SC26-3759-1

Program Product

OS Assembler H Programmer's Guide

IBM

SC26-3759-1

Program Product

OS Assembler H Programmer's Guide

Program Number 5734-AS1

IBM

Second Edition (June, 1972)

This is a major revision of, and obsoletes, SC26-3759-0 and Technical Newsletters SN33-8095 and SN33-8124.

This edition applies to version 4 of the OS Assembler H Program Product, Program Number 5734-AS1 and to all subsequent modifications until otherwise indicated in new editions or Technical Newsletters. Changes to the text and to illustrations are indicated by a vertical line to the left of the change.

Changes are continually made to the information 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.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

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

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

©Copyright International Business Machines Corporation 1970, 1971, 1972

Preface

This publication tells how to use Assembler H. It describes assembler options, cataloged job control language procedures, assembler listing and output, assembler data sets, error diagnostic facilities, sample programs, and programming techniques and considerations.

Assembler H is an assembler language processor for the Operating System. It performs high-speed assemblies on an IBM System/360 Model 40 or higher and on an IBM System/370 Model 145 or higher with at least 256K bytes of main storage.

This manual has the following main sections:

- Using the Assembler
- Assembler Listing Description
- Assembler Diagnostic Facilities
- Programming Considerations

"Using the Assembler" describes the EXEC statement PARM field option, the data sets used by the assembler, and the job control language cataloged procedures supplied by IBM. The cataloged procedures can be used to assemble, linkage edit or load, and execute an assembler program.

"Assembler Listing Description" describes each field of the assembly listing. "Assembler Diagnostic Facilities" describes the purpose and format of error messages, MNOTEs, and the MHELP macro trace facility. "Programming Considerations" discusses various topics, such as standard entry and exit procedures for problem programs.

Appendix A is a sample program which demonstrates many of the assembler language features, especially those unique to Assembler H. Appendix B is a sample MHELP macro trace and dump. Appendix C describes the object module output formats. Appendix D tells how to call the assembler dynamically from problem programs.

This publication is intended for all Assembler H programmers. To use this publication, you should be familiar with the assembler language and with the basic concepts and facilities of the Operating System, especially job control language, data management services, supervisor services, and the linkage editor and loader. To use this publication effectively, the reader should be familiar with the OS Introduction, Order Number GC28-6534 or have the equivalent knowledge.

Assembler Publications

The following publication contains a brief description of Assembler H and how it differs from lower level OS assemblers:

| OS Assembler H General Information Manual, Order Number GC26-3758.

The following publications describe the assembler language and the information required to run Assembler H programs:

| OS/VS and DOS/VS Assembler Language, Order Number GC33-4010.

OS Assembler Language, Order Number GC28-6514.

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

OS Assembler H Language, Order Number GC26-3771.

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

OS Assembler H Messages, Order Number SC26-3770.

The Messages manual provides an explanation of each of the diagnostic and abnormal termination messages issued by Assembler H and suggests how you should respond in each case.

The following publications contain information used tc install and maintain Assembler H:

OS Assembler H System Information, Order Number GC26-3768.

The System Information manual consists of three self-contained chapters on performance estimates, storage estimates, and system generation of Assembler H.

OS Assembler H Logic, Order Number LY26-3760.

The Logic manual describes the design logic and functional characteristics of Assembler H.

Operating System Publications

The following OS books are referenced in this publication:

OS/VS JCL Reference, Order Number GC28-0618, or

OS Job Control Language Reference, Order Number GC28-6704.

OS/VS Linkage Editor and Loader, Order Number GC26-3803, or

OS Loader and Linkage Editor, Order Number GC28-6538.

OS/VS Supervisor Services and Macros, Order Number GC27-6979, or

OS Supervisor Services and Macro Instructions, Order Number GC28-6646.

OS/VS Utilities, Order Number GC35-0005, or

OS Utilities, Order Number GC28-6586.

Contents

Figures

Using the Assembler

This section describes the assembly time options available to the assembler language programmer, the data sets used by the assembler, and the cataloged procedures of job control language supplied by IBM to simplify assembling, linkage editing or loading, and execution of assembly language programs. The job control language is described in detail in the Job Control Language Reference publication.

Assembler Options

Assembler H offers a number of optional facilities. For example, you can suppress printing of the assembly listing or parts of the listing, and you can specify whether you want an object deck or an object module. You select the options by including appropriate keywords in the PARM field of the EXEC statement that invokes the assembler. There are two types of options:

- Simple pairs of keywords: a positive form (such as OBJECT) that requests a facility, and an alternative negative form (such as NOOBJECT) that rejects that facility.

- Keywords that permit you to assign a value to a function (such as LINECOUNT(50).

Each of these options has a standard or default value which is used for the assembly if you do not specify an alternative value. The default values are explained in the following section, "Default Options."

If you are using a cataloged procedure, you must include the PARM field in the EXEC statement that invokes the procedure. You must also qualify the keyword (PARM) with the name of the step within the procedure that invokes the compiler. For example:

```
//    EXEC    ASMHC,PARM.C='OBJECT,NODECK'
```

The section "Overriding Statements in Cataloged Procedures" contains more examples on how to specify options in a cataloged procedure.

PARM is a keyword parameter: code PARM= followed by the list of options, separating the options by commas and enclosing the entire list within single quotes or parentheses. If you specify only one option and it does not include any special characters, the enclosing quotes or parentheses can be omitted. The option list must not be longer than 100 characters, including the separating commas. You may specify the options in any order. If contradictory options are used (for example, LIST and NOLIST), the rightmost option (in this case, NOLIST) is used.

The assembler options are:

```
        (DECK,   OBJECT,   LIST,  TEST,  'XREF(FULL/SHORT)',                       ALIGN,  RENT,
PARM=     or        or      or     or           or             'LINECOUNT(nn)',    or      or
        (NODECK,NOOBJECT, NOLIST,NOTEST,NOXREF,                                    NOALIGN,NORENT,

        ESD,  RLD,  BATCH,
         or    or     or     'SYSPARM(string),FLAG(nnn)')
        NOESD,NORLD,NOBATCH,
```

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

OBJECT -- The object module is placed on the device specified in the SYSLIN DD statement.

Note: The OBJECT and DECK options are independent of each other. Both or neither can be specified. The output on SYSLIN and SYSPUNCH is identical except that the control program closes SYSLIN with a disposition of LEAVE and SYSPUNCH with a disposition of REREAD.

ESD -- The assembler produces the External Symbol Dictionary as part of the listing.

RLD -- The assembler produces the Relocation Dictionary as part of the listing.

BATCH -- The assembler will do multiple (batch) assemblies under the control of a single set of job control language cards. The source decks must be placed together with no intervening /* card; a single /* card must follow the final source deck.

LIST -- An assembler listing is produced. Note that the NOLIST option overrides the ESD, RLD, and XREF options.

TEST -- The object module contains the special source symbol table required by the test translator (TESTRAN) routine.

XREF(FULL) -- The assembler listing will contain a cross reference table of all symbols used in the assembly. This includes symbols that are defined but never referenced. The assembler listing will also contain cross reference table of literals used in the assembly.

XREF(SHORT) -- The assembler listing will contain a cross reference table of all symbols that are referenced in the assembly. Any symbols defined but not referenced are not included in the table. The assembler listing will also contain a cross reference table of literals used in the assembly.

RENT -- The assembler checks for a possible coding violation of program reenterability.

LINECOUNT(nn) -- The number of lines to be printed between headings in the listing is nn. The permissible range is 1 to 99 lines.

NOALIGN -- The assembler suppresses the diagnostic message "IEV033 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 that are not aligned on the proper (halfword) boundary or for data boundary violations for privileged instructions such as LPSW. DC, DS, DXD, or CXD constants, usually causing alignment, are not aligned. See the "Special CPU Programming Considerations" section for information on alignment requirements.

ALIGN -- The assembler does not suppress the alignment error diagnostic message; all alignment errors are diagnosed.

FLAG(nnn) -- Error diagnostic messages below severity code nnn will not appear in the listing. Diagnostic messages can have severity codes of 0, 4, 8, 12, 16, or 20 (0 is the least severe). MNOTEs can have a severity code of 0 through 255.

For example, FLAG(8) will suppress messages for severity codes 0 through 7.

SYSPARM(string) -- 'string' is the value of the system variable symbol &SYSPARM. The assembler uses &SYSPARM as a read-only SETC variable. If no value is specified for the SYSPARM option, &SYSPARM will be a null (empty) character string. The function of &SYSPARM is explained in the Assembler H Language Specifications and in OS/VS and DOS/VS Assembler Language.

Due tc JCL restrictions, you cannot specify a SYSPARM value longer than 56 characters (as explained in Note 1). Twc quotes are needed to represent a single quote, and two ampersands to represent a single ampersand. For example:

```
PARM='OBJECT,SYSPARM((&&AM,''EO).FY)'
```

assigns the following value to &SYSPARM:

```
(&AM,'EO).FY .
```

Any parentheses inside the string must be paired. If you call the assembler from a problem program (dynamic invocation), SYSPARM can be up to 256 characters long.

Note 1: The restrictions imposed upon the PARM field limit the maximum length of the SYSPARM value to 56 characters. Consider the following example:

```
//  EXEC  ASMFC,PARM.C=(OBJECT,NODECK,
//  'SYSPARM(ABCD.........................................)')
↑   ↑        ↑                                             ↑
col 1  col 4   col 13              56 bytes                col 68
```

Since SYSPARM uses parentheses, it must be surrounded by quotes. Thus, it cannot be continued onto a continuation card. The leftmost column that can be used is column 4 on a continue card. A quote and the keyword must appear on that line as well as the closing quotes. In addition, either a right parenthesis, indicating the end of the PARM field, or a comma, indicating that the PARM field is continued on the next card, must be coded before or in the last column of the statement field (column 71).

Note 2: Even though the formats of some of the options previously supported by Assembler H have been changed, you can use the old formats for the following options: ALGN (now ALIGN), NOALGN (NOALIGN), LINECNT=nn (LINECOUNT(nn)), LOAD (OBJECT), and NOLOAD (NOOBJECT).

Default Options

If you do not code an option in the PARM field, the assembler assumes a default option. The following default options are included when Assembler H is shipped by IBM:

```
PARM=(DECK,NOOBJECT,LIST,NOTEST,'XREF(FULL),LINECOUNT(55)',ALIGN,NOBATCH,'SYSPARM( ), FLAG(0)')
```

However, these may not be the default options in effect in your installation. The defaults can be respecified when Assembler H is installed. For example, NOCECK can be made the default in place of DECK. Also, a default option can be specified during installation so that you cannot override it.

The cataloged procedures described in this book assume the default entries. The section "Overriding Statements in Cataloged Procedures" tells you how to override them. First, however, check whether any default options have been changed or whether there are any you cannot override at your installation.

Assembler Data Sets

Assembler H requires the following data sets, as shown in Figure 1:

- SYSUT1 -- utility data set used as intermediate external storage.
- SYSIN -- an input data set containing the source statements to be processed.

In addition, the following four data sets may be required:

- SYSLIB -- a data set containing macro definitions (for macro definitions not defined in the source program) and/cr source code to be called for through COPY assembler instructions.
- SYSPRINT -- a data set containing the assembly listing (unless the NOLIST option is specified).
- SYSPUNCH -- a data set containing object module output, usually for punching (unless the NODECK option is specified).
- SYSLIN -- a data set containing object module cutput usually for the linkage editor (only if the OBJECT option is specified).

The above data sets are described in the following text. The CDname that normally must be used in the DD statement describing the data set appears as the heading for each description. The characteristics of these data sets, those set by the assembler and those you can override, are shown in Figures 2 and 3.

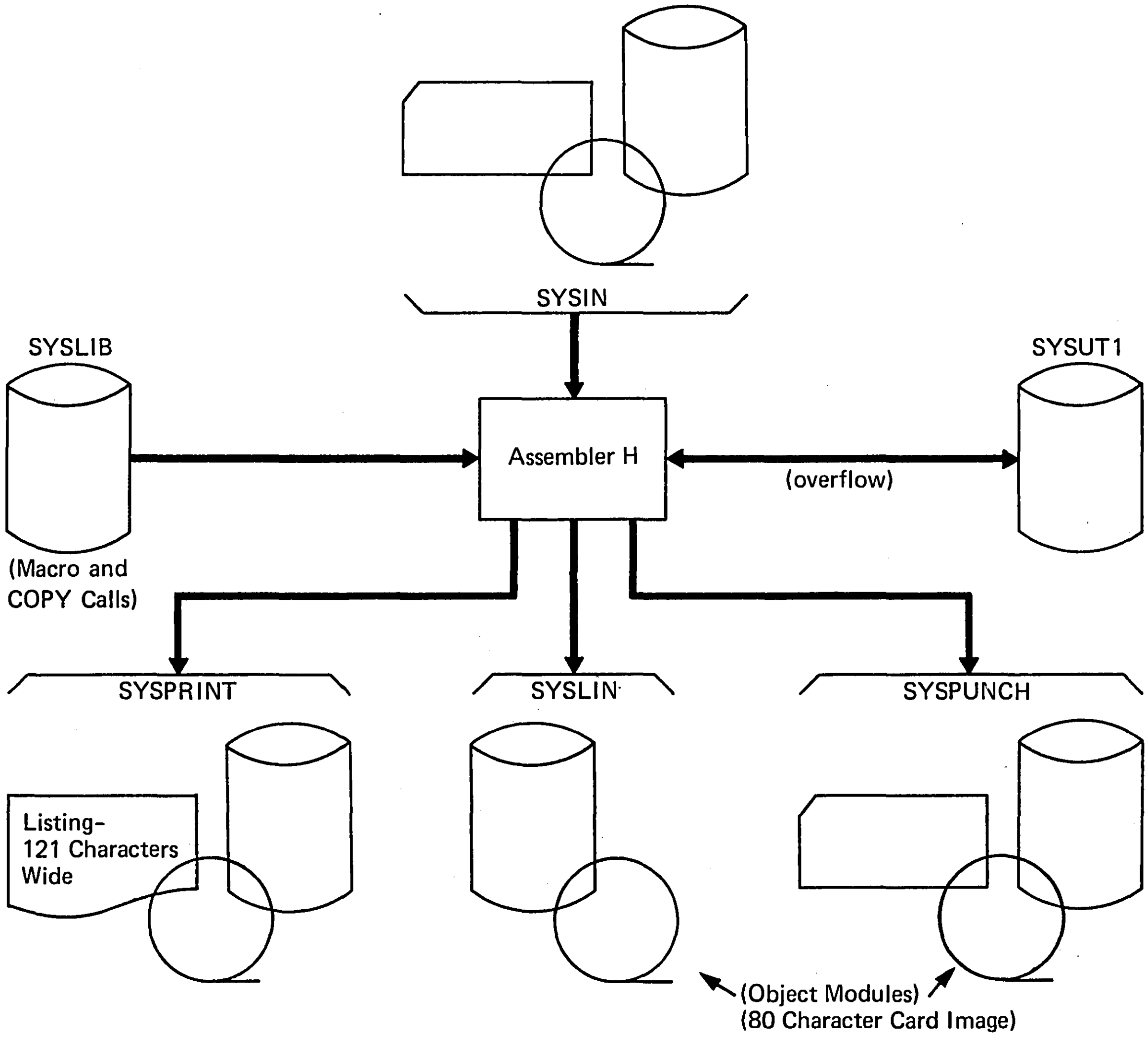


Figure 1. Assembler H Data Sets

DDNAME SYSUT1

The assembler uses this utility data set as an intermediate external storage device when processing the source program. The input/output device assigned to this data set must be a direct access device. The assembler does not support a multi-volume utility data set. The IBM 2321 Data Cell is not supported for this data set.

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 that you have designated. The DD statement describing this data set appears in the input stream. The IBM-supplied procedures do not contain this statement.

DDNAME 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; each macro definition or sequence of assembler language statements is a separate member, with the member name being the macro instruction mnemonic or COPY operand name.

The data set may be defined as SYS1.MACLIB or your private macro definition or COPY library. SYS1.MACLIB contains macro definitions for the system macro instructions provided by IBM. Your private library may be concatenated with SYS1.MACLIB. The two libraries must have the same logical record length (80 bytes), but the blocking factors may be different. The DD statement for the library with the largest blocksize must appear first in the job control language for the assembly (that is, before any other library DD statements). The Job Control Language Reference publication, explains the concatenation of 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, or direct-access storage device. 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 SYSLIN

This is a direct-access storage device, magnetic tape, or card punch data set used by the assembler. It contains the same cutput text as SYSPUNCH. It is used as input for the linkage editor.

Data Set	SYSUT1	SYSPUNCH	SYSPRINT	SYSLIN	SYSIN	SYSLIB
Access Method	BSAM	BSAM	BSAM	BSAM	BSAM	BPAM
Logical Record Length (LRECL)	fixed at BLKSIZE	fixed at 80	fixed at 121	fixed at 80	fixed at 80	fixed at 80
Block Size (BLKSIZE)	①	②	②	②	②	③
Record Format (RECFM)	④	④⑥	⑤⑥	④⑥	④⑥	④⑥
Number of channel Programs (NCP)	①	⑦	⑦	⑦	⑦	Not Applicable

① You can specify a blocksize (BLKSIZE) between 2008 and 5100 bytes in the DD statement or in the data set label. BLKSIZE should be a multiple of 8; if it is not, it will be rounded to the next lower multiple of 8. If you do not specify BLKSIZE, the assembler sets a default blocksize based on the device used for SYSUT1 as follows:

2301 Drum	5016 bytes
2302 Disk	4984 bytes
2303 Drum	4888 bytes
2305 Drum model 1	4280 bytes
2305 Drum model 2	4688 bytes
2311 Disk	3624 bytes
2314 Disk	3520 bytes
3330 Disk	4208 bytes

The Storage Estimates chapter of the System Information manual, Order Number SC26-3768, discusses the reasons for changing the default blocksize.

② If specified, BLKSIZE must equal LRECL or a multiple of LRECL. If BLKSIZE is not specified, it is set equal to LRECL. If BLKSIZE is not a multiple of LRECL, it is truncated.

③ BLKSIZE be specified in the DD statement or the data set label as a multiple of LRECL.

④ Set by the assembler to F or FB if necessary.

⑤ Set by the assembler to FM or FBM if necessary.

⑥ You may specify B, S, or T.

⑦ You can specify the number of channel programs (NCP) used by any assembler data set except SYSUT1 and SYSLIB. The NCP of SYSUT1 is fixed at 1. The assembler, however, can change your NCP specification under certain conditions. Figure 3 shows how NCP is calculated. Note that if the NCP is greater than 2, chained I/O request scheduling is set by the assembler.

Figure 2. Assembler Cata Set Characteristics

Number of Channel Programs (NCP)

The number of channel programs can be specified by the user or set by the assembler. The number will vary depending upon whether or not a unit record device is used. The following table shows how the NCP selection is made.

	Unit record device	No unit record device
NCP specified ≥ 2	User specified	User specified
NCP specified = 1	Computed[1]	User specified (= 1)
NCP not specified	Computed[1]	Computed[1]

Figure 3. Number of Channel Program (NCP) Selection

[1] For SYSPRINT data set, the NCP set by the assembler is the larger of 1210/BLKSIZE or 2.
For SYSIN data set, the NCP set by the assembler is the larger of 800/BLKSIZE or 2.
For SYSLIN or SYSPUNCH data set, the NCP set by the assembler is the larger of 240/BLKSIZE or 2.

Note: If the NCP is greater than 2, chained I/O scheduling is set by the assembler.

Return Codes

Assembler H issues return codes for use with the COND parameter of the JOB and EXEC job control language statements. The COND parameter enables you to skip or execute a job step depending on the results (indicated by the return code) of a previous job step. It is explained in the Job Control Language Reference publication.

The return code issued by the assembler is the highest severity code that is associated with any error detected in the assembly or with any MNOTE message produced by the source program or macro instructions. See the Assembler H Messages publication, for a listing of the assembler errors and their severity codes.

Cataloged Procedures

Often the same set of job control statements is used over and over again (for example, to specify the compilation, linkage editing, and execution of many different programs). To save programming time and to reduce the possibility of error, sets of standard series of EXEC and DD statements can be prepared once and 'cataloged' in a system library. Such a set of statements is termed a cataloged procedure and can be invoked by one of the following statements:

```
//stepname EXEC      procname

//stepname EXEC      PROC=procname
```

The specified procedure is read from the procedure library (SYS1.PROCLIB) and merged with the job control statements that follow this EXEC statement.

This section describes four IBM cataloged procedures: a procedure for assembling (ASMHC), a procedure for assembling and linkage editing (ASMHCL), a procedure for assembling, linkage editing, and executing (ASMHCLG), and a procedure for assembling and loader-executing (ASMHCG).

CATALOGED PROCEDURE FOR ASSEMBLY (ASMHC)

This procedure consists of one job step: assembly. The name ASMHC 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. An example of the statements entered in the input stream to use this procedure is:

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

The statements of the ASMHC procedure are read from the procedure library and merged into the input stream.

Figure 4 shows the statements that make up the ASMHC procedure.

```
1 //C            EXEC   PGM=IEV90,REGION=200K

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

3 //SYSUT1       DD     UNIT=(SYSDA,SEP=SYSLIB),SPACE=(CYL,(10,5)),DSN=&SYSUT1

4 //SYSPUNCH     DD     SYSOUT=B,DCB=(BLKSIZE=800),SPACE=(CYL,(5,5,0))

5 //SYSPRINT     DD     SYSOUT=A,DCB=(BLKSIZE=3509),UNIT=(,SEP=(SYSUT1,SYSPUNCH))

   ::::::::
```

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 IEV90 identifies Assembler H.

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

3 This statement specifies the assembler utility data set. The device classname used here, SYSDA, represents a direct-access unit. The I/O unit assigned to this name is specified by the installation when the operating system is generated. A unit name such as 2311 may be substituted for SYSDA.

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

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

Figure 4. Cataloged Procedure for Assembly (ASMHC)

CATALOGED PROCEDURE FOR ASSEMBLY AND LINKAGE EDITING (ASMHCL)

This procedure consists of two job steps: assembly and linkage editing. The name ASMHCL 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 illustrates input to the assembler via the input job stream. SYSLIN contains the output from the assembly step and the input to the linkage edit step. It can be concatenated with additional input to the linkage editor as shown in the example. This additional input can be linkage editor control statements or other object modules.

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

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

Figure 5 shows the statements that make up the ASMHCL procedure. Only those statements not previously discussed are explained.

```
  //C          EXEC   PGM=IEV90,PARM=OBJECT,REGION=200K

  //SYSLIB     DD     DSN=SYS1.MACLIB,DISP=SHR

  //SYSUT1     DD     UNIT=(SYSDA,SEP=SYSLIB),SPACE=(CYL,(10,5)),DSN=&SYSUT1

  //SYSPUNCH   DD     SYSOUT=B,DCB=(BLKSIZE=800),SPACE=(CYL,(5,5,0))

  //SYSPRINT   DD     SYSOUT=A,DCB=(BLKSIZE=3509),UNIT=(,SEP=(SYSUT1,SYSPUNCH))

1 //SYSLIN     DD     DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(5,5,0)),             *
  //                  DCB=(BLKSIZE=400),DSN=&&LOADSET

2 //L          EXEC   PGM=IEWL,PARM='MAP,LET,LIST,NCAL',REGION=96K,COND=(8,LT,C)

3 //SYSLIN     DD     DSN=&&LOADSET,DISP=(OLD,DELETE)

4 //           DD     DDNAME=SYSIN

5 //SYSLMOD    DD     DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(2,1,2)),DSN=&GOSET(GO)

6 //SYSUT1     DD     UNIT=SYSDA,SPACE=(CYL,(3,2)),DSN=&SYSUT1

7 //SYSPRINT   DD     SYSOUT=A,DCB=(RECFM=FB,BLKSIZE=3509)
   --------
```

1 In this procedure the SYSLIN 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, a 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 as the same one (SYSLIN) produced as output from the assembler.

4 This statement is used to concatenate any input to the linkage editor from the input stream (object decks and/or linkage editor control statements) 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 DSN 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 DSN 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 5. Cataloged Procedure for Assembling and Linkage Editing (ASMHCL)

CATALOGED PROCEDURE FOR ASSEMBLY, LINKAGE EDITING, AND EXECUTION (ASMHCLG)

This procedure consists of three job steps: assembly, linkage editing, and execution.

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

The name ASMHCLG must be used to call this procedure. An assembler listing, an object deck, and a linkage editor listing are produced.

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

```
//jobname      JOB
//stepname     EXEC PROC=ASMHCLG
//C.SYSIN      DD    *
                I
                I
                I
       source program statements
                I
                I
                I
/*                                   }
//L.SYSIN       DD    *              }
                I                    }
                I                    }  necessary only if linkage
                I                    }  editor is to combine modules
       object module or              }  or read linkage editor control
       linkage editor                }  information from the job stream
       control statements            }
                I                    }
/*              I                    }
//G.ddname      DD   (parameters)        }
//G.ddname      DD   (parameters)        }
//G.ddname      DD    *                  }
                I                        }
                I                        }  only if necessary
                I                        }
       problem program input             }
                I                        }
                I                        }
/*                                       }
```

```
  //C          EXEC  PGM=IEV90,PARM=OBJECT,REGION=200K
  //SYSLIB     DD    DSN=SYS1.MACLIB,DISP=SHR
  //SYSUT1     DD    UNIT=(SYSDA,SEP=SYSLIB),SPACE=(CYL,(10,5)),DSN=&SYSUT1
  //SYSPUNCH   DD    SYSOUT=B,DCB=(BLKSIZE=800),SPACE=(CYL,(5,5,0))
  //SYSPRINT   DD    SYSOUT=A,DCB=(BLKSIZE=3509),UNIT=(,SEP=(SYSUT1,SYSPUNCH))
  //SYSLIN     DD    DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(5,5,0)),            *
  //                 DCB=(BLKSIZE=400),DSN=&&LOADSET
1 //L          EXEC  PGM=IEWL,PARM='MAP,LET,LIST,NCAL',REGION=96K,COND=(8,LT,C)
  //SYSLIN     DD    DSN=&&LOADSET,DISP=(OLD,DELETE)
  //           DD    DDNAME=SYSIN
2 //SYSLMOD    DD    DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(2,1,2)),DSN=&GOSET(GO)
  //SYSUT1     DD    UNIT=SYSDA,SPACE=(CYL,(3,2)),DSN=&SYSUT1
  //SYSPRINT   DD    SYSOUT=A,DCB=(RECFM=FB,BLKSIZE=3509)
3 //G          EXEC  PGM=*.L.SYSLMOD,COND= ((8,LT,C),(4,LT,L))
   ::::::::
```

[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 *.L.SYSLMOD identifies the program to be executed as being in the data set described in job step L by the DD statement named SYSLMOD.

Figure 6. Cataloged Procedure for Assembly, Linkage Editing and Execution (ASMHCLG)

CATALOGED PROCEDURE FOR ASSEMBLY AND LOADER EXECUTION (ASMHCG)

This procedure consists of two job steps: assembly and loader execution. Loader-execution is a combination of linkage editing and loading the program for execution. Load modules for program libraries are not produced.

```
  //C            EXEC    PGM=IEV90,PARM=OBJECT,REGION=200K

  //SYSLIB       DD      DSN=SYS1.MACLIB,DISP=SHR

  //SYSUT1       DD      UNIT=(SYSDA,SEP=SYSLIB),SPACE=(CYL,(10,5)),DSN=&SYSUT1

  //SYSPUNCH     DD      SYSOUT=B,DCB=(BLKSIZE=800),SPACE=(CYL,(5,5,0))

  //SYSPRINT     DD      SYSOUT=A,DCB=(BLKSIZE=3509),UNIT=(,SEP=(SYSUT1,SYSPUNCH))

  //SYSLIN       DD      DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(5,5,0)),                *
  //                     DCB=(BLKSIZE=400),DSN=&&LOADSET

1 //G            EXEC    PGM=LOADER,PARM='MAP,LET,PRINT,NOCALL'

2 //SYSLIN       DD      DSN=&&LOADSET,DISP=(OLD,DELETE)

  //             DD      DDNAME=SYSIN

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 and print the map and diagnostics. The NOCALL option is the same as NCAL for the linkage editor and the LET option is the same as for the 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 7. Cataloged Procedure for Assembly and Loader-Execution (ASMHCG)

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

The name ASMHCG 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=ASMHCG
//C.SYSIN       DD   *
                |
                |
                |
     source program
                |
                |
/*
//G.ddname      DD   (parameters)   \
//G.ddname      DD   (parameters)    |
//G.ddname      DD   *               |
                |                    |
                |                     > only 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

Any EXEC parameter (except PGM) can be overridden. For example, the PARM= and COND= parameters can be added or, if present, respecified 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.

If the procedure consists of more than one job step, a PARM.procstepname= or COND.procstepname= parameter may be entered for each step. The entries must be in order (PARM.procstep1=, PARM.procstep2=, 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 //procstepname.DDname in the name field. "Procstepname" refers to the job step in which the statement identified by "DDname" appears.

Note: If more than one DD statement in a procedure is to be overridden, the overriding statements must be in the same order as the statements in the procedure.

Examples

In the assembly procedure ASMHC (Figure 4), the production of a punched object deck could be suppressed and the UNIT= and SPACE= parameters of data set SYSUT1 respecified, by including the following statements in the input stream:

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

In procedure ASMHCLG (Figure 6), 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=ASMHCLG,                      X
//                      PARM.C=(NOLIST,OBJECT),            X
//                      COND.L=(8,LT,stepname.C)
```

For this execution of procedure ASMHCLG, no assembler listing would be produced, and execution of the linkage editor job step //L would be suppressed if the return code issued by the assembler (step C) were greater than 8.

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

The following listing shows how to use the procedure ASMHCL (Figure 5) 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 tape labeled TAPE10, with a data set name of PROG1 and a blocking factor of 5.

3. Block the SYSLIN output of the assembler and use it as input to the linkage editor with a blocking factor of 5.

4. Linkage edit the module only if there are no errors in the assembler (COND=0).

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

```
//jobname       JOB
//stepname      EXEC    PROC=ASMHCL,                          X
//                      COND.L=(0,NE,stepname.C)
//C.SYSPRINT    DD      DSNAME=PROG1,UNIT=TAPE,               X
//                      VOLUME=SER=TAPE10,DCB=(BLKSIZE=605)
//C.SYSLIN      DD      DCB=(BLKSIZE=800)
//C.SYSIN       DD      UNIT=282,LABEL=(,NL),                 X
//                      DCB=(RECFM=FBS,BLKSIZE=800)
//L.SYSIN       DD      DCB=stepname.C.SYSLIN
//L.SYSLMOD     DD      DSNAME=USER.LIBRARY(PROG),DISP=OLD
/*
```

Note: The order of appearance of overriding DDnames for job step C corresponds to the order of DDnames in the procedure; that is, SYSPRINT precedes SYSLIN within step C. The DDname C.SYSIN was placed last because SYSIN does not occur at all within step C. These points are covered in the Job Control Language Reference manual.

The following example shows assembly of two programs, linkage editing of the two assemblies into one load module, and execution of the load module. The input stream appears as follows:

```
//stepname1     EXEC    PROC=ASMHC,PARM.C=OBJECT
//C.SYSLIN      DD      DSNAME=&LOADSET,UNIT=SYSSQ,         X
//                      SPACE=(80,(100,50)),                X
//                      DISP=(MOD,PASS),DCB=(BLKSIZE=800)
//C.SYSIN       DD      *
                        |
                        |
                        |
                        source program 1 statements
                        |
                        |
/*
//stepname2     EXEC    PROC=ASMHCLG
//C.SYSLIN      DD      DCB=(BLKSIZE=800),DISP=(MOD,PASS)
//C.SYSIN       DD      *
                        |
                        |
                        |
                        source program 2 statements
                        |
                        |
                        |
/*
//L.SYSLIN      DD      DCB=BLKSIZE=800
//L.SYSIN       DD      *
                ENTRY   PROG
/*
//G.ddname              dd cards for G step
```

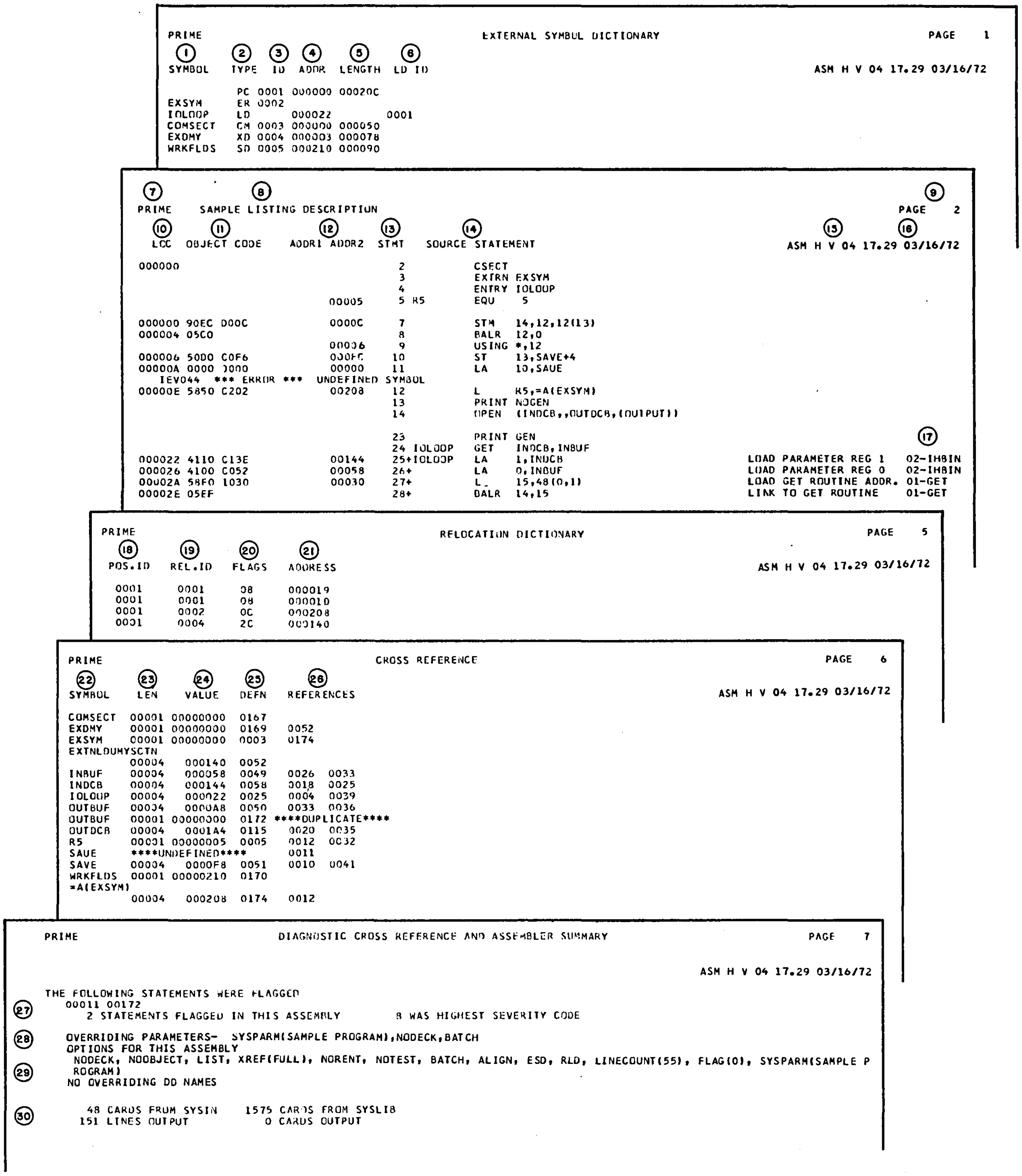
The Job Control Language Reference manual provides additional description of overriding techniques.

Assembler Listing

The assembler H listing consists of up to five sections, ordered as follows:

- External symbol dictionary
- Source and object program
- Relocation dictionary
- Symbol and literal cross reference
- Diagnostic cross reference and assembler summary

Figure 8 shows each section of the listing. Each item marked with a circled number is explained in the following section.



```
PRIME                                          EXTERNAL SYMBOL DICTIONARY                         PAGE    1

SYMBOL    TYPE  ID   ADDR   LENGTH  LD ID                                       ASM H V 04 17.29 03/16/72

          PC 0001 000000 00020C
EXSYM     ER 0002
IOLOOP    LD      000022         0001
COMSECT   CM 0003 000000 000050
EXDMY     XD 0004 000003 000078
WRKFLDS   SD 0005 000210 000090
```

```
PRIME     SAMPLE LISTING DESCRIPTION                                                                PAGE    2
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                          ASM H V 04 17.29 03/16/72

000000                               2            CSECT
                                     3            EXTRN EXSYM
                                     4            ENTRY IOLOOP
                         00005       5 R5         EQU    5

000000 90EC D00C         0000C       7            STM    14,12,12(13)
000004 05C0                          8            BALR   12,0
                         00006       9            USING *,12
000006 50D0 C0F6         000FC      10            ST     13,SAVE+4
00000A 0000 0000         00000      11            LA     10,SAUE
   IEV044  *** ERROR ***  UNDEFINED SYMBOL
00000E 5850 C202         00208      12            L      R5,=A(EXSYM)
                                    13            PRINT NOGEN
                                    14            OPEN   (INDCB,,OUTDCB,(OUTPUT))

                                    23            PRINT GEN
                                    24 IOLOOP     GET    INDCB,INBUF
000022 4110 C13E         00144      25+IOLOOP     LA     1,INDCB                     LOAD PARAMETER REG 1   02-IHBIN
000026 4100 C052         00058      26+           LA     0,INBUF                     LOAD PARAMETER REG 0   02-IHBIN
00002A 58F0 1030         00030      27+           L_     15,48(0,1)                  LOAD GET ROUTINE ADDR. 01-GET
00002E 05EF                         28+           BALR   14,15                       LINK TO GET ROUTINE    01-GET
```

```
PRIME                                        RELOCATION DICTIONARY                      PAGE    5
 POS.ID    REL.ID    FLAGS    ADDRESS                                       ASM H V 04 17.29 03/16/72

  0001      0001      08      000019
  0001      0001      08      000010
  0001      0002      0C      000208
  0001      0004      2C      000140
```

```
PRIME                                    CROSS REFERENCE                         PAGE    6
SYMBOL     LEN     VALUE    DEFN    REFERENCES                          ASM H V 04 17.29 03/16/72

COMSECT   00001 00000000   0167
EXDMY     00001 00000000   0169    0052
EXSYM     00001 00000000   0003    0174
EXTNLDUMYSCTN
          00004   000140   0052
INBUF     00004   000058   0049    0026  0033
INDCB     00004   000144   0058    0018  0025
IOLOUP    00004   000022   0025    0004  0039
OUTBUF    00004   0000A8   0050    0033  0036
OUTBUF    00001 00000000   0172  ****DUPLICATE****
OUTDCB    00004   0001A4   0115    0020  0035
R5        00001 00000005   0005    0012  0032
SAUE      ****UNDEFINED****        0011
SAVE      00004   0000F8   0051    0010  0041
WRKFLDS   00001 00000210   0170
=A(EXSYM)
          00004   000208   0174    0012
```

```
   PRIME                           DIAGNOSTIC CROSS REFERENCE AND ASSEMBLER SUMMARY                      PAGE    7

                                                                                          ASM H V 04 17.29 03/16/72
   THE FOLLOWING STATEMENTS WERE FLAGGED
      00011 00172
         2 STATEMENTS FLAGGED IN THIS ASSEMBLY        8 WAS HIGHEST SEVERITY CODE

      OVERRIDING PARAMETERS-  SYSPARM(SAMPLE PROGRAM),NODECK,BATCH
      OPTIONS FOR THIS ASSEMBLY
       NODECK, NOOBJECT, LIST, XREF(FULL), NORENT, NOTEST, BATCH, ALIGN, ESD, RLD, LINECOUNT(55), FLAG(0), SYSPARM(SAMPLE P
       ROGRAM)
      NO OVERRIDING DD NAMES

         48 CARDS FROM SYSIN     1575 CARDS FROM SYSLIB
        151 LINES OUTPUT            0 CARDS OUTPUT
```

Figure 8. Assembler H Listing

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 entry, shown in Figure 9 along with their associated fields. The circled numbers refer to the corresponding headings in the sample listing (Figure 8). The Xs indicate entries accompanying each type designation.

① SYMBOL	② TYPE	③ ID	④ ADDR	⑤ LENGTH	⑥ LD ID
X	SD	X	X	X	–
X	LD	–	X	–	X
X	ER	X	–	–	–
–	PC	X	X	X	–
X	CM	X	X	X	–
X	XD	X	X	X	–
X	WX	X	–	–	–

Figure 9. Types of ESD Entries

①The name of every external dummy section, control section, entry point, and external symbol.

②The type designator for the entry, as shown in the table. The type designators are defined as:

SD -- Control section definition. The symbol appeared in the name field of a CSECT or START statement.

LD -- Label definition. The symbol appeared as the operand of an ENTRY statement.

ER -- External reference. The symbol appeared as the operand of an EXTRN statement, or was declared as a V-type address constant.

PC -- Unnamed control section definition (private code). A CSECT or START statement that commences a control section does not have a symbol in the name field, or a control section is commenced (by any instruction which affects the location counter) before a CSECT or START is encountered.

CM -- Common control section definition. The symbol appeared in the name field of a COM statement.

XD -- External dummy section. The symbol appeared in the name field of a DXD statement or a Q-type address constant. (The external dummy section is also called a pseudo register in the <u>Loader and Linkage Editor</u> manual.

WX -- Weak external reference. The symbol appeared as an operand in a WXTRN statement.

③The external symbol dictionary identification number (ESDID). The number is a unique four-digit hexadecimal number identifying the entry. It is used in combination with the LD entry of the ESD and in the relocation dictionary for referencing the ESD.

④The address of the symbol (in hexadecimal notation) for SD- and

LD-type entries, and blanks 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. For example, 7 indicates doubleword alignment.

⑤ The assembled length, in bytes, of the control section (in hexadecimal notation).

⑥ For an LD-type entry, the ESDID of the control section in which the symbol was defined.

Source and Object Program

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

⑦ The one to eight-character deck identification, if any. It is obtained from the name field of the first named TITLE statement. The assembler prints the deck identification and date (item 16) on every page of the listing.

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

⑨ The listing page number.

⑩ The assembled address (in hexadecimal notation) of the object code.

- For ORG statements, the location-counter value before the ORG is placed in the location column and the location counter value after the ORG is placed in the object code field.
- If the END statement contains an operand, the operand value (transfer address) appears in the location field (LOC).
- In the case of LOCTR, COM, CSECT, and DSECT statements, the location field contains the current address of these control sections.
- In the case of EXTRN, WXTRN, ENTRY, and DXD instructions, the location field and object code field are blank.
- For a USING statement, the location field contains the value of the first operand. It is four bytes long.
- For LTORG statements, the location field contains the location assigned to the literal pool.
- For an EQU statement, the location field contains the value assigned. It is four bytes long.

⑪ 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). Only the first eight bytes of a constant will appear in the listing if PRINT NODATA is in effect, unless the statement has continuation cards. The entire constant appears if PRINT DATA is in effect. (See the PRINT assembler instruction in the Assembler Language publication.)

⑫ Effective addresses (each the result of adding together a base register value and a displacement value):

The field headed ADDR1 contains the effective address for the first operand of an SS instruction.

The field headed ADDR2 contains the effective address of the last 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 The statement number. A plus sign (+) to the right of the number indicates that the statement was generated as the result of macro call processing. An unnumbered statement with a plus sign (+) is the result of open code substitution.

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

- 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 PRINT, which is always printed.

- Macro definitions obtained from SYSLIB are not listed unless the macro definition is included in the source program by means of a COPY statement.

- The statements generated as the result of a macro call follow the macro call in the listing unless PRINT NOGEN is in effect.

- Assembler and 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.

- All error diagnostic messages appear in line except those suppressed by the FLAG option. The "Assembler Diagnostics Facilities" section describes how error messages and MNOTEs are handled.

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

- Whenever possible, a generated statement is printed in the same format as the corresponding macro definition (model) statement. The starting columns of the operation, operand, and comments fields are preserved unless they are displaced by field substitution, as shown in the following example:

```
Source Statements:     &C   SETC        'ABCDEFGHIJK'
                       &C   LA          1,4
Generated Statement:   ABCDEFGHIJK LA 1,4
```

It is possible for a generated statement to occupy three or more continuation lines on the listing. In this way generated statements are unlike source statements, which are restricted to two continuation lines.

⑮ The version identifier of Assembler H.

⑯ The current date (date run is made).

⑰ The identification-sequence field from the source statement. For a macro-generated statement, this field contains infcrmation identifying the origin of the statement. The first two columns define the level of the macro call.

For a library macro call, the last five columns contain the first five characters of the macro name. For a macro whose definition is in the source program (including one read by a COPY statement), the last five characters contain the line number of the model staterrent in the definition from which the generated statement is derived. This information can be an important diagnostic aid in analyzing output resulting from macro calls within macro calls.

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 The external symbol dictionary ID number assigned tc the ESD entry for the control section in which the address constant is used as an operand.

⑲ The external symbol dictionary ID number assigned to the ESD entry for the control section in which the referenced syrrbol is defined.

⑳ The two-digit hexadecimal number represented by the characters in this field is interpreted as follows:

- 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 indicates that the entry 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	Bit 3
00 = 1 byte	0 = +	Always 0
01 = 2 bytes	1 = -	
10 = 3 bytes		
11 = 4 bytes		

㉑ The assembled address of the field where the address constant is stored.

Cross Reference

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

㉒ The symbols or literals.

㉓ The length (in decimal notation), in bytes, of the field represented by the symbol. The length of a literal is always 1.

㉔ Either the address the symbol or literal represents, or a value to which the symbol is equated. The value is three bytes long, except for the following, which are four bytes long: CSECT, DSECT, START, COM, DXD, EQU, LOCTR, EXTRN, WXTRN, and a duplicate symbol.

㉕ The number of the statement in which the symbol or literal was defined.

㉖ The statement numbers of statements in which the symbol or literal appears as an operand. In the case of a duplicate symbol or literal, 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 nct appear in the cross-reference listing.
- Cross-reference entries for symbols used in a literal refer to the assembled literal in the literal pool. Look up the literals in the cross reference to find where the symbols are used.
- 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 fields 23, 24, and 25 with the message:

    ****UNDEFINED****.

Diagnostic Cross Reference and Assembler Summary

㉗ The statement number of each statement flagged with an error message or MNOTE appears in this list. The number of statements flagged and the highest non-zero severity code encountered is also printed. The highest severity code is equal to the assembler return code.

If no errors are encountered, the following statement is printed:

    NO STATEMENTS FLAGGED IN THIS ASSEMBLY

See the section "Error Diagnostics" for a complete discussion of how error messages and MNOTEs are handled.

㉘ A list of the options in effect for this assembly is printed. The options specified by the programmer in the PARM field to override the assembler default options are also printed.

㉙ If the assembler has been called by a problem program (See Appendix D) and any standard (default) DDnames have been overridden, both the default DDnames and the overriding DDnames are listed. Otherwise, this statement appears:

NO OVERRIDING DD NAMES

30 The assembler prints the number of records read frcm SYSIN and SYSLIB and the number of records written on SYSPUNCH. The assembler also prints the number of lines written on SYSPRINT. 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 (for example, for a SPACE 2, no blank records are generated). The assembler does not generate a blank record to force a page eject.

Assembler Diagnostic Facilities

The diagnostic facilities for Assembler H include diagnostic messages for assembly errors, diagnostic or explanatory messages issued by the source program or by macro definitions (MNOTEs), a macro trace and dump facility (MHELP), and messages and dumps issued by the assembler in case it terminates abnormally.

This section briefly describes these facilities. The assembly error diagnostic messages and abnormal assembly termination messages are described in detail in the OS Assembler H Messages bock.

Assembly Error Diagnostic Messages

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

The messages do not follow the statement in error when:

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

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

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

A typical error diagnostic message is:

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

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

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

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

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

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

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

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

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

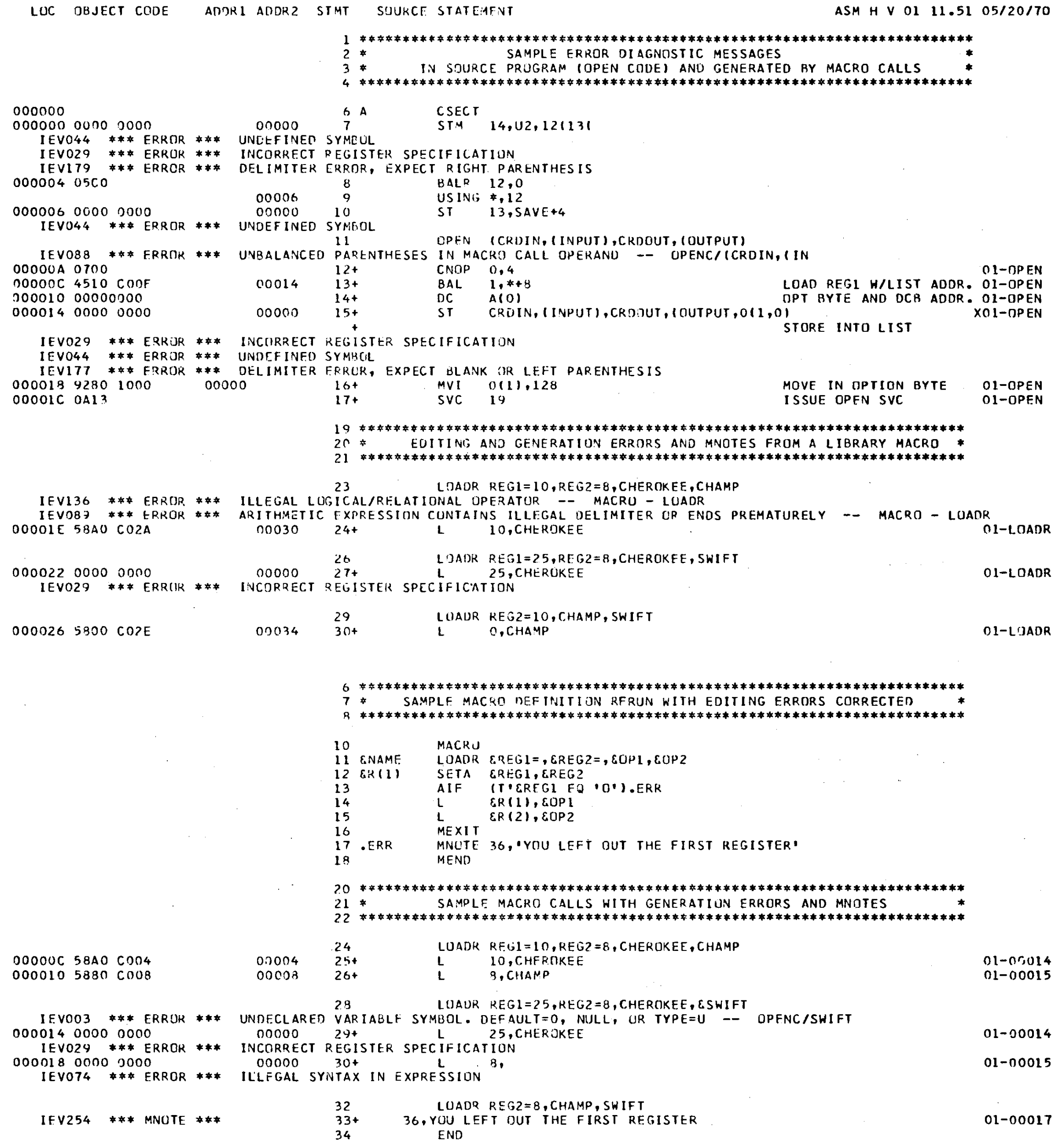
If a diagnostic message fcllows a conditional assembly statement in the source program, the following items will be appended to the error message:

- The word "OPENC"

- The SET symbol or value string associated with the error

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

Figure 10 is an example of Assembler H handling of errcr messages.



```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                      ASM H V 01 11.51 05/20/70

                                       1 ***********************************************************************
                                       2 *                SAMPLE ERROR DIAGNOSTIC MESSAGES                     *
                                       3 *      IN SOURCE PROGRAM (OPEN CODE) AND GENERATED BY MACRO CALLS     *
                                       4 ***********************************************************************

000000                                 6 A        CSECT
000000 0000 0000            00000      7          STM   14,U2,12(13(
   IEV044  *** ERROR ***   UNDEFINED SYMBOL
   IEV029  *** ERROR ***   INCORRECT REGISTER SPECIFICATION
   IEV179  *** ERROR ***   DELIMITER ERROR, EXPECT RIGHT PARENTHESIS
000004 05C0                            8          BALR  12,0
                            00006      9          USING *,12
000006 0000 0000            00000     10          ST    13,SAVE+4
   IEV044  *** ERROR ***   UNDEFINED SYMBOL
                                      11          OPEN  (CRDIN,(INPUT),CRDOUT,(OUTPUT)
   IEV088  *** ERROR ***   UNBALANCED PARENTHESES IN MACRO CALL OPERAND  --  OPENC/(CRDIN,(IN
00000A 0700                           12+         CNOP  0,4                                                      01-OPEN
00000C 4510 C00F            00014     13+         BAL   1,*+8                             LOAD REG1 W/LIST ADDR. 01-OPEN
000010 00000000                       14+         DC    A(0)                              OPT BYTE AND DCB ADDR. 01-OPEN
000014 0000 0000            00000     15+         ST    CRDIN,(INPUT),CRDOUT,(OUTPUT,0(1,0)                    X01-OPEN
                                        +                                                 STORE INTO LIST
   IEV029  *** ERROR ***   INCORRECT REGISTER SPECIFICATION
   IEV044  *** ERROR ***   UNDEFINED SYMBOL
   IEV177  *** ERROR ***   DELIMITER ERROR, EXPECT BLANK OR LEFT PARENTHESIS
000018 9280 1000      00000           16+         MVI   0(1),128                          MOVE IN OPTION BYTE    01-OPEN
00001C 0A13                           17+         SVC   19                                ISSUE OPEN SVC         01-OPEN

                                      19 ***********************************************************************
                                      20 *     EDITING AND GENERATION ERRORS AND MNOTES FROM A LIBRARY MACRO   *
                                      21 ***********************************************************************

                                      23          LOADR REG1=10,REG2=8,CHEROKEE,CHAMP
   IEV136  *** ERROR ***   ILLEGAL LOGICAL/RELATIONAL OPERATOR  --  MACRO - LOADR
   IEV089  *** ERROR ***   ARITHMETIC EXPRESSION CONTAINS ILLEGAL DELIMITER OR ENDS PREMATURELY  --  MACRO - LOADR
00001E 58A0 C02A            00030     24+         L     10,CHEROKEE                                              01-LOADR

                                      26          LOADR REG1=25,REG2=8,CHEROKEE,SWIFT
000022 0000 0000            00000     27+         L     25,CHEROKEE                                              01-LOADR
   IEV029  *** ERROR ***   INCORRECT REGISTER SPECIFICATION

                                      29          LOADR REG2=10,CHAMP,SWIFT
000026 5800 C02E            00034     30+         L     0,CHAMP                                                  01-LOADR


                                       6 ***********************************************************************
                                       7 *    SAMPLE MACRO DEFINITION RERUN WITH EDITING ERRORS CORRECTED      *
                                       8 ***********************************************************************

                                      10          MACRO
                                      11 &NAME    LOADR &REG1=,&REG2=,&OP1,&OP2
                                      12 &R(1)    SETA  &REG1,&REG2
                                      13          AIF   (T'&REG1 EQ 'O').ERR
                                      14          L     &R(1),&OP1
                                      15          L     &R(2),&OP2
                                      16          MEXIT
                                      17 .ERR     MNOTE 36,'YOU LEFT OUT THE FIRST REGISTER'
                                      18          MEND

                                      20 ***********************************************************************
                                      21 *        SAMPLE MACRO CALLS WITH GENERATION ERRORS AND MNOTES         *
                                      22 ***********************************************************************

                                      24          LOADR REG1=10,REG2=8,CHEROKEE,CHAMP
00000C 58A0 C004            00004     25+         L     10,CHEROKEE                                              01-00014
000010 5880 C008            00008     26+         L     8,CHAMP                                                  01-00015

                                      28          LOADR REG1=25,REG2=8,CHEROKEE,&SWIFT
   IEV003  *** ERROR ***   UNDECLARED VARIABLE SYMBOL. DEFAULT=0, NULL, OR TYPE=U  --  OPENC/SWIFT
000014 0000 0000            00000     29+         L     25,CHEROKEE                                              01-00014
   IEV029  *** ERROR ***   INCORRECT REGISTER SPECIFICATION
000018 0000 0000            00000     30+         L     8,                                                       01-00015
   IEV074  *** ERROR ***   ILLEGAL SYNTAX IN EXPRESSION

                                      32          LOADR REG2=8,CHAMP,SWIFT
   IEV254  *** MNOTE ***              33+     36,YOU LEFT OUT THE FIRST REGISTER                                 01-00017
                                      34          END
```

Figure 10. Sample Error Diagnostic Messages

MNOTEs

An MNOTE statement is included in a macro definition or in the source program. It causes the assembler to generate an inline error or informational message.

An MNOTE appears in the listing as follows:

```
IEV254 ***MNOTE***   severity code, message
```

Unless it has a severity code of * or the severity code is omitted, the statement number or the MNOTE is listed in the diagnostic cross reference.

Suppression of Error Messages and MNOTEs

Error messages and MNOTEs below a specified severity level can be optionally suppressed by declaring in the EXEC statement: PARM='FLAG(n)' (where "n" is the selected severity level).

Abnormal Assembly Termination

Whenever the assembly cannot be completed, Assembler H provides a message and, in some cases, a specially formatted dump for diagnostic information. This may indicate an assembler malfunction or it may indicate a programmer error. The statement causing the error is identified and, if possible, the assembly listing up to the point of the error is printed. The OS Assembler H Messages book, describes the abnormal termination messages. The messages give enough information to (1) correct the error and reassemble your program, or (2) determine that the error is an assembler malfunction.

The CS Assembler H Logic manual, gives a complete explanation of the format and contents of the abnormal termination dump.

Macro Trace Facility (MHELP)

The MHELP instruction controls a set of trace and dump facilities. Options are selected by an absolute expression in the MHELP operand field. MHELP statements can occur anywhere in open code or in macro definitions. MHELP options remain in effect continuously until superseded by another MHELP statement. Appendix B is a sample MHELP trace and dump.

Macro Call Trace

(MHELP B'1' or MHELP 1). This option provides a one-line trace for each macro call, giving the name of the called macro, its nested depth, and its &SYSNDX (total number of macro calls) value.

Note: This trace is provided upon entry into the macro. No trace is provided if error conditions prevent entry into the macro.

Macro Branch Trace

(MHELP B'10', or MHELP 2). This option provides a one-line trace for each AGO and true AIF conditional-assembly statement within a macro. It gives the model-statement numbers of the "branched from" and "branched to" statements, and the name of the macro in which the branch occurs. This trace option is suppressed for library macros.

Macro Entry Dump

(MHELP B'10000', or MHELP 16), This option dumps parameter values from the macro dictionary when the macro is called.

Macro Exit Dump

(MHELP B'10000', or MHELP 8). This option dumps SET symbol values from the macro dictionary upon encountering a MEND or MEXIT statement.

Macro AIF Dump

(MHELP B'100', or MHELP 4). This option dumps SET symbol values from the macro dictionary immediately before each AIF statement that is encountered.

Global Suppression

(MHELP B'100000', or MHELP 32). This option suppresses global SET symbols in the two preceding options, MHELP 4 and MHELP 8.

MHELP Suppression

(MHELP B'10000000', or MHELP 128). This option suppresses all currently active MHELP options.

Combining Options

Multiple options can be obtained by combining the option codes in one MHELP operand. For example, call and branch traces can be invoked by MHELP B'11', MHELP 2+1, or MHELP 3.

MHELP Control on &SYSNDX

The MHELP operand field is actually mapped into a full word. Previously defined MHELP codes correspond to the fourth byte of this fullword.

&SYSNDX control is turned on by any bit in the third byte (operand values 256-65535 inclusive). Then, when &SYSNDX (total number of macro calls) exceeds the value of the fullword which contains the MHELP operand value, control is forced to stay at the open-code level, by in effect making every statement in a macro behave like a MEXIT. Open code macro calls are honored, but with an immediate exit back to open code.

Examples:

```
MHELP 256      Limit &SYSNDX to 256.
MHELP 1        Trace macro calls.
MHELP 256+1    Trace calls and limit &SYSNDX to 257.
MHELP 65536    No effect.  No bits in bytes 3,4.
MHELP 65792    Limit &SYSNDX to 65792.
```

When the value of &SYSNDX reaches its limit, the message "ACTR EXCEEDED -- &SYSNDX" is issued.

Programming Considerations

This section discusses some topics in assembler language programming.

Saving and Restoring General Register Contents

A problem program should save the values contained in the general registers 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 should be the first statement in the program. It stores the contents of registers 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 contains the address of an area in which the general contents should be saved.

If the program calls any subprograms, 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 of 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 contents of the registers. A complete discussion of the SAVE and RETURN macro instructions and the saving and restoring of registers is contained in the Supervisor Services and Macro Instructions publication.

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

You indicate completion of an assembler language source program by using the RETURN system macro instruction to pass control from the terminating program to the program that initiated it. The initiating program may be the Operating System or, if a subprogram issued the RETURN, the program that called the subprogram.

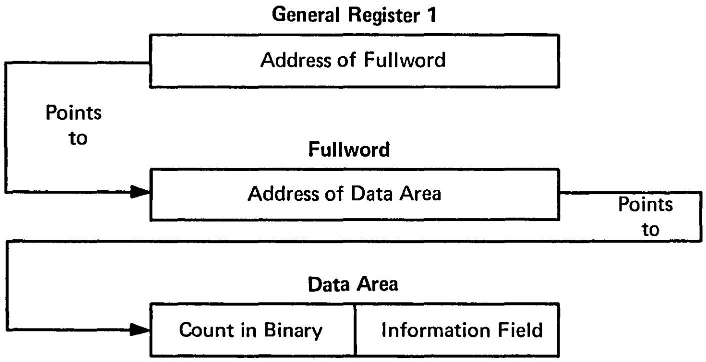
In addition to indicating program completion and restcring register contents, the RETURN macro instruction may also pass a return code -- a condition indicator that may be used by the program receiving control. If the return is to the operating system, the return code is compared against the condition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

The RETURN system macro instruction is discussed in detail in | Supervisor Services and Macro Instructions.

PARM Field Access

Access to information in the PARM field of and EXEC statement is gained through general register 1. When control is given to the problem program, general register 1 contains the address of a fullword 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 fullword 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 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   JOE
//stepname  EXEC      PGM=IEBUPDTE,PARM=MOD
//SYSUT1    CC        DSNAME=SYS1.MACLIB,DISP=OLD
//SYSUT2    CC        DSNAME=SYS1.MACLIB,DISP=OLD
//SYSPRINT  CC        SYSOUT=A
//SYSIN     DD        *
            .
            .
            .
IEBUPCTE control statements and source statements cr
macro definitions to be added to the macro library
(SYS1.MACLIB)
            .
            .
            .
/* (deliriter 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 pcint must have been identified originally as an external symbol; that is, it must have appeared as an entry in the external symbol dicticnary. External symbol identification is done automatically by the assembler if the entry point is the name of a control section or SIART statement; otherwise, an assembler ENTRY statement must be used tc identify the entry point name as an external symbol.

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

- By placing the entry pcint symbol in the operand field of an EXTRN statement and an ENC statement in the new object rcdule.

- By using an END statement in the new object module to designate a new entry point in the 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 Loader and Linkage Editor publication.

Object Module Linkage

Object modules, whether generated by the assembler or ancther language processor, 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 linkage an assembler language main program to subprograms produced by another processor. The Supervisor Services and Macro Instructions publication, contains additional details concerning linkage conventions and the CALL system macro instruction.

Figure 11 is an example of statements used tc establish the assembler language program linkage to FORTRAN and COBOL subprograms.

If any input/cutput operations are performed by called subprograms, appropriate CC statements for the data sets used by the subprograms must be supplied. See the appropriate language programmer's guide for an explanation of the DD statements and special data set record formats used for the processor.

```
  ENTRPT SAVE   (14,12)
         LR     12,15
         USING  ENTRPT,12
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 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-language 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.

NOTE: 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 ompilation. The FORTRAN IV Library publication, Order Number GC28-6596, 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. Sample Assembler Linkage Statements for FORTRAN or COBOL Subprograms

Special CPU Programming Considerations

You should be aware of operational differences between the Model 85, Model 91, and Model 195 and other System/360 models. The primary differences are:

- Non-sequential instruction execution -- 91 and 195
- Extended precision machine instructions -- 85 and 195
- Unaligned operands -- 85 and 195

CONTROLLING INSTRUCTION EXECUTION SEQUENCE

The Model 91 and Model 195 maintain a logical consistency with respect to their own operations, including the beginning and ending of I/C operations, but they do 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 instruction sequence.

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

```
      BCR          N,0       N=1,15
```

causes instruction decoding to halt until the instructions that have already been decoded are executed. (This action is called a pipe-line drain.) On the Model 91 and Model 195, 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, because it may affect the performance of the CPU.

Isolating an instruction by preceding it and following it with a BCR N,0 instruction eliminates multiple imprecise interruptions from more than one instruction by virtue of the pipe-line drain effect. However, because 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 you that you 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.

Note: BCR 0,0 does not cause a pipe-line drain.

EXTENDED-PRECISION MACHINE INSTRUCTIONS

The extended-precision arithmetic instructions and the rounding instructions of the Model 85 and the Model 195 are shown below. A complete description of these instructions is in the IBM System/360 Principles of Operation.

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

A program containing the extended-precision instructions cannot be executed successfully on another System/360 model unless those instructions are converted into others that can be executed by the non-Model 85 or Model 195 machine. The OPSYN assembler instruction helps provide a facility for doing this.

OPSYN is described in the OS Assembler H Language Specifications manual or, for VS, in the Assembler Language manual.

A type L DC instruction can be used to specify an extended-precision (16-byte) floating-point constant. The DC instruction is described in the Assembler Language manual.

UNALIGNED (BYTE-ORIENTED) OPERANDS

The Model 85 and Model 195 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 "IEV033 ALIGNMENT ERROR". A PARM option in the EXEC statement, ALIGN or NOALIGN, 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 Model 85 and Model 195 byte-oriented operand feature cannot be run on another machine, nor can it run successfully under the Operating System if it violates any alignment restrictions imposed by the Operating System.

Appendix A. Sample Program

The sample program included with Assembler H when it is received from IBM is described in this appendix. This program demonstrates some basic assembler language, macro, and conditional assembly features, most of which are unique to Assembler H. The circled letters in the description below refer to corresponding letters in the listing that follows the description.

Ⓐ The job control language for the assembly consists of the IBM-supplied cataloged procedure ASMHC and the statements needed to use the procedure and supply input to the assembler. Note that three of the default PARM options are overridden in the EXEC statement that calls the procedure.

By using the BATCH (multiple assembly) option, this sample program, the sample program in Appendix B, and the listings in Figure 8 and Figure 10 were assembled with one set of JCL cards. Object modules were not punched for any of the assemblies because the NODECK option is specified. The character string specified in the SYSPARM option is available to each assembly. The character string is displayed in this program by using the system variable symbol &SYSPARM (statement 144).

Ⓑ The External Symbol Dictionary shows a named common statement. The named common section is defined in statement 158.

Ⓒ Statement 10: Save the current status of the PRINT statement (ON,NODATA,GEN).

Statement 11: Leave ON in effect, modify the other two options to DATA,NOGEN.

Statement 12: Macro call; note that the expansion (statement 10) is not printed.

Statement 14: All 28 bytes of data are displayed to the two-operand DC.

Statement 15: Restore prior status of PRINT.

Statements 17 and 18: The generated output of the macro WTO is shown and only the first 8 bytes of data are displayed.

Ⓓ Statements 14 and 18: Multiple constants are allowed in hexadecimal and binary DC operands, and neither symbol in the duplication factor has been defined yet. Definition occurs in statements 108 and 109.

Ⓔ Statements 26, 28, 136, and 155 illustrate use of the LOCTR assembler instruction. This feature allows one to break control sections down into sub-control sections. It may be used in CSECT, DSECT, and COM. LOCTR has many of the features of a control section; for example, all of the first LOCTR in a section is assigned space, then the second, and so on. The name of the control section automatically names the first LOCTR section. Thus LOCTR A is begun, or resumed, at statements 2, 28, and 155. Note that the location counter value shown each time is the resumed value of the LOCTR. On the other hand, various LOCTR sections within a control section have common addressing as far as USING statements are concerned, subject to the computed displacement falling within 0 through 4095. In the sample, CONSTANT is in LOCTR DEECEES but the instruction referencing it (statement 25) has no addressing problems.

Ⓕ Three-operand EQU. Here, we are assigning: (a) the value of B5 (not yet defined) to A8, (b) the length attribute of A5 to A8, and (c) the type attribute of A7 to A8. If no operand is present in an EQU statement, the type attribute is U and the length attribute is that of the first term in the operand expression. Symbols present in the label and/cr operand field must be previously defined. Note that it is not pcssible to express the type attribute of A7 directly in the EQU statement. The EQU statement at 38 could have been written

```
A8 EQU B5,2,C'L'

A8 EQU B5,X'2',X'D3'
```

Ⓖ Set symbols &LA8 and &TA8 have not been declared in a LCL or GBL statement prior to their use here. Therefore, they are defaulted to local variable symbols, as follows: &LA8 is a LCLA SET symbol because it appears in the name field of a SETA; &TA8 is a LCLC SET symbol because it is first used in a SETC.

Ⓗ MNOTE may appear in open code. As such, they have all properties of MNOTEs inside macros, including substitution.

Ⓘ A SETC expression may have a duplication factor. The SETA expression must be enclosed in parentheses and immediately precede the character string, the substring notation, or the type attribute reference.

Ⓙ Statements 57-60 illustrate 4-byte self-defining values and unary + and -. The value of X will appear later in a literal address constant (see statement 162).

Ⓚ The programmer macro DEMO is defined well after the start of the assembly. Macros can be defined at any point and, having been defined and/or expanded, can be redefined. Note that the parameters on the prototype are a mixture of keywords and positional operands. &SYSLIST may be used. The positional parameters are identified and numbered 1, 2, 3 from left to right; keywcrds are skipped cver.

Ⓛ Statement 70 illustrates the extended SET feature (as well as implicit declaration of &LOC(1) as a LCLC). Both &LOC(1) and &LOC(2) are assigned values. One SETA, SETB, or SETC statement can then do the work of many.

Ⓜ Statement 72 is a model statement with a symbolic parameter in its operation field. This statement will be edited as if it is a macro call; at this time, each operand will be denoted as positional or keyword. At macro call time, it will not be possible to reverse this decision. Even though treated as a macro, it is still expanded as a machine or assembler operation.

Ⓝ Statement 74 illustrates the computed AGO statement. Control will pass to .MNOTE1 if &KEY2 is 1, to .MNOTE2 if &KEY2 is 2, to .MNOTE3 if &KEY2 is 3 or will fall through to the model statement at 75 otherwise.

Ⓞ Statement 77 illustrates the extended AIF facility. This statement is written in the alternate format. The logical expressions are examined from left to right. Control passes to the sequence symbol corresponding to the first true expression encountered, else falls through to the next model statement.

Ⓟ Statement 87 contains a subscripted created SET symbol in the name field. Exclusive of the subscript notation, these SET symbols have the form &(e) where e is an expression made up of character strings and/or variable symbols. When such a symbol is encountered at expansion time, the assembler evaluates e and attempts to use &(value) in place of &(e). Looking ahead, we see that DEMO is used as a macro instruction in statement 97 and &KEY1=C. Thus, the 'e' in this case is X&KEY1 which has the value XC. Finally, the macro-generator will use &XC(2) as the name field of this model statement. In statement 108, note that &XC(2) equals TRANSYLVANIA (statement 96). Finally, in the sequence field of statement 108, we see that this statement is a level 01 expansion of a programmer macro and the corresponding model statement is statement number 87.

Created SET symbols may be used wherever regular SET symbols are used in declarations, name fields or operands of SET statements, in model statements, etc. Likewise, they are subject to all the restrictions of regular SET symbols. In the programmer macro DEMO, it would not have been valid to have the statement GBLC &(X&KEY1)(1) because, in statement 71, &XA, &XB, and &XC are declared as global variable symbols and &(X&KEY1)(2) becomes &XC(2) unless, of course, &KEY1 was assigned something other than the value A, B, or C in the macro instruction DEMO, statement 97. In that case, we would need a global declaration statement if we wanted &(X&KEY1) to be a global SET symbol. Because global declarations are processed at generation time and then only if the statement is encountered, we would insert the following statements between, say, statements 71 and 72.

```
         AIF      ('&KEY1' EQ 'A' OR '&KEY1' EQ 'B' OR '&KEY1' EQ 'C').
         GBLC &(X&KEY1)(1)
.SKIP    ANOP
```

As the macro is defined, &(X&KEY1) will be a global SETC if &KEY1 is A, B, or C; otherwise it will be a LCLC or, possibly, a LCLA. In the macro, if &(X&KEY1) becomes a local, it will have a null or zero value.

Ⓠ In statements 93 and 94, note that &XA is declared as a subscripted global SETC variable with a maximum subscript of 1 and, in the next statement (an extended SET statement), we store something into &XA(2). There is no contradiction here. The statement GBLC &XA(1) marks &XA as a subscripted global SETC symbol. Any decimal self-defined number (1 through 2147483647) can be used. Furthermore, only a nominal amount of space is set aside in the global dictionary -- this space is open-ended and will be increased on demand and only on demand.

Ⓡ Statement 97 is the macro instruction DEMO. Note that &P1 has the value WRITE. Therefore, the model statement at statement 72 becomes an inner macro, WRITE, producing the code at statements 98-103. The sequence field of these statements contains 03-IHBRD, indicating that they are generated by a level 03 macro (DEMO is 01, WRITE is 02) named IHBRDWRS. It is an inner macro called by WRITE.

Ⓢ Statements 108 and 109 contain some ordinary symbols longer than eight characters. The limit for ordinary symbols, operation codes (for programmer and library macros and op codes defined through OPSYN), variable symbols, and sequence symbols is sixty-three characters (including the & and . in the latter two instances, respectively). Most long symbols will probably be nearer to eight than sixty-three characters in length. Extremely long symbols are simply too difficult to write, especially if the symbol is used frequently. The requirement that the operation field be present in the first statement of a continued statement is still in effect. Furthermore, names of START, CSECT, EXTRN, WXTRN, ENTRY, etc. symbols are still restricted to eight characters.

Ⓣ Library macros may be inserted into the source stream as programmer macros by use of a COPY statement. The result (statements 118-126) is essentially a programmer macro definition. When a library macro is brought in and expanded by use of a macro instruction, the assembler (1) looks the macro up by its member-name and (2) verifies that this same name is used in the operation field of the prototype statement. Therefore, for example, DCB has to be catalogued as DCB. However, as COPY code, the member name bears no relationship to any of the statements in the member. Thus, several variations of a given macro could be stored as a library under separate names, then copied in at various places in a single assembly as needed. (Assembler H allows you to define and redefine a macro any number of times).

Ⓤ In statement 129, MARK is made a synonym for NOTE. To identify NOTE as a macro, it has to be used as either a system macro call (that is, from a macro library) or a programmer macro definition prior to its use in the operand field of an OPSYN statement. The COPY code at 118-126 is a programmer macro definition. The macro instruction at statement 130 is MARK. We can use MARK and NOTE interchangeably. If desired, we could remove NOTE as a macro definition in the following way:

```
MARK          OPSYN     NOTE
NOTE          OPSYN
```

We could then refer to the macro only as MARK.

Ⓥ Statement 144 demonstrates &SYSTIME, &SYSDATE and &SYSPARM. The values for the first two are the same as we use on the heading line. The value for &SYSPARM is the value passed in the PARM field of the EXEC statement or the default value assigned to &SYSPARM when Assembler H is installed.

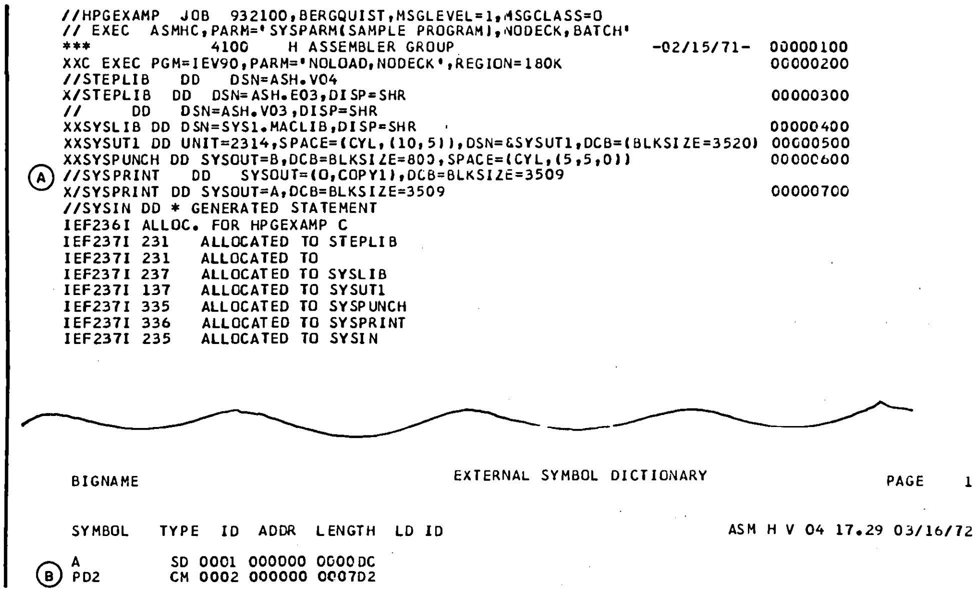
Ⓦ System variable symbols &SYSLOC and &SYSECT are displayed. The sequence field indicates that the model statements are statements 148 and 149.

Ⓧ Illustration of named COMMON. You can establish addressability for a named COMMON section with :

```
USING         section-name, register
```

You can address data in a blank COMMON section by labelling a statement <u>after</u> the COMMON statement and using relative addressing.

Ⓨ If there are literals outstanding when the END statement is encountered, they are assigned to the LOCTR currently in effect for the first control section in the assembly. This may or may not put the literals at the end of the first control section. In this sample assembly, the first control section, A, has two LOCTRs, A and DEECEES. Because A is active (at statement 155), the literals are assembled there. You always have the ability to control placement of literal pools by means of the LTORG statement. Note that X'FFFFFFE8' is used for the contents of A(A5,X), statement 162. The symbol X was assigned the value (4*-6) by an EQU in statement 60.



```
   //HPGEXAMP  JOB  932100,BERGQUIST,MSGLEVEL=1,MSGCLASS=O
   // EXEC  ASMHC,PARM='SYSPARM(SAMPLE PROGRAM),NODECK,BATCH'
   ***             4100    H ASSEMBLER GROUP                      -02/15/71-  00000100
   XXC EXEC PGM=IEV90,PARM='NOLOAD,NODECK',REGION=180K                        00000200
   //STEPLIB   DD   DSN=ASH.V04
   X/STEPLIB  DD  DSN=ASH.E03,DISP=SHR                                        00000300
   //      DD   DSN=ASH.V03,DISP=SHR
   XXSYSLIB DD DSN=SYS1.MACLIB,DISP=SHR                                       00000400
   XXSYSUT1 DD UNIT=2314,SPACE=(CYL,(10,5)),DSN=&SYSUT1,DCB=(BLKSIZE=3520)   00000500
   XXSYSPUNCH DD SYSOUT=B,DCB=BLKSIZE=800,SPACE=(CYL,(5,5,0))                 00000600
Ⓐ  //SYSPRINT   DD   SYSOUT=(O,COPY1),DCB=BLKSIZE=3509
   X/SYSPRINT DD SYSOUT=A,DCB=BLKSIZE=3509                                    00000700
   //SYSIN DD * GENERATED STATEMENT
   IEF236I ALLOC. FOR HPGEXAMP C
   IEF237I 231   ALLOCATED TO STEPLIB
   IEF237I 231   ALLOCATED TO
   IEF237I 237   ALLOCATED TO SYSLIB
   IEF237I 137   ALLOCATED TO SYSUT1
   IEF237I 335   ALLOCATED TO SYSPUNCH
   IEF237I 336   ALLOCATED TO SYSPRINT
   IEF237I 235   ALLOCATED TO SYSIN


   BIGNAME                                  EXTERNAL SYMBOL DICTIONARY                        PAGE    1

   SYMBOL   TYPE  ID  ADDR  LENGTH  LD ID                                ASM H V 04 17.29 03/16/72

   A          SD 0001 000000 0000DC
Ⓑ  PD2        CM 0002 000000 0007D2
```

```
BIGNAME  SAMPLE PROGRAM.  1ST TITLE STATEMENT HAS NO NAME, 2ND ONE DOES                                       PAGE    2

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  ASM H V 04 17.29 03/16/72

000000                                2 A         CSECT                                                        00020000
                            00000     3           USING *,8                                                    00030000

                                      5 ********************************************************************** 00050000
                                      6 *                  PUSH  AND POP  STATEMENTS                         * 00060000
                                  Ⓒ   7 * PUSH DOWN THE PRINT STATEMENT, REPLACE IT, RETRIEVE ORIGINAL       * 00070000
                                      8 ********************************************************************** 00080000

                                     10           PUSH  PRINT     SAVE DEFAULT SETTING '  PRINT ON,NODATA,GEN'  00100000
                                     11           PRINT NOGEN,DATA                                             00110000
                                     12           WTO   MF=(E,(1))                       EXPANSION NOT SHOWN    00120000
C00002 01230ABC0102030A           Ⓓ  14  DC X'123,ABC',(REALLYLONGSYMBOL-TRANSYLVANIA)B'1,10,11,1010,1011,1100' 00130000
C0000A 0B0C0102030A0B0C
000012 0102030A0B0C0102
00001A 030A0B0C
                                     15           POP   PRINT                      RESTORE DEFAULT PRINT SETTING   00140000
                                     16           WTO   MF=(E,(1))                       EXPANSION SHOWN        00150000
00001E 0A23                          17+          SVC   35                                   ISSUE SVC      01-WTO
000020 01230ABC0102030A              18  DC X'123,ABC',(REALLYLONGSYMBOL-TRANSYLVANIA)B'1,10,11,1010,1011,1100' 00160000

                                     20 ********************************************************************** 00180000
                                  Ⓔ  21 *                           LOCTR  INSTRUCTION                       * 00190000
                                     22 * LOCTR ALLOWS 'REMOTE' ASSEMBLY OF CONSTANT                         * 00200000
                                     23 ********************************************************************** 00210000

00003C 5850 8098            00098    25           L     5,CONSTANT                                             00230000
000098                               26 DEECEES  LOCTR                                                         00240000
000098 00000005                      27 CONSTANT DC     F'5'      CONSTANT CODED HERE, ASSEMBLED BEHIND LOCTR A  00250000
000040                               28 A        LOCTR                     RETURN TO 1ST LOCTR IN CSECT A      00260000

                                     30 ********************************************************************** 00280000
                                  Ⓕ  31 *  3 OPERAND EQUATE WITH FORWARD REFERENCE IN 1ST OPERAND            * 00290000
                                     32 ********************************************************************** 00300000

000040 1812                          34 A5        LR    1,2               L'A5 = 2, T'A5 = I                   00320000
                                     35           PRINT DATA                                                   00330000
000042 000000000000
C00048 413243F6A8885A30              36 A7 DC L'3.14159265358979323846264338327950288841972' L'A7 = 16,T'A7 = L  00340000
00C050 338D313198A2E037
                                     37 &TYPE     SETC  T'A7                                                   00350000
                                     38 A8        EQU   B5,L'A5,C'&TYPE'                                       00360000
                            000A0      +A8        EQU   B5,L'A5,C'L'                                           00360000
```

```
BIGNAME  SAMPLE PROGRAM.  1ST TITLE STATEMENT HAS NO NAME, 2ND GNE DOES                                       PAGE    3

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    ASM H V 04 17.29 03/16/72
                                    40 **********************************************************************  00380000
                                    41 * IMPLICIT DECLARATION OF LOCALS &A, &C -- USE OF SETC DUP FACTOR TO *  00390000
                                    42 *   PRODUCE SETC STRING LONGER THAN 8, MNOTE IN OPEN CODE            *  00400000
                                    43 **********************************************************************  00410000

                                    45 &LA8     SETA  L'A8                                                     00430000
                                 G  46 &TA8     SETC  T'A8                                                     00440000
                                    47          MNOTE *,'LENGTH OF A8 = &LA8, TYPE OF A8 = &TA8'               00450000
                                 H    +*,LENGTH OF A8 = 2, TYPE OF A8 = L                                      00450000

                                    49 &A       SETA  2                                                        00470000
                                 I  50 &C       SETC  (&A+3)'STRING,'                                          00480000
                                    51          MNOTE *,'&&C HAS VALUE = &C'                                   00490000
                                      +*,&C HAS VALUE = STRING,STRING,STRING,STRING,STRING,                    00490000

                                    53 **********************************************************************  00510000
                                 J  54 * EXAMPLES OF 4 BYTE SELF-DEFINED TERMS, UNARY + AND -               *  00520000
                                    55 **********************************************************************  00530000

000058 7FFFFFFFC1C2C3C4             57          DC    A(2147483647,C'ABCD',X'FFFFFFFF')                        00550000
000060 FFFFFFFF
000064 181D                         58          LR    -1+2,16+-3                                               00560000

                          FFFFFFE8  60 X        EQU   4*-6                                                     00580000
```

```
BIGNAME  INSERT PROGRAMMER MACRO IN SOURCE STREAM NOW                                                         PAGE     4

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                     ASM H V 04 17.29 03/16/72

                                     62 ********************************************************************** 00600000
                                     63 *  MIXED KEYWORDS AND POSITIONAL PARAMETERS, EXTENDED AGO AND AIF    * 00610000
                                     64 *     STATEMENTS, DECLARATION AND USE OF SUBSCRIPTED SET SYMBOLS,   * 00620000
                                     65 *       USE OF CREATED SET SYMBOLS, EXTENDED SET STATEMENTS          * 00630000
                                     66 ********************************************************************** 00640000

                                  Ⓚ  68          MACRO                                                         00660000
                                     69          DEMO  &P1,&KEY1=A,&P2,&KEY2=1,&P3,&KEY3=3,&P4                 00670000
                                  Ⓛ  70 &LOC(1)  SETC  '2','3'        &LOC IS DIMENSIONED LCLC BY DEFAULT      00680000
                                     71          GBLC  &XA(5),&XB(20),&XC(1)                                   00690000
                                  Ⓜ  72          &P1   &SYSLIST(4),&SYSLIST(5),&SYSLIST(6),MF=E                00700000
                                     73 &N       SETA  1                                                       00710000
                                  Ⓝ  74          AGO   (&KEY2).MNOTE1,.MNOTE2,.MNOTE3                          00720000
                                     75 &N       SETA  2                                                       00730000
                                     76          MNOTE *,'&&KEY2 NOT 1,2, OR 3---USE &&KEY3 IN PLACE OF IT'    00740000
                                  Ⓞ  77          AIF   (&KEY3 EQ 1).MNOTE1,                                   X00750000
                                                       (&KEY3 EQ 2).MNOTE2,(&KEY3 EQ 3).MNOTE3                 00760000
                                     78          MNOTE *,'BOTH &&KEY2 AND &&KEY3 FAIL TO QUALIFY'              00770000
                                     79          AGO   .COMMON                                                 00780000
                                     80 .MNOTE1  MNOTE *,'&&KEY&LOC(&N) = 1'                                   00790000
                                     81          AGO   .COMMON                                                 00800000
                                     82 .MNOTE2  MNOTE *,'&&KEY&LOC(&N) = 2'                                   00810000
                                     83          AGO   .COMMON                                                 00820000
                                     84 .MNOTE3  MNOTE *,'&&KEY&LOC(&N) = 3'                                   00830000
                                     85 .COMMON  L     5,8(,10)       NOTE THAT OPCODES, OPERANDS & COMMENTS   00840000
                                     86 &XB(2) SR 9,10                  ON MODEL STATEMENTS                    00850000
                                  Ⓟ  87 &(X&KEY1)(2) LM 12,13,=A(A5,X)    ARE KEPT IN PLACE UNLESS DISPLACED   00860000
                                     88 &P2 ST 7,&P3                        AS A RESULT OF SUBSTITUTION        00870000
                                     89          MEND                                                          00880000

                                     91 *****          DEMO  MACRO  INSTRUCTION (CALL)                         00900000
                                  Ⓠ
                                     93          GBLC  &XA(1),&XB(2),&XC(3)                                    00920000
                                     94 &XA(1)   SETC  'A','MISSISSIPPI'                                       00930000
                                     95 &XB(1)   SETC  'B','SUSQUEHANNA'                                       00940000
                                  Ⓡ  96 &XC(1)   SETC  'C','TRANSYLVANIA'                                      00950000
                                     97          DEMO  KEY3=2,WRITE,REALLYLONGSYMBOL,                         M00960000
                                                       A8+8*(B5-CONSTANT-7)(3),KEY1=C,(6),SF,                 N00970000
                                                       (8),KEY2=7                                              00980000
000066 1816                          98+         LR    1,6                               LOAD DECB ADDRESS     03-IHBRD
000068 9220 1005      00005          99+         MVI   5(1),X'20'             SET TYPE FIELD                   03-IHBRD
00006C 5081 0008            00008   100+         ST    8,8(1,0)                          STORE DCB ADDRESS     03-IHBRD
000070 58F1 0008            00008   101+         L     15,8(1,0)           LOAD DCB ADDRESS                    03-IHBRD
000074 58F0 F030            00030   102+         L     15,48(0,15)                       LOAD RDWR ROUTINE ADDR 03-IHBRD
000078 05EF                         103+         BALR  14,15                             LINK TO RDWR ROUTINE  03-IHBRD
                                    104+*,&KEY2 NOT 1,2, OR 3---USE &KEY3 IN PLACE OF IT                       01-00076
                                    105+*,&KEY3 = 2                                                            01-00082
00007A 5850 A008            00008   106+         L     5,8(,10)       NOTE THAT OPCODES, OPERANDS & COMMENTS   01-00085
00007E 1B9A                         107+SUSQUEHANNA SR 9,10             ON MODEL STATEMENTS                    01-00086
000080 98CD 8090            00090Ⓢ  108+TRANSYLVANIA LM 12,13,=A(A5,X)    ARE KEPT IN PLACE UNLESS DISPLACED   01-00087
000084 5073 80A8            000A8   109+REALLYLONGSYMBOL ST 7,A8+8*(B5-CONSTANT-7)(3)                         X01-00088
                                       +                            AS A RESULT OF SUBSTITUTION
```

```
BIGNAME  INSERT PROGRAMMER MACRO IN SOURCE STREAM NOW                                                   PAGE    5

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                               ASM H V 04 17.29 03/16/72

                                    111 *********************************************************************** 01000000
                                    112 *  COPY 'NOTE' MACRO IN FROM MACLIB, RENAME IT 'MARK', CALL IT UNDER *  01010000
                                    113 *     ITS ALIAS -- IN EXPANSION OF MARK, NOTICE REFERENCE BACK TO     *  01020000
                                    114 *       DEFINITION STATEMENTS IN 'COLUMNS' 76-80 OF EXPANSION         *  01030000
                                    115 *********************************************************************** 01040000

                                 Ⓣ  117          COPY  NOTE                                                    01060000
                                    118          MACRO                                                         00020000
                                    119 &NAME    NOTE  &DCB,&DUMMY=                                            00040017
                                    120          AIF   ('&DCB' EQ '').ERR                                      00060000
                                    121 &NAME    IHBINNRA &DCB                                                 00080000
                                    122          L     15,84(0,1)                      LOAD NOTE  RTN ADDRESS C0100000
                                    123          BALR  14,15                           LINK TO NOTE  ROUTINE  00120000
                                    124          MEXIT                                                         00140000
                                    125 .ERR     IHBERMAC 6                                                    00160000
                                    126          MEND                                                          00180000


                                 Ⓤ  129 MARK     OPSYN NOTE      COMMENTS OF GENERATED STATEMENTS OCCUPY SAME  01090000
                                    130          MARK  (6)          'COLUMNS' AS THOSE IN MODEL STATEMENTS     01100000
000088 1816                         131+         LR    1,6                             LOAD PARAMETER REG 1   02-IHBIN
00008A 58F0 1054             00054  132+         L     15,84(0,1)                      LOAD NOTE  RTN ADDRESS 01-00122
00008E 05EF                         133+         BALR  14,15                           LINK TO NOTE  ROUTINE  01-00123

                                    135 *********************************************************************** 01120000
00009C                              136 DEECEES  LOCTR        SWITCH TO ALTERNATE LOCATION COUNTER             01130000
00009C 00000000
C000A0 0B0000A000000050             137 B5       CCW   X'0B',B5,0,80                                           01140000

                                    139 *********************************************************************** 01160000
                                    140 *  DISPLAY OF &SYSTIME, &SYSDATE, &SYSPARM AND &SYSLOC                 *  01170000
                                    141 *********************************************************************** 01180000

                                 Ⓥ  143          PRINT NODATA                                                  01200000
                                    144   DC  C'TIME = &SYSTIME, DATE = &SYSDATE, PARM = &SYSPARM'              01210000
0000A8 E3C9D4C5407E40F1               +  DC  C'TIME = 17.29, DATE = 03/16/72, PARM = SAMPLE PROGRAM'        01210000

                                    146          MACRO                                                         01230000
                                    147          LOCATE                                                        01240000
                                    148 &SYSECT  CSECT      DISPLAY OF CURRENT CONTROL SECTION                 01250000
                                    149 &SYSLOC  LOCTR           AND LOCATION COUNTER                          01260000
                                    150          MEND                                                          01270000
                                 Ⓦ
                                    152          LOCATE                                                        01290000
0000DC                              153+A        CSECT      DISPLAY OF CURRENT CONTROL SECTION                 01-00148
0000DC                              154+DEECEES  LOCTR           AND LOCATION COUNTER                          01-00149
000C90                              155 A        LOCTR                                                         01300000

                                 Ⓧ  157 *********************************************************************** 01320000
C00000                              158 PD2      COM                          NAMED COMMON THROWN IN FOR GOOD MEASURE  01330000
C00000                              159          DS    500F                                                    01340000
0007D0 1867                         160          LR    6,7                                                     01350000
                                 Ⓨ  161          END                                                           01360000
000090 00000040FFFFFFE8             162                =A(A5,X)
```

```
BIGNAME                                           RELOCATION DICTIONARY                                      PAGE    6

  POS.ID     REL.ID    FLAGS     ADDRESS                                                     ASM H V 04 17.29 03/16/72

   0001       0001      0C       000C90
   0001       0001      08       000CA1
```

```
BIGNAME                                      CROSS REFERENCE                                               PAGE    7

SYMBOL     LEN     VALUE    DEFN     REFERENCES                                             ASM H V 04 17.29 03/16/72

A          00001 00000000   0002     0028   0153   0155
A5         00002   000040   0034     0038   0162
A7         00016   000048   0036
A8         00002 000000A0   0038     0109
B5         00008   0000A0   0137     0038   0109   0137
CONSTANT   00004   000098   0027     0025   0109
DEECEES    00001 00000098   0026     0136   0154
PD2        00001 00000000   0158
REALLYLONGSYMBOL
           00004   000084   0109     0014   0018
SUSQUEHANNA
           00002   00007E   0107
TRANSYLVANIA
           00004   000080   0108     0014   0018
X          00001 FFFFFFE8   0060     0162
=A(A5,X)   00004   000090   0162     0108
```

```
BIGNAME                           DIAGNOSTIC CROSS REFERENCE AND ASSEMBLER SUMMARY                      PAGE    8

                                                                                        ASM H V 04 17.29 03/16/72
    NO STATEMENTS FLAGGED IN THIS ASSEMBLY

  OVERRIDING PARAMETERS-  SYSPARM(SAMPLE PROGRAM),NODECK,BATCH
  OPTIONS FOR THIS ASSEMBLY
   NODECK, NOOBJECT, LIST, XREF(FULL), NORENT, NOTEST, BATCH, ALIGN, ESD, RLD, LINECOUNT(55), FLAG(0), SYSPARM(SAMPLE P
   ROGRAM)
  NO OVERRIDING DD NAMES

    136 CARDS FROM SYSIN       524 CARDS FROM SYSLIB
    198 LINES OUTPUT             0 CARDS OUTPUT
```

Appendix B. Sample Macro Trace and Dump (MHELP)

The Macro Trace and Dump (MHELP) facility is a useful means of debugging macro definitions. MHELP can be used anywhere in the source program or in macro definitions. MHELP is processed during macro generation. It is completely dynamic; you can branch around the MHELP statements by using AIF or AGO statements. Therefore, its use can be controlled by symbolic parameters and SET symbols.

The following sample program illustrates the five primary functions of MHELP. Since most of the information produced is unrelated to statement numbers, the dumps and traces in the listing are marked with circled numbers. Most dumps refer to statement numbers. If you request MHELP information about a library macro definition, the first five characters of the macro name will appear in place of the statement number. To get the statement numbers, you should use COPY to copy the library definition into the source program prior to the macro call.

MACRO CALL TRACE (MHELP 1)

Item 1A illustrates an outer macro call, 1B an inner one. In each case, the amount of information given is brief. This trace is given after successful entry into the macro; no dump is given if error conditions prevent an entry.

MACRO ENTRY DUMP (MHELP 16)

This provides values of system variable symbols and symbolic parameters at the time the macro is called. The following numbering system is used:

Number	Item
000	&SYSNDX
001	&SYSECT
002	&SYSLOC
003	&SYSTIME
004	&SYSDATE
005	&SYSPARM
006	NAME FIELD ON MACRO INSTRUCTION

If there are NKW keyword parameters, they follow in order of appearance on the prototype statement.

007	1st keyword value
008	2nd keyword value
.	.
.	.
.	.
006+NKW	NKWth keyword value

If there are NPP positional parameters, they follow in order of appearance in the macro instruction.

007+NKW	1st positional parameter values
008+NKW	2nd positional parameter values
.	
.	
.	
006+NKW+NPP	NPPth positional parameter values

For example, item 16A has one keyword parameter (&OFFSET) and one positional parameter. The value of the keyword parameter appears opposite 110006, the positional parameter, opposite 110007. In both the prototype (statement 3) and the macro instruction (statement 54), the positional parameter appears in the first operand field, the keyword in the second. A length appears between the NUM and VALUE fields. A length of NUL indicates the corresponding item is empty.

Item 16B illustrates an inner call containing zero keywords, and two positional parameters.

MACRO AIF DUMP (MHELP 4)

Items 4A, 4B, 4C, ... are examples of these dumps. Each such dump includes a complete set of unsubscripted SET symbols with values. This list covers all unsubscripted variable symbols which appear in the name field of a SET statement in the macro definition. Values of elements of dimensioned SET symbols are not displayed.

MACRO BRANCH TRACE (MHELP 2)

This provides a one-line trace for each AGO and true AIF branch within a programmer macro. In any such branch, the "branched from" statement number, the "branched to" statement number and the macro name are included. Note, in example 2A, the "branched to" statement number indicated is not that of the ANOP statement bearing the target sequence symbol but that of the statement following it. The branch trace facility is suspended when library macros are expanded and MHELP 2 is in effect. To obtain a macro branch trace for such a macro, one would have to insert a COPY "macro-name" statement in the source deck at some point prior to the MHELP 2 statement of interest.

MACRO EXIT DUMP (MHELP 8)

This provides a dump of the same group of SET symbols as are included in the Macro AIF dump when a MEXIT or MEND is encountered.

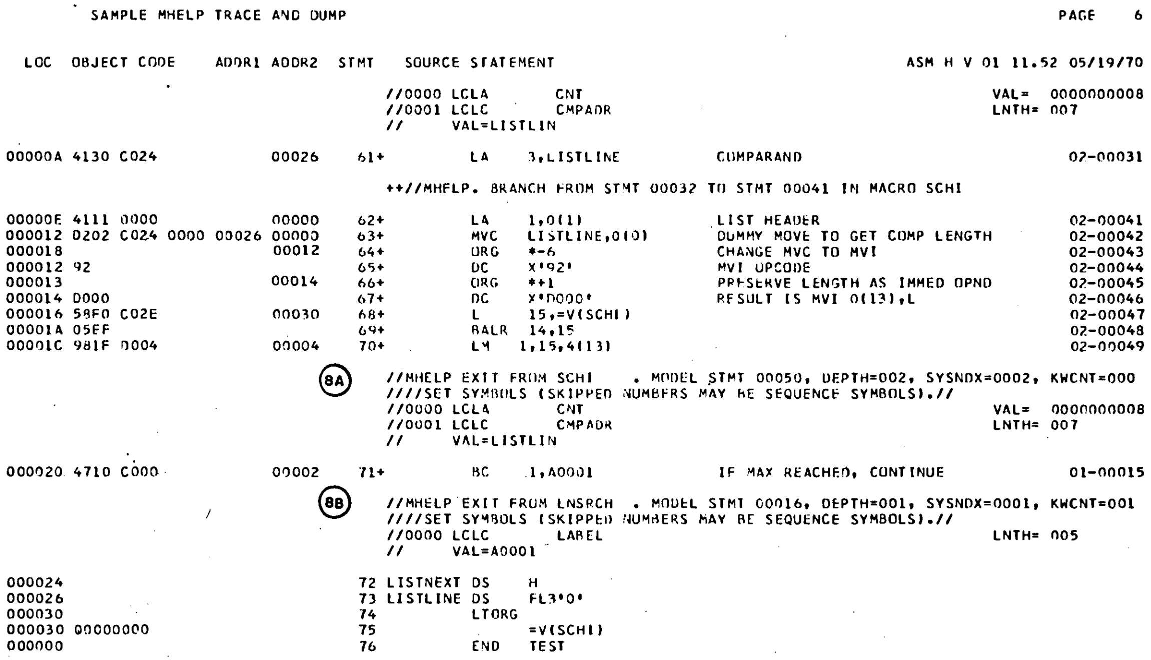
Note that local and/or global variable symbols are not displayed at any point unless they appear in the current macro explicitly as SET symbols.

```
      SAMPLE MHELP TRACE AND DUMP                                                                  PAGE    2

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                              ASM H V 01 11.52 05/19/70
                                      2 *    INCLUDE MACRO DEFINITIONS TO BE TRACED IN THE SOURCE PROGRAM

000000                                4          CSECT
                                      5          COPY  LNSRCH
                                      6          MACRO
                                      7 &NAME   LNSRCH &ARG,&OFFSET=STNUM8-STCHAIN
                                      8          LCLC  &LABEL
                                      9 &LABEL   SETC  'A&SYSNDX'           GENERATE SYMBOL
                                     10          AIF   (T'&NAME EQ 'O').SKIP
                                     11 &LABEL   SETC  '&NAME'              IF MACRO CALL HAS LABEL, USE IT
                                     12 .SKIP    ANOP                       INSTEAD OF GENERATED SYMBOL
                                     13 &LABEL   LA    0,&OFFSET            LOAD REG. 0
                                     14          SCHI  &ARG,0(1)            SEARCH
                                     15          BC    1,&LABEL             IF MAX REACHED, CONTINUE
                                     16          MEND

                                     18          COPY  SCHI
                                     19          MACRO
                                     20 &NM      SCHI  &COMP,&LIST
                                     21          LCLA  &CNT
                                     22          LCLC  &CMPADR
                                     23 &CNT     SETA  1
                                     24 &NM      STM   1,15,4(13)
                                     25 .TEST    ANOP
                                     26 &CMPADR  SETC  '&CMPADR'.'&COMP'(&CNT,1)
                                     27          AIF   ('&COMP'(&CNT,1) EQ '(').LPAR
                                     28 &CNT     SETA  &CNT+1
                                     29          AIF   (&CNT LT K'&COMP).TEST
                                     30 .NOLNTH  ANOP
                                     31          LA    3,&COMP              COMPARAND
                                     32          AGO   .CONTIN
                                     33 .LPAR    AIF   ('&COMP'(&CNT+1,1) EQ ',').FINISH
                                     34 &CNT     SETA  &CNT+1
                                     35          AIF   (&CNT LT K'&COMP).LPAR
                                     36          AGO   .NOLNTH
                                     37 .FINISH  ANOP
                                     38 &CMPADR  SETC  '&CMPADR'.'&COMP'(&CNT+2,K'&COMP-&CNT)
                                     39          LA    3,&CMPADR            COMPARAND SANS LENGTH
                                     40 .CONTIN  ANOP
                                     41          LA    1,&LIST              LIST HEADER
                                     42          MVC   &COMP,0(0)           DUMMY MOVE TO GET COMP LENGTH
                                     43          ORG   *-6                  CHANGE MVC TO MVI
                                     44          DC    X'92'                MVI OPCODE
                                     45          ORG   *+1                  PRESERVE LENGTH AS IMMED OPND
                                     46          DC    X'D000'              RESULT IS MVI 0(13),L
                                     47          L     15,=V(SCHI)
                                     48          BALR  14,15
                                     49          LM   1,15,4(13)
                                     50          MEXIT
                                     51          MEND
```

```
        SAMPLE MHELP TRACE AND DUMP                                                                          PAGE    3

  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                         ASM H V 01 11.52 05/19/70

000000                                53 TEST      CSECT
000000 05C0                           54           BALR  12,0
                             00002    55           USING *,12


                                      57           MHELP B'11111'            REQUEST ALL MHELP FUNCTIONS
                                      58           LNSRCH LISTLINE,OFFSET=LISTLINE-LISTNEXT
                               (1A)        ++//MHELP. CALL TO MACRO LNSRCH  . DEPTH=001, SYSNDX=0001, STMT 00058

                               (16A)       //MHELP ENTRY TO  LNSRCH   . MODEL STMT 00000, DEPTH=001, SYSNDX=0001, KWCNT=001
                                           ////PARAMETERS (SYSNDX,SYSECT,SYSLOC,SYSTIME,SYSDATE,SYSPARM,NAME,KWS,PPS) ///
                                           //NUM  LNTH  VALUE (64 CHARS/LINE)
                                           //0000  004  0001
                                           //0001  004  TEST
                                           //0002  004  TEST
                                           //0003  005  11.52
                                           //0004  008  05/19/70
                                           //0005  014  SAMPLE*PROGRAM
                                           //0006  NUL
                                           //0007  017  LISTLINE-LISTNEXT
                                           //0008  008  LISTLINE

                               (4A)        //MHELP AIF IN    LNSRCH   . MODEL STMT 00010, DEPTH=001, SYSNDX=0001, KWCNT=001
                                           ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                           //0000 LCLC       LABEL                                            LNTH= 005
                                           //      VAL=A0001

                               (2A)        ++//MHELP. BRANCH FROM STMT 00010 TO STMT 00013 IN MACRO LNSRCH
000002 4100 0002             00002    59+A0001     LA    0,LISTLINE-LISTNEXT LOAD REG. 0                          01-00013
                               (1B)        ++//MHELP. CALL TO MACRO SCHI    . DEPTH=002, SYSNDX=0002, STMT 00014

                               (16B)       //MHELP ENTRY TO  SCHI     . MODEL STMT 00000, DEPTH=002, SYSNDX=0002, KWCNT=000
                                           ////PARAMETERS (SYSNDX,SYSECT,SYSLOC,SYSTIME,SYSDATE,SYSPARM,NAME,KWS,PPS) ///
                                           //NUM  LNTH  VALUE (64 CHARS/LINE)
                                           //0000  004  0002
                                           //0001  004  TEST
                                           //0002  004  TEST
                                           //0003  005  11.52
                                           //0004  008  05/19/70
                                           //0005  014  SAMPLE*PROGRAM
                                           //0006  NUL
                                           //0007  008  LISTLINE
                                           //0008  004  0(1)

000006 901F 0004             00004    60+          STM   1,15,4(13)                                               02-00024

                               (4B)        //MHELP AIF IN    SCHI     . MODEL STMT 00027, DEPTH=002, SYSNDX=0002, KWCNT=000
```

```
         SAMPLE MHELP TRACE AND DUMP                                                                   PAGE    4

LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  ASM H V 01 11.52 05/19/70
                                       ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                       //0000 LCLA        CNT                                         VAL=  0000000001
                                       //0001 LCLC        CMPADR                                      LNTH= 001
                                       //      VAL=L

                                 (4C)  //MHELP AIF IN     SCHI     . MODEL STMT 00029, DEPTH=002, SYSNDX=0002, KWCNT=000
                                       ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                       //0000 LCLA        CNT                                         VAL=  0000000002
                                       //0001 LCLC        CMPADR                                      LNTH= 001
                                       //      VAL=L

                                 (2B)  ++//MHELP. BRANCH FROM STMT 00029 TO STMT 00026 IN MACRO SCHI

                                 (4D)  //MHELP AIF IN     SCHI     . MODEL STMT 00027, DEPTH=002, SYSNDX=0002, KWCNT=000
                                       ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                       //0000 LCLA        CNT                                         VAL=  0000000002
                                       //0001 LCLC        CMPADR                                      LNTH= 002
                                       //      VAL=LI

                                 (4E)  //MHELP AIF IN     SCHI     . MODEL STMT 00029, DEPTH=002, SYSNDX=0002, KWCNT=000
                                       ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                       //0000 LCLA        CNT                                         VAL=  0000000003
                                       //0001 LCLC        CMPADR                                      LNTH= 002
                                       //      VAL=LI

                                 (2C)  ++//MHELP. BRANCH FROM STMT 00029 TO STMT 00026 IN MACRO SCHI

                                       //MHELP AIF IN     SCHI     . MODEL STMT 00027, DEPTH=002, SYSNDX=0002, KWCNT=000
                                       ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                       //0000 LCLA        CNT                                         VAL=  0000000003
                                       //0001 LCLC        CMPADR                                      LNTH= 003
                                       //      VAL=LIS

                                       //MHELP AIF IN     SCHI     . MODEL STMT 00029, DEPTH=002, SYSNDX=0002, KWCNT=000
                                       ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                       //0000 LCLA        CNT                                         VAL=  0000000004
                                       //0001 LCLC        CMPADR                                      LNTH= 003
                                       //      VAL=LIS

                                       ++//MHELP. BRANCH FROM STMT 00029 TO STMT 00026 IN MACRO SCHI

                                       //MHELP AIF IN     SCHI     . MODEL STMT 00027, DEPTH=002, SYSNDX=0002, KWCNT=000
                                       ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                       //0000 LCLA        CNT                                         VAL=  0000000004
                                       //0001 LCLC        CMPADR                                      LNTH= 004
                                       //      VAL=LIST
```

```
     SAMPLE MHELP TRACE AND DUMP                                                                          PAGE    5

LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                     ASM H V 01 11.52 05/19/70


                                         //MHELP AIF IN     SCHI    . MODEL STMT 00029, DEPTH=002, SYSNDX=0002, KWCNT=000
                                         ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                         //0000 LCLA        CNT                                         VAL=  0000000005
                                         //0001 LCLC        CMPADR                                      LNTH= 004
                                         //      VAL=LIST

                                        ++//MHELP. BRANCH FROM STMT 00029 TO STMT 00026 IN MACRO SCHI

                                         //MHELP AIF IN     SCHI    . MODEL STMT 00027, DEPTH=002, SYSNDX=0002, KWCNT=000
                                         ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                         //0000 LCLA        CNT                                         VAL=  0000000005
                                         //0001 LCLC        CMPADR                                      LNTH= 005
                                         //      VAL=LISTL

                                         //MHELP AIF IN     SCHI    . MODEL STMT 00029, DEPTH=002, SYSNDX=0002, KWCNT=000
                                         ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                         //0000 LCLA        CNT                                         VAL=  0000000006
                                         //0001 LCLC        CMPADR                                      LNTH= 005
                                         //      VAL=LISTL

                                        ++//MHELP. BRANCH FROM STMT 00029 TO STMT 00026 IN MACRO SCHI

                                         //MHELP AIF IN     SCHI    . MODEL STMT 00027, DEPTH=002, SYSNDX=0002, KWCNT=000
                                         ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                         //0000 LCLA        CNT                                         VAL=  0000000006
                                         //0001 LCLC        CMPADR                                      LNTH= 006
                                         //      VAL=LISTLI

                                         //MHELP AIF IN     SCHI    . MODEL STMT 00029, DEPTH=002, SYSNDX=0002, KWCNT=000
                                         ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                         //0000 LCLA        CNT                                         VAL=  0000000007
                                         //0001 LCLC        CMPADR                                      LNTH= 006
                                         //      VAL=LISTLI

                                        ++//MHELP. BRANCH FROM STMT 00029 TO STMT 00026 IN MACRO SCHI

                                         //MHELP AIF IN     SCHI    . MODEL STMT 00027, DEPTH=002, SYSNDX=0002, KWCNT=000
                                         ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                         //0000 LCLA        CNT                                         VAL=  0000000007
                                         //0001 LCLC        CMPADR                                      LNTH= 007
                                         //      VAL=LISTLIN

                                         //MHELP AIF IN     SCHI    , MODEL STMT 00029, DEPTH=002, SYSNDX=0002, KWCNT=000
                                         ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
```



```
         SAMPLE MHELP TRACE AND DUMP                                                                          PAGE    6

  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                         ASM H V 01 11.52 05/19/70
                                        //0000 LCLA       CNT                                              VAL=  0000000008
                                        //0001 LCLC       CMPADR                                           LNTH= 007
                                        //      VAL=LISTLIN

00000A 4130 C024            00026    61+         LA    3,LISTLINE          COMPARAND                            02-00031

                                        ++//MHFLP. BRANCH FROM STMT 00032 TO STMT 00041 IN MACRO SCHI

00000E 4111 0000            00000    62+         LA    1,0(1)              LIST HEADER                          02-00041
000012 D202 C024 0000 00026 00000    63+         MVC   LISTLINE,0(0)       DUMMY MOVE TO GET COMP LENGTH        02-00042
000018                      00012    64+         ORG   *-6                 CHANGE MVC TO MVI                    02-00043
000012 92                            65+         DC    X'92'               MVI OPCODE                           02-00044
000013                      00014    66+         ORG   *+1                 PRESERVE LENGTH AS IMMED OPND        02-00045
000014 D000                          67+         DC    X'D000'             RESULT IS MVI 0(13),L                02-00046
000016 58F0 C02E            00030    68+         L     15,=V(SCHI)                                              02-00047
00001A 05EF                          69+         BALR  14,15                                                    02-00048
00001C 981F D004            00004    70+         LM    1,15,4(13)                                               02-00049

                                  8A    //MHELP EXIT FROM SCHI     . MODEL STMT 00050, DEPTH=002, SYSNDX=0002, KWCNT=000
                                        ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                        //0000 LCLA       CNT                                              VAL=  0000000008
                                        //0001 LCLC       CMPADR                                           LNTH= 007
                                        //      VAL=LISTLIN

000020 4710 C000            00002    71+         BC    1,A0001             IF MAX REACHED, CONTINUE             01-00015

                                  8B    //MHELP EXIT FROM LNSRCH   . MODEL STMT 00016, DEPTH=001, SYSNDX=0001, KWCNT=001
                                        ////SET SYMBOLS (SKIPPED NUMBERS MAY BE SEQUENCE SYMBOLS).//
                                        //0000 LCLC       LABEL                                            LNTH= 005
                                        //      VAL=A0001

000024                               72 LISTNEXT DS    H
000026                               73 LISTLINE DS    FL3'0'
000030                               74          LTORG
000030 00000000                      75                =V(SCHI)
000000                               76          END   TEST
```

Appendix C. Object Deck Output

ESD Card Format

The format of the ESC 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 (columns 17-64)
13-14	Blank
15-16	ESDID of first SD, XD, CM, PC, ER, or WX in variable field
17-64	Variable field. One to three 16-byte items of the following format: 8 bytes -- Name 1 byte -- ESC 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
73-80	Deck IC and/or sequence number -- The deck ID is the name from the first TITLE statement that has a non-blank name field. The name can be 1 to 8 characters long. If the name is less than 8 characters long or if there is no name, 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.)

TEXT (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 (columns 17-72)
13-14	Blank
15-16	ESDID
17-72	56-byte information field
73-80	Deck ID and/or sequence number -- The deck ID is the name from the first TITLE statement that has a non-blank name field. The name can be 1 to 8 characters long. If the name is less than 8 characters long or if there is no name, 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.)

RLD Card Format

The format of the RLD card is as follows:

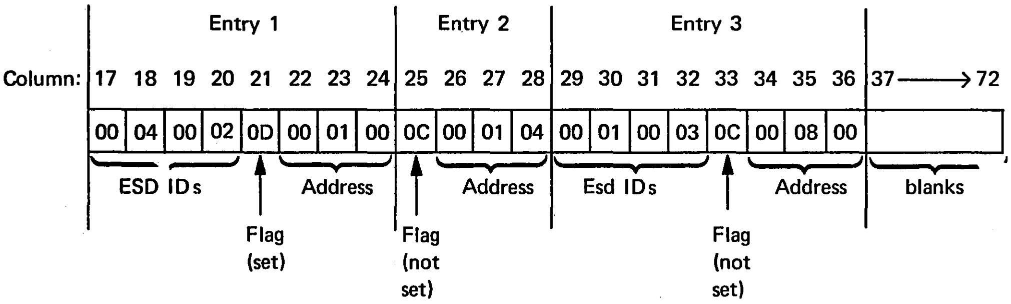
Columns	Contents
1	12-2-9 punch
2-4	RLD
5-10	Blank
11-12	Data field count -- number of bytes of information in data field (columns 17-72)
13-16	Blank
17-72	Data field:
17-18	Relocation ESDID
19-20	Position ESDID
21	Flag byte
22-24	Absolute address to be relocated
25-72	Remaining RLD entries
73-80	Deck ID and/or sequence number -- The deck ID is the name from the first TITLE statement that has a non-blank name field. The name can be 1 to 8 characters long. If the name is less than 8 characters long or if there is no name, 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.)

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 contain the following information:

	Position ESDID	Relocation ESDID	Flag	Address
Entry 1	02	04	0C	000100
Entry 2	02	04	0C	000104
Entry 3	03	01	0C	000800

then, columns 17-72 of the RLD card would be as follows:



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	

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	Number of IDR items that follow (EBCDIC1 or EBCDIC2)
34-52	Translator Identification, version and modification level (such as 0301), and date of the assembly (yyddd)
53-71	When present, they are the same format as columns 34-52
73-80	Deck ID and/or sequence number -- The deck ID is the name from the first TITLE statement that has a non-blank name field. The name can be 1 to 8 characters long. If the name is less than 8 characters long or if there is no name, 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.)

TESTRAN (SYM) Card Format

If you request it, the assembler punches out symbolic information for TESTRAN concerning the assembled program. This output appears ahead of all loader text. The format of the card images for TESTRAN output 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 (columns 17-72)
13-16	Blank
17-72	Variable field (see below)
73-80	Deck ID and/or sequence number -- The deck ID is the name from the first TITLE statement that has a non-blank name field. The name can be 1 to 8 characters long. If the name is less than 8 characters long or if there is no name, 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.)

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 12. The contents of the fields within an individual entry are as follows:

1. Organization (1 byte)

   Bit 0:
   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 = 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 1

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

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

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

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

00	= character
04	= hexadecimal
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
38	= floating point, extended.

5. Length (2 bytes for character, hexadecimal, decimal, 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, P and Z type data, and only if scale is non-zerc.

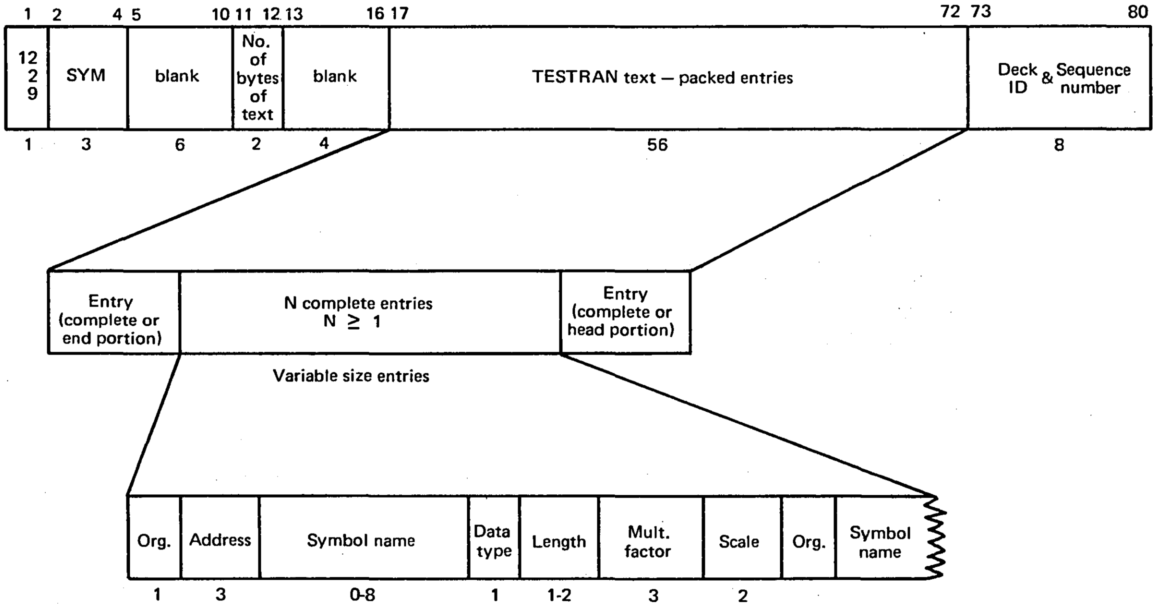


Figure 12. TESTRAN SYM Card Format

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, LINKAGE, XCTL, or ATTACH, macro instruction. 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, LINKAGE, or ATTACH, you may supply:

1) The assembler options

2) The DDnames of the data sets to be used during processing.

Name	Operation	Operand
[symbol]	CALL	IEV90,(optionlist [,ddnamelist]),VL

	LINK
	ATTACH
		EP=IEV90,
		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 from the next 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, 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 8 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	SYSLIN
2	not applicable
3	not applicable
4	SYSLIB
5	SYSIN
6	SYSPRINT
7	SYSPUNCH
8	SYSUT1

Note: An overriding DDname specified when Assembler H was added to the Operating System occupies the same place in the above list as the IBM-supplied DDname it overrides. The overriding DDname can itself be overridden during invocation. For example, if SYSWORK1 replaced SYSUT1, it occupies position 8 in the above list. SYSWORK1 can be overridden by another name during invocation.

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

The Job Control Language Reference manual provides additional description of overriding techniques.

Index

SC26-3759-1

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 H
Programmer's Guide

Order No. SC26-3759-1

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.)

SC26-3759-1

Your comments, please . . .

Your answers to the questions on the back of this form, together with your comments, will help us to produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

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)

SC26-3759-1

OS Assembler H Prog. Guide Printed in U. S. A SC26-3759-1

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)