60449800

CONTROL DATA CORPORATION

CYBER LOADER VERSION 1 INSTANT

CDC®
OPERATING SYSTEMS:
NOS 2
NOS/BE 1

REVISION RECORD

Revision	Description
A	Original printing. This manual is a successor publication to publication number 60372200 for users of the NOS 1.0, NOS/BE 1.0, and SCOPE 2.1 operating systems.
B (05/05/77)	This revision documents Version 1.3 of the CYBER Loader for the NOS 1 and NOS/BE 1 operating systems. (References to the SCOPE 2 operating system have been removed.) This revision is effective with release of PSR level 446.
C (08/01/79)	This revision documents Versions 1.4 and 1.5 of the CYBER Loader. Version 1.4 introduced the Fast Overlay Loader, multiple entry points in the main overlay, a new DEBUG PRESET option and 8 lines/inch for map output. Version 1.5 introduced SEGLOAD common blocks and PTEXT table. This revision is effective with the release of PSR level 498.
D (04/25/86)	CYBER Loader for NOS 2 and NOS/BE 1 operating systems. (References to the NOS 1 operating system have been removed.) This revision is effective with release of PSR level 647.

REVISION LETTERS I, O, Q, AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
P. O. BOX 3492
SUNNYVALE, CALIFORNIA 94088-3492

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
Front Cover	-
Title Page	-
ii	D
iii/iv	D
v/vi	D
vii	D
viii	D
1 thru 3	C
4	D
5	D
6 thru 8	C
9	D
10	C
10.1	D
10.2	D
10.3/10.4	D
11 thru 15	C
16	D
17	D
18	C
19	D
20 thru 30	C
31	D
32	D
32.1/32.2	D
33 thru 36	C
37	D
38	D
39 thru 50	C
51 thru 54	D
54.1/54.2	D
55 thru 60	C
Back Cover	-

PREFACE

This manual summarizes features of the CONTROL DATA® CYBER Loader, which is included as part of the following operating systems:

NOS/BE 1 for the CDC® CYBER 170 Series, CYBER 70 Models 71, 72, 73, 74, and 6000 Series Computer Systems

NOS 2 for the CDC CYBER 180 Series; CYBER 170 Series; and CYBER 70 Models 71, 72, 73, and 74

The user is assumed to be familiar with the operating system and computer system in use.

More detailed information can be found in the CYBER Loader Version 1 Reference Manual, publication number 60429800.

United States sites can order CONTROL DATA publications from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103. Other sites should order CDC manuals through their local country sales office.

CONTENTS

TABLES

TYPES OF LOADING

ABSOLUTE LOADING

Involves no relocation of addresses or linkage symbols.

Consists of loading one or two blocks. One block resides in central memory; one is optional and resides in extended core storage (ECS).

RELOCATABLE LOADING

Involves the actual connecting of one or more blocks defined through programs.

Relocates addresses and links externals.

Permits loading of programs produced by independent compilations and assemblies.

BASIC LOADING

Involves cases in which all of the object code is loaded into memory concurrently, resulting in a single core image.

SEGMENTATION

Permits selected portions of the program to be in core concurrently. Different core images reside in the same area of memory at different times.

Loading is dynamic and results from implicit calls.

OVERLAY GENERATION

Permits selected portions of the program to be in core concurrently.

Provides an overlay structure compatible with previous 6000 series loaders.

Loading results from explicit loader calls.

CAPSULES

Provides for the dynamic loading and unloading of specially processed relocatable code from an executing program.

The executing program issues an explicit call to the Fast Dynamic Loading (FDL) routine, which in turn loads and unloads the capsules.

Capsules must be formed into groups and placed in a library before they can be loaded by the FDL routine.

OVCAPS

Provides for the dynamic loading and unloading of portions of overlayed programs.

OVCAPs are loaded and unloaded by a call to the FDL routine from a currently loaded overlay.

OVCAPs are generated during an overlay generation load sequence and are analogous to primary overlays.

NAME CALL STATEMENT

Formats:

name.

name($p_1, \ldots, p_n$)

name — A file name or entry point name within a program of one to seven characters that identifies a program that is to be loaded into central memory.

p_i — Parameters to be passed to the loaded program.

A name call statement is the last or only statement in a load sequence. The name type is determined by a search of:

Local files

Global library set

Local library set

Default system library

LOADER (VERB) STATEMENTS

The first field in a verb statement represents an explicitly defined loader command. Execution of a verb statement causes subsequent statements to be interpreted as loader control statements until either an EXECUTE, NOGO, or name call is encountered.

Once in the loader control sequence, operating system control statements are interpreted as name call statements. Exceptions are MAP and REDUCE, which the system recognizes within the loader control sequence for compatibility with previous systems; DMP is recognized but ignored.

An EXIT statement must not appear inside a load sequence.

Loader statements are listed below in alphabetical order.

CAPSULE(pname$_1, \ldots,$pname$_n$)

Specifies the components of capsules to be generated. The program names (pname$_i$) list in order the programs that signal the start of a new capsule.

EXECUTE.

EXECUTE(eptname)

EXECUTE(eptname,p_1, . . . ,p_n)

EXECUTE(,p_1, . . . ,p_n)

Causes completion of the load followed immediately by execution of the loaded program at an entry point (eptname) optionally specified. Unsatisfied externals are satisfied from the library set. p_i are optional execution time parameters.

GROUP(groupname)

Specifies the name of a capsule group. It applies to the capsules specified by the CAPSULE statements following the GROUP statement.

LDSET(option$_1$, . . . ,option$_2$)

Provides user control of a variety of load operations.

Options are described in table 1 and are specified in one of the following forms:

key

key=param

key=param$_1$/ . . . /param$_n$

LDSET options apply for the current load sequence only.

LIBLOAD(libname,eptname$_1$, . . . ,eptname$_n$)

Requests a physical load of one or more programs from a library (libname) according to entry point names (eptname$_i$).

LOAD(lfn$_1$, . . . ,lfn$_n$)

Requests physical load of either object programs from each of the files (lfn$_i$) specified in the sequence or a single absolute overlay from a single file. A file name can be optionally accompanied by a rewind indicator (/R or /NR).

NOGO.

NOGO(lfn)

NOGO(lfn,eptname$_1$, . . . ,eptname$_n$)

Specifies no execution and optionally saves the core image on the named file (lfn). One or more entry point names (eptname$_i$) can be specified for inclusion in the overlay as program entry points of the absolute program.

SATISFY.

SATISFY(libname$_1$, . . . ,libname$_2$)

Provides for the satisfaction of unsatisfied externals prior to normal satisfaction at load completion, and optionally names user-specified libraries (libname$_i$) from which the externals are to be satisfied.

SEGLOAD(p_1,p_2,p_3)

Specifies that the load is to be a segmented load.

SLOAD(lfn,name$_1$, . . . ,name$_n$)

Requests physical load of selected programs (name$_i$) from a local file (lfn). The file name is optionally accompanied by a rewind indicator (/R or /NR).

TABLE 1. LDSET OPTIONS

Key	Parameters	Description
COMMON	lcbname,/ . . . /lcbname$_n$	Assigns named labeled common blocks to segments such that the blocks will be available to all segments that reference them. Applies to SEGLOAD only.
COMMON		Causes all labeled common blocks to be available.
EPT and NOEPT	eptname$_1$/ . . . /eptname$_n$	Provides control over the entry points of capsules, overlays, and OVCAPs.
ERR	p	Selects one of three methods of handling errors. If no ERR option is specified, the installation default is used. Catastrophic errors always result in job abortion. Also, there are informative errors that never result in job abortions. p — Significance ALL — The program is aborted for errors that are: Fatal, Nonfatal, Catastrophic FATAL — The program is aborted for errors that are: Fatal, Catastrophic NONE — Only catastrophic errors cause job abortion. For any other type of error, processing continues if possible.

FILES or STAT	$lfn_1/\ldots/lfn_n$	Permits a CYBER Record Manager user to assure that library programs are loaded for the processing of specified files. The FILES or STAT option is treated as a no-op in segment generation; its application is the same as in USE for overlay generation.
LIB	$libname_1/$ $\ldots/libname_n$	Specifies one or more libraries comprising the local library set.
LIB		Causes the local library set to be cleared. NOTE The global library set is altered by use of the LIBRARY control statement.
MAP	p/lfn /lfn p	Controls the generation of the load map. lfn Specifies file to receive the map. Default is OUTPUT. lfn is not rewound. p Specifies map contents: omitted Current job default, as set by the last MAP or installation default N No map S Statistics B Block map E Entry point map X Entry point cross-references Map options S, B, E, and X can be concatenated (for example, LDSET,MAP=BX).

TABLE 1. LDSET OPTIONS (Contd)

Key	Parameters	Description
OMIT	$eptname_1$/ . . . /$eptname_n$	Specifies that named entry points are to remain unsatisfied. An OMIT request is in effect from the time it is encountered until the end of the load or until superseded by a USE request; for example, OMIT(XYZ) later followed by USE(XYZ).
PD	p	Provides control over print density of load map. Must be set to 6 or 8 before printing starts.
PRESET and PRESETA	p	Specifies the values to which unused memory is set prior to execution of the loaded program. The loader also presets memory at various times during loading, always using the most recent preset selection. For PRESETA, the lower 17 bits (central memory) or lower 24 bits (ECS) of each word contain its address. For example, if PRESETA=ONES were specified with locations $RA+1000_8$ and $RA+1001_8$ unused, they would be set to: 7777 7777 7777 7740 1000 and 7777 7777 7777 7740 1001

		Under NOS/BE 1, p can be an octal number of 1 to 20 digits, optionally prefixed by + or - and optionally suffixed by the letter B. Under all operating systems, p can be one of the following keywords: p — Octal Preset Value NONE — No presetting for ECS; same as ZERO for CM ZERO — 0000 0000 0000 0000 0000 ONES — 7777 7777 7777 7777 7777 INDEF — 1777 0000 0000 0000 0000 INF — 3777 0000 0000 0000 0000 NGINDEF — 6000 0000 0000 0000 0000 NGINF — 4000 0000 0000 0000 0000 ALTZERO — 2525 2525 2525 2525 2525 ALTONES — 5252 5252 5252 5252 5252 DEBUG — 6000 0000 0004 0040 0000
PS	p	Provides control over page size of load map. Map is printed at p lines per page. $10 \leq p \leq 1\,000\,000$.
REWIND and NOREWIN		Alters the default option for rewinding of files before loading. The selection of /R and /NR on the LOAD and SLOAD statements takes precedence over this option.

TABLE 1. LDSET OPTIONS (Contd)

Key	Parameters	Description
SUBST	$pair_1$/ . . . /$pair_n$	$pair_i$ Pair of entry point names in the form: $eptname_1$-$eptname_2$ Changes the external references to entry point names to other entry point names. For example, a reference to external $eptname_1$ becomes a reference to external $eptname_2$.
USE	$eptname_1$/ . . . /$eptname_n$	Forces the loading of object programs to ensure that specified entry points are included in the load. The loader loads the programs the next time the loader satisfies externals, either as a result of a SATISFY statement or load completion. If the loader is unable to find an entry point name in the libraries searched, it flags a nonfatal error. During overlay generation, a USE applies only to the overlay begun by the last OVERLAY directive read from the last or only file named in the most recent LOAD directive. If the USE occurs prior to the first LOAD directive, it applies to the first overlay.
USEP	$pname_1$/ . . . $pname_n$	Causes the indicated programs to be loaded whether or not they are needed to satisfy external references. The loader loads the programs the next time the loader satisfies externals, either as a result of a SATISFY statement or load completion.

LOADER-RELATED CONTROL STATEMENTS

MAP STATEMENT

Specifies the default option for load maps for load sequences requested by the job.

Formats:

```
MAP.
```

```
MAP(p)
```

p specifies type of map:

omitted	Resets load map option to installation default.
OFF	No map.
PART	Statistics and block map.
ON	Statistics, block map, and entry point cross-reference.
FULL	Statistics, block map, entry point map, and entry point cross-reference.

LIBRARY STATEMENT

Specifies a set of global libraries that are searched to satisfy externals and name call statements. Also specifies the order in which the libraries are to be considered.

Formats:

```
LIBRARY.
```

```
LIBRARY(libname1,...,libnamen)
```

```
LIBRARY(libname1,...,libnamen/p)
```

p Specifies one of the following options:

A	Add libraries to global library set.
D	Delete libraries from global library set.
R	Replace global library set with specified library set.

REDUCE STATEMENT

Specifies whether or not the loader is to determine field length assignment.

Formats:

REDUCE.

NOS/BE batch and NOS only.

REDUCE(ON)

NOS/BE Intercom only.

REDUCE(-)

NOS only.

REDUCE(OFF)

NOS/BE Intercom only.

REDUCE. or REDUCE(ON)

Specifies that the loader assigns field length.

REDUCE(-) or REDUCE(OFF)

Inhibits reduction of the field length by the loader.

Controls the amount of field length that is used to execute program.

Formats:

```
RFL(n)
```

```
RFL(n,m)
```

```
RFL(CM=n,EC=m)
```

n Central memory field length in octal.

m ECS field length in octal.

CALLING THE LOADER

Formats:

LOCATION	OPERATION	VARIABLE SUBFIELDS
	LOADER	paddr
	LOADER	paddr, CMM

paddr The address of LDREQ BEGIN.

CMM Indicates that the Common Memory Manager (CMM) is to be used during loading. It must be specified if CMM is active at the time of the call.

The expanded form of the first format of the LOADER macro call is:

LOCATION	OPERATION	VARIABLE SUBFIELDS
	RJ	=XLOADER=
	CON	paddr

The expanded form of the second format of the LOADER macro call is:

LOCATION	OPERATION	VARIABLE SUBFIELDS
+	RJ	=XLOADER.
-	VFD	30/paddr

FORMATTING THE REQUEST TABLE

Rules:

A request table must accompany each loader call (LOADER macro).

The table must consist of a header followed by one or more loader requests in internal form.

The table is either generated by the programmer directly or generated through LDREQ macro calls.

The first word of the table is the paddr specified in the loader call.

The required sequence of LDREQ macro calls is:

LOCATION	OPERATION	VARIABLE SUBFIELDS
paddr	LDREQ	BEGIN, fwasc, . . . Generates header
	LDREQ	option, . . .
	.	.
	.	.
	.	.
	LDREQ	option, . . .
	LDREQ	END

Generates requests in internal format

Terminates table

paddr Table first word address.

LDREQ options are listed below in alphabetical order.

BEGIN,fwasc,lwasc,fwalc,lwalc

Generates the table header.

fwasc Loadable area in central memory. Default is the value in RA+65_8 (bits 17 through 00).

lwasc Last word address+1 in central memory. Default is RA+FL.

fwalc Loadable area in ECS. Default is RA+65_8 (bits 58 through 36). Bit 59 of paddr+1 is set if fwalc is not blank.

lwalc Last word address+1 of ECS. Default is ECSRA+ECSFL. Bit 29 of paddr+1 is set if lwalc is not blank.

CMLOAD,fwa,end

Specifies load input in central memory, beginning at fwa and ending at end.

DMP,p1,p2

Requests a dump within a loader sequence. This option is intended for use by a system analyst. See the operating system reference manual for parameters.

ECLOAD,fwa,end

Specifies load input in ECS beginning at fwa and ending at end.

END

Terminates the loader request table.

ENTRY,(eptname$_1$, . . . ,eptname$_n$)

Allows the executing program to obtain addresses of entry points that are currently being loaded and/or have previously been loaded.

NOTE

A user call that includes an ENTRY request must not specify limits of the loadable area such that the user call request area is overwritten. The updated request table must be available to the program after the user call is completed.

EXECUTE

EXECUTE,eptname

EXECUTE,eptname,(p$_1$, . . . ,p$_n$)

Provides similar capability and has the same parameters as the EXECUTE statement. The EXECUTE option differs from the EXECUTE statement, as follows:

Libraries are not searched automatically

Blank common is established only if it is declared in the load for the first time

Field length is not altered

The SATISFY request is required before an EXECUTE request if libraries are to be searched to satisfy external references.

FILES,(lfn$_1$, . . . ,lfn$_n$)

Provides the same capability and has the same subparameters as the FILES option of the LDSET statement (table 1).

LIB

LIB,(libname$_1$, . . . ,libname$_n$)

Provides the same capability and has the same subparameters as the LIB option of the LDSET statement (table 1).

LIBLOAD,libname,(eptname$_1$, . . . ,eptname$_n$)

Provides the same capability and has the same parameters as the LIBLOAD statement.

LOAD,(lfn$_1$, . . . ,lfn$_n$)

Provides the same capability and has the same parameters as the LOAD statement.

MAP,p,lfn

Provides the same capability and has the same parameters as the MAP option of the LDSET statement (table 1).

NOGO

Requests load completion. Control is returned to the calling program, not the newly loaded program. Externals are not satisfied unless satisfied through a SATISFY request.

OMIT,(eptname$_1$, . . . ,eptname$_n$)

Provides the same capability and has the same subparameters as the OMIT option of the LDSET statement (table 1).

PASSLOC,((id$_1$,t$_1$,b$_1$,a$_1$), . . . ,(id$_n$,t$_n$,b$_n$,a$_n$))

Allows an executing program to supply addresses to the loader. PASSLOC is an LDREQ option only.

PRESET,p

PRESETA,p

Provides the same capability and has the same subparameters as the PRESET and PRESETA options of the LDSET statement (table 1).

SATISFY

SATISFY,(libname$_1$, . . . ,libname$_n$)

Must be used to satisfy externals at load completion. In all other respects, this option is the same as the SATISFY statement.

SLOAD,lfn,(name$_1$, . . . ,name$_n$)

Provides the same capability and has the same parameters as the SLOAD statement.

SUBST,(pair$_1$, . . . ,pair$_n$)

Provides the same capability and has the same subparameters as the SUBST option on the LDSET statement (table 1).

USE,(eptname$_1$, . . . ,eptname$_n$)

Provides the same capability and has the same subparameters as the USE option of the LDSET statement (table 1).

USEP,(pname$_1$, . . . ,pname$_n$)

Provides the same capability and has the same subparameters as the USEP option of the LDSET statement (table 1).

OVERLAYS

FEATURES

The overlay structure is compatible with that of previous loaders.

Only main overlays can contain multiple entry points.

Loading of an overlay results from an explicit user call.

Organization of object modules into overlays depends on the sequential grouping of programs on the load file.

Three levels of overlays are possible. One main overlay is at the lowest level, up to 63 overlays are allowed at the primary level, and up to 63 overlays are allowed at the secondary level for each overlay at the primary level.

The main overlay can be loaded in various ways. Other overlays must be loaded by explicit calls from a currently loaded overlay.

Memory references between one overlay and another can be downward only (except for the initial entry to a just loaded overlay). That is, programs in the main overlay cannot reference entry points in any other overlay; programs in a primary overlay can reference entry points in the main overlay; and programs in the secondary overlay can reference entry points in either the main overlay or the associated primary overlay.

OVERLAY DIRECTIVE

Formats:

OVERLAY(lfn,0,0,origin,OV=n)

OVERLAY(ℓ_1,ℓ_2,origin)

lfn — Optionally names the file on which the overlay is to be written. (A file specified on a NOGO statement overrides the file named on the OVERLAY directive.) If lfn is not specified, the overlay is written on either the file named on the NOGO statement, the file specified on a previous OVERLAY directive, or file ABS.

ℓ_1 — Primary level number, 0 through 77_8.

ℓ_2 Secondary level number, 0 through 77_8.

origin Optional parameter specifying the origin of the overlay; not allowed for the (0,0) overlay. The forms of the origin parameter are:

Cnnnnn The overlay is loaded nnnnnn$_8$ words from the start of blank common.

0=nnnnnn The overlay is loaded at the address specified. nnnnnn must be greater than or equal to 110_8.

0=eptname The overlay is loaded at the entry point specified.

0=eptname ± nnnnnn The overlay is loaded at the entry point specified, offset by ±nnnnnn.

OV=n Optional parameter specifying the overlay generator is to generate an overlay structure suitable for Fast Overlay Loader (FOL). n specifies decimal number, 0 - 2000, of higher level overlays or OVCAPs in overlay structure. OV=n can only be specified on the (0,0) overlay.

REQUESTING OVERLAY LOADING AND EXECUTION

Format:

LOCATION	OPERATION	VARIABLE SUBFIELDS
	LOADREQ	paddr,rcl,flag

paddr An address expression, specifying the first word of the parameter area.

rcl Auto-recall indicator (optional):

null Control returns immediately.

not null Control returns after the call is completed.

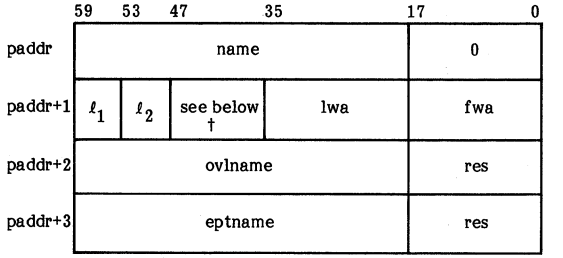
flag A keyword indicating special action. Required if the Common Memory Manager is active at the time of the call. The keywords are:

null No special action; CMM must not be active.

CMM Indicates an overlay load by a job that uses CMM. This call activates CMM if it is not already active.

DATA Indicates that the loaded overlay is to be treated as data, such as a system text. This call can be made when CMM is either active or inactive, and neither activates nor deactivates CMM if it is present. Both fwa and lwa must be supplied in the parameter area, and the e bit must not be set.

Parameter area before the call:



name A library name, overlay name, or entry point name (see below), left-justified with zero fill.

ℓ_1 Primary overlay level.

ℓ_2 Secondary overlay level.

†n Number of words-2 in the request (bits 47 and 46).

†res Used by CDC (bit 45).

†s,d Used by CMM (bits 43 and 44).

†u Load option (bit 42).

† v Overlay load flag (bit 41).

† res Used by CDC (bit 37 through 40).

† e Automatic execute flag (bit 36).

lwa The last word address available for load.

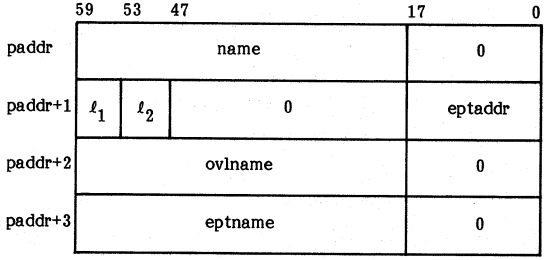
fwa The first word address available for load.

ovlname The overlay name, left-justified with zero fill.

eptname The entry point name, left-justified with zero fill.

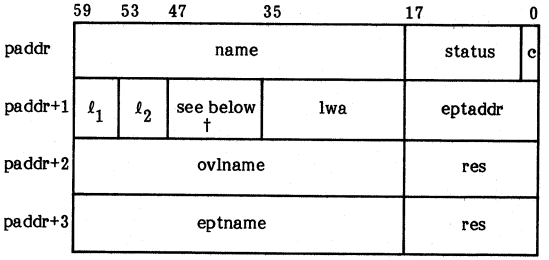
Fields specified as res can be used only by CDC.

Parameter area after an overlay load under NOS 2:



eptaddr Entry point address of the overlay. If n is 2, eptaddr is the address of eptname.

Parameter area after an overlay load under NOS/BE 1:



†n Number of words-2 in the request (bits 47 and 46).

†res Used by CDC (bit 45).

†s,d Used by CMM (bits 43 and 44).

†u Load option (bit 42).

†v Overlay load flag (bit 41).

†res Used by CDC (bit 38 through 40).

†ne Nonfatal error flag (bit 37).

†fe Fatal error flag (bit 36).

status Contents of the field depend on ne and fe, as follows:

ne	fe	Contents of Status Field
0	0	Zero
0	1	Fatal error code
1	0	Error code for first nonfatal error
1	1	Fatal error code

c Completion flag:
1 = Call is completed

ne Nonfatal error flag:
0 = No nonfatal errors occurred
1 = One or more nonfatal errors occurred

fe Fatal error flag:
0 = No fatal errors occurred
1 = A fatal error occurred

SEGMENTATION

FEATURES

A segment can have more than one entry point.

Segment loads are implicit. Execution of an instruction that refers to an entry point in a currently nonloaded segment automatically results in calling the SEGLOAD resident program, which assumes control of loading segments.

A segment load can involve more than one level. This feature allows gaps in memory between segments that are logically connected.

Object programs can be selected from more than one source file.

Calls for the SEGLOAD loader can be made through the control statement only.

SEQUENCE OF EVENTS

1. Directive translation
2. Object program indexing
3. Object program processing
4. Execution
5. Segment loading under control of SEGLOAD resident (SEGRES)

SEGLOAD CONTROL STATEMENT

Format:

SEGLOAD(p_1, . . . ,p_n)

p_i Represents the following optional parameters in any order:

I=lfn	Identifies the source of directives; default is INPUT
B=lfn	Identifies the binary output file; default is ABS
LO=c_1c_2	Identifies list options for the segment loader, depending on the value of c_1c_2, as follows:
0	Neither tree diagram nor directive list
D	Directive list only
T	Tree diagram only

DT TD omitted	Both directive list and tree diagram

If the LO parameter is omitted, the list options depend on the load map produced. If MAP(OFF) or LDSET(MAP=N) is specified, LO=0 is assumed; otherwise, LO=DT is assumed.

Statements allowed in the same loader sequence as the SEGLOAD control statement are:

name*call

LOAD

LIBLOAD

SLOAD

EXECUTE

NOGO

LDSET

The rules for programs on files are:

No program can span record boundaries

Any embedded directive (for example, OVERLAY, which is ignored) must occur in a separate record

DIRECTIVES

Syntax:

General	Consists of four fields: label, verb, specification, and comment, and uses columns 1 through 72 of a line.
Continuation	Comma in column 1.
Comment	Asterisk in column 1.
Page eject	Slash in column 1; remainder of the directive is ignored.
Label field	Begins in column 1 or 2 and is terminated by a blank. One to seven characters.
Verb field	First nonblank character after label; or, if columns 1 and 2 are blank, the first nonblank character of the line. Terminated by a blank.
Specification field	One or more subfields, beginning with the first nonblank character after verb. Each subfield contains one to seven characters and is terminated by , - ( ) or blank.

Comment field	Begins after a specification field specia character followed by one or more blanks.

Directives are listed below in alphabetical order:

Label	Verb	Specification
$bname_i$	COMMON	$bname_1, \ldots, bname_n$ Renders the labeled common block addressable by all segments.
	END	$eptname_1, \ldots, eptname_n$ Signifies the end of directives and optionally, specifies entry points at which execution can begin.
$bname_g$	EQUAL	$bname_1, \ldots, bname_n$ Renders blocks named in the specification field synonymous with the global block named in the label field.
segname	GLOBAL	$bname_1, \ldots, bname_n$(-SAVE) Renders a labeled common block accessible by all segments that do not overwrite it, and optionally preserves and restores its contents. The named blocks are in segname.
segname	INCLUDE	$program_1, \ldots, program_n$ Forces inclusion of object programs into a specified segment, overriding normal assignment. If segname is omitted, programs are placed in the root segment. segname must be on a TREE directive.
	LEVEL	Divides memory and allows trees to be spatially independent.
tname	TREE	expression Organizes segments into a tree structure. tname is an optional name by which the tree can be referenced in other TREE directives. expression is a string composed of segment names and/or tree names linked by - and , and grouped by parentheses.

)EBUG AIDS

RAP CONTROL STATEMENT

'ormat:

TRAP(p_1,p_2)

p_i Represents the following optional parameters:

I=lfn	Source of directives; default is INPUT
L=lfn	List output file; default is TRAPS

'RAP can only be used with basic loads.

)IRECTIVES

'yntax:

General	Consists of three fields: label, verb, and specification, and uses columns 1 through 72 of a line.
Continuation	Comma in column 1.
Comment	Asterisk in column 1.
Page eject	Slash in column 1; the remainder of the directive is ignored.
Label field	Starts in column 1, contains one to seven characters, and is terminated by , - ( ) or blank. The field is empty if column 1 is blank.
Verb field	First nonblank character after label, or first character of the line. Terminated by , - ( ) or blank.
Specification	First nonblank character after the verb field. Specification consists of a list of parameters separated by commas or blanks.

The TRAP directives are listed below; the parameters are summarized in tables 2, 3, 4, and 5.

FRAME

TRACK

TABLE 2. FRAME DIRECTIVE PARAMETERS

Parameter		Default (No Parameter Specified)	Description
Keyword	Phrase		
AT	BLOCK name octal CM address PROGRAM name octal CM address PROG name octal CM address ABS CM octal CM address	None (parameter is required).	Specifies the address of the instruction at which the dump is to be taken.
FROM	BLOCK name octal address PROGRAM name octal address PROG name octal address ABS CM octal address ABS ECS octal address	0	First word address for snapshot dump.
FOR	Decimal number	fl-fwa (used only if the first word address is in central memory). No default for ECS.	Specifies the number of words to be dumped.
START	Decimal number	1	Iteration of the AT instruction, at which the first dump is to be taken.

EVERY	Decimal number	1	Interval after which the dump is to be repeated.
UNTIL	Decimal number	131071	Last interaction at which the dump is to be taken.
REG		No register dump.	Presence of REG specifies that the dump is to include the contents of registers.

TABLE 3. PHRASES FOR FRAME PARAMETERS

Phrase		Value Type	Description
Keyword	Value		
PROGRAM or PROG	name	Alphanumeric program block name, one to seven characters.	Name of the program block containing the address specified by AT or FROM.

BLOCK	name	Alphanumeric block name, one to seven characters. Blank common is indicated as:
		// Central memory (CM)
		//L ECS (FROM only)
			Name of the block containing the address specified by AT or FROM. //L applied to FROM only.

ABS	type	Core memory type: CM ECS	Type of memory containing the absolute address specified by AT or FROM. If type is omitted, CM is assumed.

TABLE 4. TRACK DIRECTIVE PARAMETERS

Parameter		Default (No Parameter Specified)	Description
Keyword	Phrase		
FROM	PROGRAM name octal address PROG name octal address BLOCK name octal address ABS CM octal address	Entry point (XFER address).	First instruction in tracing range.
TO	PROGRAM name octal address PROG name octal address BLOCK name octal address ABS CM octal address	Last word address of the program.	Last instruction in tracing range.
WHEN	PROGRAM pname octal address PROG pname octal address BLOCK bname octal address ABS CM octal address ABS ECS octal address Ai Register Bi ($0 \leq i \leq 7$) Xi P	A change to any location or register causes a dump.	Dump condition: a dump is taken if the contents of the specified location or register change value. P must change through a jump. Normal incrementation of P does not cause a dump.

START	Decimal number		range at which to take the first dump.
EVERY	Decimal number	1	Interval after which the dump is to be repeated.
UNTIL	Decimal number	131071	Last iteration at which the dump is to be taken.

TABLE 5. PHRASES FOR TRACK PARAMETERS

Phrase		Value Type	Description
Keyword	Value		
PROGRAM or PROG	name	Alphanumeric program block name, one to seven characters.	Program block referred to by the FROM, TO, or WHEN parameter.

BLOCK	name	Alphanumeric block name, one to seven characters. Blank common is indicated as:
		// CM
		//L ECS
			Common block referred to by the FROM, TO, or WHEN parameter.

ABS	type	Core memory type: CM ECS	Type of memory containing the absolute address specified by the FROM, TO, or WHEN parameter. If type is omitted, CM is assumed.

FEATURES

Every capsule must be a member of a group.

Capsules must be placed in libraries before they can be loaded by an executing program.

An executing program must call an FDL subroutine, specifying a group name and a capsule name, to load or unload a capsule.

Entry points and external references of capsules in the same group are automatically linked or delinked when the capsule is loaded or unloaded by the FDL subroutine.

Statically loaded code can interface with dynamically loaded code through the use of PASSLOC tables.

A capsule can be loaded statically as input to basic loads, overlay generation loads, segment generation loads, and all other situations where standard relocatables are allowed.

GENERATING CAPSULES

GROUP statement format:

GROUP(groupname)

groupname — A name of one to seven characters. Any valid program name acceptable to the loader can be used.

CAPSULE statement format:

CAPSULE(pname$_1$, . . . ,pname$_n$)

pname$_i$ — The names of programs from the loadfile, listed in order, that signal the start of a new capsule. These programs must be loaded by either LOAD or SLOAD statements.

LOADING AND UNLOADING CAPSULES

The calling sequence for loading capsules is:

FDL.LDC - LOAD CAPSULE.

ENTRY (X1) = 42/GROUP NAME, 18/SIZE ESTIMATE.
(X2) = CAPSULE NAME.
(X3) = LIBRARY LIST ADDRESS.
(X4) = PASSLOC/ENTRY LIST ADDRESS.

EXIT (B1) = 1.
(X6) = ERROR CODE:
0 = NO ERROR.
1 = BAD LIBRARY LIST.
2 = UNKNOWN GROUP NAME.
3 = UNKNOWN CAPSULE NAME.
4 = BAD CAPSULE FORMAT.
5 = BAD PASSLOC/ENTRY FORMAT.
6 = CAPSULE ALREADY IN MEMORY.
7 = CAPSULE / OVCAP CONFUSION.

SAVES A0, X0, B2, B3, X5.

CALLS CMM.ALF, CMM.CSF, CMM.FRF, CMM.SLF, SYS=.

GROUP NAME	The name of the capsule group, left-justified with zero fill.
SIZE ESTIMATE	The number of capsules in the capsule group; zero if not known.
CAPSULE NAME	The name of the capsule, left-justified with zero fill.
LIBRARY LIST ADDRESS	The address of a list of libraries.
PASSLOC/ENTRY LIST ADDRESS	The address of a list of PASSLOC/ENTRY table addresses. The list must be in ascending display code order.

The calling sequence for unloading capsules is:

FDL.ULC - UNLOAD CAPSULE.

ENTRY (X1) = 42/GROUP NAME, 18/IGNORED.
(X2) = CAPSULE NAME.
(X4) = PASSLOC/ENTRY LIST ADDRESS.

EXIT (B1) = 1.
(X6) = ERROR CODE:
0 = NO ERROR.
2 = UNKNOWN GROUP NAME.
3 = UNKNOWN CAPSULE NAME.
4 = BAD CAPSULE FORMAT.
5 = BAD PASSLOC/ENTRY FORMAT.

SAVES A0, X0, B2, B3, X5.

CALLS CMM.FRF.

GROUP NAME	The name of the capsule group, left-justified with zero fill.
CAPSULE NAME	The name of the capsule, left-justified with zero fill.
PASSLOC/ENTRY LIST ADDRESS	The address of a list of PASSLOC/ENTRY table addresses.

FREEING UNUSED MEMORY

The calling sequence for compacting group directories:

FDL.CGD - COMPACT GROUP DIRECTORY

ENTRY NONE.

EXIT (B1) = 1.

SAVES A0, X0, B2, B3, X5.

CALLS CMM.CSF, CMM.FRF, CMM.SLF, MVE=.

The calling sequence for unloading group directories:

FDL.UGD - UNLOAD GROUP DIRECTORY.

ENTRY (X1) = 42/GROUPNAME,18/IGNORED.

EXIT (B1) = 1.
(X6) = ERROR CODE:
0 - NO ERROR.
2 - UNKNOWN GROUP NAME.
10B - GROUP HAS CAPSULES.

SAVES A0, X0, B2, B3, X5.

CALLS CMM.FRF.

GROUP NAME The name of the capsule group, left-justified with zero fill.

LINKING AND DELINKING CAPSULES

The calling sequence for linking and relocating capsules is:

RJ = XFDL.REL

ENTRY (X2) = 42/0, 18/FWA.
(X4) = PASSLOC/ENTRY LIST ADDRESS.

EXIT (B1) = 1.
(X6) = ERROR CODE:
0 =NO ERROR.
4 =BAD CAPSULE FORMAT.
5 =BAD PASSLOC/ENTRY FORMAT.

SAVES A0, X0, B2, B3, X5.

FWA The address of the first word of the capsule.

PASSLOC/ ENTRY LIST ADDRESS The address of a list of PASSLOC/ENTRY table addresses. The list contains the address of a PASSLOC/ENTRY table in the lower 18 bits of each word. The list is terminated by a zero word.

The calling sequence for delinking capsules is:

RJ = XFDL.UNR

ENTRY (X2) = 42/0, 18/FWA.
(X4) = PASSLOC/ENTRY LIST ADDRESS.

EXIT (B1) = 1.
(X6) = ERROR CODE:
0 =NO ERROR.
4 =BAD CAPSULE FORMAT.
5 =BAD PASSLOC/ENTRY FORMAT.

SAVES	A0, X0, B2, B3, X5.
FWA	The address of the first word of the capsule.
PASSLOC/ ENTRY LIST ADDRESS	The address of a list of PASSLOC/ENTRY table addresses. The list contains the address of a PASSLOC/ENTRY table in the lower 18 bits of each word. The list is terminated by a zero word.

OVCAPS

OVCAPs have the same features as capsules except that they cannot be loaded by the basic loader in a relocatable load sequence.

OVCAP directive formats:

```
OVCAP.
```

```
OVCAP(lfn)
```

lfn — File name on which the OVCAP is to be written; must be the same file on which the (0,0) overlay is written.

If no file name is specified on either the NOGO statement or the OVCAP directive, the binary is written to file ABS.

The calling sequence for loading OVCAPs is:

FDL.LOC - LOAD OVERLAY-CAPSULE.

ENTRY (X1) = 42/GROUP NAME, 18/SIZE ESTIMATE.
(X2) = OVERLAY-CAPSULE NAME.
(X3) = LIBRARY LIST ADDRESS.
(X4) = PASSLOC/ENTRY LIST ADDRESS.

EXIT (B1) = 1.
(X6) = ERROR CODE:
0 = NO ERROR.
1 = BAD LIBRARY LIST.
2 = UNKNOWN GROUP NAME.
3 = UNKNOWN OVERLAY-CAPSULE NAME.
4 = BAD OVERLAY-CAPSULE FORMAT.
5 = BAD PASSLOC/ENTRY FORMAT.
6 = OVERLAY-CAPSULE ALREADY IN MEMORY.
7 = CAPSULE / OVCAP CONFUSION.

SAVES A0, X0, B2, B3, X5.

CALLS FDL RESIDENT AND FOL RESIDENT.

GROUP NAME	The name of the OVCAP group, left-justified with zero fill.
SIZE ESTIMATE	The number of members in the OVCAP group; zero if not known.
OVERLAY-CAPSULE NAME	The name of the OVCAP, left-justified with zero fill.
LIBRARY LIST ADDRESS	The address of a list of libraries.
PASSLOC/ENTRY LIST ADDRESS	The address of a list of PASSLOC/ENTRY table addresses.

The calling sequence for unloading OVCAPs is:

FDL.UOC - UNLOAD OVERLAY-CAPSULE.

ENTRY (X2) = OVERLAY-CAPSULE NAME.
(X4) = PASSLOC/ENTRY LIST ADDRESS.

EXIT (B1) = 1.
(X6) = ERROR CODE:
0 = NO ERROR.
3 = UNKNOWN OVERLAY-CAPSULE NAME.
4 = BAD OVERLAY-CAPSULE FORMAT.
5 = BAD PASSLOC/ENTRY FORMAT.

SAVES A0, X0, B2, X3, X5.

CALLS FDL RESIDENT.

)VERLAY- :APSULE NAME	The name of the OVCAP, left-justified with zero fill.
'ASSLOC/ENTRY .IST ADDRESS	The address of a list of PASSLOC/ENTRY table addresses.

:NTRY TABLE GENERATION

:NTHDR MACRO

Location	Operation	Variable Subfields
label	ENTHDR	

label An optional label.

:NT MACRO

Location	Operation	Variable Subfields
label	ENT	eptname,addr

label An optional label.

eptname Entry point name.

addr Optional; entry point address. If addr is not specified, then Yeptname will be used.

BINARY TABLE FORMATS

All fields designated res in this section are reserved for use by CDC.

RELOCATABLE PROGRAM

PRFX table

LDSET table (optional)

PTEXT table

PIDL table

ENTR, TEXT, REPL, FILL, LINK, XTEXT, SYMBOL, LINE, PTEXT, XREPL, XFILL, and XLINK tables

XFER table (optional)

ABSOLUTE CENTRAL PROCESSOR PROGRAM

PRFX table

ASCM, EASCM, ACPM, or EACPM table

PERIPHERAL PROCESSOR PROGRAM

PRFX table

6PPM table

SYSTEM TEXT OVERLAY

PRFX table

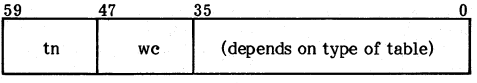
ASCM table with $\ell_1=\ell_2=1$ and origin=entry=0

CAPSULE

PRFX table

CAPSULE table

HEADER WORD



Binary number designating the type of table:

tn (octal)	Mnemonic	Table Type
aabb†	6PPM	6000 PPU program or overlay; bits 59 through 42 contain three alphanumeric characters in display code
3400	PIDL	Program identification and length
3500	PTEXT	Partial-word relocatable text
3600	ENTR	Entry point definitions
3700	XTEXT	Extended relocatable text
4000	TEXT	Relocatable text
4100	XFILL	Extended relocation fill
4200	FILL	Relocation fill
4300	REPL	Replication of text
4400	LINK	External reference linkage
4500	XLINK	Extended external reference linkage
4600	XFER	Transfer point
4700	XREPL	Extended replication of text
5000†	ASCM	Absolute CPU program or overlay
5100†	EASCM	Absolute CPU program with multiple entry points
5300†	ACPM	Absolute program or overlay with ECS data
5400†	EACPM	Absolute program or overlay with ECS data and multiple entry points
5600	SYMBOL	Symbol information
5700	LINE	Line number information and object code addresses
6000†	CAPSULE	Relocatable capsule or overlay-capsule (OVCAP)
6600 & 6700		Reserved for installation use
7000	LDSET	Object directive
7700	PRFX	Prefix

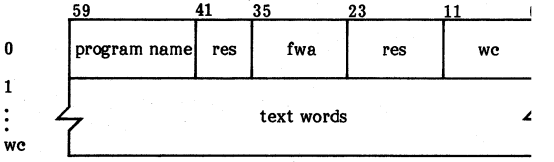
---

†The header word varies from the general format by not containing a word count. This table must be the last or only table of its program or overlay and is terminated by an end-of-record.

All other tn values 3300 through 7777 are reserved tc CDC.

wc Number of 60-bit words in the table, not counting th header word.

6PPM TABL

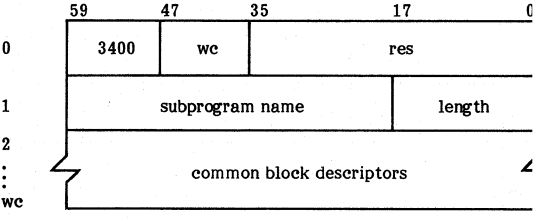


program name Three letters and/or digits in display code.

fwa Address of the byte in PPU memory into whicl the first byte of the table header word is to be loaded; the first text byte is loaded at PPU memory address fwa+5.

wc Number of 60-bit text words; the number of 12-bit bytes is five times wc.

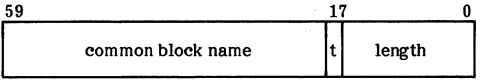
PIDL TABLE (3400



subprogram name Same as in the PRFX table. A subprogram name can be the same as a common block name and/or an entry point name without confusion.

length Number of 60-bit words in the subprogram's local central memory block; if this is zero, the block length is determined by the largest address into which subsequent tables load text.

)escriptor:



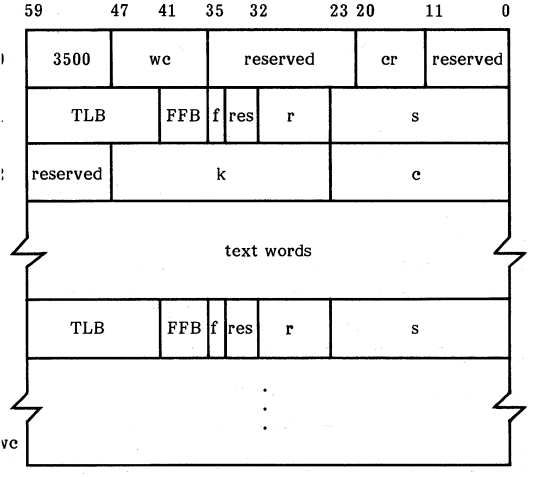
Type of block:

0 = central memory

1 = ECS

ength — If t is 0, block size is length 60-bit words. If t is 1, the block size is 8 x length 60-bit words.

'TEXT TABLE (3500)



cr — Conditional relocation base designator. The PTEXT table is ignored if cr refers to a common block that was first declared by an earlier subprogram.

TLB — Text length in bits.

FFB First bit in first word to receive first bit of text words.

f =0 text words follow immediately

=1 next word is replication indicator, followed immediately by text words

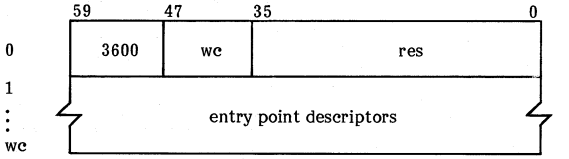
r Relocation base designator.

s Relative first word address; cannot exceed 377777 octal.

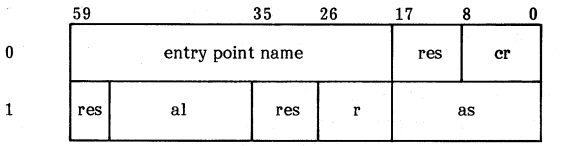
k Increment in bits between replicated copies of text words. If k=0, loader assumes TLB.

c Replication count. If c=0, loader assumes 1.

ENTR TABLE (3600)



Descriptor:



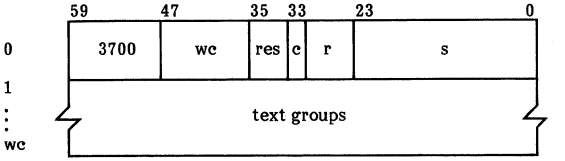
cr Relocation base indicator, in the same format as r. The descriptor is ignored if cr refers to a common block that was first declared by an earlier subprogram.

al Relative value of the entry point if r refers to an ECS block.

r Relocation base designator.

as Relative value of the entry point if r does not refer to an ECS block.

XTEXT TABLE (3700)



c Conditional Flag:

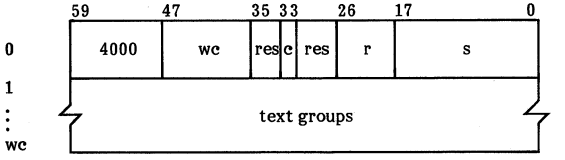
0 = Load text unconditionally

1 = Ignore this XTEXT table if r refers to a common block that was first declared by an earlier subprogram

r Relocation base designator.

s Relative first word address; cannot exceed 7777777_8.

TEXT TABLE (4000)



c Conditional flag:

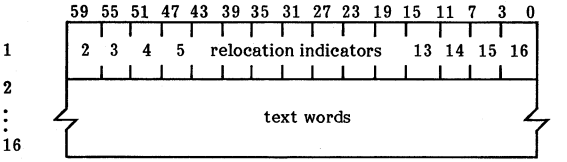
0 = Load text unconditionally

1 = Ignore this TEXT table if r refers to a common block that was first declared by an earlier subprogram

r Relocation base designator.

s Relative first word address; cannot exceed 377777_8.

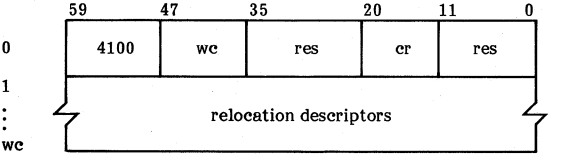
Text group:



Relocation bytes:

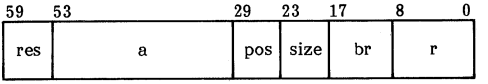
000x	No relocation
10xx	Upper address, positive relocation
11xx	Upper address, negative relocation
010x	Middle address, positive relocation
011x	Middle address, negative relocation
1x10	Lower address, positive relocation
1x11	Lower address, negative relocation
0010	Lower address, positive relocation
0011	Lower address, negative relocation

XFILL TABLE (4100)



cr The conditional relocation base designator. The XFILL table is ignored if cr refers to a common block that was first declared by an earlier subprogram.

Descriptor:



a — Relative address of the text word; can be as large as 7777777_8.

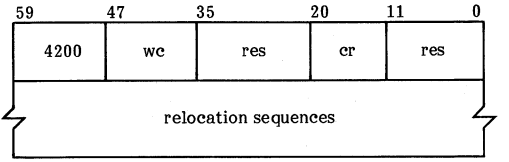
pos — Bit position of the low-order bit of the address field in the text word.

size — Address field length in bits.

br — Base address designator for the relocation quantity.

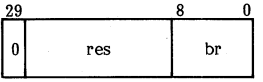
r — Relocation base designator for the text word address; r cannot be 2 (negative program relocation) and cannot refer to a blank common block.

FILL TABLE (4200)



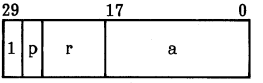
cr — Conditional relocation base designator. The FILL table is ignored if cr refers to a common block that was first declared by an earlier subprogram.

Header byte:



br — Relocation base designator.

Trailer byte:



p The address field to be relocated, designated as follows:

00 = Lower address (bits 17 through 0)

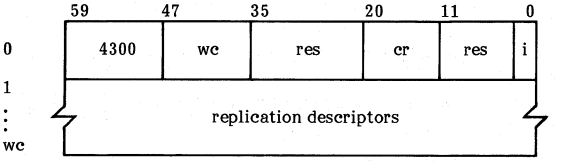
01 = Middle address (bits 32 through 15)

10 = Upper address (bits 47 through 30)

r Relocation base designator.

a Relative address; cannot exceed 377777_8.

REPL TABLE (4300)



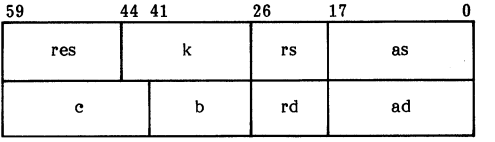
cr Conditional relocation base designator. The REPL table is ignored if cr refers to a common block that was first declared by an earlier subprogram.

i Immediate flag:

0 = The REPL table is saved when encountered and processed when the end of the current load file or library is reached

1 = The REPL table is processed immediately when encountered

Descriptor:



k — Destination address increment: k is added to destination address (D) after each copy. If k is 0, the loader uses block size (b) for increment size.

rs — Relocation base designator for the source address (S).

as — Relative source address; cannot exceed 377777_8.

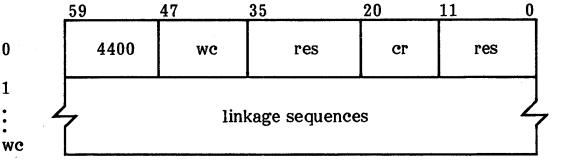
c — Count; the number of times the block is copied. If c is 0 or 1, the loader makes one copy.

b — Block size (number of words to be copied) in 60-bit words. If b is 0 or 1, the loader copies one word.

rd — Relocation base designator for destination address D.

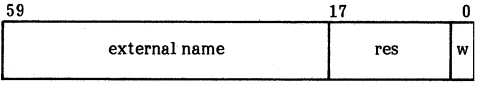
ad — Relative destination address; cannot exceed 377777_8.

LINK TABLE (4400)



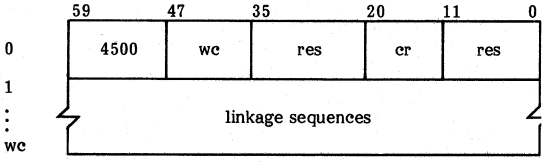
cr — Conditional relocation base designator. The LINK table is ignored if cr refers to a common block that was first declared by an earlier subprogram.

Header byte:



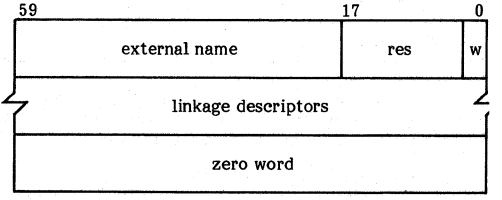
w Weak external flag:
0 = strong
1 = weak

XLINK TABLE (4500)



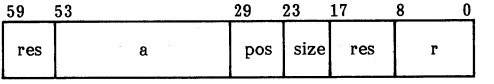
cr Conditional relocation base designator. The XLINK table is ignored if cr refers to a common block that was first declared by an earlier subprogram.

Linkage sequence:



w Weak external flag:
0 = strong
1 = weak

Linkage descriptor:



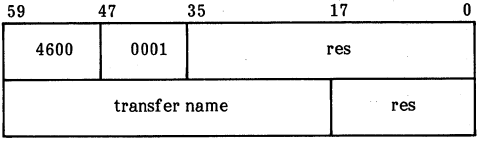
a — Relative address of the text word; can be as large as 7777777_8.

pos — Bit position of the low-order bit of the address field in the text word.

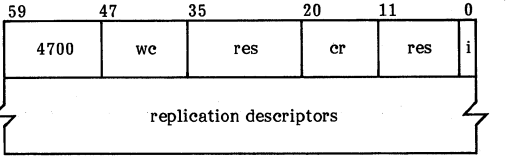
size — Address field length in bits.

r — Relocation base designator for the text word address; r cannot be 2 (negative program relocation) and cannot refer to a blank common block.

XFER TABLE (4600)



XREPL TABLE (4700)



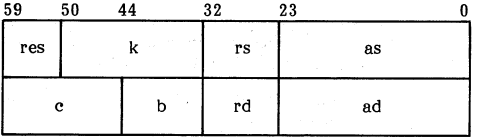
cr — Conditional relocation base designator. The XREPL table is ignored if cr refers to a common block that was first declared by an earlier subprogram.

i — Immediate flag:

0 = The XREPL table is saved when encountered and processed when the end of the current load file or library is reached

1 = The XREPL table is processed immediately when encountered

Replication descriptor:



k — Destination address increment: k is added to destination address (D) after each copy. If k is 0, the loader uses block size (b) for increment size.

rs — Relocation base designator for the source address (S).

as — Relative source address; cannot exceed 7777777_8.

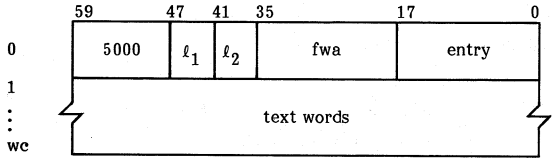
c — Count; the number of times the block is copied. If c is 0 or 1, the loader makes one copy. Cannot exceed 32767.

b — Block size (number of words to be copied) in 60-bit words. If b is 0 or 1, the loader copies one word. Cannot exceed 4095.

rd — Relocation base designator for destination address D.

ad — Relative destination address; cannot exceed 7777777_8.

ASCM TABLE (5000)



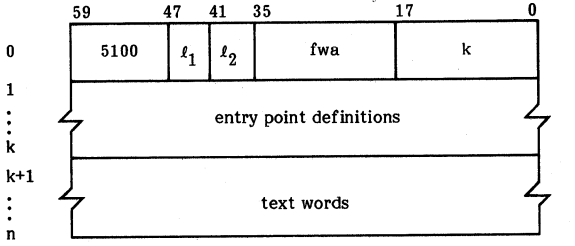
ℓ_1 Primary overlay level number.

ℓ_2 Secondary overlay level number.

fwa Address of the word in central memory into which the table header word is to be loaded; the first text word is loaded at central memory address fwa+1.

entry The address at which execution of the program or overlay is to begin.

EASCM TABLE (5100)

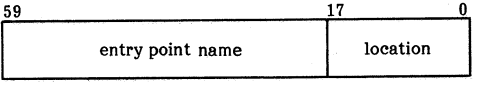


ℓ_1,ℓ_2 Same as in the ASCM table. If ℓ_1=ℓ_2=0, the same restrictions on fwa apply.

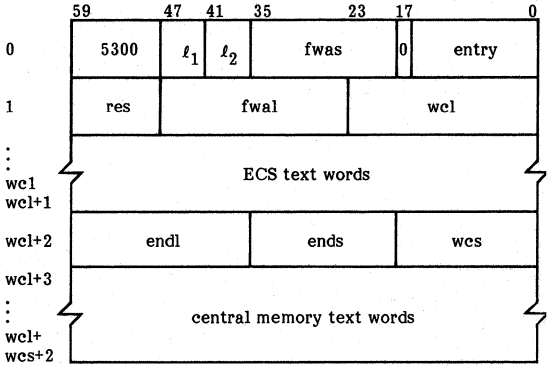
fwa Address of the word in central memory into which the table header word is to be loaded. The first entry point definition is loaded at central memory address fwa+1, and the first text word at fwa+k+1.

k Number of entry point definitions.

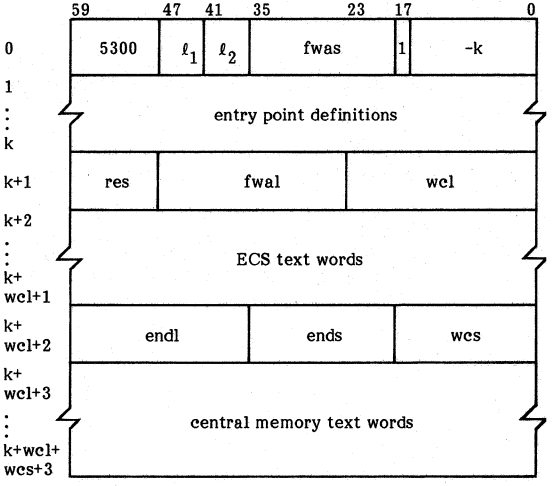
Entry point definition:



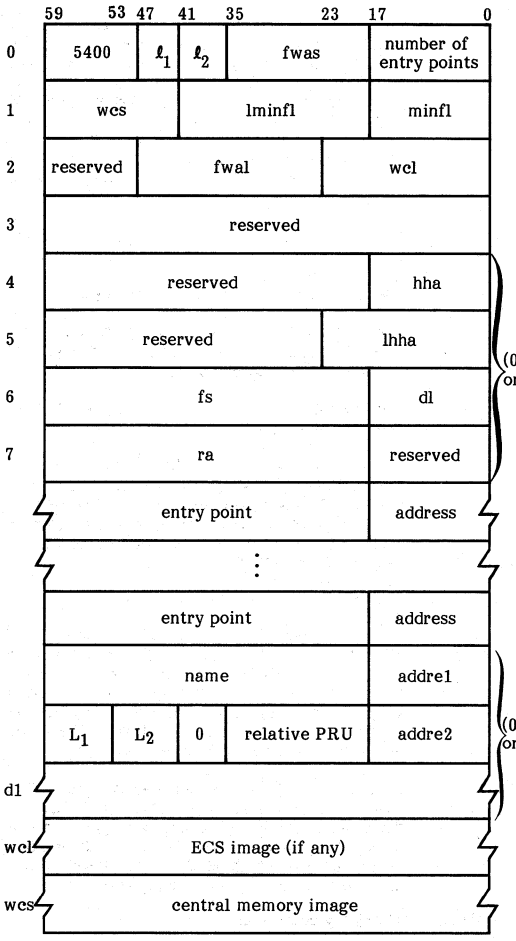
Format 1 (one unnamed entry point):



Format 2 (one or more named entry points):



$\ell_1\ell_2$	Overlay level number.
fwas	Address of the word in central memory into which the table header word is to be loaded. For format 1, the first central memory text word is loaded at address fwas+1. For format 2, the first entry point definition is loaded at address fwas+1, and the first central memory text word at fwas+k+1. If $\ell_1+\ell_2=0$, fwas must not be less than 100_8.
entry	The address at which execution of the program or overlay is to begin.
k	Number of entry point definitions. The complement of k is stored in the table to distinguish format 2 from format 1.
entry point definitions	Same as in the EASCM table.
fwal	Address of the word in ECS into which the first ECS text word (if any) is to be loaded.
wcl	Number of ECS text words; can be zero.
endl	The greatest address+1 of the fixed ECS area used by this program. If endl is 0, endl=fwal+wcl is assumed.
ends	The greatest address+1 of the fixed central memory area used by this program. If ends is 0, ends=fwas+wcs+1 is assumed.
wcs	Number of central memory text words. If wcs is 0, the central memory text words comprise all that remains of the ACPM table, which is terminated by an end-of-record.


59
53
47
41
35
23
17
0
0
5400
ℓ_1
ℓ_2
fwas
number of entry points
1
wcs
lminfl
minfl
2
reserved
fwal
wcl
3
reserved
4
reserved
hha
5
reserved
lhha
6
fs
dl
7
ra
reserved
(0,0) only
entry point
address
entry point
address
name
addre1
L_1
L_2
0
relative PRU
addre2
(0,0) only
d1
wcl
ECS image (if any)
wcs
central memory image

ℓ_1,ℓ_2	Overlay level. Words 4 through 7 of the table header are present only for level (0,0) overlays.
fwas	The address of the word in central memory into which word 0 of the table header is to be loaded. The entry point list, the FOL directory, and then the central memory image are to be loaded immediately following the header. If $\ell_1=\ell_2=0$, fwa must be equal to 100.
fwal	The address of the word in ECS, if any, into which the first word of the ECS image is to be loaded.
wcs	The number of words in the central memory image.
wcl	The number of words in the ECS image.
minfl	The minimum central memory field length needed to execute the overlay; equivalent to the lwa+1 of the overlay.
lminfl	The minimum ECS field length needed to execute the overlay.
hha	The highest high address for central memory; the minimum field length needed to execute any legal combination of overlays generated as part of this overlay structure. This number is derived from the values of minfl specified for each of the overlays.
lhha	The highest high address for ECS.
entry points	If the overlay was generated because of OVERLAY directives in the object stream, or because of a NOGO control statement specifying a file name only, the overlay will have a single entry point. Its name will be the same as that of the overlay and its address will be that of the last transfer address encountered. If the overlay was generated because of a NOGO control statement (and no OVERLAY directives) specifying a file name and one or more entry points, the overlay will contain those entry points named on the NOGO statement with their respective addresses.
address	The entry point address.
dl	The fast overlay directory length (FOL). Two words are used for each entry.

fs — The file specification entry set into 5400 table when a FOL overlay structure (0,0) overlay is loaded.

ra — The random address set in the 5400 table when a FOL overlay structure (0,0) overlay is loaded.

The following fields are repeated for each overlay on the FOL directory:

name — The name of the overlay or OVCAP.

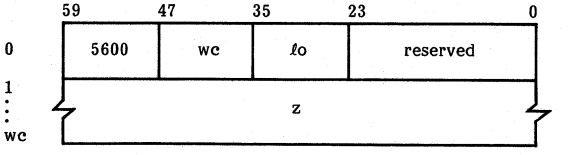
addre1 — The address of the first word of the overlay. Zero if an OVCAP.

L_1 L_2 — Overlay levels of the remaining overlay in the FOL directory.

relative PRU — The PRU address relative to the (0,0) overlay.

addre2 — The address of the last word of the overlay plus 1. Length if an OVCAP.

SYMBOL TABLE (5600)

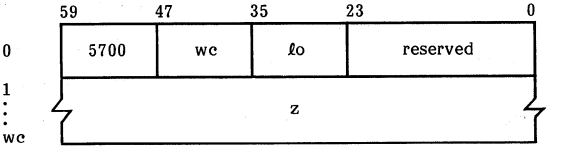


ℓo — Language ordinal, for example:

2=FTN4
4=FTN5
8=BASIC

z — Symbol table entry. Each entry is 2 words long. The content of this field is irrelevant to the loader.

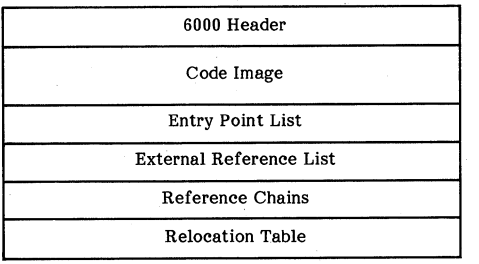
LINE NUMBER TABLE (5700)



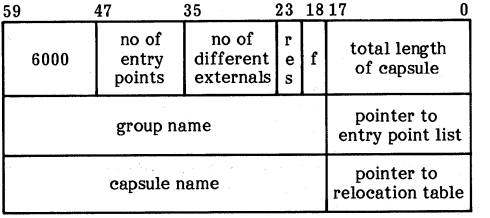
ℓo Language ordinal.

z Line number table entry. The content of this field is irrelevant to the loader.

CAPSULE TABLE (6000)

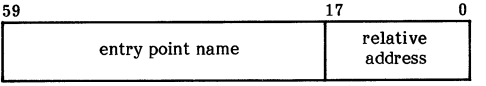


Header:

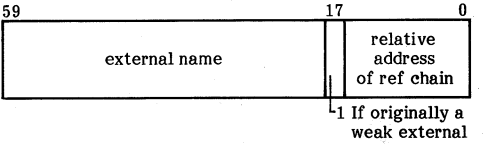


f = 0 - Capsule
1 - OVCAP

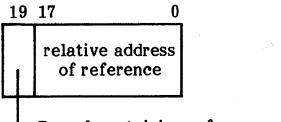
Entry point list:



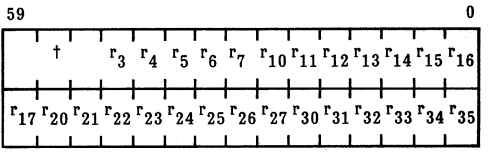
External reference list:



Reference chain:

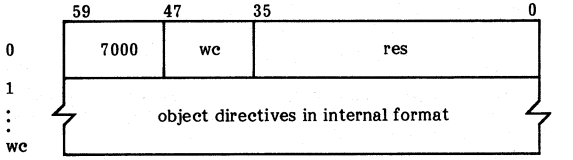


Relocation table:



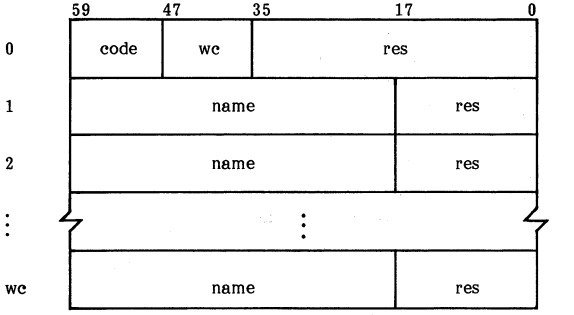
†Zero, indicating that the three header words are not to be relocated.

LDSET TABLE (7000)



The LDSET table options are listed below.

LIB (0010)



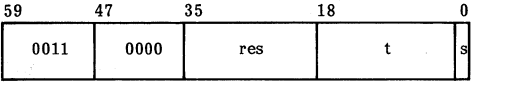
code Identifies the table type.

name The name passed by the LDSET option.

wc Number of names in the table.

MAP (0011)

Format 1:



s Significance of the t flag:

0 = Ignore t and write the default map type

1 = Write the map type specified by t

t Map type octal code. These options are equivalent to LDSET(MAP=) options. Any combination of the following can be specified:

1 (bit 1) = Statistics (S)

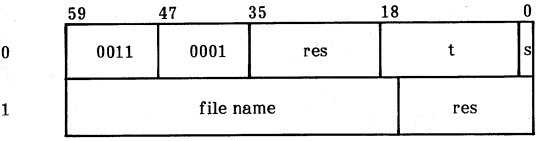
2 (bit 2) = Blocks (B)

4 (bit 3) = Entry points (E)

10 (bit 4) = Entry point cross-references (X)

The map is written on file OUTPUT.

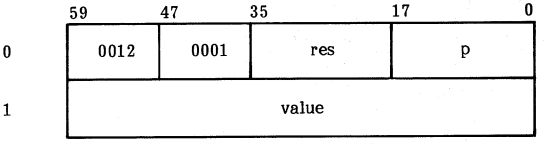
Format 2:



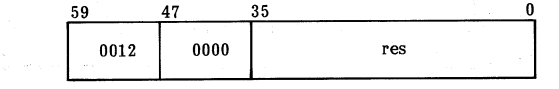
Format 2 has the same interpretation of s and t as format 1. In addition, the map is written on the specified file.

PRESET/PRESETA (0012)

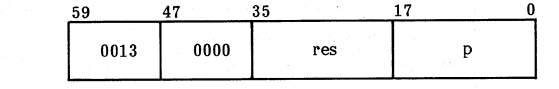
Format 1:



Format 2:

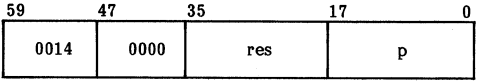


ERR (0013)



p	Severity
0	ALL
1	FATAL
2	NONE

REWIND/NOREWIN (0014)



p	Position
0	Rewind
1	No rewind

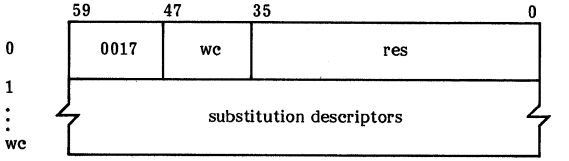
USEP (0015)

Same format as the LIB (0010) option.

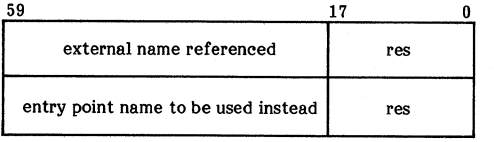
USE (0016)

Same format as the LIB (0010) option.

SUBST (0017)



Substitution descriptor:



OMIT (0020)

Same format as the LIB (0010) option.

EPT (0025)

Same format as the LIB (0010) option.

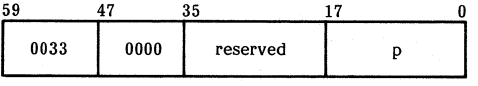
NOEPT (0026)

Same format as the LIB (0010) option.

COMMON (0032)

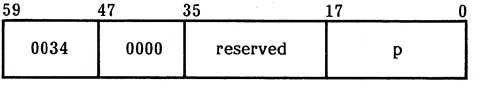
Same format as the LIB (0010) option.

PD (0033)



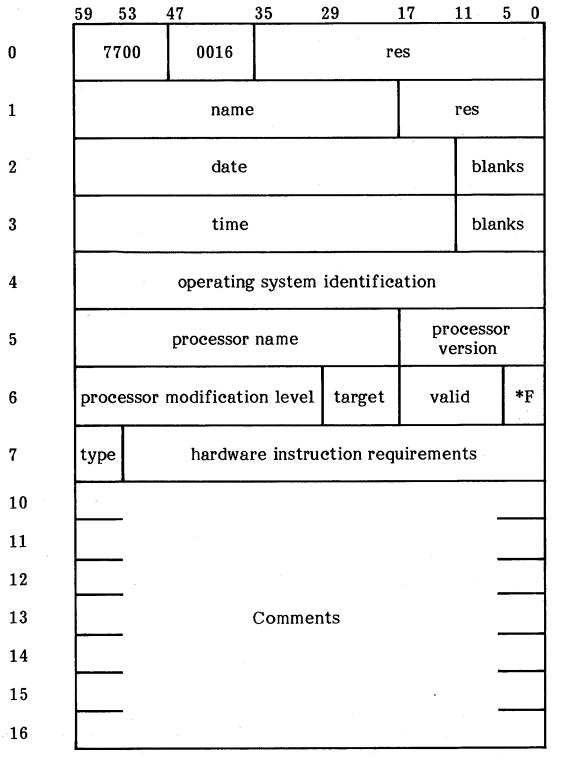
p Print density for load map (6 or 8 lines/inch).

PS(0034)



p Page size for load map (lines/page). Must be at least 10.

PRFX TABLE (7700)



CONTROL DATA CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO.
MINNEAPOLIS, MINN. 55440

SALES OFFICES AND SERVICE CENTERS
IN MAJOR CITIES THROUGHOUT THE WORLD