH

Explanation: A relocatable expression has been used to specify length.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing

available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU005 S-TYPE CONSTANT IN LITERAL

Explanation: An S-type address constant may not be specified in a literal.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU006 INVALID ORIGIN

Explanation: The location counter has been reset to a value less than the starting address of the control section; ORG operand is not a simply relocatable expression or specifies an address outside the control section.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, dc the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU007 LOCATION COUNTER ERROR

Explanation: Either the location counter has exceeded 2^{24}-1, or passed out of control section in negative direction (3 byte arithmetic).

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU008 INVALID DISPLACEMENT

Explanation: The displacement in an explicit address is not an absolute value within the range of 0 to 4095.
Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU009 MISSING OPERAND

Explanation: Statement requires an operand entry and none is present.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU010 INCORRECT SPECIFICATION OF REGISTER OR MASK

Explanation: One of the following:
- The register or mask field specification is not an absolute value.
- The register or mask field specified is not in the range 0 - 15.
- An odd register is specified where an even register is required (applies to multiply, divide and shift instructions).
- The register specified is not a floating point register (applies to floating point instructions).
- The register specified is not an extended precision floating point register (applies to extended precision floating point instructions).

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU011 SCALE MODIFIER ERROR

Explanation: The scale modifier is not an absolute expression or is too large, negative scale modifier for floating point,

* in scale modifier expression; invalid syntax or illegally specified scale modifier.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU012 RELOCATABLE SCALE MODIFIER

Explanation: A relocatable expression has been used to specify the scale modifier.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU013 EXPONENT MODIFIER ERROR

Explanation: The exponent is not specified as an absolute expression or is out of range; * in exponent modifier expression; invalid syntax; illegally specified exponent modifier.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU014 RELOCATABLE EXPONENT MODIFIER

Explanation: A relocatable expression has been used to specify the exponent modifier.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU015 INVALID LITERAL USAGE

Explanation: A valid literal is used illegally, e.g., it specifies a receiving field or a register, or it is a Q-type constant.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU016 INVALID NAME

Explanation: A name entry is incorrectly specified, e.g., it contains more than 8 characters, it does not begin with a letter, it has a special character embedded, or -- if the statement is OPSYN -- the name entry is not an ordinary symbol or is an assembler operation mnemonic.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU017 DATA ITEM TOO LARGE

Explanation: The constant is too large for the data type or for the explicit length; operand field for packed DC exceeds 31 characters and for zoned DC exceeds 16 characters (excluding decimal points).

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU018 INVALID SYMBOL

Explanation: The symbol is specified invalidly, e.g., it is longer than 8 characters, or -- if the statement is OPSYN -- the name entry is not an ordinary symbol or is an assembler operation mnemonic.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU019 EXTERNAL SYMBOL ERROR

Explanation: One of the following:
- A symbol appears in the name field of both a CSECT and a DSECT statement.
- A symbol appearing the name field of a DXD instruction also appears in the name field of another DXD instruction, in the operand field of an EXTRN of WXTRN instruction, or in the name field of a CSECT or DSECT statement.
- A symbol appearing the operand field of an EXTRN or WXTRN instruction also appears in the operand field of the same or another EXTRN or WXTRN instruction, or in the name field of a DXD, CSECT, or DSECT instruction.
- A symbol previously encountered in the name field of a statement other than those mentioned above, appears in the operand field of an EXTRN or WXTRN instruction or in the name field of a DXD, CSECT, or DSECT instruction.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU020 INVALID IMMEDIATE FIELD

Explanation: The value of the immediate operand exceeds 255 (or 9 for SRP) or the operand is not of an acceptable type.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU021 SYMBOL NOT PREVIOUSLY DEFINED

Explanation: An expression requiring that all symbols be previously defined contains at least one symbol not so defined.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU022 ESDTABLE OVERFLOW

Explanation: The combined number of control sections and dummy sections plus the number of unique symbols in EXTRN and WXTRN statements and V-type constants exceeds 255. (A DSECT which appears as XD makes two entries).

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU023 PREVIOUSLY DEFINED NAME

Explanation: The symbol which appears in the name field has appeared in the name field of a previous statement.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU024 UNDEFINED SYMBOL

Explanation: A symbol being referenced has not been defined in the program.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU025 RELOCATABILITY ERROR

Explanation: A relocatable or complex relocatable expression is specified where an absolute expression is required, an absolute expression or complex relocatable expression is specified where a relocatable expression is required, or a relocatable term is involved in multiplication or division.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU026 TOO MANY LEVELS OF PARENTHESES

Explanation: An expression specifies more than 5 levels of parentheses.

Severity code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU027 TOO MANY TERMS

Explanation: More than 16 terms are specified in an expression.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU028 REGISTER NOT USED

Explanation: A register specified in a DROP statement is not currently in use.

Severity Code: 4

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU029 CCW ERROR

Explanation: Bits 37-39 of the CCW are set to non-zero.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU030 INVALID CNOP

Explanation: An invalid combination of operands is specified in a CNOP instruction.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU031 UNKNOWN TYPE

Explanation: Incorrect type designation is specified in a DC, DS, or literal. If the DOS option is specified, type Q will be flagged as unknown.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro

definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU032 OP-CODE NOT ALLOWED TO BE GENERATED

Explanation: Operation code not allowed if source statement has been obtained through substitution of a value for a variable symbol.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU033 ALIGNMENT ERROR

Explanation: Referenced address is not aligned to the proper boundary for this instruction, e.g., the location of the START operand is not a multiple of 8. Note: If a register is explicitly specified in the reference, e.g., as in L 3,3(REG4), no message is issued.

Severity Code: 4

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU034 INVALID OP-CODE

Explanation: Syntax error, e.g., more than 8 characters in operation field, not followed by blank on first card image, op code missing.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU035 ADDRESSABILITY ERROR

Explanation: The referenced address does not fall within the range of a USING instruction.

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU036 (No message is assigned to this number)

IEU037 MNOTE STATEMENT

Explanation: This indicates that an MNOTE statement has been generated from a macro definition. The text and severity code of the MNOTE statement will be found in line in the listing.

Severity Code: Variable

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU038 ENTRY ERROR

Explanation: A symbol in the operand of an ENTRY statement appears in more than one ENTRY statement, it is undefined, it is defined in a dummy section or in a blank common control section, or it is equated to a symbol defined by an EXTRN or WXTRN statement.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU039 INVALID DELIMITER

Explanation: This message can be caused by any syntax error, e.g., missing delimiter, special character used which is not a valid delimiter, delimiter used illegally, operand missing, i.e., nothing between delimiters, unpaired parentheses, embedded blank in expression.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU040 STATEMENT IS TOO LONG

Explanation: There are more than 236 characters in a generated statement.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU041 UNDECLARED VARIABLE SYMBOL

Explanation: Variable symbol is not declared in a define SET symbol statement or in a macro prototype.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU042 SINGLE TERM LOGICAL EXPRESSION IS NOT A SETB SYMBOL

Explanation: The single term logical expression has not been declared as a SETB symbol.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU043 SET SYMBOL PREVIOUSLY DEFINED

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU044 SET SYMBOL USAGE INCONSISTENT WITH DECLARATION

Explanation: A SET symbol has been declared as undimensioned, but is subscripted, or has been declared dimensioned, but is unsubscripted.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU045 ILLEGAL SYMBOLIC PARAMETER

Explanation: An attribute has been requested for a variable symbol which is not a legal symbolic parameter.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU046 AT LEAST ONE RELOCATABLE Y TYPE CONSTANT IN ASSEMBLY

Explanation: One or more relocatable Y-type constants in assembly; relocation may result in address greater than 2 bytes in length.

Severity Code: 4

Programmer Response: Make sure the source code is correct and reassemble if

necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU047 SEQUENCE SYMBOL PREVIOUSLY DEFINED

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU048 SYMBOLIC PARAMETER PREVIOUSLY DEFINED OR SYSTEM VARIABLE SYMBOL DECLARED AS SYMBOLIC PARAMETER

| Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU049 VARIABLE SYMBOL MATCHES A PARAMETER

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU050 INCONSISTENT GLOBAL DECLARATIONS

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

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU051 MACRO DEFINITION PREVIOUSLY DEFINED

Explanation: Prototype operation field is the same as a machine or assembler instruction or a previous prototype. This message is not produced when a programmer macro matches a system macro. The programmer macro will be assembled with no indication of the corresponding system macro.

| Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU052 NAME FIELD CONTAINS ILLEGAL SET SYMBOL

Explanation: SET symbol in name field does not correspond to SET statement type.

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU053 GLOBAL DICTIONARY FULL

Explanation: The global dictionary is full, assembly terminated.

Severity Code: 12

Programmer Response: Probable user error. Take one or more of the following steps and then rerun the job:
- Split the assembly into two or more parts and assemble each separately.
- Allocate more core for the assembler (the global and local dictionaries together can occupy up to 64K).
- Run the assembly under Assembler E, unless it includes features not allowed by Assembler E. (Due to its dictionary building algorithm, Assembler E can handle more symbols with a given size dictionary than can Assembler F.)
- Specify a smaller SYSLIB blocksize.

Thus, if BLKSIZE=1800 or BLKSIZE=1200, reblock the library to the size chosen, and try the assembly again.

If the problem recurs, do the following before calling IBM for programming support:

- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU054 LOCAL DICTIONARY FULL

Explanation: The local dictionary is full, current macro aborted and the macro instruction is flagged as undefined. If in open code, assembly terminated.

Severity Code: 12

Programmer Response: Probable user error. Take one or more of the following steps and then rerun the jobs.

- Split the assembly into two or more parts and assemble each separately.
- Allocate more core for the assembler (the global and local dictionaries together can occupy up to 64K).
- Run the assembly under Assembler E, unless it includes features not allowed by Assembler E. (Due to its dictionary building algorithm, Assembler E can handle more symbols with a given size dictionary than can Assembler F.)
- Specify a smaller SYSLIB blocksize. Thus, if BLKSIZE=1800 or BLKSIZE=1200, reblock the library to the size chosen, and try the assembly again.
- Specify smaller SYSUT1 blocksize.

If the problem recurs, do the following before calling IBM for programming support:

- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU055 INVALID ASSEMBLER OPTION(S) ON THE EXECUTE CARD

Severity Code: 8

Programmer Response: Probable user error. Make sure all assembler options specified are correct and reassemble if necessary. If problem recurs, do the following before calling IBM:

- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Have the user source program, user macro definitions, and associated listings available.

IEU056 ARITHMETIC OVERFLOW

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

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU057 SUBSCRIPT NOT WITHIN DIMENSIONS

Explanation: (1) Subscript of &SYSLIST or symbolic parameter exceeds 200 or is negative. (2) Subscript of symbolic parameter is zero. (3) Subscript of SET symbol exceeds dimension specified in GBLx/LCLx statement.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU058 RE-ENTRANT CHECK FAILED

Explanation: An instruction has been detected, which, when executed, might store data into a control section or a common area. This message is generated only when requested via control cards and merely indicates a possible re-entrant error. The statement number is not given in the message.

Severity Code: 4

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU059 UNDEFINED SEQUENCE SYMBOL

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro

definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU060 ILLEGAL ATTRIBUTE NOTATION

Explanation: L', S', or I' requested for a parameter whose type attribute dces not allow these attributes to be requested.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU061 ACTR COUNTER EXCEEDED

Explanation: Conditional assembly loop counter exceeded; conditional assembly terminated.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, dc the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU062 GENERATED STRING GREATER THAN 255 CHARACTERS

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU063 EXPRESSION 1 OF SUBSTRING IS ZERO OR MINUS

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, dc the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU064 EXPRESSICN 2 CF SUBSTRING IS ZERO OR MINUS

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU065 INVALID OR ILLEGAL TERM IN ARITHMETIC EXPRESSICN

Explanation: The value of a SETC symbol used in the arithmetic expression is not composed of decimal digits, or the parameter is not a self-defining term.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

| IEU066 UNDEFINED CR DUPLICATE KEYWORD OPERAND

Explanation: The same keyword operand occurs more than once in the macro instruction; a keyword is not defined in a prototype statement.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU067 EXPRESSION 1 OF SUBSTRING GREATER THAN LENGTH CF CHARACTER EXPRESSION

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling

IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU068 GENERATION TIME DICTIONARY AREA OVERFLOWED

Severity Code: 12

Programmer Response: Probable user error. Take one or more of the following steps and then rerun the job:
- Split the assembly into two or more parts and assemble each separately.
- Allocate more core the assembler (the global and local dictionaries together can occupy up to 64K).
- Run the assembly under Assembler E, unless it includes features not allowed by Assembler E. (Due to its dictionary building algotithm, Assembler E can handle more symbols with a given size dictionary then can Assembler F.)
- Specify a smaller SYSLIB blocksize. Thus, if BLKSIZE=1800 or BLKSIZE=1200, reblock the library to the size chosen, and try the assembly again.
- Specify smaller SYSUT1 blocksize.

If the problem recurs, do the following before calling IBM for programming support:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU069 VALUE OF EXPRESSION 2 OF SUBSTRING GREATER THAN 8

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU070 FLOATING POINT CHARACTERISTIC OUT OF RANGE

Explanation: Exponent too large for length of defining field, exponent modifier has caused loss of all significant digits.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU071 ILLEGAL OCCURRENCE OF LCL, GBL, OR ACTR STATEMENT

Explanation: LCL, GBL, or ACTR statement not in proper place in the program.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU072 ILLEGAL RANGE ON ISEQ STATEMENT

Explanation: One or more columns to be sequence checked are between the "begin" and "end" columns of the statement.

Severity Code: 4

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU073 ILLEGAL NAME FIELD

Explanation: (1) The name field is blank in a statement where a name is required. (2) A name is present where no name is allowed. (3) The wrong type of symbol is in the name field (e.g., an ordinary symbol in a conditional assembly statement).

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU074 ILLEGAL STATEMENT IN COPY CODE OR SYSTEM MACRO

Explanation: A statement being copied was a COPY, END, ICTL, ISEQ, MACRO, MEND, OPSYN, or a model statement in a macro containing an END, PRINT, COPY, ISEQ, ICTL, OPSYN.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU075 ILLEGAL STATEMENT OUTSIDE OF A MACRO DEFINITION

Explanation: Statement allowed only in a macro definition encountered outside macro definitions (in open code), e.g., period asterisk (.*), MNOTE statement.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU076 SEQUENCE ERROR

Explanation: Sequence error discovered by the sequence checking mechanism initiated by an ISEQ instruction.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU077 ILLEGAL CONTINUATION CARD

Explanation: Either there are too many continuation cards, or there are non-blanks between the begin and continue columns on the continuation card.

Severity Code: 8

Programmer Response: Probable user error. Make sure source is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU078 INCOMPATIBLE ASSEMBLER OPTIONS ON THE EXECUTE CARD

Explanation: One of the following:
- The DOS assembler option has been specified along with LOAD, TEST, RENT, TERM, or NOALGN. The assembler has used the options specified.
- The NUM or STMT option has been specified along with NOTERM. The assembler has not produced any SYSTERM output.

Severity Code: 0

Programmer Response: Make sure all assembler options specified are correct and reassemble if necessary. If problem recurs, do the following before calling IBM:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Have the user source program, user macro definitions, and associated listings available.

IEU079 ILLEGAL STATEMENT IN MACRO DEFINITION

Explanation: This operation is not allowed within a macro definition.

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU080 ILLEGAL START CARD

Explanation: Statements affecting or depending upon the location counter have been encountered before a START statement.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU081 ILLEGAL FORMAT IN GBL OR LCL STATEMENTS

Explanation: An operand is not a variable symbol.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU082 ILLEGAL DIMENSION SPECIFICATION IN GBL OR LCL STATEMENT

Explanation: Dimension is other than 1 to 2500.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU083 SET STATEMENT NAME FIELD NOT A VARIABLE SYMBOL

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU084 ILLEGAL OPERAND FIELD FORMAT

Explanation: Syntax invalid, e.g., AIF statement operand does not start with a left parenthesis; operand of AGO is not a sequence symbol; operand of PUNCH, TITLE, MNOTE not enclosed in quotes.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU085 INVALID SYNTAX IN EXPRESSION

Explanation: Invalid delimiter, too many terms in expression, too many levels of parentheses, two operators in succession, two terms in succession, or illegal character.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU086 ILLEGAL USAGE OF SYSTEM VARIABLE SYMBOL

Explanation: A system variable symbol appears in the name field of a SET statement, is declared in a GBL or LCL statement, or is an unsubscripted &SYSLIST in a context other than N'&SYSLIST.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU087 NO ENDING APOSTROPHE

Explanation: There is an unpaired apostrophe or ampersand in the statement.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU088 UNDEFINED OPERATION CODE

Explanation: Symbol in operation code field does not correspond to a valid machine or assembler operation code or to any operation code in a macro prototype statement, or a SYSLIB data set has not been provided. If the statement is OPSYN, the operand entry is not a defined machine or extended operation code, or the operand entry is omitted and the name entry is not a defined machine or extended operation code. If the DOS option is in effect, DXD and CXD operation codes will be flagged as undefined.

Severity Code: 12

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If problem recurs, do the following before calling IBM:

- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY STATEMENT.

IEU089 INVALID ATTRIBUTE NOTATION

Explanation: Syntax error inside a macro definition, e.g., the argument of the attribute reference is not a symbolic parameter.

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM.

- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU090 INVALID SUBSCRIPT

Explanation: Syntax error, e.g., double subscript where single subscript is required or vice versa; not right parenthesis after subscript.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU091 INVALID SELF-DEFINING TERM

Explanation: Value is too large or is inconsistent with the data type, i.e., severity code of MNOTE statement greater than 255.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU092 INVALID FORMAT FOR VARIABLE SYMBOL

Explanation: The first character after the ampersand is not alphabetic, or the variable symbol contains more than 8 characters, or failure to use double ampersand in TITLE card or character self-defining term.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU093 UNBALANCED PARENTHESIS OR EXCESSIVE LEFT PARENTHESES

Explanation: End of statement or card encountered before all parenthesis levels are satisfied. May be caused by embedded blank or other unexpected terminator, or failure to have a punch in continuation column.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU094 INVALID OR ILLEGAL NAME OR OPERATION IN PROTOTYPE STATEMENT

Explanation: Name not blank or variable symbol, or variable symbol in name field is subscripted, or violation of rules for forming variable symbol (must begin with ampersand (&) followed by 1-7 letters and/or numbers first of which must be a letter), or statement following 'MACRO' is not a valid prototype statement.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU095 ENTRY TABLE OVERFLOW

Explanation: Number of ENTRY symbols, i.e., ENTRY instruction operands, exceeds 100.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU096 MACRO INSTRUCTION OR PROTOTYPE OPERAND EXCEEDS 255 CHARACTERS IN LENGTH

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU097 INVALID FORMAT IN MACRO INSTRUCTION OPERAND OR PROTOTYPE PARAMETER

Explanation: This message can be caused by:

1. Illegal "=".
2. A single "&" appears somewhere in the standard value assigned to a prototype keyword parameter.
3. First character of a prototype parameter is not "&".
4. Prototype parameter is a subscripted variable symbol.
5. Invalid use of alternate format in prototype statement, e.g.,

```
10       16                   72
PROTO    &A,&B,
                 or
PROTO    &A,&B,               X
         &C
```

6. Unintelligible prototype parameter, e.g., "&A*" or "&A&&."
7. Illegal (non-assembler) character appears in prototype parameter or macro instruction operand.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU098 EXCESSIVE NUMBER OF OPERANDS OR PARAMETERS

Explanation: Either the prototype has more than 200 parameters, or the macro instruction has more than 200 operands.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU099 POSITIONAL MACRO INSTRUCTION OPERAND, PROTOTYPE PARAMETER OR EXTRA COMMA FOLLOWS KEYWORD

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU100 STATEMENT COMPLEXITY EXCEEDED

Explanation: More than 32 operands in a DC, DS, DXD, or literal DC, or more than 50 terms in a statement.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
• Have the user source program, user macro definitions and associated listing available.
• If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU101 EOD ON SYSIN

Explanation: EOD before END card.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
• Have the user source program, user macro definitions and associated listing available.
• If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU102 INVALID OR ILLEGAL ICTL

Explanation: The operands of the ICTL are out of range, or the ICTL is not the first statement in the input deck. (Assembly is terminated and further input is ignored.)

Severity Code: 16

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
• Have the user source program, user macro definitions and associated listing available.
• If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU103 ILLEGAL NAME IN OPERAND FIELD OF COPY CARD

Explanation: Syntax error, e.g., symbol has more than 8 characters or has an illegal character.

Severity Code: 12

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
• Have the user source program, user macro definitions, and associated listings available.
• If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU104 COPY CODE NOT FOUNC

Explanation: The operand of a COPY statement specified COPY text which cannot be found in the library.

Severity Code: 12

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If problem recurs, do the following before calling IBM:
• Make sure the SYSLIB CC statement is included.
• Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
• Have the user source program, user macro definitions, and associated listings available.
• If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU105 EOD ON SYSTEM MACRO LIBRARY

Explanation: EOC before MENC card.

Severity Code: 12

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
• Have the user source program, user macro definitions, and associated listings available.
• If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU106 NOT NAME OF CSECT OR CXD

Explanation: Referenced symbol expected to be DSECT name, but it is not.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
• Have the user source program, user macro definitions and associated listing available.
• If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PCS member specified in the COPY statement.

IEU107 INVALID OPERAND

Explanation: Invalid syntax in DC operand, e.g., invalid hexadecimal character in hexadecimal CC; operand string too long for X, B, C, DC's; operand unrecognizable, contains invalid value, or incorrectly specified.

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU108 PREMATURE EOD

Explanation: Indicates an internal assembler error; should not occur.

Severity Code: 16

Programmer Response: Reassemble; if the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEU109 PRECISION LOST

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU110 EXPRESSION VALUE TOO LARGE

Explanation: Value of expression not in range than -16777216 to +16777215. Expressions in EQU and ORG statements are flagged if (1) they include terms previously defined as negative values, or (2) positive terms give a result of more than three bytes in magnitude. The error indication may be erroneous due to (1) the treatment of negative values as three-byte positive values, or (2) the effect of large positive values on the location counter if a control section begins with a START statement having an operand greater than zero, or a control section is divided into subsections.

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU111 OPEN FAILED FOR SYSGO, NOLOAD OPTION USED

Explanation: DD statement incorrect or missing.

Severity Code: 16

Programmer Response: Probable user error. If necessary supply missing DD statement or make sure that information on DD statement is correct and reassemble. If problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEU112 OPEN FAILED FOR SYSPUNCH, NODECK OPTION USED

Explanation: DD statement incorrect or missing.

Severity Code: 16

Programmer Response: Probable user error. If necessary supply missing DD statement or make sure that information on DD statement is correct and reassemble. If problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEU113 OPEN FAILED FOR SYSTERM, NOTERM OPTION USED

Explanation: DD statement incorrect or missing.

Severity Code: 0

Programmer Response: Probable user error. If necessary supply missing DD statement or make sure that information on DD statement is correct and reassemble. If problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEU116 ILLEGAL OPSYN

Explanation: An OPSYN statement may be preceded only by an ICTL instruction or another OPSYN statement.

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU117 OPSYN TABLE OVERFLOW

Explanation: No room exists in symbol table for this and following OPSYN definitions; generated operation codes may not be processed correctly.

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU996I ASSEMBLY TERMINATED INSUFFICIENT STORAGE

Explanation: One of the following:
- The partition or region size is less than the mimimum required by the assembler.
- The blocksize specified for the utility data sets is too large for available main storage.

System Action: Assembly is terminated.

Severity Code: 20

IEU997I OPEN FAILED FOR SYSPRINT, NOLIST OPTION USED

Explanation: DD statement incorrect or missing.

System Action: Processing continues.

Severity Code: 0

Programmer Response: Probable user error. If necessary supply the missing DD statement or make sure that information on the DD statement is correct; reassemble. If problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEU998I ASSEMBLY TERMINATED, OPEN FAILED FOR DATA SET (ddname)

Explanation: DD statement(s) for data set(s) SYSIN, SYSUT1, SYSUT2, SYSUT3, and/or SYSPRINT incorrect or missing.

System Action: Assembly is terminated.

Severity Code: 20

Programmer Response: Probable user error. Supply missing DD statement(s) or make sure that information on DD statement(s) is correct; reassemble. If problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEU999I ASSEMBLY TERMINATED, jobname, stepname, unit address, device type, ddname, operation attempted, error description (bytes 107 through 128 of the SYNADAF message buffer; this area is described in OS Data Management Macro Instructions.

Explanation: Indicates a permanent I/O error. This message is produced by the SYNADAF macro instruction.

System Action: Assembly is terminated.

Severity Code: 20

Programmer Response: Reassemble. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

Appendix B. Object Deck Output

TXT Card Format

The format of the TXT cards is as follows:

Columns	Contents
1	12-2-9 punch
2-4	TXT
5	Blank
6-8	Relative address of first instruction on card
9-10	Blank
11-12	Byte count -- number of bytes in information field (cc 17-72)
13-14	Blank
15-16	ESDID
17-72	56-byte information field
73-76	Deck ID (from first TITLE card)
77-80	Card sequence number

RLD Card Format

The format of the RLD card is as follows:

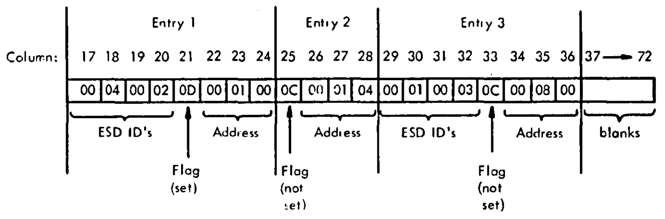
Columns	Contents
1	12-2-9 punch
2-4	RLD
5-10	Blank
11-12	Data field count -- number of bytes of information in data field (cc 17-72)
13-16	Blank
17-72	Data field:
17-18	Relocation ESDID
19-20	Position ESDID
21	Flag byte
22-24	Absolute address to be relocated
25-72	Remaining RLD entries
73-76	Deck ID (from first TITLE card)
77-80	Card sequence number

If the rightmost bit of the flag byte is set, the following RLD entry has the same Relocation ESDID and Position ESDID, and this information will not be repeated; if the rightmost bit of the flag byte is not set, the next RLD entry has a different Relocation ESDID and/or Position ESDID, and both ESDIDs will be recorded.

For example, if the RLD Entries 1, 2, and 3 of the program listing (Appendix C) contain the following information:

	Pos. ESDID	Rel. ESDID	Flag	Address
Entry 1	02	04	0C	000100
Entry 2	02	04	0C	000104
Entry 3	03	01	0C	000800

Columns 17-36 of the RLD card would appear as follows:



Entry 1
Entry 2
Entry 3
Column: 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 → 72
00 04 00 02 0D 00 01 00 0C 00 01 04 00 01 00 03 0C 00 08 00
ESD ID's
Address
Address
ESD ID's
Address
blanks
Flag (set)
Flag (not set)
Flag (not set)

ESD Card Format

The format of the ESD card is as follows:

Columns	Contents
1	12-2-9 punch
2-4	ESD
5-10	Blank
11-12	Variable field count -- number of bytes of information in variable field (cc 17-64)
13-14	Blank
15-16	ESDID of first SD, XD, CM, WX, PC, or ER in variable field
17-64	Variable field. One to three 16-byte items of the following format: 8 bytes -- Name, padded with blanks 1 byte -- ESD type code The hex value is: 00 SD 01 LD 02 ER 04 PC* 05 CM 06 XD (PR) 0A WX 3 bytes -- Address 1 byte -- Alignment if XD; otherwise blank 3 bytes -- Length, LDID, or blank
65-72	Blank
73-76	Deck ID (from first TITLE card)
77-80	Card sequence number

END Card Format

The format of the END card is as follows:

Columns	Contents
1	12-2-9 punch
2-4	END
5	Blank
6-8	Entry address from operand of END card in source deck (blank if no operand)
9-14	Blank
15-16	ESDID of entry point (blank if no operand)
17-32	Blank
33	1
34-43	Order number of the assembler: S360AS037.
44-45	Version level of the assembler.
46-47	Modification level of the assembler.
48-49	Last two digits of the year in which the assembly was run.
50-52	Julian day of the year in which the assembly was run.
53-72	Normally not used.
73-80	Deck ID and/or sequence number. The deck ID is the name field from the first named TITLE statement. The name can be one to eight alphameric characters long. If there is no name or the name is less than eight characters long, the remaining columns contain a card sequence number. (Columns 73-80 of cards produced by PUNCH or REPRO statements do not contain a deck ID or a sequence number.)

SYM Card Format

If requested by the user, the assembler punches out SYM cards with symbolic information concerning the assembled program. These cards can be used by the TESTRAN routine or the TSO Test command processor. The cards are located between the ESD and TXT cards. The format of SYM cards is as follows:

Columns	Contents
1	12-2-9 punch
2-4	SYM
5-10	Blank
11-12	Variable field count -- number of bytes of text in variable field (cc 17-72)
13-16	Blank
17-72	Variable field (see below)
73-76	Deck ID (from first TITLE card)
77-80	Card sequence number

The variable field (columns 17-72) contains up to 56 bytes of TESTRAN text. The items making the text are packed together, consequently only the last card may contain less than 56 bytes of text in the variable field. The formats of a text card and an individual text item are shown in Figure 19. The contents of the fields within an individual entry are as follows:

1. Organization (1 byte)
   0 = non-data type
   1 = data type

   Bits 1-3 (if non-data type):
   000 = space
   001 = control section
   010 = dummy control section
   011 = common
   100 = machine instruction
   101 = CCW

   Bit 1 (if data type):
   0 = no multiplicity
   1 = multiplicity (indicates presence of M field)

   Bit 2 (if data type):
   0 = independent (not a packed or zoned decimal constant)
   1 = cluster (packed or zoned decimal constant)

   Bit 3 (if data type):
   0 = no scaling
   1 = scaling (indicates presence of S field)

   Bit 4:
   0 = name present
   1 = name not present

   Bits 5-7:
   Length of name minus one

2. Address (3 bytes) - displacement from beginning of control section

3. Symbol Name (0-8 bytes) - symbolic name of particular item

Note: The following fields are only present for data-type items.

4. Data Type (1 byte) - contents in hexadecimal

   00 = character
   04 = hexadecimal, L-type data
   08 = binary
   10 = fixed point, full
   14 = fixed point, half
   18 = floating point, short
   1C = floating point, long
   20 = A-type or Q-type data
   24 = Y-type data
   28 = S-type data
   2C = V-type data
   30 = packed decimal
   34 = zoned decimal

5. Length (2 bytes for character, hexadecimal, or binary items; 1 byte

for other types) - length of data item minus 1

6. Multiplicity - M field (3 bytes) - equals 1 if not present

7. Scale -signed integer - S field (2 bytes) - present only for F, H, E, D, L, P and Z type data, and only if scale is non-zero.

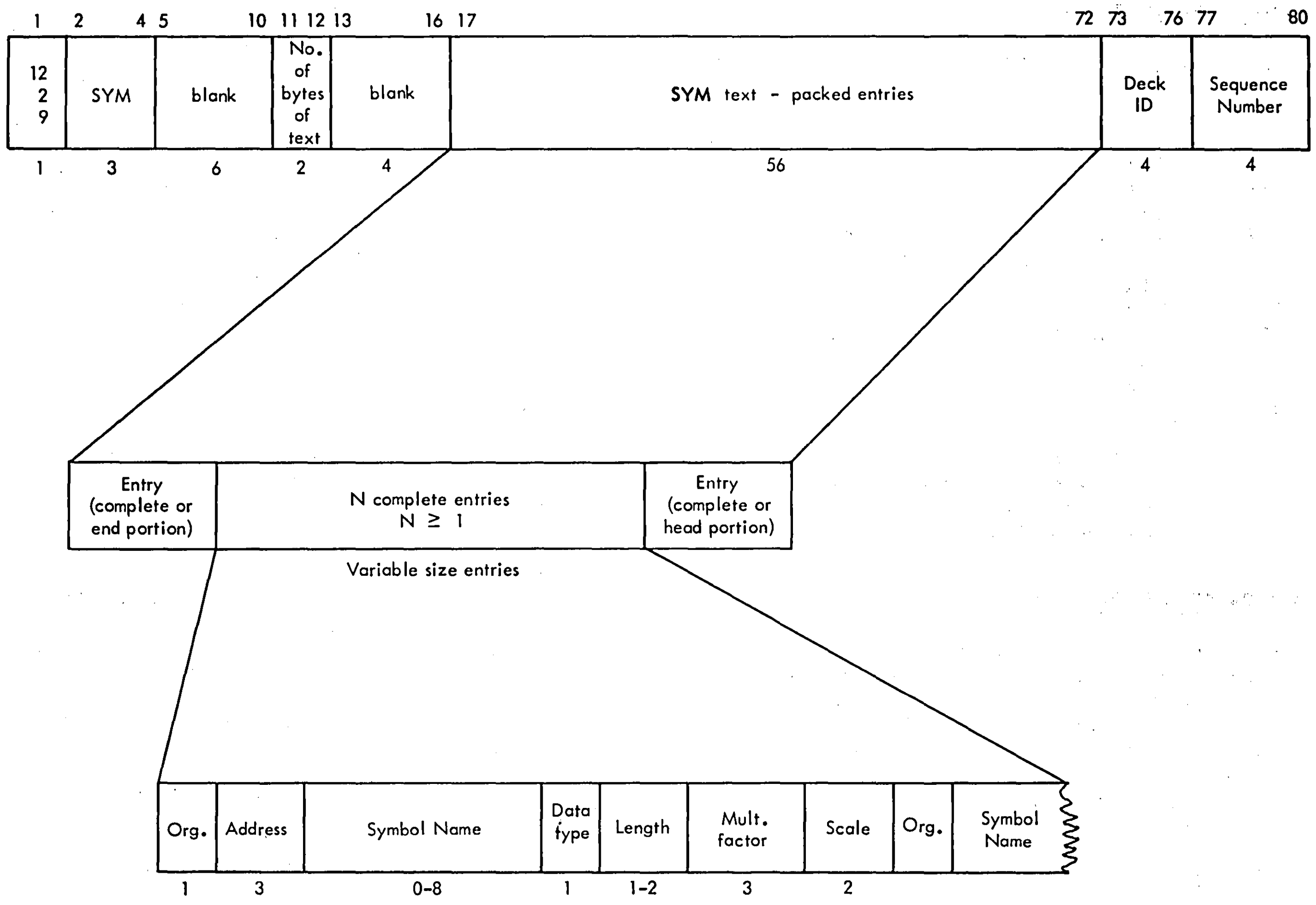


Figure 19. SYM Card Format

Appendix C. Assembler F Program Listing

The Assembler F listing shown in this appendix results from assembling the source program documented in an appendix to the OS Assembler Language publication. For easy reference to the explanations that appear in the section "The Assembler Listing", the headings on the listing are numbered.

Since there were no errors in the assembly, a diagnostic list was not produced. Each of the following pages represents one printer-produced listing page.

```
❼                                                                                                  ❾
EXAM      ❷   ❸   ❹      ❺      ❻             EXTERNAL SYMBOL DICTIONARY                           PAGE    1
SYMBOL   TYPE ID  ADDR  LENGTH LD ID                                                         14.56 10/13/71


SAMPLR    SD  01 000000 0003B8
```

```
 7          8                                                                                                        9
EXAM    SAMPLE PROGRAM                                                                                      PAGE    1
  10        11           12         13         14                                                        15        16
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                              F01OCT71  10/13/71
                                                                                                                  17
                                      2          PRINT DATA                                                    01000019
                                      3 *                                                                      01500019
                                      4 *        THIS IS THE MACRO DEFINITION                                  02000019
                                      5 *                                                                      02500019
                                      6          MACRO                                                         03000019
                                      7          MOVE  &TO,&FROM                                               03500019
                                      8 .*                                                                     04000019
                                      9 .*       DEFINE SETC SYMBOL                                            04500019
                                     10 .*                                                                     05000019
                                     11          LCLC  &TYPE                                                   05500019
                                     12 .*                                                                     06000019
                                     13 .*       CHECK NUMBER OF OPERANDS                                      06500019
                                     14 .*                                                                     07000019
                                     15          AIF   (N'&SYSLIST NE 2).ERROR1                                07500019
                                     16 .*                                                                     08000019
                                     17 .*       CHECK TYPE ATTRIBUTES OF OPERANDS                             08500019
                                     18 .*                                                                     09000019
                                     19          AIF   (T'&TO NE T'&FROM).ERROR2                               09500019
                                     20          AIF   (T'&TO EQ 'C' OR T'&TO EQ 'G' OR T'&TO EQ 'K').TYPECGK  10000019
                                     21          AIF   (T'&TO EQ 'D' OR T'&TO EQ 'E' OR T'&TO EQ 'H').TYPEDEH  10500019
                                     22          AIF   (T'&TO EQ 'F').MOVE                                     11000019
                                     23          AGO   .ER OR3                                                 11500019
                                     24 .TYPEDEH ANOP                                                          12000019
                                     25 .*                                                                     12500019
                                     26 .*       ASSIGN TYPE ATTRIBUTE TO SETC SYMBOL                          13000019
                                     27 .*                                                                     13500019
                                     28 &TYPE    SETC  T'&TO                                                   14000019
                                     29 .MOVE    ANOP                                                          14500019
                                     30 *        NEXT TWO STATEMENTS GENERATED FOR MOVE MACRC                  15000019
                                     31          L&TYPE   2,&FROM                                              15500019
                                     32          ST&TYPE  2,&TO                                                16000019
                                     33          MEXIT                                                         16500019
                                     34 .*                                                                     17000019
                                     35 .*       CHECK LENGTH ATTRIBUTES OF OPERANDS                           17500019
                                     36 .*                                                                     18000019
                                     37 .TYPECGK AIF   (L'&TO NE L'&FROM OR L'&TO GT 256).ERROR4               18500019
                                     38 *        NEXT STATEMENT GENERATED FOR MOVE MACRO                       19000019
                                     39          MVC   &TO,&FROM                                               19500019
                                     40          MEXIT                                                         20000019
                                     41 .*                                                                     20500019
                                     42 .*       ERROR MESSAGES FOR INVALID MOVE MACRO INSTRUCTIONS            21000019
                                     43 .*                                                                     21500019
                                     44 .ERROR1  MNOTE 1,'IMPROPER NUMBER OF OPERANDS, NO STATEMENTS GENERATED' 22000019
                                     45          MEXIT                                                         22500019
                                     46 .ERROR2  MNOTE 1,'OPERAND TYPES DIFFERENT, NO STATEMENTS GENERATED'    23000019
                                     47          MEXIT                                                         23500019
                                     48 .ERROR3  MNOTE 1,'IMPROPER OPERAND TYPES, NO STATEMENTS GENERATED'     24000019
                                     49          MEXIT                                                         24500019
                                     50 .ERROR4  MNOTE 1,'IMPROPER OPERAND LENGTHS, NO STATEMENTS GENERATED'   25000019
                                     51          MEND                                                          25500019
                                     52 *                                                                      26000019
                                     53 *        MAIN ROUTINE                                                  26500019
                                     54 *                                                                      27000019
000000                               55 SAMPLR   CSECT                                                         27500019
                                     56 BEGIN    SAVE  (14,12),,*                                              28000019
000000 47F0 F00A           0000A     57+BEGIN    B     10(0,15) BRANCH AROUND ID
000004 05                            58+         DC    AL1(5)
000005 C2C5C7C9D5                    59+         DC    CL5 BEGIN' IDENTIFIER
00000A 90EC D00C           0000C     60+         STM   14,12,12(13) SAVE REGISTERS
00000E 05C0                          61          BALR  R12,0        ESTABLISH ADDRESSABILITY OF PROGRAM      28500019
000010                               62          USING *,R12        AND TELL THE ASSEMBLER WHAT BASE TO USE  29000019
000010 50D0 C0B8           000C8     63          ST    13,SAVE13                                               29500019
000014 9857 C390           003A0     64          LM    R5,R7,=A(LISTAREA,16,LISTEND)  LOAD LIST AREA PARAMETERS 30000019
000000                               65          USING LIST,R5     REGISTER 5 POINTS TO THE LIST               30500019
000018 45E0 C0BE           000CE     66 MORE     BAL   R14,SEARCH  FIND LIST ENTRY IN TABLE                    31000019
00001C 9180 C0BC     000CC           67          TM    SWITCH,NONE CHECK TO SEE IF NAME WAS FOUND              31500019
000020 4710 C0B0           000C0     68          BO    NOTTHERE    BRANCH IF NOT                               32000019
000000                               69          USING TABLE,R1    REGISTER 1 NOW POINTS TO TABLE ENTRY        32500019
                                     70          MOVE  TSWITCH,LSWITCH         MOVE FUNCTIONS                  33000019
                                     71+*        NEXT STATEMENT GENERATED FOR MOVE MACRO
```

```
7      8                                                                                                         9
EXAM   SAMPLE PROGRAM                                                                                          PAGE    2
  10      11            12          13      14                                                            15       16
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                              F01OCT71  10/13/71

000024 D200 1003 5008 00003 00008    72+        MVC   TSWITCH,LSWITCH                                             17
                                     73         MOVE  TNUMBER,LNUMBER                FROM LIST ENTRY            33500019
                                     74++*      NEXT STATEMENT GENERATED FOR MOVE MACRO
00002A D202 1000 5009 00000 00009    75+        MVC   TNUMBER,LNUMBER
                                     76         MOVE  TADDRESS,LADDRESS                   TO TABLE ENTRY        34000019
                                     77++*      NEXT TWO STATEMENTS GENERATED FOR MOVE MACRO
000030 5820 500C            0000C    78+        L     2,LADDRESS
000034 5020 1004            00004    79+        ST    2,TADDRESS
000038 8756 C008            00018    80 LISTLOOP BXLE  R5,R6,MORE   LOOP THROUGH THE LIST                       34500019
00003C D5EF C240 C0F0 00250 00100    81         CLC   TESTTABL(240),TABLAREA                                    35000019
000042 4770 C07C            0008C    82         BNE   NOTRIGHT                                                  35500019
000046 D55F C330 C1E0 00340 001F0    83         CLC   TESTLIST(96),LISTAREA                                     36000019
00004C 4770 C07C            0008C    84         BNE   NOTRIGHT                                                  36500019
                                     85         WTO   'ASSEMBLER SAMPLE PROGRAM SUCCESSFUL'                     37000019
000050                               86+        CNOP  0,4
000050 4510 C06C            0007C    87+        BAL   1,IHB0005A BRANCH AROUND MESSAGE
000054 0027                          88+        DC    AL2(IHB0005-*) MESSAGE LENGTH
000056 0000                          89+        DC    B'0000000000000000' MCSFLAGS FIELD
000058 C1E2E2C5D4C2D3C5              90+        DC    C'ASSEMBLER SAMPLE PROGRAM SUCCESSFUL' MESSAGE
000060 D940E2C1D4D7D3C5
000068 40D7D9D6C7D9C1D4
000070 40E2E4C3C3C5E2E2
000078 C6E4D3
00007B                               91+IHB0005  EQU   *
00007C                               92+IHB0005A DS    0H
00007C 0A23                          93+        SVC   35 ISSUE SVC
00007E 58D0 C0B8            000C8    94 EXIT     L     R13,SAVE13                                                37500019
                                     95         RETURN (14,12),RC=0                                             38000019
000082 98EC D00C            0000C    96+        LM    14,12,12(13)  RESTORE THE REGISTERS
000086 41F0 0000            00000    97+        LA    15,0(0,0) LOAD RETURN CODE
00008A 07FE                          98+        BR    14 RETURN
                                     99 *                                                                       38500019
                                    100 NOTRIGHT WTO   'ASSEMBLER SAMPLE PROGRAM UNSUCCESSFUL'                   39000019
00008C                              101+        CNOP  0,4
00008C 4510 C0AA            000BA   102+NOTRIGHT BAL   1,IHB0007A BRANCH AROUND MESSAGE
000090 0029                         103+        DC    AL2(IHB0007-*) MESSAGE LENGTH
000092 0000                         104+        DC    B'0000000000000000' MCSFLAGS FIELD
000094 C1E2E2C5D4C2D3C5             105+        DC    C'ASSEMBLER SAMPLE PROGRAM UNSUCCESSFUL' MESSAGE
00009C D940E2C1D4D7D3C5
0000A4 40D7D9D6C7D9C1D4
0000AC 40E4D5E2E4C3C3C5
0000B4 E2E2C6E4D3
0000B9                              106+IHB0007  EQU   *
0000BA                              107+IHB0007A DS    0H
0000BA 0A23                         108+        SVC   35 ISSUE SVC
0000BC 47F0 C06E            0007E   109         B     EXIT                                                      39500019
0000C0 9680 5008      00008         110 NOTTHERE OI    LSWITCH,NONE TURN ON SWITCH IN LIST ENTRY                 40000019
0000C4 47F0 C028            00038   111         B     LISTLOOP     GO BACK AND LOOP                             40500019
0000C8 00000000                     112 SAVE13   DC    F'0'                                                      41000019
0000CC 00                           113 SWITCH   DC    X'00'                                                     41500019
000080                              114 NONE     EQU   X'80'                                                     42000019
                                    115 *                                                                       42500019
                                    116 *               BINARY SEARCH ROUTINE                                   43000019
                                    117 *                                                                       43500019
0000CD 00
0000CE 947F C0BC      000CC         118 SEARCH   NI    SWITCH,255-NONE TURN OFF NOT FOUND SWITCH                 44000019
0000D2 9813 C39C            003AC   119         LM    R1,R3,=F'128,4,128' LOAD TABLE PARAMETERS                 44500019
0000D6 4111 C0E0            000F0   120         LA    R1,TABLAREA-16(R1)   GET ADDRESS OF MIDDLE ENTRY          45000019
0000DA 8830 0001            00001   121 LOOP     SRL   R3,1                 DIVIDE INCREMENT BY 2                45500019
0000DE D507 5000 1008 00000 00008   122         CLC   LNAME,TNAME          COMPARE LIST ENTRY WITH TABLE ENTRY  46000019
0000E4 4720 C0E4            000F4   123         BH    HIGHER               BRANCH IF SHOULD BE HIGHER IN TABLE  46500019
0000E8 078E                         124         BCR   8,R14                EXIT IF FOUND                        47000019
                                    125         SR    R1,R3                OTHERWISE IT IS LOWER IN THE TABLE  X47500019
0000EA 1B13                                                              SO SUBTRACT INCREMENT             48000019
0000EC 4620 C0CA            000DA   126         BCT   R2,LOOP              LOOP 4 TIMES                         48500019
0000F0 47F0 C0EA            000FA   127         B     NOTFOUND             ARGUMENT IS NOT IN THE TABLE         49000019
0000F4 1A13                         128 HIGHER   AR    R1,R3                ADD INCREMENT                        49500019
0000F6 4620 C0CA            000DA   129         BCT   R2,LOOP              LOOP 4 TIMES                         50000019
0000FA 9680 C0BC      000CC         130 NOTFOUND OI    SWITCH,NONE          TURN ON NOT FOUND SWITCH             50500019
0000FE 07FE                         131         BR    R14                  EXIT                                 51000019
```

```
❼         ❽                                                                                        ❾
EXAM     SAMPLE PROGRAM                                                                           PAGE    3
  ❿        ⓫                    ⓬           ⓭          ⓮                                          ⓯        ⓰
 LOC    OBJECT CODE        ADDR1 ADDR2  STMT   SOURCE STATEMENT                               F01OCT71  10/13/71
                                                                                                        ⓱
                                         132 *                                                           51500019
                                         133 *        THIS IS THE TABLE                                  52000019
                                         134 *                                                           52500019
000100                                   135           DS    0D                                          53000019
000100 0000000000000000                  136 TABLAREA  DC    XL8'0',CL8'ALPHA'                           53500019
000108 C1D3D7C8C1404040
000110 0000000000000000                  137           DC    XL8'0',CL8'BETA'                            54000019
000118 C2C5E3C140404040
000120 0000000000000000                  138           DC    XL8'0',CL8'DELTA'                           54500019
000128 C4C5D3E3C1404040
000130 0000000000000000                  139           DC    XL8'0',CL8'EPSILON'                         55000019
000138 C5D7E2C9D3D6D540
000140 0000000000000000                  140           DC    XL8'0',CL8'ETA'                             55500019
000148 C5E3C14040404040
000150 0000000000000000                  141           DC    XL8'0',CL8'GAMMA'                           56000019
000158 C7C1D4D4C1404040
000160 0000000000000000                  142           DC    XL8'0',CL8'IOTA'                            56500019
000168 C9D6E3C140404040
000170 0000000000000000                  143           DC    XL8'0',CL8'KAPPA'                           57000019
000178 D2C1D7D7C1404040
000180 0000000000000000                  144           DC    XL8'0',CL8'LAMBDA'                          57500019
000188 D3C1D4C2C4C14040
000190 0000000000000000                  145           DC    XL8'0',CL8'MU'                              58000019
000198 D4E4404040404040
0001A0 0000000000000000                  146           DC    XL8'0',CL8'NU'                              58500019
0001A8 D5E4404040404040
0001B0 0000000000000000                  147           DC    XL8'0',CL8'OMICRON'                         59000019
0001B8 D6D4C9C3D9D6D540
0001C0 0000000000000000                  148           DC    XL8'0',CL8'PHI'                             59500019
0001C8 D7C8C94040404040
0001D0 0000000000000000                  149           DC    XL8'0',CL8'SIGMA'                           60000019
0001D8 E2C9C7D4C1404040
0001E0 0000000000000000                  150           DC    XL8'0',CL8'ZETA'                            60500019
0001E8 E9C5E3C140404040
                                         151 *                                                           61000019
                                         152 *        THIS IS THE LIST                                   61500019
                                         153 *                                                           62000019
0001F0 D3C1D4C2C4C14040                  154 LISTAREA  DC    CL8'LAMBDA',X'0A',FL3'29',A(BEGIN)          62500019
0001F8 0A00001D00000000
000200 E9C5E3C140404040                  155           DC    CL8'ZETA',X'05',FL3'5',A(LOOP)              63000019
000208 05000005000000DA
000210 E3C8C5E3C1404040                  156           DC    CL8'THETA',X'02',FL3'45',A(BEGIN)           63500019
000218 0200002D00000000
000220 E3C1E44040404040                  157           DC    CL8'TAU',X'00',FL3'0',A(1)                  64000019
000228 0000000000000001
000230 D3C9E2E340404040                  158           DC    CL8'LIST',X'1F',FL3'465',A(0)               64500019
000238 1F0001D100000000
000240 C1D3D7C8C1404040                  159 LISTEND   DC    CL8'ALPHA',X'00',FL3'1',A(123)              65000019
000248 000000010000007B
                                         160 *                                                           65500019
                                         161 *        THIS IS THE CONTROL TABLE                          66000019
                                         162 *                                                           66500019
000250                                   163           DS    0D                                          67000019
000250 000001000000007B                  164 TESTTABL  DC    FL3'1',X'00',A(123),CL8'ALPHA'              67500019
000258 C1D3D7C8C1404040
000260 0000000000000000                  165           DC    XL8'0',CL8'BETA'                            68000019
000268 C2C5E3C140404040
000270 0000000000000000                  166           DC    XL8'0',CL8'DELTA'                           68500019
000278 C4C5D3E3C1404040
000280 0000000000000000                  167           DC    XL8'0',CL8'EPSILON'                         69000019
000288 C5D7E2C9D3D6D540
000290 0000000000000000                  168           DC    XL8'0',CL8'ETA'                             69500019
000298 C5E3C14040404040
0002A0 0000000000000000                  169           DC    XL8'0',CL8'GAMMA'                           70000019
0002A8 C7C1D4D4C1404040
0002B0 0000000000000000                  170           DC    XL8'0',CL8'IOTA'                            70500019
0002B8 C9D6E3C140404040
0002C0 0000000000000000                  171           DC    XL8'0',CL8'KAPPA'                           71000019
0002C8 D2C1D7D7C1404040
0002D0 00001D0A00000000                  172           DC    FL3'29',X'0A',A(BEGIN),CL8'LAMBDA'          71500019
```

```
 ❼       ❽                                                                                  ❾
EXAM     SAMPLE PROGRAM                                                                    PAGE    4
  ❿      ⓫                   ⓬          ⓭        ⓮                                ⓯        ⓰
  LOC    OBJECT CODE     ADDR1 ADDR2  STMT    SOURCE STATEMENT                        F01OCT71  10/13/71
                                                                                                   ⓱
0002D8 D3C1D4C2C4C14040
0002E0 0000000000000000                173            DC     XL8'0',CL8'MU'                       72000019
0002E8 D4E4404040404040
0002F0 0000000000000000                174            DC     XL8'0',CL8'NU'                       72500019
0002F8 D5E4404040404040
000300 0000000000000000                175            DC     XL8'0',CL8'OMICRON'                  73000019
000308 D6D4C9C3D9D6D540
000310 0000000000000000                176            DC     XL8'0',CL8'PHI'                      73500019
000318 D7C8C94040404040
000320 0000000000000000                177            DC     XL8'0',CL8'SIGMA'                    74000019
000328 E2C9C7D4C1404040
000330 00000505000000DA                178            DC     FL3'5',X'05',A(LOOP),CL8'ZETA'       74500019
000338 E9C5E3C140404040
                                       179 *                                                      75000019
                                       180 *          THIS IS THE CONTROL LIST                    75500019
                                       181 *                                                      76000019
000340 D3C1D4C2C4C14040                182 TESTLIST   DC     CL8'LAMBDA',X'0A',FL3'29',A(BEGIN)   76500019
000348 0A00001D00000000
000350 E9C5E3C140404040                183            DC     CL8'ZETA',X'05',FL3'5',A(LOOP)       77000019
000358 05000005000000DA
000360 E3C8C5E3C1404040                184            DC     CL8'THETA',X'82',FL3'45',A(BEGIN)    77500019
000368 8200002D00000000
000370 E3C1E44040404040                185            DC     CL8'TAU',X'80',FL3'0',A(1)           78000019
000378 8000000000000001
000380 D3C9E2E340404040                186            DC     CL8'LIST',X'9F',FL3'465',A(0)        78500019
000388 9F0001D100000000
000390 C1D3D7C8C1404040                187            DC     CL8'ALPHA',X'00',FL3'1',A(123)       79000019
000398 000000010000007B
                                       188 *                                                      79500019
                                       189 *          THESE ARE THE SYMBOLIC REGISTERS            80000019
                                       190 *                                                      80500019
000000                                 191 R0         EQU    0                                    81000019
000001                                 192 R1         EQU    1                                    81500019
000002                                 193 R2         EQU    2                                    82000019
000003                                 194 R3         EQU    3                                    82500019
000005                                 195 R5         EQU    5                                    83000019
000006                                 196 R6         EQU    6                                    83500019
000007                                 197 R7         EQU    7                                    84000019
00000C                                 198 R12        EQU    12                                   84500019
00000D                                 199 R13        EQU    13                                   85000019
00000E                                 200 R14        EQU    14                                   85500019
00000F                                 201 R15        EQU    15                                   86000019
                                       202 *                                                      86500019
                                       203 *          T IS IS THE FORMAT DEFINITION OF LIST ENTRYS 87000019
                                       204 *                                                      87500019
000000                                 205 LIST       DSECT                                       88000019
000000                                 206 LNAME      DS     CL8                                  88500019
000008                                 207 LSWITCH    DS     C                                    89000019
000009                                 208 LNUMBER    DS     FL3                                  89500019
00000C                                 209 LADDRESS   DS     F                                    90000019
                                       210 *                                                      90500019
                                       211 *          THIS IS THE FORMAT DEFINITION OF TABLE ENTRYS 91000019
                                       212 *                                                      91500019
000000                                 213 TABLE      DSECT                                       92000019
000000                                 214 TNUMBER    DS     FL3                                  92500019
000003                                 215 TSWITCH    DS     C                                    93000019
000004                                 216 TADDRESS   DS     F                                    93500019
000008                                 217 TNAME      DS     CL8                                  94000019
000000                                 218            END    BEGIN                                94500019


0003A0 000001F000000010                219                   =A(LISTAREA,16,LISTEND)
0003A8 00000240
0003AC 0000008000000004                220                   =F'128,4,128'
0003B4 00000080
```

```
 7                                                                                              9
EXAM                                          RELOCATION DICTIONARY                          PAGE    1
 18        19        20        21                                                             16
POS.ID    REL.ID    FLAGS    ADDRESS                                                       10/13/71

  01        01        0C      0001FC
  01        01        0C      00020C
  01        01        0C      00021C
  01        01        0C      0002D4
  01        01        0C      000334
  01        01        0C      00034C
  01        01        0C      00035C
  01        01        0C      00036C
  01        01        0C      0003A0
  01        01        0C      0003A8
```

```
7                                                                                    9
EXAM                                       CROSS-REFERENCE                        PAGE    1
 22         23     24     25         26                                              16
SYMBOL     LEN   VALUE  DEFN      REFERENCES                                       10/13/71

BEGIN     00004 000000 00057    0154  0156  0172  0182  0184  0218
EXIT      00004 00007E 00094    0109
HIGHER    00002 0000F4 00128    0123
IHB0005   00001 00007B 00091    0088
IHB0005A  00002 00007C 00092    0087
IHB0007   00001 0000B9 00106    0103
IHB0007A  00002 0000BA 00107    0102
LADDRESS  00004 00000C 00209    0078
LIST      00001 000000 00205    0065
LISTAREA  00008 0001F0 00154    0064  0083  0219
LISTEND   00008 000240 00159    0064  0219
LISTLOOP  00004 000038 00080    0111
LNAME     00008 000000 00206    0122
LNUMBER   00003 000009 00208    0075
LOOP      00004 0000DA 00121    0126  0129  0155  0178  0183
LSWITCH   00001 000008 00207    0072  0110
MORE      00004 000018 00066    0080
NONE      00001 000080 00114    0067  0110  0118  0130
NOTFOUND  00004 0000FA 00130    0127
NOTRIGHT  00004 00008C 00102    0082  0084
NOTTHERE  00004 0000C0 00110    0068
R0        00001 000000 00191
R1        00001 000001 00192    0069  0119  0120  0120  0125  0128
R12       00001 00000C 00198    0061  0062
R13       00001 00000D 00199    0094
R14       00001 00000E 00200    0066  0124  0131
R15       00001 00000F 00201
R2        00001 000002 00193    0126  0129
R3        00001 000003 00194    0119  0121  0125  0128
R5        00001 000005 00195    0064  0065  0080
R6        00001 000006 00196    0080
R7        00001 000007 00197    0064
SAMPLR    00001 000000 00055
SAVE13    00004 0000C8 00112    0063  0094
SEARCH    00004 0000CE 00118    0066
SWITCH    00001 0000CC 00113    0067  0118  0130
TABLAREA  00008 000100 00136    0081  0120
TABLE     00001 000000 00213    0069
TADDRESS  00004 000004 00216    0079
TESTLIST  00008 000340 00182    0083
TESTTABL  00003 000250 00164    0081
TNAME     00008 000008 00217    0122
TNUMBER   00003 000000 00214    0075
TSWITCH   00001 000003 00215    0072


NO STATEMENTS FLAGGED IN THIS ASSEMBLY
*STATISTICS*     SOURCE RECORDS (SYSIN) =    189      SOURCE RECORDS (SYSLIB) =    833
*OPTIONS IN EFFECT*   LIST, DECK, NOLOAD, NORENT, XREF, NOTEST, ALGN, OS, NOTERM, LINECNT =   70
  350 PRINTED LINES
```

Appendix D. Dynamic Invocation of the Assembler

The Assembler can be invoked by a problem program at execution time through the use of the CALL, LINK, XCTL, or ATTACH macro instructions. If the XCTL macro instruction is used to invoke the Assembler, then no user options may be stated. The Assembler will use the standard default, as set during system generation, for each option.

If the Assembler is invoked by CALL, LINK, or ATTACH, the user may supply:

1) The Assembler options
2) The ddnames of the data sets to be used during processing

Name	Operation	Operand
[symbol]	CALL	IEUASM, (optionlist [,ddnamelist]), VL
	{LINK / ATTACH}	EP=IEUASM, PARAM=(optionlist [,ddnamelist]), VL=1

EP - specifies the symbolic name of the Assembler. The entry point at which execution is to begin is determined by the control program (from the library directory entry).

PARAM - specifies, as a sublist, address parameters to be passed from the problem program to the Assembler. The first word in the address parameter list contains the address of the option list. The second word contains the address of the ddname list.

optionlist - specifies the address of a variable length list containing the options. This address must be written even if no option list is provided.

The option list must begin on a halfword boundary. The first two bytes contain a count of the number of bytes in the remainder of the list. If no options are specified, the count must be zero. The option list is free form with each field separated by a comma. No blanks or zeros should appear in the list.

ddnamelist - specifies the address of a variable length list containing alternate ddnames for the data sets used during compiler processing. If standard ddnames are used, then this operand may be omitted.

The ddname list must begin on a halfword boundary. The first two bytes contain a count of the number of bytes in the remainder of the list. Each name of less than eight bytes must be left-justified and padded with blanks. If an alternate ddname is omitted, the standard name will be assumed. If the name is omitted within the list, the 8-byte entry must contain binary zeros. Names can be omitted from the end merely by shortening the list. The sequence of the 8-byte entries in the ddname list is as follows:

Entry	Alternate Name
1	not applicable
2	not applicable
3	not applicable
4	SYSLIB
5	SYSIN
6	SYSPRINT
7	SYSPUNCH
8	SYSUT1
9	SYSUT2
10	SYSUT3
11	SYSGO
12	SYSTERM

VL - specifies that the sign bit is to be set to 1 in the last word of the address parameter list.

Appendix E. The SYSTERM Listing

The SYSTERM data set is designed to give the user of a remote terminal under the Time Sharing Option (TSO) quick access to the assembler diagnostics. It lists the diagnosed statement immediately fcllcwed by an errcr message, which tells the programmer what is wrong with the statement that has been flagged. To help identify the position of the statement in the program, SYSTERM also has facilities for printing the line number field (NUM option) and the statement number assigned by the assembler in front of the flagged statement. (STMT option).

The Assembler option TERM specifies that the assembler will write diagnostic information on the SYSTERM data set. If the programmer does not want the line number to be written, he should alsc specify the NONUM option. To prevent the statement number on the listing from being printed, he should specify the NOSTMT option in the PARM field of the EXEC card.

The format of the flagged statement on SYSTERM is:

Line No(s) (option NUM)	Statement No (option STMT)	Source record(s) (columns 1-72 of the source statement lines)

If a statement contains continuation lines it will occupy several lines on the listing, each identified by a line number (if option NUM is in effect). If a statement in error is discovered during the expansion of a macro, or of any inner macro called by the outer macro, the first line of the outer macro is listed before the flagged statement. If a statement is flagged during open code conditional assembly, the first line of the model statement will be listed before the statement in error.

Figures 20 and 21 illustrate the content and format of SYSTERM output. Figure 20 shows the source statement section of a SYSPRINT listing, and Figure 21 shows the SYSTERM listing produced during the same assembly. This example exemplifies the rules given above. Options TERM, NUM, and STMT have been in effect during this assembly.

The SYSTERM listing starts with the statement ASSEMBLER (F) DONE. At the end of the listing some diagnostic information is given: nnn STATEMENTS FLAGGED IN THIS ASSEMBLY, which indicates the total number of source statements in error, and nn WAS HIGHEST SEVERITY CODE, which specifies the maximum severity code encountered. This figure is equal to the return code passed by the assembler to the supervisor.

```
                                                                                          PAGE    1

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                              F01OCT71   9/27/71

                                      1           MACRO
                                      2           GENF  &P,&L
                                      3           LCLA  &K
                                      4 .LOOP     ANOP
                                      5 &K        SETA  &K+1
                                      6 &P&L(&K)  CC    F'&L(&K)'
                                      7           AIF   (&K LT N'&L).LCCP
                                      8 .DCNE     MEND
                                      9           GELC  &C
000000                               1C SAMPL2    CSECT
                                     11           SAVE  (14,12)        ALL REGS ARE SAVEC IN SLPERVISCR SAVEAREA
COCCCC                               12+          CS    0H
COCCCC 9CEC DCCC           CCCCC     13+          STM   14,12,12(13) SAVE REGISTERS
000004 05C0                          14           BALR  R12,0
CCCCC6                               15           USING *,R12          SET UP BASE REGISTER
                                     16 &C        SETC  'B'
000006 0000 0C00           CCCCC     17           L     R2,END         ENC CF AREA
        *** ERRCR ***
                                     18           LA    R3,A           THIS IS A CUMMY COMMENT                *
                                                                       TC SHCW A                              *
                                                                       STATEMENT CCNTAINING TOC               *
00000A CCCC 0C00           00000                                       MANY CONTINUATICN CARDS
        *** ERROR ***
00000E 5840 C022           0C028     19           L     R4,F0          ZERC CONSTANT FOR RESETTING AREA
000012 5043 0000           00000     20 LOOP      ST    R4,C(R3)
COCC16 4130 3C04           CCC04     21           LA    R3,4(,R3)      RESET AREA A
00001A 1923                          22           CR    R2,R3
00001C 4770 C00C           00012     23           BNE   LOOP
                                     24           AIF   ('A' EQ 'Q').GO
                                     25           SR    &C,&C          OPEN CODE MODEL STATEMENT              *
                                                                       WITH A CCNTINUATICN CARC
                                                  SR    B,B OPEN CODE MODEL STATEMENT                         X
COC02C CCCC                                                 WITH A CONTINUATION CARD
        *** ERROR ***
                                     26 .GO       RETURN (14,12)       EXIT FROM RCLTINE
COC022 98EC D0OC           0000C     27+          LM    14,12,12(13) RESTORE THE REGISTERS
000026 C7FE                          28+          BR    14 RETURN
                                     29 *
                                     30 *                   CONSTANTS AND AREAS HAVE BEEN CMITTED CN PURPCSE
                                     31 *
                                     32           GENF  F,0            GENERATION OF CONSTANTS
000028 00000000                      33+F0        DC    F'C'
                                     34           GENF  1,234          EXAMPLE OF MORE THAN CNE CARD          *
                                                                       IN A MACRO INSTRUCTION
00002C 000000EA                      35+1234      DC    F'234'
        *** ERROR ***
COCC02                               36 R2        ECU   2
000003                               37 R3        EQU   3
000004                               38 R4        EQU   4
COCCCC                               39 R12       ECU   12
                                     40           ENC
```

Figure 20. SYSPRINT Source Statement Listing

```
ASSEMBLER (F) DONE
             17            L     R2,END         END OF AREA
IEU024  NEAR OPERAND COLUMN   4--UNDEFINED SYMBOL
             18            LA    R3,A           THIS IS A DUMMY COMMENT                    *
                                                TO SHOW A                                  *
                                                STATEMENT CONTAINING TOO                   *
                                                MANY CONTINUATION CARDS
IEU077  ILLEGAL CONTINUATION CARD
IEU024  NEAR OPERAND COLUMN   4--UNDEFINED SYMBOL
             25            SR    &Q,&Q          OPEN CODE MODEL STATEMENT                  *
                           SR    B,B OPEN CODE MODEL STATEMENT                             X
                                    WITH A CONTINUATION CARD
IEU024  NEAR OPERAND COLUMN   1--UNDEFINED SYMBOL
IEU024  NEAR OPERAND COLUMN   3--UNDEFINED SYMBOL
             34            GENF  1,234          EXAMPLE OF MORE THAN ONE CARD              *
             35 +1234      DC    F'234'
IEU016  INVALID NAME
    4 STATEMENTS FLAGGED IN THIS ASSEMBLY
    8 WAS HIGHEST SEVERITY CODE
*OPTIONS IN EFFECT*   LIST, NODECK, NOLOAD, NORENT, XREF, NOTEST, ALGN, OS, TERM, NUM, STMT, LINECNT =  70
```

Figure 21. SYSTERM Assembly Output Listing
(Produced for the source statements shown in Figure 20.)

Index

Indexes to system reference library manuals are consolidated in the publication OS Master Index to Reference Manuals, Order No. GC28-6644. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

GC26-3756-6

OS Assembler (F) Programmer's Guide Printed in U. S. A. GC26-3756-6

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]

OS Assembler (F)
Programmer's Guide

GC26-3756-6

READER'S COMMENT FORM

CUT ALONG DOTTED LINE

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 request, please contact your IBM representative or the IBM Branch Office serving your locality.*

Reply requested:
Yes ☐
No ☐

Name: ____________________
Job Title: ____________________
Address: ____________________
____________________ Zip ____________________

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.)

GC26-3756-6

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

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 L
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)

Technical Newsletter

This Newsletter No. GN33-8144
Date July 15, 1972

Base Publication No. GC26-3756-6
File No. S360-21 (OS)

Previous Newsletters None

OS ASSEMBLER (F) PROGRAMMER'S GUIDE

This Technical Newsletter, a part of version 21 of IBM System/360 Operating System provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are listed below.

41-44	59-62
51-56	Reader's Comment Form
56.1 (added)	Reader's Comment Reply

A change to the text or to an illustration is indicated by a vertical line to the left of the change.

Summary of Amendments

Minor technical corrections.

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

IBM Nordic Laboratory, Publications Development, Box 962, S-181 09 Lidingö 9, Sweden

Printed in U.S.A.

ampersand (&) followed by 1-7 letters and/or numbers first of which must be a letter), or statement following 'MACRO' is not a valid prototype statement.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU095 ENTRY TABLE OVERFLOW

Explanation: Number of ENTRY symbols, i.e., ENTRY instruction operands, exceeds 100.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU096 MACRO INSTRUCTION OR PROTOTYPE OPERAND EXCEEDS 255 CHARACTERS IN LENGTH

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU097 INVALID FORMAT IN MACRO INSTRUCTION OPERAND OR PROTOTYPE PARAMETER

Explanation: This message can be caused by:

1. Illegal "=".
1. A single "&" appears somewhere in the standard value assigned to a prototype keyword parameter.
3. First character of a prototype parameter is not "&".
4. Prototype parameter is a subscripted variable symbol.
5. Invalid use of alternate format in prototype statement, e.g.,
```
10        16                  71
PROTO     &A,&B,
                  or
PROTO     &A,&B,              X
          &C
```
6. Unintelligible prototype parameter, e.g., "&A*" or "&A&&."
7. Illegal (non-assembler) character appears in prototype parameter or macro instruction operand.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU098 EXCESSIVE NUMBER OF OPERANDS OR PARAMETERS

Explanation: Either the prototype has more than 200 parameters, or the macro instruction has more than 100 operands.

Severity Code: 11

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU099 POSITIONAL MACRO INSTRUCTION OPERANE, PROTOTYPE PARAMETER OR EXTRA COMMA FOLLOWS KEYWORD

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU100 STATEMENT COMPLEXITY EXCEEDED

Explanation: More than 32 operands in a DC, DS, DXD, or literal DC, or more than 50 terms in a statement.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc cbtain a copy of the PDS member specified in the COPY statement.

IEU101 EOD ON SYSIN

Explanation: EOD before END card.

Severity Code: 12

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, dc the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc cbtain a copy of the PDS member specified in the COPY statement.

IEU102 INVALIE OR ILLEGAL ICTL

Explanation: The operands of the ICTL are out of range, or the ICTL is not the first statement in the input deck. (Assembly is terminated and further input is igncred.)

Severity Code: 16

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, dc the following before calling IBM:
- Have the user source program, user macro definitions and associated listing available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc cbtain a copy of the PDS member specified in the COPY statement.

IEU103 ILLEGAL NAME IN OPERAND FIELD OF COPY CARD

Explanation: Syntax error, e.g., symbol has more than 8 characters or has an illegal character.

Severity Code: 12

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program tc obtain a copy of the PDS member specified in the COPY statement.

IEU104 COPY CODE NOT FOUND

Explanation: The operand of a COPY statement specified COPY text which cannot be found in the library.

Severity Code: 12

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If problem recurs, do the following before calling IBM:
- Make sure the SYSLIB DD statement is included.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Have the user source prcgram, user macro definitions, and associated listings available.
- If the CCPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the CCPY statement.

IEU105 EOD CN SYSTEM MACRO LIBRARY

Explanation: EOD before MEND card.

Severity Code: 12

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source prcgram, user macro definitions, and associated listings available.
- If the CCPY statement was used, execute the IEBPTPCH utility prcgram to obtain a copy of the PDS member specified in the CCPY statement.

IEU106 NOT NAME OF DSECT CR DXD

Explanation: Referenced symbol expected to be DSECT name, but it is not.

Severity Code: 8

Programmer Response: Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source prcgram, user macro definitions and associated listing available.
- If the CCPY statement was used, execute the IEBPTPCH utility prcgram to obtain a copy of the PDS member specified in the CCPY statement.

IEU107 INVALID OPERAND

Explanation: Invalid syntax in DC operand, e.g., invalid hexadecimal character in hexadecimal DC; operand string too long for X, B, C, DC's; operand unrecognizable, contains invalid value, or incorrectly specified.

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU108 PREMATURE EOD

Explanation: Indicates an internal assembler error; should not occur.

Severity Code: 16

Programmer Response: Reassemble; if the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEU109 PRECISION LOST

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU110 EXPRESSION VALUE TOO LARGE

Explanation: Value of expression not in range than -16777216 to +16777215. Expressions in EQU and ORG statements are flagged if (1) they include terms previously defined as negative values, or (2) positive terms give a result of more than three bytes in magnitude. The error indication may be erroneous due to (1) the treatment of negative values as three-byte positive values, or (1) the effect of large positive values on the location counter if a control section begins with a START statement having an operand greater than zero, or a control section is divided into subsections.

Severity Code: 8

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU111 OPEN FAILED FOR SYSGO, NOLOAD OPTION USED

Explanation: DD statement incorrect or missing.

Severity Code: 16

Programmer Response: Probable user error. If necessary supply missing DD statement or make sure that information on DD statement is correct and reassemble. If problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEU112 OPEN FAILED FOR SYSPUNCH, NODECK OPTION USED

Explanation: DD statement incorrect or missing.

Severity Code: 16

Programmer Response: Probable user error. If necessary supply missing DD statement or make sure that information on DD statement is correct and reassemble. If problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEU113 OPEN FAILED FOR SYSTERM, NOTERM OPTION USED

Explanation: DD statement incorrect or missing.

Severity Code: 0

Programmer Response: Probable user error. If necessary supply missing DD statement or make sure that information on DD statement is correct and reassemble. If problem recurs, do the following before calling IBM:

- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEU116 ILLEGAL OPSYN

Severity Code: 8

Explanation: An OPSYN statement may be preceded only by an ICTL instruction or another OPSYN statement.

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU117 OPSYN TABLE OVERFLOW

Explanation: No room exists in symbol table for this and following OPSYN definitions; generated operation codes may not be processed correctly.

Programmer Response: Probable user error. Make sure the source code is correct and reassemble if necessary. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.

IEU996I ASSEMBLY TERMINATED INSUFFICIENT STORAGE

Explanation: One of the following:
- The partition or region size is less than the mimimum required by the assembler.
- The blocksize specified for the utility data sets is too large for available main storage.

System Action: Assembly is terminated.

Severity Code: 20

IEU997I OPEN FAILED FOR SYSPRINT, NOLIST OPTION USED

Explanation: DD statement incorrect or missing.

System Action: Processing continues.

Severity Code: 0

Programmer Response: Probable user error. If necessary supply the missing DD statement or make sure that information on the DD statement is correct; reassemble. If problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEU998I ASSEMBLY TERMINATED, OPEN FAILED FOR DATA SET (ddname)

Explanation: DD statement(s) for data set(s) SYSIN, SYSUT1, SYSUT2, SYSUT3, and/or SYSPRINT incorrect or missing.

System Action: Assembly is terminated.

Severity Code: 20

Programmer Response: Probable user error. Supply missing DD statement(s) or make sure that information on DD statement(s) is correct; reassemble. If problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEU999I ASSEMBLY TERMINATED, jobname, stepname, unit address, device type, ddname, operation attempted, error description (bytes 107 through 128 of the SYNADAF message buffer; this area is described in OS Data Management Macro Instructions.

Explanation: Indicates a permanent I/O error. This message is produced by the SYNADAF macro instruction.

System Action: Assembly is terminated.

Severity Code: 20

Programmer Response: Reassemble. If the problem recurs, do the following before calling IBM:
- Have the user source program, user macro definitions, and associated listings available.
- If the COPY statement was used, execute the IEBPTPCH utility program to obtain a copy of the PDS member specified in the COPY statement.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

Appendix B. Object Deck Output

TXT Card Format

The format of the TXT cards is as fcllows:

Columns	Contents
1	12-2-9 punch
2-4	TXT
5	Blank
6-8	Relative address of first instruction on card
9-10	Blank
11-12	Byte count -- number of bytes in information field (cc 17-72)
13-14	Blank
15-16	ESDID
17-72	56-byte information field
73-76	Deck ID (from first TITLE card)
77-80	Card sequence number

RLD Card Format

The format of the RLD card is as follows:

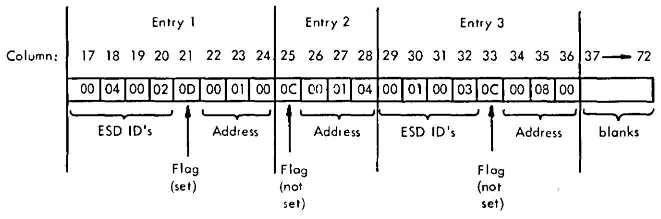
Columns	Contents
1	12-2-9 punch
2-4	RLC
5-10	Blank
11-12	Data field count -- number of bytes of information in data field (cc 17-72)
13-16	Blank
17-72	Data field:
17-18	Relocation ESDID
19-20	Position ESDID
21	Flag byte
22-24	Absolute address to be relocated
25-72	Remaining RLD entries
73-76	Deck ID (from first TITLE card)
77-80	Card sequence number

If the rightmost bit of the flag byte is set, the following RLD entry has the same Relocation ESCIC and Position ESDID, and this information will not be repeated; if the rightmost bit of the flag byte is not set, the next RLC entry has a different Relocation ESDID and/or Position ESCIC, and both ESCICs will be recorded.

For example, if the RLD Entries 1, 2, and 3 of the program listing (Appendix C) contain the following information:

	Pos. ESDID	Rel. ESDID	Flag	Address
Entry 1	02	04	0C	000100
Entry 2	02	04	0C	000104
Entry 3	03	01	0C	000800

Columns 17-36 of the RLD card wculd appear as follows:



Entry 1
Entry 2
Entry 3
Column: 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 → 72
00 04 00 02 0D 00 01 00 0C 00 01 04 00 01 00 03 0C 00 08 00
ESD ID's
Address
Address
ESD ID's
Address
blanks
Flag (set)
Flag (not set)
Flag (not set)

ESD Card Format

The format of the ESD card is as fcllows:

Columns	Contents		
1	12-2-9 punch		
2-4	ESD		
5-10	Blank		
11-12	Variable field ccunt -- number of bytes of information in variable field (cc 17-64)		
13-14	Blank		
15-16	ESDID of first SD, XD, CM, WX, PC, or ER in variable field		
17-64	Variable field. One to three 16-byte items of the following format:		
	8 bytes	--	Name, padded with blanks
	1 byte	--	ESD type code The hex value is:
			00 SD
			01 LD
			02 ER
			04 PC
			05 CM
			06 XD (PR)
			0A WX
	3 bytes	--	Address
	1 byte	--	Alignment if XD; otherwise blank
	3 bytes	--	Length, LDID, or blank
65-72	Blank		
73-76	Deck ID (from first TITLE card)		
77-80	Card sequence number		

END Card Format

The format of the END card is as follows:

Columns	Contents
1	12-2-9 punch
2-4	END
5	Blank
6-8	Entry address from operand of END card in source deck (blank if no operand)
9-14	Blank
15-16	ESDID of entry point (blank if no operand)
17-39	Blank
40-62	Version of the assembler (e.g., F 14FEB66, time of the assembly (hh.mm), and date of the assembly (mm/dd/yy). (See "Assembler Listing" section.)

SYM Card Format

If requested by the user, the assembler punches out SYM cards with symbolic information concerning the assembled program. These cards can be used by the TESTRAN routine or the TSO Test command processor. The cards are located between the ESD and TXT cards. The format of SYM cards is as follows:

Columns	Contents
1	12-2-9 punch
2-4	SYM
5-10	Blank
11-12	Variable field count -- number of bytes of text in variable field (cc 17-72)
13-16	Blank
17-72	Variable field (see below)
73-76	Deck ID (from first TITLE card)
77-80	Card sequence number

The variable field (columns 17-72) contains up to 56 bytes of TESTRAN text. The items making the text are packed together, consequently only the last card may contain less than 56 bytes of text in the variable field. The formats of a text card and an individual text item are shown in Figure 19. The contents of the fields within an individual entry are as follows:

1. Organization (1 byte)
   0 = non-data type
   1 = data type

   Bits 1-3 (if non-data type):
   000 = space
   001 = control section
   010 = dummy control section
   011 = common
   100 = machine instruction
   101 = CCW

   Bit 1 (if data type):
   0 = no multiplicity
   1 = multiplicity (indicates presence of M field)

   Bit 2 (if data type):
   0 = independent (not a packed or zoned decimal constant)
   1 = cluster (packed or zoned decimal constant)

   Bit 3 (if data type):
   0 = no scaling
   1 = scaling (indicates presence of S field)

   Bit 4:
   0 = name present
   1 = name not present

   Bits 5-7:
   Length of name minus one

2. Address (3 bytes) - displacement from beginning of control section

3. Symbol Name (0-8 bytes) - symbolic name of particular item

Note: The following fields are only present for data-type items.

4. Data Type (1 byte) - contents in hexadecimal

   00 = character
   04 = hexadecimal, L-type data
   08 = binary
   10 = fixed point, full
   14 = fixed point, half
   18 = floating point, short
   1C = floating point, long
   20 = A-type or Q-type data
   24 = Y-type data
   28 = S-type data
   2C = V-type data
   30 = packed decimal
   34 = zoned decimal

5. Length (2 bytes for character, hexadecimal, or binary items; 1 byte for other types) - length of data item minus 1

6. Multiplicity - M field (3 bytes) - equals 1 if not present

7. Scale -signed integer - S field (2 bytes) - present only for F, H, E, D, L, P and Z type data, and only if scale is non-zero.