96836000

CONTROL DATA
CORPORATION

Sheryl Mosbarger

CYBER CROSS SYSTEM VERSION 1 REFERENCE MANUAL

CONTROL DATA®
255X HOST COMMUNICATIONS PROCESSORS
CYBER 170 SYSTEMS
CYBER 70 COMPUTER SYSTEMS
MODELS 72, 73, 74
6000 COMPUTER SYSTEMS
CYBER 18 COMPUTER SYSTEMS

REVISION RECORD

REVISION	DESCRIPTION
A (12/75)	Manual released.
B (4/76)	This revision reflects NOS/BE 1.1 changes and a manual name change. See the List of Effective Pages for pages changed in this revision.
Publication No. 96836000	

Address comments concerning this manual to:
Control Data Corporation
Publications and Graphics Division
4455 Eastgate Mall
La Jolla, California 92037
or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision	SFC[†]
Cover	-	
Revision Record	B	
iii	B	
v	B	
vii	B	
1-1	B	
2-1	B	
2-2	B	
2-3	B	
2-4	A	
2-5	B	
2-6	B	
2-7	B	
2-8	B	
2-9	B	
2-10	A	
2-11	B	
2-12	B	
2-13	B	
2-14	B	
Glossary-1	B	
A-1	B	
B-1	B	
B-2	B	
B-3	B	
Comment sheet	B	
Envelope	-	
Back cover	-	

Page	Revision	SFC[†]

[†] Software Feature Change

PREFACE

This manual describes the calls and loading procedures for the CYBER Cross System. This software operates on the CONTROL DATA® CYBER 170/70/6000 computers, under control of the NOS or NOS/BE 1 operating system. The object code generated by the CDC 170/70/6000 computers is executed on the CYBER 18 computer systems and the 255x Host Communications Processors.

Information applicable to the Host Operating System can be found in the Literature Distribution Service catalog. More detailed information regarding parameters and formats can be found in the following publications.

Description	Publication Number
NOS/BE 1 Reference Manual	60493800
NOS 1 Reference Manual, Volume 1	60435400
NOS 1 Reference Manual, Volume 2	60445300
CYBER Cross System PASCAL Compiler Reference Manual	96836100
CYBER Cross System Macro Assembler Reference Manual	96836500
CYBER Cross System Micro Assembler Reference Manual	96836400
CYBER Cross System Link Editor and Library Maintenance Programs Reference Manual	60471200

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features and parameters.

CONTENTS

FIGURES

INTRODUCTION 1

The CYBER Cross System provides a means of developing software on a CYBER 170/70/6000 computer for CYBER 18 computers or a CDC 255x Host Communications Processor. Programs may be written in three languages: PASCAL, a high-level language based on ALGOL; macro assembler language; and micro assembler language.

Source programs are maintained in libraries by using the system routine UPDATE. Object programs can be maintained on mass storage libraries by using the CYBER Cross System Library Maintenance program (MPLIB). The Link Editor is provided to build load modules that are loaded and executed on the CYBER 18 computers or CDC 255x processor.

CYBER CROSS SYSTEM COMPONENTS 2

The CYBER Cross System consists of the following programs:

- PASCAL Compiler
- Macro Assembler
- Micro Assembler
- Library Maintenance
- Link Editor

The following is a description of each of these programs.

2.1 PASCAL COMPILER

The PASCAL Compiler is available to the user for implementing programs in the higher-level language, PASCAL. Parameters may be passed to the compiler on the compiler call card and through comments in the PASCAL source program. (Refer to the PASCAL reference manual for further information.)

2.1.1 CALL CARD PARAMETERS

The following are the call card parameters:

P = lfn

Where: lfn is the name of the file on which the CYBER PASCAL source program resides. If blank, INPUT is assumed.

L = lfn

Where: lfn is the name of the file on which the compiler writes the source listing. If blank, OUTPUT is assumed.

O

When this option is used, a listing of the object code is written to the file specified by the L option. The default is no object code listing.

CSET = n

Where: CSET is the character set option. If n = 63 or blank, the 63-character set is specified. If n = 64, the 64-character set is specified.

<u>Examples:</u>

PASCAL.

Where: P = INPUT

L = OUTPUT

(no object listing is output)

CSET = 63

PASCAL (P=COMPILE, O)

Where: P = COMPILE

L = OUTPUT

(object listing is output)

CSET = 63

2.1.2 COMMENT OPTIONS

Compiler options appearing in PASCAL source language comments are preceded by a dollar sign ($), which is the first character in the comment. An option followed by a plus sign turns the option on; an option followed by a minus sign turns the option off. The comment-type options are:

A Check for out-of-range assignments to subrange variables.

B Generate SJQ rather than RTJ.

C For compiler maintenance only.

D Check for division by zero.

E Empty (no space) tag fields in variant records.

G Interrupts inhibited during execution of entry/exit code in recursive procedures.

H Print page headings.

I Compile non-interruptible procedures.

J Eject page.

M Generate object code to be loaded and executed under MSOS.

N Suppress the generation of ENTs during processing of the global declarations.

O Check for stack area overflow.

R Compile recursive procedures.

S Suppress the source listing.

T Set options A, D, O, V, X (on or off).

V Check for dynamic variable area overflow.

X Check for out-of-range array indices.

Y Generate run-anywhere code.

The following is an example of use of the comment options:

```
↱$A-, D-, E+, O+, V+↓
```

2.2 MACRO ASSEMBLER

The Macro Assembler is available to the user for implementing programs written in macro assembler language. Parameters may be passed to the assembler on the assembler call card. (Refer to the Macro Assembler reference manual for further information.)

The Macro Assembler call card has the following format:

ASSEM($p_1, p_2, \ldots, p_n$)

The parameters for p that may be specified on the call card are listed below. These may be in any order and must be in one of the following forms:

Omitted Default value assumed

p Option letter alone

p = x Substitute x as the effective value of the parameter p

Parameters	Meaning
I	Input file name
L	Output file name
B	Binary file name
T	Tidy tab columns (six-digit number specifying columns for operation code, address, and comment field on assembly listing).
LO	List options.
NR	Do not rewind input file before assembly.
M	Build assembly text.

Parameters	Meaning
F	Call assembly text from system file.
G	Call assembly text from user file.

The parameter options are as follows:

Input file name

Blank	Input on file INPUT
I	Input on file COMPILE
I=lfn	Input on file lfn

Output file name

Blank	List on file OUTPUT
L	List on file OUTPUT
L=lfn	List on file lfn
L=0	No list

Binary file name

Blank	Binary output on file LGO
B	Binary output on file LGO
B=lfn	Binary output on file lfn
B=0	No binary output

Tidy tab columns

Blank	Tabs set at 11, 18, and 31
T	Tabs set at 11, 18, and 31
T=abcdef	Tabs set at ab, cd, and ef (a through f are decimal digits)

List option

Blank	Options B, M, R, and T selected
LO	Options B, M, R, and T selected
LO=xxxx	Where xxxx may be any combination of the following:

B	List BSS/BZS blocks on banner page
C	List program control (SPC, EJT, etc.).
D	Suppress comment cards.
E	Process EJT as eject.
I	List code skipped by IFA pseudo operation.
L	List macro cross-reference.
M	List all entries on multiword instructions.
R	List full reference map.
S	List abbreviated reference map.
T	Tidy listing columns.
X	Expand macro code.

If more than seven list options are desired, a second LO must be specified.

No rewind

Blank	Rewind input file.
NR	Do not rewind input file.

Build assembly text

Blank	No assembly text built.
M=lfn	Output assembly text to file lfn.

Call assembly text

Blank	No assembly text called.
F	Assembly text called from system file SMAC17.
F=lfn	Assembly text called from system file lfn.
G	Assembly text called from user file SMAC17.
G=lfn	Assembly text called from user file lfn.

To call an assembly text, it must have been built previous to this run and must reside on the file called.

2.3 MICRO ASSEMBLER

The Micro Assembler is available to the user for implementing programs written in micro assembler language. Parameters may be passed to the assembler on the assembler call card and through pseudo operation codes embedded in the source program. (Refer to the Micro Assembler reference manual for further information.)

2.3.1 CALL CARD PARAMETERS

The following are the call card parameters:

p1 The logical file name of the file on which the micro-program source resides. If specified, it is the first parameter on the call card. If blank, INPUT is assumed.

p2 The logical file name of the file on which the assembler writes the source listing. If specified, it is the second parameter on the call card. If blank, OUTPUT is assumed.

p3 The logical file name of the file on which the assembler writes the object output. If specified, it is the third parameter on the call card. If blank, MP17BO is assumed.

Examples:

```
     MASSEM.

Where:    p1 is INPUT
          p2 is OUTPUT
          p3 is MP17BO

     MASSEM(COMPILE,,LGO)

Where:    p1 is COMPILE
          p2 is OUTPUT
          p3 is LGO
```

2.3.2 PSEUDO OPERATIONS

The assembler provides eight pseudo operations that govern assembler options. The pseudo operations must be punched in card columns 11 through 14. These cards may appear anywhere in the source program. The following are the pseudo operations:

CMP1 — Timing information generated by the assembler for code following this pseudo operation assumes the CYBER 18 is operating in twos complement mode.

CMP2 — Timing information generated by the assembler for code following this pseudo operation assumes the CYBER 18 is operating in ones complement mode.

ABS — The assembler produces an absolute binary tape on file P3.

DEAD — The assembler produces an absolute Hollerith deck suitable for loading into micro memory via the micro-processor panel interface. Output is written to file P3.

RELO — Produce binary output in a relocatable form compatible with MSOS. Output is written to file P3. The name punched on the NAM block is obtained from columns 17 through 22 of the IDENT card.

ENT — Define an entry point at relative program location zero. The entry point name is found in card columns 17 through 22 on the ENT card. ENT and RELO pseudo operations should be used together.

ZMAP — Produce a map of all locations within a program that are zero. The map is written on file P2.

PMAP — Produce a list containing both the address of the first instruction following each ORG pseudo instruction in the program and the number of the card that caused the location to be assembled.

2.4 LIBRARY MAINTENANCE PROGRAM

The library maintenance program (MPLIB) is used to create and maintain libraries of object programs. The object programs are produced by the PASCAL Compiler, the Macro Assembler, and the Micro Assembler.

2.4.1 CALL CARD PARAMETERS

The following are the call card parameters:

p1	The logical file name of an object program file. If specified, it is the first parameter on the call card. If blank, LGO is assumed.
p2	The logical file name of the directives file. If specified, it is the second parameter on the call card. If blank, INPUT is assumed.
p3	The logical file name of the print file. If specified, it is the third parameter on the call card. If blank, OUTPUT is assumed.
p4	The logical file name of an old library file, previously created by MPLIB. If specified, it is the fourth parameter on the call card. If blank, OLDLIB is assumed.
p5	The logical file name of a new library file that is to be created by MPLIB during the current run. If specified, it is the fifth parameter on the call card. If blank, NEWLIB is assumed.

Examples:

```
    MPLIB.

Where:  p1 is LGO
        p2 is INPUT
        p3 is OUTPUT
        p4 is OLDLIB
        p5 is NEWLIB

    MPLIB(,,,OLDOBJ,NEWOBJ)

Where:  p1 is LGO
        p2 is INPUT
        p3 is OUTPUT
        p4 is OLDOBJ
        p5 is NEWOBJ
```

2.4.2 DIRECTIVES

The following commands are used in the Library Maintenance program:

*ALL. This directive causes MPLIB to copy all programs on the LGO file to the new library file.

*PUT, pgm. This directive causes MPLIB to copy the object program named pgm on the LGO file to the new library file.

*DEL, pgm. This directive causes MPLIB to suppress the copying of the object program named pgm on the old library file to the new library file.

*LST, lib. This directive causes MPLIB to list the contents of a library. If lib = OLD, then the old library is listed. If lib = NEW, then the new library is listed. Both *LST, OLD and *LST, NEW directives may be input in the same run.

*END. This directive indicates the end of the directives.

Examples:

1.
```
*ALL.
*LST,NEW.
*END.
```

2.
```
*PUT,PROGA.
*PUT,PROGB.
*DEL,PROGC.
*LST,OLD.
*LST,NEW.
*END.
```

Note that the END directive is optional and that terminating directives with a period is optional.

2.4.3 FILES

2.4.3.1 LGO File

MPLIB rewrites the LGO file as a random access file. Object programs that replace programs on the old library are copied to the new library during the processing of the old library. New object programs that were not on the old library are appended to the end of the new library in the order in which they occur on the LGO file.

2.4.3.2 Directives File

Directives begin in card column 1 and consist of an asterisk followed by the directive, which may optionally be terminated by a period. Note that directives do not need to be input; in which case, the contents of the old library are copied to the new library without alteration.

2.4.3.3 Output File

Directives are listed as they are read. If an LST directive is input, then a library listing follows the directives listing. If both *LST,OLD and *LST,NEW directives are input, then the old library is listed first followed by the new library. A library listing consists of program name, length, and entry points for each object program in the library.

2.4.3.4 Old Library File

The old library file is a library file previously created by MPLIB. Once a library file has been created, it cannot be modified by MPLIB. Note that an old library file need not be present (for instance, during a creation run).

2.4.3.5 New Library File

A new library file is always created during an MPLIB run.

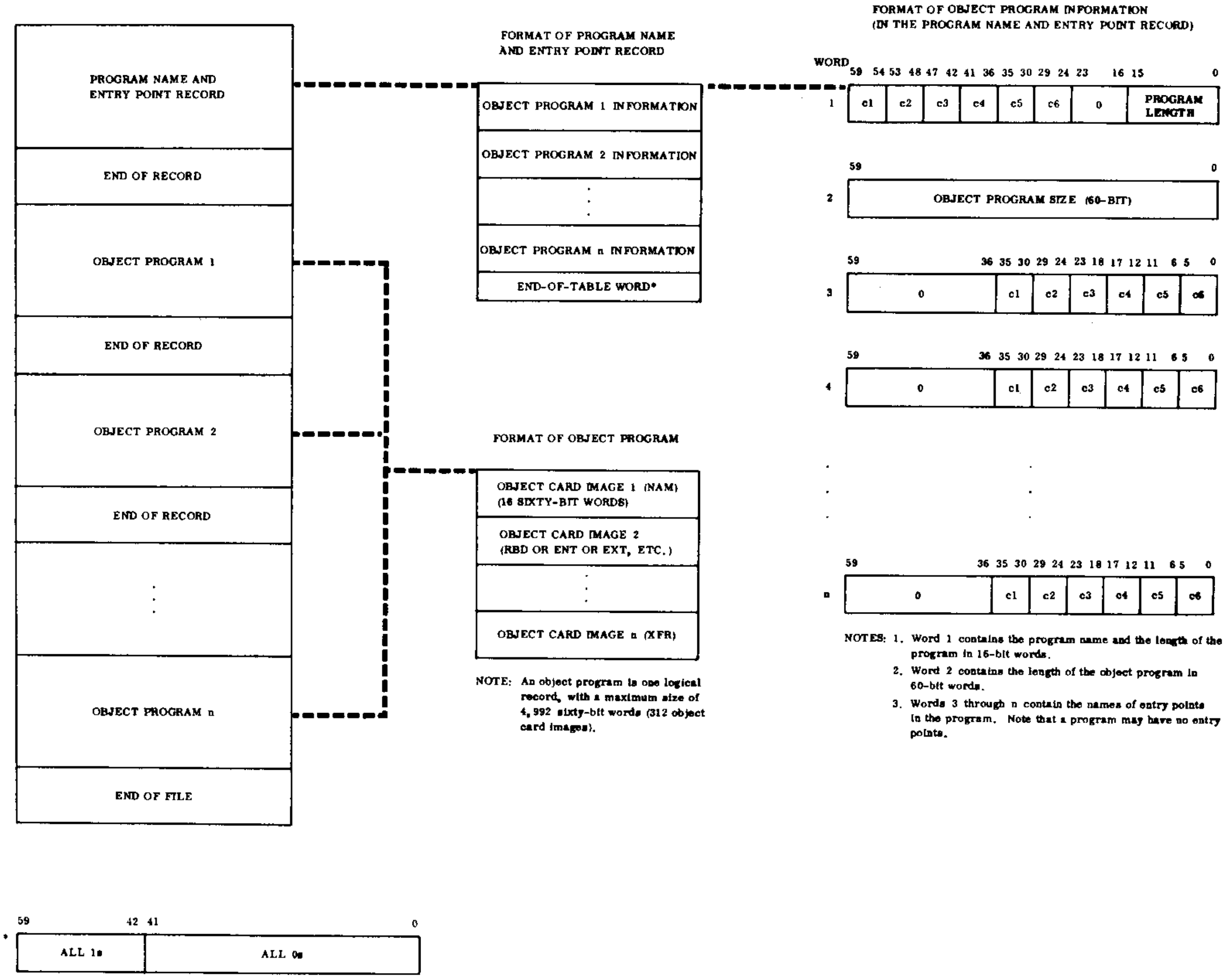
2.4.3.6 Library File Format

Library files are sequential files. The first record on a library file is a program name and entry point record; the object programs follow with each program being one logical record. Refer to Figure 2-1.

2.5 LINK EDITOR

The Link Editor accepts object text modules from the various CYBER Cross System translators and generates an absolutized memory image file suitable for loading into the processor. The Link Editor executes in two separately callable phases:

- Link phase (MPLINK)
- Edit phase (MPEDIT)


PROGRAM NAME AND ENTRY POINT RECORD
END OF RECORD
OBJECT PROGRAM 1
END OF RECORD
OBJECT PROGRAM 2
END OF RECORD
. . .
OBJECT PROGRAM n
END OF FILE
FORMAT OF PROGRAM NAME AND ENTRY POINT RECORD
OBJECT PROGRAM 1 INFORMATION
OBJECT PROGRAM 2 INFORMATION
. . .
OBJECT PROGRAM n INFORMATION
END-OF-TABLE WORD*
FORMAT OF OBJECT PROGRAM
OBJECT CARD IMAGE 1 (NAM) (16 SIXTY-BIT WORDS)
OBJECT CARD IMAGE 2 (RBD OR ENT OR EXT, ETC.)
. . .
OBJECT CARD IMAGE n (XFR)
FORMAT OF OBJECT PROGRAM INFORMATION (IN THE PROGRAM NAME AND ENTRY POINT RECORD)
WORD
1
59 54 53 48 47 42 41 36 35 30 29 24 23 16 15 0
c1 c2 c3 c4 c5 c6 0 PROGRAM LENGTH
2
59 0
OBJECT PROGRAM SIZE (60-BIT)
3
59 36 35 30 29 24 23 18 17 12 11 6 5 0
0 c1 c2 c3 c4 c5 c6
4
59 36 35 30 29 24 23 18 17 12 11 6 5 0
0 c1 c2 c3 c4 c5 c6
. . .
n
59 36 35 30 29 24 23 18 17 12 11 6 5 0
0 c1 c2 c3 c4 c5 c6
*
59 42 41 0
ALL 1s
ALL 0s

NOTE: An object program is one logical record, with a maximum size of 4,992 sixty-bit words (312 object card images).

NOTES:
1. Word 1 contains the program name and the length of the program in 16-bit words.
2. Word 2 contains the length of the object program in 60-bit words.
3. Words 3 through n contain the names of entry points in the program. Note that a program may have no entry points.

Figure 2-1. Format of Library File

2.5.1 LINK PHASE

The link phase inputs a set of directives which establish the object text modules to be linked and directs their mapping into the processor's memory.

2.5.1.1 Call Card Parameters

The link phase call card has the following format:

MPLINK(D = inlfn, R = outlfn, CSET = n)

Where: inlfn is the name of the file on which the input directives reside. If blank, INPUT is assumed.

outlfn is the name of the file to which the link phase generated output listings is written. If blank, OUTPUT is assumed.

n is the character set option. If n = 63 is specified, the CDC Display Code 63-character set is utilized. If n = 64 is specified, the CDC Display Code 64-character set is utilized. If blank, n = 63 is assumed.

<u>Examples:</u>

MPLINK.

Where the default conditions are used; that is,

D = INPUT	(Default)
R = OUTPUT	(Default)
CSET = 63	(Default)

MPLINK(D = COMPILE,CSET = 64)

Where:

D = COMPILE	
R = OUTPUT	(Default)
CSET = 64	

2.5.1.2 Directives

All directives are delimited by a beginning asterisk character (*) and a terminating period character (.). The link phase directives are:

*COM Define blank common area. Establishes the limits of the area that may be designated with a macro assembly language COM data storage allocation.

*COR Memory size. Establishes the highest macro memory address that may be assigned.

*DAT Define named common area. Establishes the limits of the area that may be designated with a macro assembly language data storage allocation statement.

*DMP Dump memory image. Generates a hexadecimal dump listing of the generated memory image load file.

*DSTK Define stack area. Establishes the limits of the stack area that is used in conjunction with PASCAL recursive procedures.

*DVAR Define dynamic variable area. Establishes the limits of the dynamic variable (that area that may be accessed via the PASCAL standard procedure, NEW).

*END Directives end. Marks the end of the link phase directives.

*ENT Define entry point. Assigns a value to a name that may be used to satisfy an external reference.

*L Link. Links one or the remaining modules on an object text library at the designated memory location.

*LIB Library file. Designates that unsatisfied externals may be resolved from a library of object text modules provided through the library maintenance facility.

*OVLY Define an overlay area. Establishes the name and memory limits for an overlay area.

*SYSID System identification. Establishes a user-assigned system identified for a memory image load file build.

*SYN External synonym. Establishes a name that is to be used in place of a designated object text name.

2.5.1.3 Files

The following file names are input to the link phase:

INPUT or D = infile The link phase directives

LGO The object text library file. This file may either be the batch of object text modules as produced by the Cross System translators or a library file prepared by the Library Maintenance program.

NEWLIB The library file prepared by the Library Maintenance program that may be used to resolve unsatisfied externals

The following file names are output by the link phase:

OUTPUT or R = outfile	The generated output listings: A copy of the input directives. Two module memory maps sorted by: -Module name -Memory location Two entry symbol lists sorted by: -Entry name -Value or address A hexadecimal dump listing of the generated memory image file (optional).
ABSOLMP	The generated memory image file
SYMTAB	The link phase symbol table. This file may be input to the edit phase for use in symbolically editing the memory image file.

2.5.2 EDIT PHASE

The edit phase provides the ability to introduce specialized data (e.g., field initial values, system dependent parameters, etc.) into the memory image file generated in the link phase. (Refer to the Link Editor reference manual for further information.)

2.5.2.1 Call Card Parameters

The edit phase call card has the following format:

```
MPEDIT(D = inlfn, R = outlfn, CSET = n)
```

Where: inlfn is the name of the file on which the edit phase source program resides. If blank, INPUT is assumed.

outlfn is the name of the file to which the edit phase generated output listings is written. If blank, OUTPUT is assumed.

n is the character set option. If n = 63 is specified, the CDC Display Code 63-character set is utilized. If n = 64 is specified, the CDC Display Code 64-character set is utilized. If blank, n = 63 is assumed.

Examples:

```
MPEDIT.
```

Where the default conditions are used; that is,

D = INPUT	(Default)
R = OUTPUT	(Default)
CSET = 63	(Default)

```
MPEDIT(D = COMPILE, CSET = 64)
```

Where: D = COMPILE
R = OUTPUT (Default)
CSET = 64

2.5.2.2 Statements

Edit values are introduced via elements referred to as assignment statements. These assignment statements plus some ancillary specifications possess a syntactical structure similar to corresponding elements available in the PASCAL programming language.

2.5.2.3 Files

The following file names are input to the edit phase:

INPUT or D = inlfn	The edit phase statements
ABSOLMP	The memory image file which is to be edited
SYMTAB	The link phase symbol table

The following file names are output by the edit phase:

OUTPUT or R = outlfn	The generated output listings: A copy of the input statement. The entry symbol list (including local symbols introduced in the edit phase) sorted by entry name (optional) A trace listing of the edited words (optional) A hexadecimal dump listing of the edited memory image file (optional)
ZAPMP	The edited memory image file

GLOSSARY

Global

Those definitions and declarations that are in effect for the entire PASCAL program.

Load module

A set of data that is executable on the micro processor. It consists of one or more absolutized object modules.

Local

Those definitions and declarations in PASCAL that are only in effect for a particular procedure or function.

Macro assembler language

A language which when translated produces a superset of the 1700 instruction set that is interpreted by the firmware.

Micro assembler language

A language which when translated produces instructions that are native to the micro processor.

PASCAL

A higher-level language based on ALGOL

Object module

An object program in CDC CYBER 18 format. It is the result of translating a source module.

Source module

A source program; it may be written in PASCAL, macro assembler language, or micro assembler language.

LOAD PROCEDURE A

There are two loaders for the CYBER Cross System:

- Seven-track magnetic tape loader
- Nine-track magnetic tape loader

The loader (seven-track or nine-track) that is present in the LGO file or in the object program library is written as the first record on the load module file (ABSOLMP) by the Link Editor.

After completion of the link edit step, ABSOLMP may be copied to the pertinent type of magnetic tape (seven-track or nine-track) using the system utility program, COPYBF. The tape is then mounted on a tape unit that is attached to the micro processor. The appropriate bootstrap (seven-track or nine-track) is input through the card reader. The bootstrap reads the loader into macro memory and transfers control to it. The loader reads and loads the remainder of the load module file into macro memory and transfers control to the transfer address specified in the trailer record.

USING THE CROSS SYSTEM B

The following examples are for NOS/BE 1.1:

1. Compile a PASCAL program:

```
ABC,CM77000,T77,P4.      0000,xxxx,xxxxxxxx,SMITH.
ATTACH(PASCAL,ID=SCDD)
PASCAL.
7/8/9
···PASCAL source program···
6/7/8/9
```

2. Assemble a macro assembler language program:

```
ABC,CM77000,T77,P4.      0000,xxxx,xxxxxxxx,SMITH.
ASSEM(F)
7/8/9
···macro assembler language program···
6/7/8/9
```

3. Assemble a micro assembler language program:

```
ABC,CM77000,T77,P4.      0000,xxxx,xxxxxxxx,SMITH.
MASSEM.
7/8/9
···micro assembler language program···
6/7/8/9
```

4. Create a new object program library from object programs produced by PASCAL compilation:

```
ABC,CM77000,T77,P4.      0000,xxxx,xxxxxxxx,SMITH.
REQUEST(NEWLIB,*PF)
ATTACH(PASCAL,ID=SCDD)
PASCAL (O,CSET=64)
FRMT.
MPLIB.
CATALOG(NEWLIB,OBJPGMLIB01,ID=PT,RP=30)
7/8/9
···PASCAL source program···
7/8/9
*ALL.
*LST,NEW.
*END.
6/7/8/9
```

NOTE

Output from the PASCAL Compiler is reformatted by the format program to be compatible with output from the Macro Assembler. The contents of the new object program library is listed.

5. Assemble and add a macro assembler language program to an object program library:

```
ABC,CM77000,T77,P4.      0000,xxxx,xxxxxxxx,SMITH.
REQUEST(NEWLIB,*PF)
ATTACH(OLDLIB,OBJPGMLIB01,ID=PT)
ASSEM(F)
MPLIB.
CATALOG(NEWLIB,OBJPGMLIB02,ID=PT,RP=30)
7/8/9
···macro assembler language program(MA1)···
7/8/9
*PUT,MA1.
*END.
6/7/8/9
```

6. Assemble and add a micro assembler language program to an object program library:

```
ABC,CM77000,T77,P4.      0000,xxxx,xxxxxxxx,SMITH.
REQUEST(NEWLIB,*PF)
ATTACH(OLDLIB,OBJPGMLIB02,ID=PT)
MASSEM(,,LGO)
MPLIB.
CATALOG(NEWLIB,OBJPGMLIB03,ID=PT,RP=30)
7/8/9
···micro assembler language program(MI1)···
7/8/9
*PUT,MI1.
*END.
6/7/8/9
```

7. List the contents of an object program library:

```
ABC,CM77000,T77,P4.      0000,xxxx,xxxxxxxx,SMITH.
ATTACH(OLDLIB,OBJPGMLIB03,ID=PT)
MPLIB.
7/8/9
*LST,OLD.
*END.
6/7/8/9
```

8. Compile a PASCAL source program and build a load module satisfying external references from an object program library:

```
ABC,CM77000, T77, P4.         0000, xxxx, xxxxxxxx, SMITH.
REQUEST(ABSOLMP, *PF)
ATTACH(NEWLIB, OBJPGMLIB03, ID=PT)
ATTACH(MPLINK,ID=SCDD)
ATTACH(PASCAL,ID=SCDD)
PASCAL(O,CSET=64)
FRMT.
REWIND(LGO)
MPLINK(CSET=64)
CATALOG(ABSOLMP,LOADMOD01, ID=PT, RP=30)
7/8/9
•••PASCAL source program•••
7/8/9
*LIB.
*DSTK,$5000, $5FFF.
*DVAR,$6000, $6FFF.
*L, , $200.
*END,MAIN$.
6/7/8/9
```

After all of the object programs on the LGO file have been read and loaded, any remaining unsatisfied external references are resolved using the object program library OBJPGMLIB03. The *DSTK and *DVAR link editor directives specify the stack area and the dynamic variable area, respectively.

9. Build a load module from an object program library with editing of the load module file.

```
ABC,CM77000, T77, P4.         0000, xxxx, xxxxxxxx, SMITH.
REQUEST(ZAPMP,*PF)
ATTACH(NEWLIB,OBJPGMLIB03, ID=PT)
ATTACH(MPLINK,ID=SCDD)
ATTACH(MPEDIT,ID=SCDD)
MPLINK(CSET=64)
REWIND(ABSOLMP,SYMTAB)
MPEDIT(CSET=64)
CATALOG(ZAPMP,LOADMOD02, ID=PT, RP=30)
7/8/9
*REL,NEWLIB.
*L,PGM1, $200.
*L,PGM2.
*L,PGM3.
*DMP.
*END.
7/8/9
•••MPEDIT program•••
6/7/8/9
```

PGM1, PGM2, and PGM3 are the names of programs in the object program library. Loading begins at address 200_{16}. The core image is written in hexadecimal to the OUTPUT file.

COMMENT SHEET

MANUAL TITLE CONTROL DATA® CYBER Cross System Version 1 Reference Manual

Reference Manual

PUBLICATION NO. 96836000 REVISION B

FROM NAME:

BUSINESS
ADDRESS:

COMMENTS: This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number.

CUT ALONG LINE

STAPLE

STAPLE

FOLD

FIRST CLASS
PERMIT NO. 333

LA JOLLA, CA.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
PUBLICATIONS AND GRAPHICS DIVISION
4455 EASTGATE MALL
LA JOLLA, CALIFORNIA 92037

CUT ALONG LINE

FOLD

STAPLE

STAPLE