PERKIN-ELMER

OS/32 LINK

Reference Manual

48-005 F01 R02

The information in this document is subject to change without notice and should not be construed as a commitment by The Perkin-Elmer Corporation. The Perkin-Elmer Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license, and it can be used or copied only in a manner permitted by that license. Any copy of the described software must include the Perkin-Elmer copyright notice. Title to and ownership of the described software and any copies thereof shall remain in The Perkin-Elmer Corporation.

The Perkin-Elmer Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Perkin-Elmer.

The Perkin-Elmer Corporation, Data Systems Group, 2 Crescent Place, Oceanport, New Jersey 07757

©1980, 1983, 1984 by The Perkin-Elmer Corporation

Printed in the United States of America

TABLE OF CONTENTS

CHAPTERS (Continued)

CHAPTERS (Continued)

APPENDIXES

FIGURES

TABLES

PREFACE

This manual describes the Perkin-Elmer linkage editor, OS/32 Link, which provides the user with the ability to link one or more object modules to produce an executable image. An image can be a task, a partial image or an operating system. This manual is intended for all users who are developing programs for execution on Perkin-Elmer 32-bit computers using the OS/32 operating system. The user should be familiar with the Perkin-Elmer OS/32 Multi-Terminal Monitor (MTM) if Link is to be used in an MTM environment (see the OS/32 Multi-Terminal Monitor (MTM) Reference Manual).

Chapter 1 provides an overview of the features of Link. Chapter 2 describes how to build, load and start the linkage editor. Chapter 3 lists and describes the active, passive and environment Link commands. Chapter 4 provides examples of Link command sequences. Chapter 5 introduces and explains virtual task management (VTM). Appendix A is the Link command summary. Appendix B is the Link message summary. Appendix C is the VTM message summary. Appendix D explains the format of an object module that is compatible with Link.

Revision F01 R02 of this manual includes changes to the VTM documentation in Chapter 5 and Appendix C. Appendix D has also been changed to document additional loader items.

This manual is intended for use with the R01-02 version of OS/32 Link and the OS/32 R07.2 software release.

For information on the contents of all Perkin-Elmer 32-bit manuals, see the 32-Bit Systems User Documentation Summary.

CHAPTER 1
OS/32 LINK

1.1 INTRODUCTION

Perkin-Elmer OS/32 Link provides the user with the ability to link one or more object modules to produce a task image or partial image that can be loaded via the OS/32 LOAD command.

Link can also build an operating system image from the object module produced by the Perkin-Elmer OS/32 Library Loader or SYSGEN/32. The resulting image can be loaded into memory using the Perkin-Elmer OS/32 Bootstrap Loader or Loader Storage Unit (LSU).

This release of Link includes the DEBUG/32 tables (DTABLES) task option. This option allows Link to separate symbolic debug data from the object code and build this data into the tables required by DEBUG/32. This release also includes the virtual task manager (VTM). VTM provides a user-transparent virtual memory capability that allows some user tasks (u-tasks) consisting of up to 16Mb of code and data to execute in as little as 128kb of memory.

OS/32 Link can be used with both the Perkin-Elmer Uniprocessor System and the Perkin-Elmer Multiprocessor System (Model 3200MPS). The multiprocessor system consists of one central processing unit (CPU) and up to nine auxiliary processing units (APUs). In a multiprocessor system, the operating system defines a set of logical processing units (LPUs) that are used to direct tasks to execution queues. Link assigns the initial LPU for each task. Link also sets APU control or queue mapping privileges when building a task, and can optionally list comments embedded in the object file. See the Link DCMD and OPTIONS commands. Also see the Perkin-Elmer Model 3200MPS Overview Manual for more information on using the Model 3200MPS System.

1.2 IMAGE FILE FORMAT

Link allocates an image file on disk and builds an image into this file or builds the image into an already existing file. The format of the image file for a task is shown in Figure 1-1.

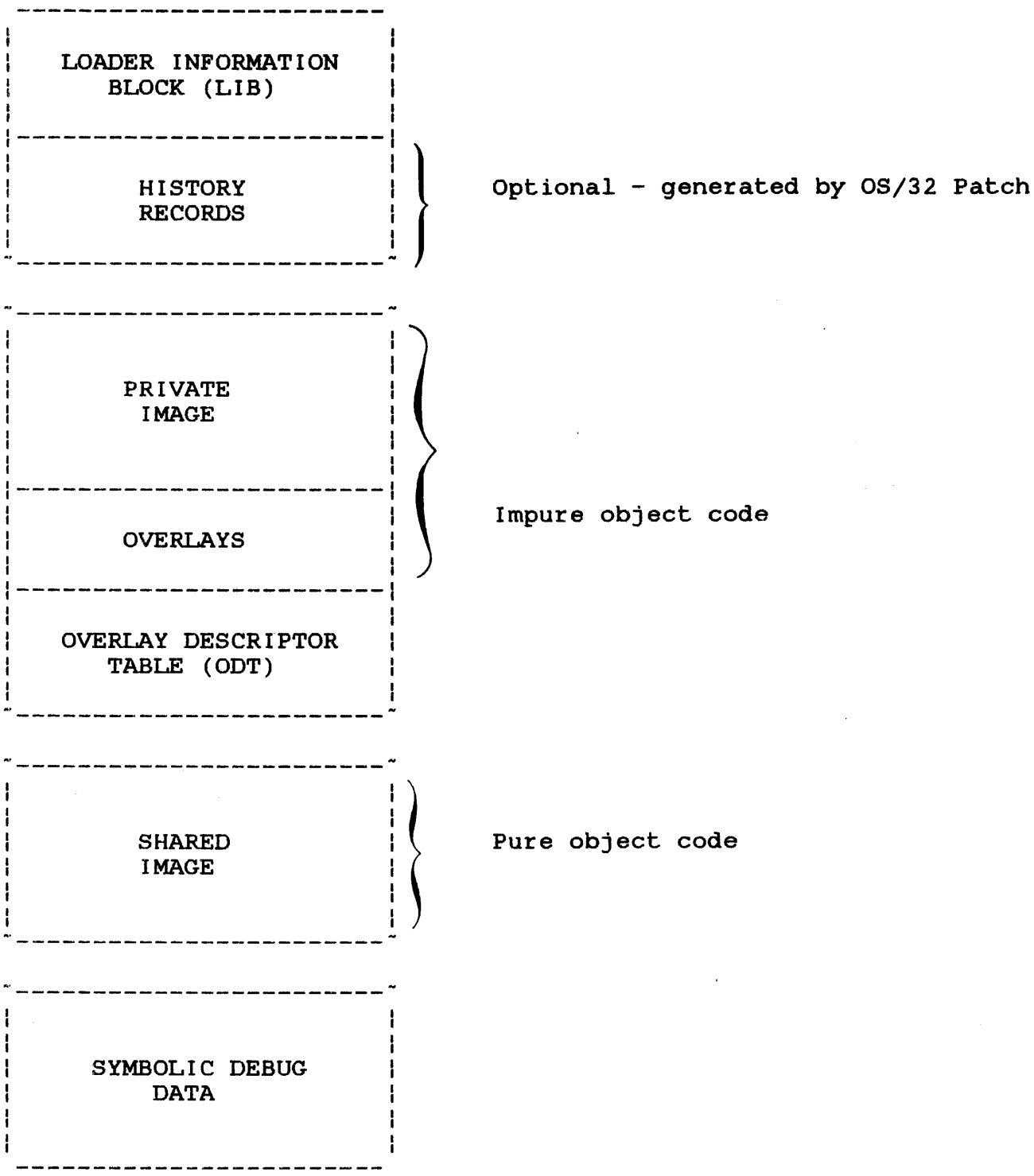


Figure 1-1 Task Image File Format

CHAPTER 2
BUILDING AND STARTING LINK

2.1 BUILDING LINK

If the Perkin-Elmer supplied ready-to-execute version of Link is to be used, no build is necessary. However, if a new version of Link is to be built, this sequence of commands builds Link as a segmented task using the Perkin-Elmer supplied version of Link:

```
ES TASK
OPTION ACPRIVILEGE,SYSSPACE=XFFFF
OPTION SEGMENTED,WORK=(X8000,XC000)
INCLUDE LINK
MAP CON:,ALPHABETIC,ADDRESS,XREF
RESOLVE HELPR072.SEG/S
BUILD LINK
END
```

The reserved workspace must be a minimum of 8kb. The more workspace allocated, the more paging to and from disk occurs. The amount of workspace specified can be overridden when Link is loaded.

2.2 LOADING LINK

Before Link can be loaded into main storage, it must be built as a task image.

2.2.1 Loading Link from the System Console

The following OS/32 LOAD command loads Link from the system console:

Format:

```
LOAD taskid [,fd] [,workspace]
```

Parameters:

taskid	is a 1- to 8-character alphanumeric string specifying the name of the task after it is loaded into main memory.
fd	is the file descriptor of the file containing the linkage editor image to be loaded into main memory. If this parameter is omitted, the default is taskid.TSK.

NOTE

The Perkin-Elmer supplied ready-to-execute version of Link is included in a file named LINK.TSK.

workspace	is a decimal number in kb specifying the additional area to be added to the root node. This value overrides the WORK= option if specified when the image was built.

NOTE

Link requires the OS/32 Help segment (.HELPRnnn.SEG/S- where nnn specifies the OS/32 revision number) to be preloaded into memory or be available on account 0 of the default system disk volume.

2.2.2 Loading Link from a Multi-Terminal Monitor (MTM) Terminal

The following MTM command loads Link from an MTM terminal:

Format:

```
LOAD fd [,workspace]
```

Parameters:

fd	is the file descriptor of the file containing the linkage editor image to be loaded into main memory.
workspace	is a decimal number in kb specifying the additional area to be added to the root node. This value overrides the WORK= option if specified when the image was built.

2.2.3 Assigning Workspace for Link

The size of the workspace increment value given when Link is loaded will control the maximum symbol table size generated by Link as shown in the following table:

WORKSPACE INCREMENT	SYMBOL TABLE MAXIMUM
0 - 7	LINK will not run
8 - 15	32 kilobytes
16 - 31	64 kilobytes
32 - 63	96 kilobytes
64 - 95	128 kilobytes
96 - 127	256 kilobytes
128 - 255	1 megabyte
256 - or greater	4 megabytes

2.3 LINK INPUT/OUTPUT (I/O) FILES

Link requires the following I/O files:

- Object files containing the compiled source code.
- Task image file to which Link outputs the task image.
- Map file to which Link sends a listing of the establishment summary and, optionally, all external programs and their addresses.
- Log file which lists all Link commands issued and any Link generated diagnostic messages.
- Command file containing commands to Link.

The Link command file can be built by a command substitution system (CSS) procedure or built as a separate file that can be specified in the START command. If no Link command file is specified in the START command, Link accepts commands interactively from the terminal or console. The BUILD command for Link automatically allocates a file, if the file does not already exist, for the task image using the filename entered, followed by the extension corresponding to the type of image (TASK, OS, partial image) being built. The log file must be preallocated by the user. The user can optionally preallocate a map file. However, LINK will allocate the map file if it does not exist.

Table 2-1 lists the logical unit (lu) assignments that are made automatically by the Link commands.

TABLE 2-1 LOGICAL UNITS ASSIGNED BY LINK

LINK COMMAND	LOGICAL UNITS ASSIGNED	I/O FILE	ACCESS
INCLUDE/LIBRARY	1	Object	SRO
BUILD	2	Task image	SRW
MAP	3	Link map	SWO
START			
,COMMAND=	5	Link command input	SRO
	7	Link command output	SWO
,LOG=	6	Log	SWO
HELP	10	Link help File	SRO

Link also assigns lu9 as needed for the temporary paging of its symbol table.

2.4 STARTING LINK

After Link is loaded into main memory, the OS/32 or MTM START command starts execution of the Link program and specifies the command and log files or devices.

Format:

START [,COMMAND=fd_1][,LOG=fd_2]

Parameters:

COMMAND= fd_1 specifies the input file or device from which Link commands are read. If this parameter is omitted, the default is the command input device (CON:). If the command input device is interactive, all messages generated by Link are sent to it. If the command input file is batch, all Link messages are sent to the file specified by the LOG parameter.

LOG=	fd_2 specifies the output file or device which all commands entered and messages generated are written. If the command input file is batch, this parameter must be specified. If the log output device is a disk file, it must have been previously allocated.

Functional Details:

After the linkage editor is started, the following message is displayed:

```
PERKIN-ELMER OS/32 LINKAGE EDITOR 03-242 Rnn-nn
```

The revision number (Rnn) indicates the revision level of Link, and the update number (-nn) indicates the update level of Link. If the command input device is interactive, the greater than (>) symbol is then displayed as a prompt indicating that the linkage editor is ready to accept commands.

CHAPTER 3
LINK COMMANDS

3.1 INTRODUCTION

There are three types of Link commands:

- Active
- Passive
- Environment

Active commands are executed as they are entered and have an immediate effect on how the image is to be built. Passive commands are executed during the build process, at which time Link processes them, making symbol table entries, etc. Although passive commands are not executed when entered, the order in which passive commands are encountered can affect the image produced by Link. This is due to the order in which items are entered into Link's internal symbol table. Environment commands affect the link session instead of the image being built. Environment commands have no affect on the image being built, but do establish the environment.

A Compatible Link Utility is included with the OS/32 software package for conversion purposes. This utility is designed for users who have extensive Link command files built using the Link command syntax documented in the R01 version of this manual. The Compatible Link Utility allows users to continue using the existing LINK command sequences in conjunction with the new enhancements included in the R02 release of the OS/32 Link Reference Manual.

Users who elect to use the Compatible Link Utility should note that there are four Link comands documented in the R01 version of this manual that differ in format from the R02 release of the Link Reference Manual. The BUILD, ESTABLISH, OPTION and SHARED commands that are supported by the Compatible Link Utility use the formats documented in the R01 version of the OS/32 Link Reference Manual.

Table 3-1 lists all the Link commands, categorizes the type and describes the function.

TABLE 3-1 LINK COMMANDS

COMMAND	TYPE ACT	TYPE PAS	TYPE ENV	MEANING
BFILE			*	Backspaces a magnetic tape or contiguous file
BUILD	*			Starts building the image
DCMD	*			Enables execution of Link commands embedded in object modules. Enables the listing of embedded auxiliary processing unit (APU) comments to the log device in the Model 3200MPS System.
END	*			Terminates the linkage editor
ESTABLISH	*			Specifies the type of image to be built
EXTERNAL		*		Specifies the names of common block(s) to be externally visible from the partial image being built.
FFILE			*	Forward spaces a magnetic tape or contiguous file
HELP			*	Lists and describes all Link commands accepted by the current revision of Link.
INCLUDE	*			Specifies the object modules to be included in the image
LIBRARY		*		Specifies the object libraries to be searched for unresolved external references
LOCAL		*		Specifies entry points that are not to be visible from outside of the partial image being built
LOG			*	Enables logging all commands, messages, and maps to the log device
MAP		*		Generates a map when the image is built

3.3 BUILD COMMAND

The BUILD command is an active command that builds the image from the object modules specified in the INCLUDE command.

Format:

BUILD fd

Parameters:

fd — is the file descriptor that is to receive the image. If the extension is omitted, the default extensions are:

- .TSK for tasks
- .IMG for partial images
- .000 for operating systems

Functional Details:

The linkage editor attempts to allocate and assign the file specified in the BUILD command. If the file does not exist, the linkage editor allocates the file. While in the interactive mode, if an error occurs during this process or the file is not specified in the BUILD command, the following message is displayed:

ENTER FILE DESCRIPTOR FOR IMAGE>

Enter the fd of the file or device to receive the image.

If a file with the fd specified already exists, Link will overwrite it automatically, without issuing any prompts.

By default, Link allocates a contiguous file for the image. Building an image to a contiguous file is significantly faster than building an image to an indexed file.

After the task is built, the Link maps are generated if the MAP command was entered. If the MAP command was not entered, the following message is displayed:

```
MAP?>
```

Enter YES(Y) or NO(N). If YES (Y) is entered, the following four messages are displayed:

- ENTER FILE DESCRIPTOR FOR MAP>

  Enter the fd of the device or file to receive the maps.

- SORTED BY ADDRESS?>

  If YES is entered, a map with all symbols already in address order is generated.

- CROSS REFERENCE?>

  If YES is entered, a cross-reference map is generated. This map lists all symbols in alphabetical order and the names of all object modules that reference each symbol.

- SORTED ALPHABETICALLY?>

  If YES is entered, a map with all symbols in alphabetical order is generated.

  If NO is entered for all of these messages, only an establishment summary is generated. See Section 3.14.

After the BUILD command is executed, the linkage editor builds the image. To only generate a Link map without saving the task image to a file, specify NULL: as the fd to the BUILD command.

Examples:

```
BUILD COM.IMG

BUILD TASK

BUILD TASK.TSK

BUILD NULL:
```

NOTE

If Link is running in batch mode and cannot allocate the file, the build process is terminated.

3.4 DCMD COMMAND

The define command (DCMD) command is an active command that, when entered without parameters, enables execution of passive Link commands in object modules included in the image. This command, at the same time, enables listing of embedded comments to the input or log device. In programs written for a Model 3200MPS System, this command entered with parameters enables or suppresses listing of APU comments to the log device.

Format:

DCMD [{ APUCOMMENT }]
     [{ NAPUCOMMENT }]

Parameters:

APUCOMMENT	enables listing of APU comments to the log device.
NAPUCOMMENT	disables listing of APU comments to the log device. This is the default.

The DCMD command enables CAL and FORTRAN programs to contain passive Link commands that will be executed when the image is built. To embed passive Link commands in a CAL program, use the CAL DCMD pseudo-op as follows:

```
DCMD C'linkedit command'
```

NOTE

This DCMD pseudo-op is not the same as the Link DCMD command.

Example of CAL code containing embedded passive Link commands:

```
MOD     PROG
        ENTRY ENTRY
        EXTRN EXTRNA
        EXTRN EXTRNB
        EXTRN EXENTRY
        DCMD  C'OPTION FLOAT'
        DCMD  C'MAP PR:,ALPHA'
        DCMD  C'*PATCH FOR SCR 1183, 1/24/83'
        DCMD  C'*APU MODULE MOD INVOKES SVC CALLS'
        PURE
ENTRY   L     0,EXTRNA
        ST    0,EXTRNB
        BAL   13,EXENTRY
        SVC   3,0
        END
```

Embedded passive Link commands are treated as if they were part of the Link command sequence. Embedded LIBRARY commands are treated as if they were entered immediately before the BUILD command; all other embedded commands are treated as if they were entered after the INCLUDE command.

If a log device is specified in the START command, all embedded passive Link commands are output to the log device with a plus sign (+) in column 1.

The DCMD command entered without any parameters also enables listing of embedded general comments to the log device. These general comments can refer to patches applied to a particular compiler or other general comments the user does not want suppressed.

In programs written for a Model 3200MPS System, some language processors, such as CAL/32 and FORTRAN VII, generate APU information comments embedded in the object files of APU tasks. These APU comment lines always begin with an asterisk (*) and the letters APU. Listing or suppression of the APU comment lines is enabled by entering the DCMD command with the APUCOMMENT or NAPUCOMMENT parameter. If the APUCOMMENT parameter is entered, all comments, including the general comments, are displayed. If the NAPUCOMMENT parameter is entered, APU comments are suppressed, but the general comments are still displayed.

When the program above is linked, the log listing will be:

```
ES TA
INCLUDE MOD
BUILD MOD
```

If the DCMD command with no parameters is entered, the log listing will be:

```
DCMD
ES TA
INCLUDE MOD
+OPTION FLOAT
+MAP PR:, ALPHA
+*PATCH FOR SCR 1183, 1/24/83
BUILD MOD
```

If the DCMD command is entered with the APUCOMMENT parameter, the log listing will be:

```
ES TA
DCMD APUCOMMENT
INCLUDE MOD
+OPTION FLOAT
+MAP PR:, ALPHA
+*'PATCH FOR SCR 1183, 1/24/83'
+*APU 'MODULE MOD INVOKES SVC CALLS'
BUILD MOD
```

Only passive Link commands can be embedded in CAL object modules. If active or environment commands are embedded in CAL object modules, they will be ignored and this message will be output:

```
COMMAND NOT PERMITTED
```

Application users in a uniprocessor system can use the DCMD command with its parameters for developing a Model 3200MPS System.

If this command is not entered, all embedded passive Link commands are executed. To turn this feature off, use the NDCMD command explained in Section 3.15.

3.5 END COMMAND

The END command is an active command that terminates the linkage editor.

Format:

END

Functional Details:

While Link is in the interactive mode, if a Link command sequence contains at least one INCLUDE command and an END command is entered before a BUILD command is entered, the following message is displayed:

BUILD IMAGE FROM PREVIOUS INPUT?>

Enter YES if the image is to be built. Enter NO if no image is to be built and the task is to be terminated. See Table 3-2 for the meaning of Link end of task codes.

TABLE 3-2 LINK END OF TASK CODES

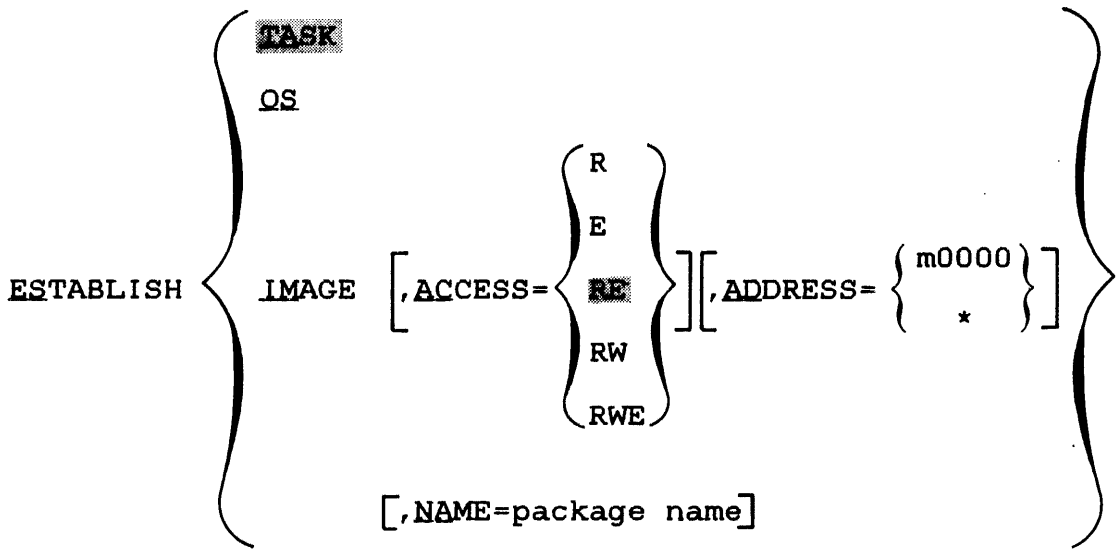
END OF TASK CODE	MEANING
0	Terminated normally
1	An error occurred that did not affect the building of the image.
2	An error occurred that affected the building of the image.
3	A severe error occurred that caused the linkage editor to abort.

3.6 ESTABLISH COMMAND

The ESTABLISH command is an active command that specifies the type of image to be built and provides a package name to a multiple segment image. The three types of images that can be built are:

- task,
- operating system, and
- partial image

Format:



Parameters:

TASK	specifies that a task image is to be built. If the ESTABLISH command or the parameters specifying the type of image are omitted, TASK is the default.
OS	specifies that an operating system image is to be built.

3.8 FFILE COMMAND

The forward file (FFILE) command is an environment command that forward spaces a magnetic tape or contiguous file a specified number of filemarks.

Format:

FFILE fd [,{n / 1}]

Parameters:

fd	is the file descriptor of the device or file to be forward spaced the specified number of filemarks.
n	is a decimal number specifying the number of filemarks to space forward. If this parameter is omitted, 1 is the default.

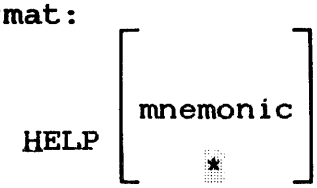
Example:

```
FF MAG1:,2
```

3.9 HELP COMMAND

The HELP command provides a list of all Link commands accepted by the latest revision of Link. HELP also describes the syntax and function of each command.

Format:



Parameters:

mnemonic	is the mnemonic for a Link command that is to be described by HELP.
*	lists all Link commands accepted by the latest revision of Link. If no parameter is specified, * is the default.

Functional Details:

If a log device has been specified in the START command for Link, HELP outputs all lists and descriptions of the Link commands to the log device.

For some commands (e.g., OPTION), the HELP information will require that more than one screen be displayed. In this case, Link will display a maximum of 23 lines, then prompt for a carriage return (CR) to continue the display. Any character except a CR will abort the remainder of the display.

Example:

```
*help

 BF(ILE)            BU(ILD)            DC(MD)             EN(D)
 ES(TABLISH)        EX(TERNAL)         FF(ILE)            H(ELP)
 IN(CLUDE)          LI(BRARY)          LOC(AL)            LOG
 MA(P)              ND(CMD)            NL(OG)             OP(TION)
 OV(ERLAY)          PA(USE)            PO(SITION)         REW(IND)
 RES(OLVE)                             TI(TLE)            VO(LUME)
 WF(ILE)
 FOR HELP ON ANY OF THE ABOVE COMMAND MNEMONICS, TYPE HELP MNEMONIC

*help map

MA(P) :  This command is a passive command that displays a map
   containing the names and addresses of symbols.

SYNTAX:  MA(P) [<FD>] [,AL(PHABETIC)] [,AD(DRESS)] [,XR(EF)]

WHERE:  <FD> is the file descriptor of the device to receive the
   map.  If this parameter is omitted, the map is sent to the log
   device.  If no log device has been specified, the maps are
   output to the command device, in interactive mode, and to
   device PR:  in batch mode.

The 'ALPHABETIC' parameter specifies that the map is to contain
    all symbols in alphabetic order.

The 'ADDRESS' parameter specifies that the map is to contain all
    symbols in address order.

The 'XREF' parameter specifies that the map is to contain all the
    names of the modules that reference each symbol, and the name
    of the module in which the symbol is defined.
```

3.10 INCLUDE COMMAND

The INCLUDE command is an active command that specifies a file containing object modules and the specific names of object modules that are to be included in the image. The INCLUDE command can be entered any number of times to include object modules from many different files.

Format:

$$\underline{\text{INCLUDE}}\ [\text{fd}]\left[\left[,\left\{\begin{matrix}\text{module}_1\\ *\end{matrix}\right\}\right]\ \left[-\left\{\begin{matrix}\text{module}_n\\ *\end{matrix}\right\}\right],\ldots,\text{module}_x\right]$$

Parameters:

fd — is the file descriptor of the file or device containing the modules to be included. If this parameter is omitted, a preassigned lul or the fd specified in the last INCLUDE command entered is used. If the extension is omitted, the default is .OBJ.

$module_1$ — is a 1- to 8-character alphanumeric string specifying the name of the next module of a range of modules to be included in the image. The first character of this string must be alphabetic if "*" or "-" is not specified. If an asterisk (*) is specified or this parameter is omitted, the next module, relative to the position of the file, is included.

$module_n$ — is a 1- to 8-character alphanumeric string specifying the name of the last module of a range of modules to be included in the image. The first character of this string must be alphabetic if "*" or "-" is not specified. If this parameter is omitted, $module_1$ is included. If an asterisk (*) or hyphen (-) with no module name is specified, all modules starting with $module_1$ to the end of the file are included.

Functional Details:

If no module names are specified, all modules in the file are included.

Object code modules specified in this command can consist only of the object code defined in Appendix D. Appendix D lists each loader item accepted by Link and describes what data may follow it.

Examples:

```
INCLUDE LIBRARY.OBJ
```

Include all modules in fd LIBRARY.OBJ.

```
INCLUDE LIBRARY.OBJ, FIRST
```

Include the object module FIRST in fd LIBRARY.OBJ.

```
INCLUDE ,SECOND-FOURTH
```

Include modules SECOND through FOURTH in the fd specified in the previous INCLUDE command.

```
INCLUDE LIBRARY.OBJ,-FOURTH,SIXTH,TENTH-*
```

Include modules FIRST through FOURTH, then module SIXTH, and module TENTH through the end of LIBRARY.OBJ

NOTE

If the NSEGMENTED option is selected, Link writes object modules to the task image in the same order as they are included. However, if SEGMENTED is specified, Link will choose the order of modules in the task image. In this case, modules will normally appear in exactly the opposite order that they were included.

3.11 LIBRARY COMMAND

The LIBRARY command is a passive command that specifies object libraries to be searched at build time to resolve external references. specified.

Format:

LIBRARY fd_1 $[,...,fd_n]$

Parameters:

fd	is the file descriptor of the library to be searched. If the extension is omitted, the default is .OBJ.

Functional Details:

The libraries specified by the LIBRARY command are searched for entry points that match unresolved external references in the image being built. When a match is found, the object module is included. Only one pass is made through the list of libraries.

When writing programs in high level languages such as FORTRAN or PASCAL, be sure to specify all user libraries before specifying a standard Perkin-Elmer Run-Time Library (RTL). This ensures that each user library routine gets resolved against the standard RTL.

Also, remember that the domain of a LIBRARY command is the entire Link command sequence (prior to the next BUILD or ESTABLISH command); i.e., its domain is not restricted to any overlay in which it might be placed. Only the order in which the libraries are specified is significant to Link.

When a program is linked, external references that were not resolved by the INCLUDE and RESOLVE commands are matched against the library(ies) entry points. All external references generated from modules included from the library cause the library modules that resolve those external references to be included, regardless of the order of the modules within the library.

Weak external references generated by the WXTRN pseudo-op are not matched against the library. These references are only resolved against entry points to modules that have been explicitly included, or have been included from a library through normal (strong) external references.

Nonlinking external references generated by the INCLD pseudo-op are matched against module names in the library.

Weak entry points in the library generated by the WNTRY pseudo-op are ignored during the library search.

A module is selected from a library for either of the following two reasons:

1. The module is named in an INCLD pseudo-op.

2. The module contains an ENTRY or a DNTRY which can be matched against an unresolved EXTRN in a previously included module.

Any weak entry points contained within this newly included module also become known to Link. These weak entry points are resolved against the list of unresolved, standard, and weak externals.

Example:

```
LI USER.LIB,F7RTL.OBJ
```

Specifies the user RTL and FORTRAN RTLs to be searched.

3.12 LOCAL COMMAND

The LOCAL command is a passive command that specifies one or more entry points in a partial image that can be referred to only by external references within that partial image. This command is valid only when establishing a partial image.

Format:

LOCAL entry point$_1$ [,...,entry point$_n$]

Parameters:

entry point	is a 1- to 8-character alphanumeric string specifying the entry point name. The first character of the string must be alphabetic.

Functional Details:

When a partial image is built, all entry points within that image can be referred to by tasks external to the partial image, unless the entry points are made local to that partial image by the LOCAL command.

Example:

```
LOC ENTRY1
```

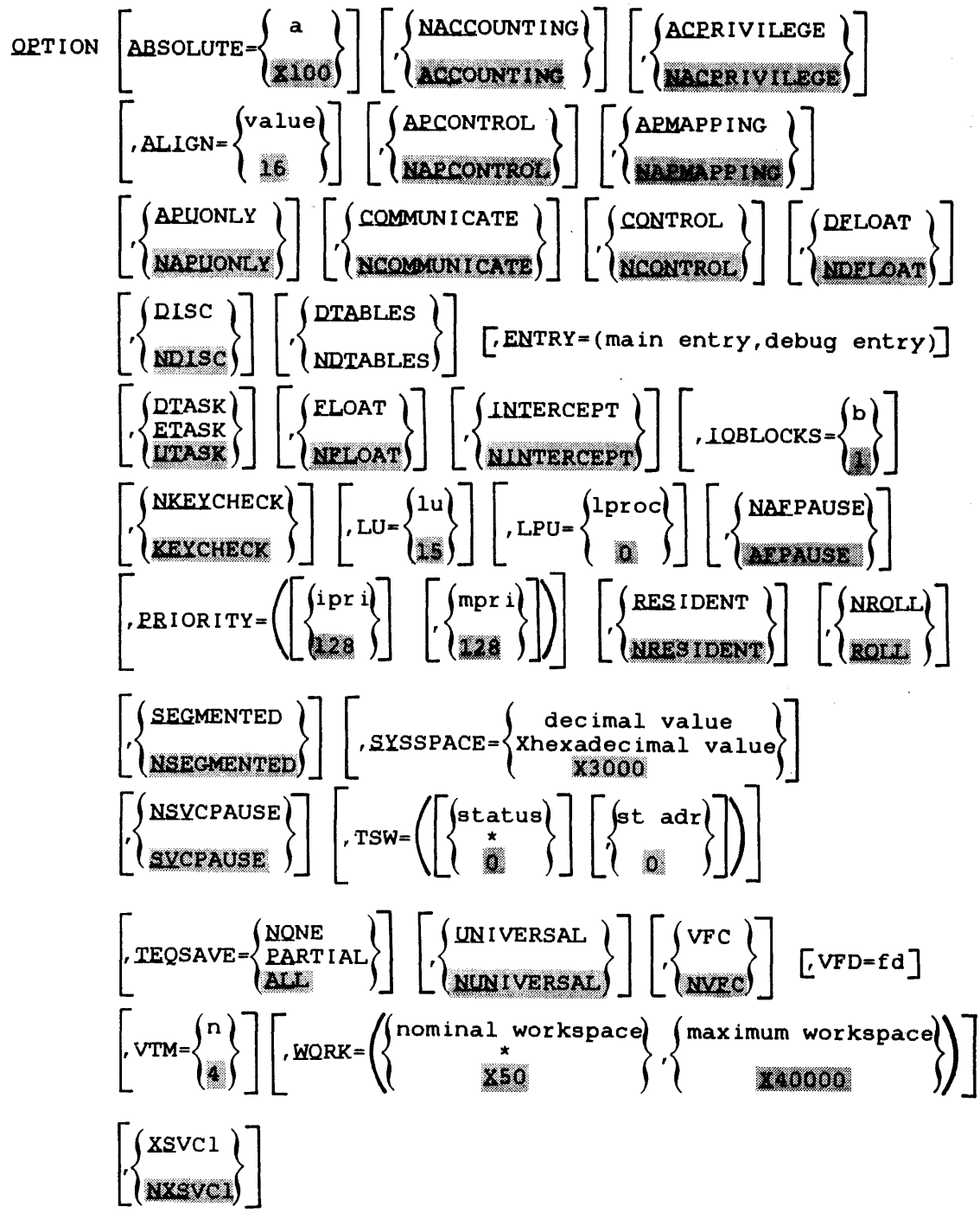
3.17 OPTION COMMAND

The OPTION command is a passive command that sets the task options that will be in effect during task execution.

CAUTION

WHEN A TASK IMAGE CREATED BY LINK IS LOADED UNDER MTM, CERTAIN MTM CONFIGURATIONS CAN OVERRIDE THE TASK OPTIONS SET BY THE OPTION COMMAND. SEE THE OS/32 MULTI-TERMINAL MONITOR (MTM) REFERENCE MANUAL FOR MORE INFORMATION.

Format:


OPTION
[ABSOLUTE={a / X100}] [,{NACCOUNTING / ACCOUNTING}] [,{ACPRIVILEGE / NACPRIVILEGE}]
[,ALIGN={value / 16}] [,{APCONTROL / NAPCONTROL}] [,{APMAPPING / NAPMAPPING}]
[,{APUONLY / NAPUONLY}] [,{COMMUNICATE / NCOMMUNICATE}] [,{CONTROL / NCONTROL}] [,{DFLOAT / NDFLOAT}]
[,{DISC / NDISC}] [,{DTABLES / NDTABLES}] [,ENTRY=(main entry,debug entry)]
[,{DTASK / ETASK / UTASK}] [,{FLOAT / NFLOAT}] [,{INTERCEPT / NINTERCEPT}] [,IOBLOCKS={b / 1}]
[,{NKEYCHECK / KEYCHECK}] [,LU={lu / 15}] [,LPU={lproc / 0}] [,{NAFPAUSE / AFPAUSE}]
[,PRIORITY=([{ipri / 128}] [,{mpri / 128}])] [,{RESIDENT / NRESIDENT}] [,{NROLL / ROLL}]
[,{SEGMENTED / NSEGMENTED}] [,SYSSPACE={decimal value / Xhexadecimal value / X3000}]
[,{NSVCPAUSE / SVCPAUSE}] [,TSW=([{status / * / 0}] [,{st adr / 0}])]
[,TEQSAVE={NONE / PARTIAL / ALL}] [,{UNIVERSAL / NUNIVERSAL}] [,{VFC / NVFC}] [,VFD=fd]
[,VTM={n / 4}] [,WORK=({nominal workspace / * / X50},{maximum workspace / X40000})]
[,{XSVC1 / NXSVC1}]

COMMUNICATE specifies that the task can perform the SVC 6 intertask communication functions. If this parameter is not specified, the task cannot communicate with other tasks.

NCOMMUNICATE prevents the task from issuing an SVC 6 for intertask communication. If the intertask communication option is not specified, NCOMMUNICATE is the default.

CONTROL specifies that the task can perform the SVC 6 intertask control functions. If this parameter is not specified, the task cannot issue an SVC 6 to control the execution of another task.

NCONTROL prevents the task from issuing an SVC 6 for intertask control. If the intertask control option is not specified, NCONTROL is the default.

DFLOAT specifies that a task can execute double precision floating point instructions. If this parameter is not specified, the task cannot execute double precision floating point instructions.

NDFLOAT prevents the task from executing double precision floating point instructions. If the double precision option is not specified, NDFLOAT is the default.

DISC is the bare disk I/O privilege option. This option allows a u-task or diagnostic task (d-task) to bypass the file manager and directly assign I/O requests to a disk device. If the disk is marked online, only assignments for shared read only (SRO) are allowed. Any other assignment is rejected, and a privilege error message is output. If the disk is marked offline, all access privileges are allowed. See the OS/32 Supervisor Call (SVC) Reference Manual for a description of the access privileges. This option has no affect on e-tasks, since they have bare disk privileges by definition.

NDISC — prevents u- and d-tasks from directly assigning I/O requests to a disk device. If the bare disk I/O privilege is not specified, NDISC is the default. This option has no affect on e-tasks.

NOTE

If a task is loaded under MTM and DISC is not specified, or DISC is specified but the task loader has the ETASK option disabled, the image is loaded without the bare disk I/O privilege.

DTABLES — causes the task loader to build the appropriate debug tables in the image for DEBUG/32. This option also increases the number of logical units used by the task, by one. However, LU=15 still appears on the Link map.

NDTABLES — prevents the task loader from building debug tables so that all debug data contained in the image is discarded. If DTABLES is not specified, debug tables are not built.

ENTRY — specifies the name of an entry point in the root node or the debug task where execution of the task image is to begin. If this option is omitted, the entry point is the starting address specified when the task was assembled or compiled.

main entry — is a standard entry point known to Link while the image is being built. Standard entry points include those for partial images but exclude data entry (DNTRY) points. If only the main entry is specified, omit the parentheses.

debug entry — is the name of the entry point for the debug task. The debug entry point specifies the location where execution of the task image will begin. In addition, the main entry or default entry is reserved for use by DEBUG/32.

DTASK — specifies that a d-task image is to be built. A d-task has its own virtual address space but can execute privileged instructions. If no task type parameter is specified, UTASK is the default.

ETASK — specifies that an e-task image is to be built. An e-task can contain only positional-independent pure and impure code and cannot reference partial images. An e-task can execute privileged instructions and reference physical and reference physical memory addresses.

UTASK — specifies that a u-task image is to be built. A u-task cannot execute privileged instructions. If no task type parameter is specified, UTASK is the default.

FLOAT — specifies that the task can execute single precision floating point instructions. If FLOAT is not specified, the task cannot execute single precision floating point instructions.

NFLOAT — prevents the task from executing single precision floating point instructions. If the single precision option is not specified, NFLOAT is the default.

INTERCEPT — specifies that the task can intercept an SVC issued by another task before the SVC is processed by the operating system. If this option is not specified, the task cannot intercept an SVC issued by another task. For more information on SVC interception, see the OS/32 System Level Programmer Reference Manual.

NINTERCEPT — prevents the task from intercepting an SVC issued by another task. If the SVC interception option is not specified, NINTERCEPT is the default.

IOBLOCKS — specifies the maximum number of I/O blocks assigned to the task. Each I/O control block can contain one queued I/O request. If this option is not specified, Link automatically assigns one I/O control block to the task.

b — is a decimal number from 1 through 65,535 indicating the number of I/O blocks assigned to the task.

NKEYCHECK — prevents the operating system from checking the file protection keys of a u- or d-task having accounting or bare disk I/O privileges. If this option is not specified, the operating system will check the file protection keys for all privileged u-tasks. NKEYCHECK has no affect on e-tasks.

KEYCHECK — causes the operating system to check the file protection keys of a u- or d-task having accounting or bare disk I/O privileges. If the file protection option is not specified, KEYCHECK is the default. KEYCHECK has no affect on e-tasks.

LU — specifies the maximum number of logical units that can be assigned to the task. If this option is not specified, the maximum number of logical units is 15.

lu — is a decimal number from 0 through 255.

LPU — specifies the logical processing unit (LPU) used to direct tasks to processors. This option is valid on a Model 3200MPS System only. Each task on Model 3200MPS System is assigned an LPU. Each LPU is logically mapped to an execution queue. Assignment of a particular LPU number results in the assignment of that task to the associated queue. The default assignment is zero, which specifies queue 0.

lproc — specifies the LPU that the task is to be assigned to. Legal values can range from decimal zero to the maximum number of LPUs present in the system (MAXLPU) up to maximum of 255. MAXLPU is a sysgen parameter. See the System Generation/32 (SYSGEN/32) Reference Manual.

NAFPAUSE — allows task execution to continue after an arithmetic fault occurs. If NAFPAUSE is not specified, task execution is suspended after an arithmetic fault.

AFPAUSE — suspends task execution after an arithmetic fault occurs. If the NAFPAUSE fault option is not specified, AFPAUSE is the default.

PRIORITY — specifies the initial and maximum priorities of the task. If this option is not specified, both the initial and maximum task priorities are 128. See the OS/32 Operator Reference Manual for an explanation of priority.

ipri — is a decimal number from 11 through 254 indicating the initial task priority. The initial priority must be greater or equal numerically to the specified maximum priority (mpri). If ipri is not specified, the default is 128.

mpri	is a decimal number from 11 through 254 indicating the maximum priority of the task. If mpri is not specified, the maximum priority is 128 (the value specified for the initial priority).
RESIDENT	specifies that the task is to remain in main memory after task execution is terminated. The task can then be restarted by the operator without issuing an OS/32 LOAD command. If this option is not specified, the task will be removed from memory after task termination.
NRESIDENT	specifies that the task is to be removed from main memory after task execution is terminated. If the RESIDENT option is not specified, NRESIDENT is the default.
NROLL	prevents the task from being rolled in and out of main memory during task execution. If this option is not specified, the task can be rolled during execution.
ROLL	specifies that the task can be rolled in and out of memory during task execution. If the NROLL option is not specified, ROLL is the default.
SEGMENTED	specifies that the pure segment of a u- or d-task can be shared when more than one copy of the task is loaded. If this option is not specified, the pure segment cannot be shared. SEGMENTED is incompatible with OPTION ETASK.
NSEGMENTED	specifies that the pure segment of a u- or d-task cannot be shared when more than one copy of the task is loaded. If the SEGMENTED option is not specified, NSEGMENTED is the default.
SYSSPACE	specifies the maximum amount of system space that a task can use during execution. System space is used for file control blocks associated with open disk files and other OS data structures associated with the task. If this option is not specified, the maximum system space that can be used is 12,288 (X3000) bytes.
decimal value	is a 1- to 7-digit decimal number specifying the maximum amount of system space.
hexadecimal value	is a 1- to 6-digit hexadecimal number preceded by an X specifying the maximum amount of system space.

NSVCPAUSE — specifies that SVC 6 is treated as a no-operation (NOP) (applies to .BG tasks only). If a background task issues an SVC 6, the operating system ignores that call and continues execution of the task. If this option is not specified, the operating system pauses the execution of a background task that issues an SVC 6.

SVCPAUSE — specifies that SVC 6 is treated as an illegal SVC (applies to .BG tasks only). If an SVC 6 is issued by a background task, the operating system pauses execution of that task. If the SVC 6 PAUSE option for background tasks is not specified, SVCPAUSE is the default.

TSW — sets the task status and starting address fields of the task status word (TSW) in the LIB. If multiple TSW options are specified, an OR operation is performed on the status field before the TSW is loaded into the final TSW for the task image. This option overrides any starting address specified by ENTRY.

status — is a 1- to 8-digit hexadecimal number indicating the initial setting of the status field of the TSW in the LIB. If the asterisk (*) is specified, the current TSW is reset to zero. If status is not specified, the initial setting of the status field is zero.

st adr — is a 1- to 6-digit hexadecimal number indicating the starting address for the task. This address overrides the starting address specified when the task was assembled or compiled as well as any starting address specified by the ENTRY option.

TEQSAVE — informs the operating system whether or not the register contents should be saved and restored when the task enters or exits a task event service routine. The parameters of this option are:

maximum workspace — is a 1- to 6-digit hexadecimal or 1- to 7-digit decimal number indicating the maximum amount of workspace that can be added by the LOAD command. If the maximum workspace is not specified, 256K (X40000) is the maximum number of bytes that can be added. The maximum workspace value is added to the maximum workspace values specified by previous OPTION WORK= commands to obtain the total maximum workspace.

XSVC1 — indicates that if the task issues an SVC 1 with bit 7 of the function code set, the options specified by the SVC 1 extended option field are to be executed for all drivers which use this field. If XSVC1 is not specified, an SVC 1 with bit 7 set performs an image I/O transfer. See the OS/32 Supervisor Call (SVC) Reference Manual for more information on the SVC 1 function code and extended options.

NXSVC1 — indicates that if the task issues an SVC 1 with bit 7 of the function code set, an image I/O transfer is performed. If the XSVC1 option is not specified, NXSVC1 is the default. See the OS/32 Supervisor Call (SVC) Reference Manual for more information on the SVC 1 function code and extended options.

Examples:

```
OPTION ACPRIVILEGE,NKEYCHECK,ALIGN=4,
       DFLOAT,LU=10,PRIORITY=(,100),
       SYSSPACE=X4000,VFC,XSVC1,
       WORK=(X100,X1000)
```

In this example, the task is to be linked as a u-task with extended file access privileges and without key checking. All object modules will be aligned to the nearest fullword boundary. The task can execute double precision floating point instructions and assign up to ten logical units. Maximum task priority is 100; initial task priority is 128. VFC is in effect for all I/O operations. The options specified by the SVC 1 extended option field are to be executed for all drivers that use this field. The task can be loaded with a maximum workspace of 4,096 bytes. If workspace is not specified in the OS/32 or MTM LOAD command, the task will be loaded with 256 bytes. Note that X precedes the hexadecimal numbers in the WORK option. Maximum system space that can be used by this task is 16,384 bytes.

```
OPTION DTABLES,ENTRY=(,DEBUG32)
```

In this example, the u-task is to be debugged using DEBUG/32. DTABLES builds the required debug tables needed to run DEBUG/32 while ENTRY specifies the name of the entry point to the debug task.

```
OPTION INTERCEPT,TEQSAVE=PARTIAL
```

This example shows the task options that apply to a u-task that is to be linked with the SVC interception software. INTERCEPT allows the u-task to intercept an SVC of another task. TEQSAVE=PARTIAL indicates that all register contents used by the task event service routine are to be saved and restored. See the OS/32 System Level Programmer Reference Manual for more information on SVC interception and the task event service routine.

```
OPTION VTM=5,VFD=PROG1.VTM
```

This example shows the task options that apply when a u-task is to run under the virtual memory manager. See Chapter 5. VTM specifies that a virtual image is to be built; VFD specifies that PROG1.VTM is to be used as a secondary storage file by the virtual task.

```
OPTION FL,RES,LU=10,WORK=X3000,TSW=(,B020),APC,APM
```

This example shows the task options that can apply when the task is to run on the APU of a Model 3200MPS System. The task can execute single precision floating point instructions; is resident; has a maximum of 10 logical units that can be assigned to it; has a maximum workspace of X3000 bytes; has a starting address field of XB020 in the LIB; can obtain APU control privileges, and APU mapping privileges in a multiprocessor system. The APC and APM options are valid on a Model 3200MPS System only.

There are two consequences to this positioning policy. The first is that named common blocks are initialized each time an overlay is fetched from disk. The second consequence is that more than one copy of a common entity can exist on separate paths in the program; i.e., two or more overlays can have their own separate and private copies of a common entity. These copies could then contain different values.

Example:

```
ES TASK
INCLUDE ROOT
POSITION COMMON=(A,B)
OVERLAY OVLY1,1
INCLUDE SUB1
INCLUDE SUB2
OVERLAY OVLY2,1
INCLUDE SUB3
```

3.21 RESOLVE COMMAND

The RESOLVE command is a passive command that specifies the name of a partial image to be referred to by the task image. The partial image can be a global entity generated at the console by the OS/32 TCOM command, a sharable segment created by Link R00, or a partial image created by Link R01.

Format:

```
RESOLVE   [fd] [,NAME=package name]

                    [  R   ]
                    [  E   ]
          ,ACCESS=  [{ RE }] [,ADDRESS=m0000]
                    [  RW  ]
                    [  RWE ]

          [,STRUCTURE=(name1 [/size1] [,...,namen][/sizen])]
          [,SIZE=([min [,max]])]
```

Parameters:

fd — is the file descriptor of the partial image. If fd is not specified, the default partial image is the global task common defined by the TCOM command. If the file extension for a partial image created by Link R01 is not specified, the default extension is .IMG. Because the default extension for sharable segments created by Link R00 is .SEG, the file extension should be specified when these segments are resolved.

NOTE

Link cannot get the size of a task common segment defined by TCOM from an image file; therefore, when the partial image is a global task common, the size of the partial image must be specified by the SIZE or STRUCTURE parameter in the RESOLVE command.

NAME= specifies the package name of the partial image. If this parameter is omitted, fd must be specified, and the default package name is the package name assigned to the partial image when it was established. When the task is loaded, the package name is matched against the names of any partial images already in main memory. If a partial image with the specified package name is not found in memory when the task is loaded, the package name is converted into an fd which is then used to locate and load a partial image.

package name is a filename.ext that identifies the partial image after it is loaded into memory. This name is matched against either the name of the global entity specified by TCOM or the package names of sharable segments or partial images.

ACCESS= specifies the access privilege of the partial image as follows:

R specifies that the task can read data within the partial image. Execution or modification of data is not allowed.

E specifies that the task can execute code within the partial image but cannot read or modify data within the image.

RE specifies that the task can read data and execute code within the partial image. Modification of data is not allowed. If the ACCESS= parameter is omitted, the default is RE.

RW specifies that the task can read and modify data within the partial image. Code execution is not allowed.

RWE specifies that the task can read and modify data and execute code within the partial image.

ADDRESS= m0000 is the starting address of the partial image. If the RESOLVE command specifies an fd for a partial image that is not address-independent, the specified address must match the address specified in the LIB of the partial image. If ADDRESS= is not specified, and the address was not specified when the partial image was established, Link automatically assigns an address to the partial image. The variable m is a hexadecimal number in the range from 0 through BF.

STRUCTURE= structures task common blocks within the partial image specified by fd. If fd is not specified, this parameter is used to structure global task common defined by the TCOM command.

$name_1 \ldots name_n$ is an 8-character alphanumeric string specifying the name of the task common block to be structured.

$size_1 \ldots size_n$ is a 1- to 6-digit hexadecimal number or a 1- to 7-digit decimal number specifying the length in bytes of the task common block. (Hexadecimal numbers must be preceded by an X; e.g., XF0.) This number must be greater than or equal to the size of the task common block specified by the program. If this number is smaller than the size specified by the program, Link outputs a warning message and uses the size specified by the program. The program size is also used if this parameter is omitted.

NOTE

If common blocks in a partial image are declared by using the EXTERNAL command when the partial image is built, STRUCTURE need not be specified when resolving against that partial image.

SIZE= specifies the minimum and maximum number of bytes of main memory that the partial image can occupy. If SIZE= and fd are not specified, the default size of the partial image is that specified by the STRUCTURE parameter. If SIZE is not specified but fd is, the default size of the partial image is the size obtained form the LIB of the partial image specified by fd.

min is a 1- to 6-digit hexadecimal number or a 1- to 7-digit decimal number specifying the minimum number of bytes of main memory that the partial image can occupy. (A hexadecimal number must be preceded by an X; e.g., XFO.)

Example:

```
INCLUDE M300:MOD5.OBJ,MSP
OVERLAY A
INCLUDE ,SUBA
OVERLAY B
INCLUDE ,SUBB
OVERLAY C
INCLUDE ,SUBC
MAP PR1:,ADDRESS
BUILD MOD5
END
```

The first INCLUDE command specifies that the object module MSP in the input file MOD5.OBJ on disc volume M300 is to be included in the image. Because this command is specified before any OVERLAY command, MSP is placed in the root node.

The first OVERLAY command defines an overlay area named A. The INCLUDE command specifies that the object module called SUBA is part of overlay A. It is contained in the object file most recently specified in an INCLUDE command (MOD5.OBJ), and it will be automatically loaded into memory when MSP calls SUBA if it is not already in memory.

The second OVERLAY command defines an overlay area named B. The INCLUDE command specifies that the object module called SUBB is part of overlay B and will be automatically loaded into the same memory area previously occupied by overlay A, if SUBB is not already loaded when MSP calls it.

The third OVERLAY and INCLUDE commands define an overlay area named C and include the object module called SUBC as part of overlay C.

The MAP command specifies that an establishment summary and a listing of the names and locations for each overlay are to be produced in address order.

The BUILD command builds the image called MOD5.TSK which consists of a root segment and an overlay area large enough to contain the largest overlay (A, B, or C).

The END command terminates the linkage editor.

4.4.2 Building a More Complex Overlayed Task Image

The following example builds an overlayed task image from the object file MOD6.OBJ which consists of a main program that calls two subroutines (SUBA and SUBB). Subroutine SUBA calls two more subroutines (SUBA1 and SUBA2). Subroutine SUBB also calls two more subroutines (SUBB1 and SUBB2). In addition to SUBA and SUBB overlaying each other, SUBA1 and SUBA2 are to be overlayed when SUBA is in memory. SUBB calls SUBB1 and SUBB2, and SUBB1 and SUBB2 are to be overlayed when SUBB is in memory. This overlay process can be accomplished by using another level of overlay areas. Figure 4-1 illustrates the overlay structure for this example.

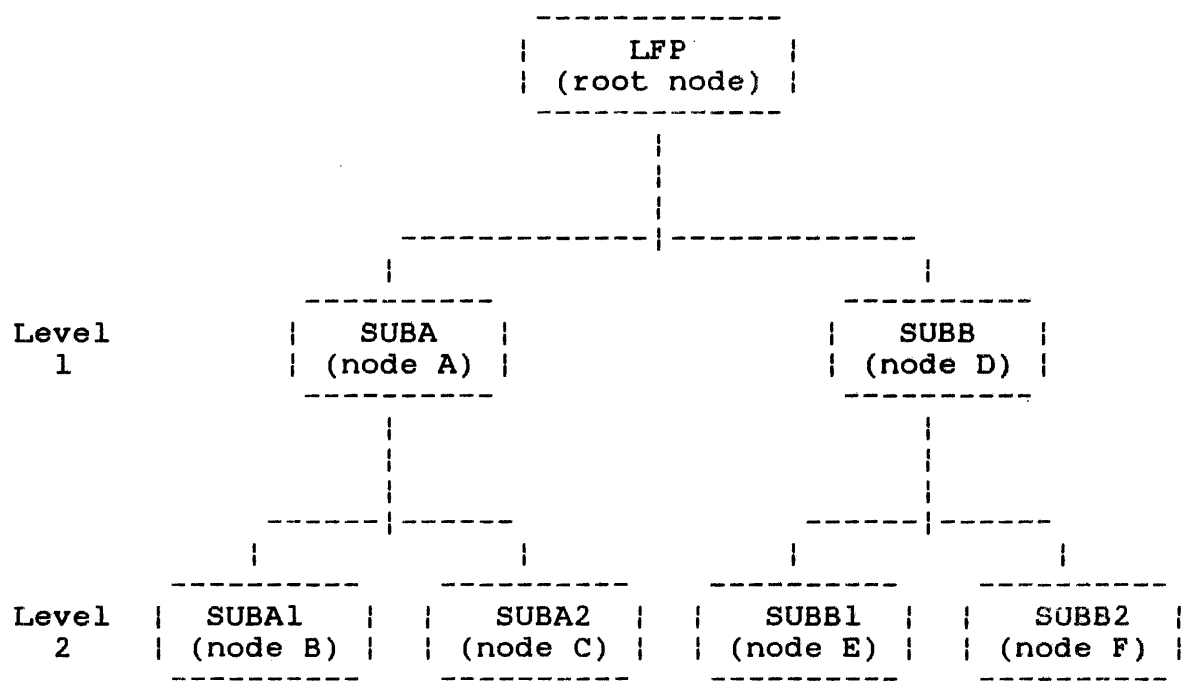


Figure 4-1 Sample Overlay Structure

A path is defined as a set of nodes (a group of routines loaded at one time is a node), one at each level, each of which is a descendant of the node at the previous level. For example, node D and node E form a path. Only nodes in the same path can be in memory at the same time and, therefore, a routine can only call routines in nodes that are in the same path as the node containing the calling routine.

CHAPTER 5
VIRTUAL TASK MANAGEMENT (VTM)

5.1 INTRODUCTION

VTM provides a virtual memory capability for large FORTRAN tasks. User tasks (u-tasks) consisting of up to 16Mb of code and data can execute in as little as 128kb of user task memory. VTM also supports common assembly language (CAL) and PASCAL programs with some code restrictions.

VTM uses the memory address translator (MAT) to optimize run-time performance. It contains run-time algorithms to provide performance for the widest possible scope of u-task characteristics. VTM employs a least recently used working set algorithm. The virtual activity of a VTM task is independent of the operating system and does not impact other tasks in the system. VTM tasks are nonrollable by default but can be made rollable.

5.2 SYSTEM REQUIREMENTS

The minimum requirements for use of this feature are any Perkin-Elmer processors equipped with MAT hardware, and OS/32 6.2 and higher. Perkin-Elmer processor Models 7/32, 8/32, and 3220 are not supported.

5.3 USER INTERFACE TO VIRTUAL TASK MANAGEMENT (VTM)

The following sections describe how to use VTM.

5.3.1 Declaring a Virtual Task Management (VTM) Task

The user declares a virtual task via the Link OPTION command:

```
OPTION  VTM[=n]
```

where n is the number of 64kb working pages desired for task memory management.

The minimum value of n is 2, the default is 4, and the maximum is 127. The number of working pages needed for reasonable performance varies depending upon the user's applications and needs.

NOTE

The VTM option and the Link overlay feature are incompatible and must not be used in the same task.

5.3.2 Virtual Task Management (VTM) Secondary Storage

An additional option may also be specified via the Link OPTION command:

```
OPTION  VFD=fd
```

where fd is a contiguous file to be used as secondary storage for the virtual task.

If the VFD option is not entered, VTM allocates a temporary contiguous file at run-time.

The specified file descriptor (fd) may be the task image file itself, in which case the task image file might be destroyed at run-time. When OPTION VFD is specified, multiple copies of the same task image cannot be run concurrently. The minimum size of fd is (CTOP/256)-255 sectors (plus 256 sectors if fd is the task image file).

5.3.3 Including the Virtual Task Management (VTM) Module

Prior to including any task modules, the user must include the VTM object module (VTM32.OBJ) supplied with the operating system package. The VTM module is approximately 8kb in size.

5.3.4 Virtual Task Workspace

All workspace required for the execution of a virtual task must be requested at Link time via the WORK option of the Link OPTION command. Additional memory cannot be obtained via the LOAD command.

5.3.5 Example of Virtual Task Management (VTM) Link Procedures

The following Link command sequence demonstrates how to build a VTM task.

Example:

```
OPTION VTM=5
OPTION DFLOAT,FLOAT,WORK=X3000
INCLUDE VTM32
INCLUDE MAIN
INCLUDE SUB1
INCLUDE SUB2
LIBRARY F7RTL
MAP PR:
BUILD FORTTASK
END
```

FORTTASK executes in five working pages, using a temporary file as secondary storage.

5.3.6 Virtual Task Managment (VTM) Logical Units

For a VTM task, the two highest numbered valid task logical units are reserved for VTM use. For example, if OPTION LU is not specified, logical units 13 and 14 are reserved for VTM.

5.3.7 Rolling of Virtual Task Management (VTM) Tasks

VTM tasks are nonrollable by default. A user can specify VTM task roll eligibility after loading and before starting the task by entering the following command:

```
MOD 104,1
```

5.3.8 Absolute Code

Absolute-origined code or data cannot extend beyond X'400' in a VTM task.

5.4 FORTRAN OPERATIONAL RULES

The following are FORTRAN operational rules for the VTM feature:

- The u-task workspace requested by the WORK option should not exceed 64kb in a virtual task. Input/output (I/O) transfers are limited to 64kb.
- Nonlanguage I/O calls made through the use of SYSIO fall under the CAL coding restrictions.

5.5 COMMON ASSEMBLY LANGUAGE (CAL) RESTRICTIONS

SVC 1 I/O buffers and SVC parameter blocks should not cross logical 64kb boundaries to ensure proper execution. It is suggested that the buffers be placed in the first 64kb of the task to avoid this possibility.

5.6 PASCAL CODE RESTRICTIONS

To ensure proper execution, file variables should be declared before any other variables in the global variable declarations of the main program. The total size of the file buffers, plus 80 bytes of control data for each file, should not exceed 64kb.

5.7 PERFORMANCE MEASUREMENT

The user can analyze the relative performance of a virtual task with different numbers of working pages using the data on the number of I/Os available in the OS/32 DISPLAY ACCOUNTING command.

NOTE

Certain tasks, by their nature, do not perform well in a virtual environment. Tasks with extensive compute bound array access in which a working set cannot be contained in the number of specified working pages might operate poorly as VTM tasks.

5.8 VIRTUAL TASK MANAGEMENT (VTM) ERROR CONDITIONS

VTM error conditions result in the task being paused or cancelled with end of task code 1 and an appropriate error message. A summary of VTM error messages is presented in Appendix C.

APPENDIX B
LINK MESSAGE SUMMARY

ADDRESS OVERFLOW AT xxxxxx

A halfword relocatable address was larger than 64kb.

ATTEMPT TO POSITION x IN A DIFFERENT PATH

An attempt was made to position a common block to a node that is not in the same path as is the node referring to it.

ATTEMPT TO POSITION x IN LOWER LEVEL NODE

An attempt was made to reposition a common block program in a lower level node.

ATTEMPT TO REFERENCE ADDRESS number
ADDRESS OUTSIDE OF ADDRESS SPACE FOR IMAGE
-FILE: vol:filename.ext/a -MODULE:module
-RECORD:number - BYTE:number

The task image being built refers to an address outside the address space of any of the known segments or partial images of the task. This message identifies the file, module, record number, and byte number of the object code that caused the error.

BUILD NOT SUPPORTED ON THIS DEVICE

A file other than an indexed, nonbuffered indexed, contiguous or extended contiguous file, or the null device was specified for building the image.

CHECKSUM ERROR FILE: x MODULE: y RECORD: z

An invalid checksum was detected while reading an object file.

COMMAND NOT PERMITTED

Command is not valid for the type of build or is not permitted as in an embedded command in an object module.

COMMON x ENCOUNTERED IN MORE THAN ONE PARTIAL IMAGE

The same common block was specified in more than one of the partial images referred to by the task.

COMMON BLOCK x, UNREFERENCED

The common block named was never referred to.

COMMON BLOCK x SPECIFIED IN POSITION COMMAND IS PART OF PARTIAL IMAGE

An attempt was made to reposition a common block that was part of a partial image by using the POSITION command.

CONTINUATION NOT PERMITTED

An attempt was made to continue a command imbedded in an object module.

ENTRY POINT x SPECIFIED IN ENTRY OPTION NOT FOUND

The ENTRY parameter of the OPTION command specified a nonexistent entry point or an entry point in other than the root node.

ENTRY POINT x SPECIFIED IN LOCAL COMMAND NOT DEFINED

The entry point named was never defined.

name SPECIFIED IN POSITION COMMAND NOT FOUND

The named common block that was specified by a POSITION command could not be found.

NODE IS NOT SUITABLE FOR OVERLAYS

This message indicates that the Link command sequence is attempting to overlay the task in a partial image or pure segment.

NUMERIC VALUE OUT OF RANGE

A numeric operand was greater than the maximum permissible value or less than the minimum permissible value.

OBJECT CODE ERROR (n) FILE: x MODULE: y RECORD: z BYTE m

An object code error occurred. If n=1, an invalid object code item exists in object record. If n=2, the object code item overflows the record. If n=3, a load program address item was expected but not encountered.

PROGRAM TRANSFER ADDRESS IN PROGRAM module IN AN OVERLAY

A program transfer address (PTA) (starting address) was specified for the task in a module that is in an overlay node. Link ignores the specified PTA and uses the task's default starting address.

OVERLAY DEFINED OUT OF ORDER

An OVERLAY command specified a level inconsistent with the rules for defining overlays.

RECORD LENGTH FOR MAP DEVICE/FILE < 64 BYTES

The device or file specified for the output of the maps has a record length of less than 64 bytes.

SEGMENT AT x OVERLAPS PREVIOUSLY DEFINED SEGMENT

The end address of an impure, pure, or shared logical segment was greater than the beginning address of another segment. See the establishment summary for the names of the segments.

SEQUENCE ERROR FILE x MODULE: y RECORD: z

A sequence number error was detected while reading an object module.

SIZE OF SEGMENT TRUNCATED TO PHYSICAL SIZE

The maximum length of the partial image specified by the SIZE parameter in the RESOLVE command is larger than any existing segment for that image. This message indicates that Link is using the size of the existing segment for the maximum partial image size rather than the maximum specified by SIZE.

TOO MANY OPERANDS

More operands than allowed were encountered.

VTM TASK WORKSPACE IS GREATER THAN 64K BYTES

When a FORTRAN task is linked as a virtual task, the user task workspace requested by the WORK option should not exceed 64kb. This message indicates that the WORK option for the FORTRAN task being linked exceeds 64kb.

VIRTUAL SYMBOL TABLE SPACE LIMIT EXCEEDED

More than 256kb of symbol table space required.

WARNING: ABSOLUTE SPACE LESS THAN 100

Less than 100 bytes of absolute code were reserved for the UDL.

WARNING: ADDRESS OF PARTIAL IMAGE SEGMENT FOR fd DOES NOT MATCH ADDRESS SPECIFIED ON RESOLVE COMMAND

This warning is output if the RESOLVE command specifies an fd and an address for an address-dependent partial image, and that address does not match the address in the loader information block (LIB) for that partial image. Link uses the address specified in the partial image's LIB.

WARNING: COMMON xxxxxx APPEARS MORE THAN ONCE IN STRUCTURE COMMAND

In the STRUCTURE parameter of the RESOLVE command, the user attempted to use the same name to define two separate common blocks. Common block names within a partial image must be unique.

APPENDIX C
VIRTUAL TASK MANAGEMENT (VTM) MESSAGE SUMMARY

INSUFFICIENT VTM WORKING PAGES

For this task, at least one additional working page is required for VTM execution.

MEM FAULT AT xxxxxx INSTR AT xxxxxx CODE=xx (task paused)

Task memory access fault. xx specifies the code that describes the type of memory error fault that occurred. These codes are defined in Table C-1.

TABLE C-1 VTM MEMORY FAULT CODES

MEMORY FAULT CODES	MEANING
00	Supervisor call (SVC) address error
01	Execute protect error
02	Write protect error
03	Read protect error
04	Access level error
07	Shared segment table size error
08	Private segment table size error

TASK FD ASGN-ERR - CODE=xx

Error in assigning task file. xx is the SVC 7 error status.

VIRT FD ALLO-ERR - CODE=xx

Error in allocating temporary file. xx is the SVC 7 error status.

VIRT FD ASGN-ERR - CODE=xx

Error in assigning VFD file. xx is the SVC 7 error status.

VIRT FD NOT CONTIG

Specified file is not contiguous.

VIRT FD TOO SMALL

Specified file is too small.

VTM RD-ERR STAT=xxxx (task paused)

Unrecoverable read error on a virtual I/O transfer. xxxx is the SVC 1 status halfword; a device independent status of 00 indicates a length of transfer error.

VTM WT-ERR STAT=xxxx (task paused)

Unrecoverable write error on a virtual I/O transfer. xxxx is the SVC 1 status halfword; a device independent status of 00 indicates a length of transfer error.

INDEX

U

V

W,X,Y,Z

MANUAL UPDATE PACKAGE COVER SHEET

MANUAL TITLE: OS/32 LINK Reference Manual

PUBLICATION NUMBER: 48-005

OLD REVISION LEVEL: F00 R02

NEW REVISION LEVEL: F01 R02

This package of affected pages updates the current version of the subject manual. New features, as well as changes, deletions and additions to information in this manual are indicated by change bars in the page margins. Please discard the indicated old pages and replace or insert them with the supplied new pages.

OLD PAGES	NEW PAGES
Title Sheet/Disclaimer, F00 R02	Title Sheet/Disclaimer, F01 R02
Sheets i through iv, F00 R02	Sheets i through iv, F01 R02
Sheet v, F00 R02	Sheet v, F01 R02
Sheet 1-1, F00 R02	Sheet 1-1, F01 R02
Sheet 1-2, F00 R02	Sheet 1-2, F01 R02
Sheet 2-1, F00 R02	Sheet 2-1, F01 R02
Sheet 2-2, F00 R02	Sheet 2-2, F01 R02
	Sheet 2-2a, F01 R02
Sheet 2-3, F00 R02	Sheet 2-3, F00 R02
Sheet 2-4, F00 R02	Sheet 2-4, F01 R02
Sheet 2-5, F00 R02	Sheet 2-5, F01 R02
Sheet 3-1, F00 R02	Sheet 3-1, F01 R02
Sheet 3-2, F00 R02	Sheet 3-2, F00 R02
Sheet 3-5, R00 R02	Sheet 3-5, F01 R02
Sheet 3-6, F00 R02	Sheet 3-6, F00 R02
Sheet 3-7, F00 R02	Sheet 3-7, F00 R02
Sheet 3-8, F00 R02	Sheet 3-8, F01 R02
Sheet 3-9, F00 R02	Sheet 3-9, F01 R02
Sheet 3-10, F00 R02	Sheet 3-10, F00 R02
Sheet 3-11, F00 R02	Sheet 3-11, F01 R02
Sheet 3-12, F00 R02	Sheet 3-12, F00 R02
Sheet 3-17, F00 R02	Sheet 3-17, F00 R02
Sheet 3-18, F00 R02	Sheet 3-18, F01 R02
Sheet 3-19, F00 R02	Sheet 3-19, F01 R02
Sheet 3-20, F00 R02	Sheet 3-20, F00 R02

OLD PAGES	NEW PAGES
Sheet 3-21, F00 R02	Sheet 3-21, F01 R02
Sheet 3-22, F00 R02	Sheet 3-22, F01 R02
Sheet 3-23, F00 R02	Sheet 3-23, F01 R02
Sheet 3-24, F00 R02	Sheet 3-24, F00 R02
Sheet 3-33, F00 R02	Sheet 3-33, F01 R02
Sheet 3-34, F00 R02	Sheet 3-34, F00 R02
Sheet 3-37, F00 R02	Sheet 3-37, F00 R02
Sheet 3-38, F00 R02	Sheet 3-38, F01 R02
Sheet 3-39, F00 R02	Sheet 3-39, F01 R02
Sheet 3-40, F00 R02	Sheet 3-40, F01 R02
Sheet 3-41, F00 R02	Sheet 3-41, F01 R02
Sheet 3-42, F00 R02	Sheet 3-42, F01 R02
Sheet 3-45, F00 R02	Sheet 3-45, F01 R02
Sheet 3-46, F00 R02	Sheet 3-46, F00 R02
Sheet 3-51, F00 R02	Sheet 3-51, F00 R02
Sheet 3-52, F00 R02	Sheet 3-52, F01 R02
Sheet 3-53, F00 R02	Sheet 3-53, F00 R02
Sheet 3-54, F00 R02	Sheet 3-54, F01 R02
	Sheet 3-54a, F01 R02
Sheet 4-5, F00 R02	Sheet 4-5, F01 R02
Sheet 4-6, F00 R02	Sheet 4-6, F00 R02
Sheets 5-1 through 5-5, F00 R02	Sheets 5-1 through 5-4, F01 R02
Sheet B-1, F00 R02	Sheet B-1, F00 R02
Sheet B-2, F00 R02	Sheet B-2, F01 R02
Sheet B-7, F00 R02	Sheet B-7, F01 R02
Sheet B-8, F00 R02	Sheet B-8, F00 R02
Sheet C-1, F00 R02	Sheet C-1, F01 R02
Sheet C-2, F00 R02	Sheet C-2, F01 R02
Sheets IND-1 through IND-3, F00 R02	Sheets IND-1 through IND-3, F01 R02

PERKIN-ELMER
Technical Systems Division

D O C U M E N T A T I O N C H A N G E N O T I C E

The purpose of this documentation change notice (DCN) is to provide a quick and efficient way of making technical changes to manuals before they are formally updated or revised.

The manual affected by these changes is:

48-005 F00 R02 OS/32 Link Reference Manual

For conversion purposes, a Compatible Link Utility (R02) is included with the OS/32 Software Package. This utility is designed to allow users who have extensive Link command files built using the Link R01 command syntax to continue to use those Link command sequences and also be able to use all of the new enhancements included in the R02 revision of Link.

The users who elect to use the Compatible Link Utility should note that there are five commands with formats that differ from the formats documented in the R02 release of Link. The formats of these commands are the same as those documented in the R01 Link Manual.

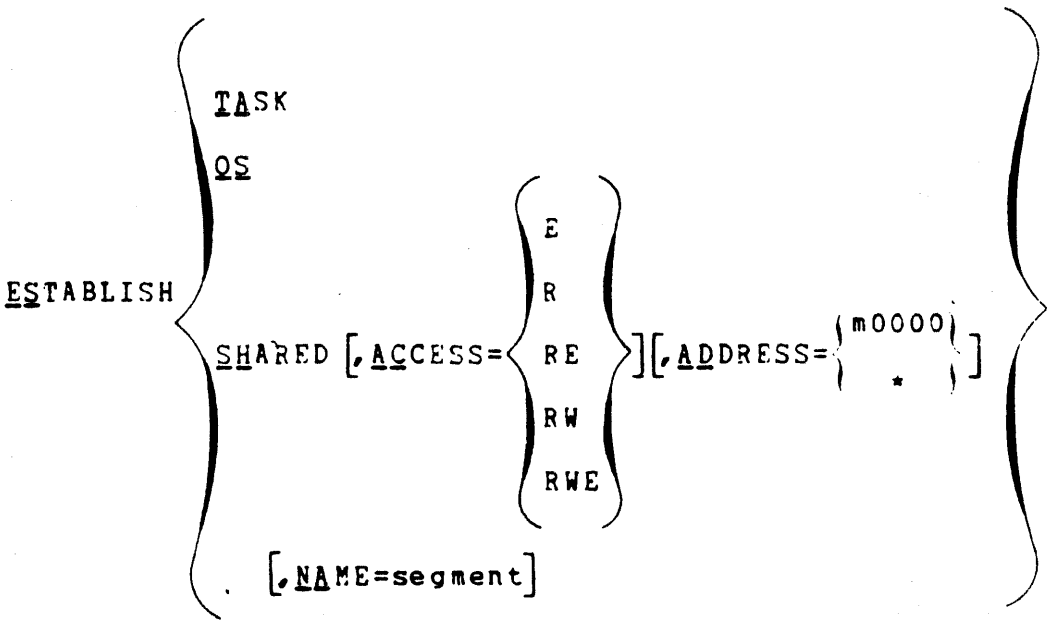
The differences between the R01 and R02 versions of these commands are as follows. Keep in mind that the R01 versions of these commands are those supported by the Compatible Link Utility.

- BUILD Command

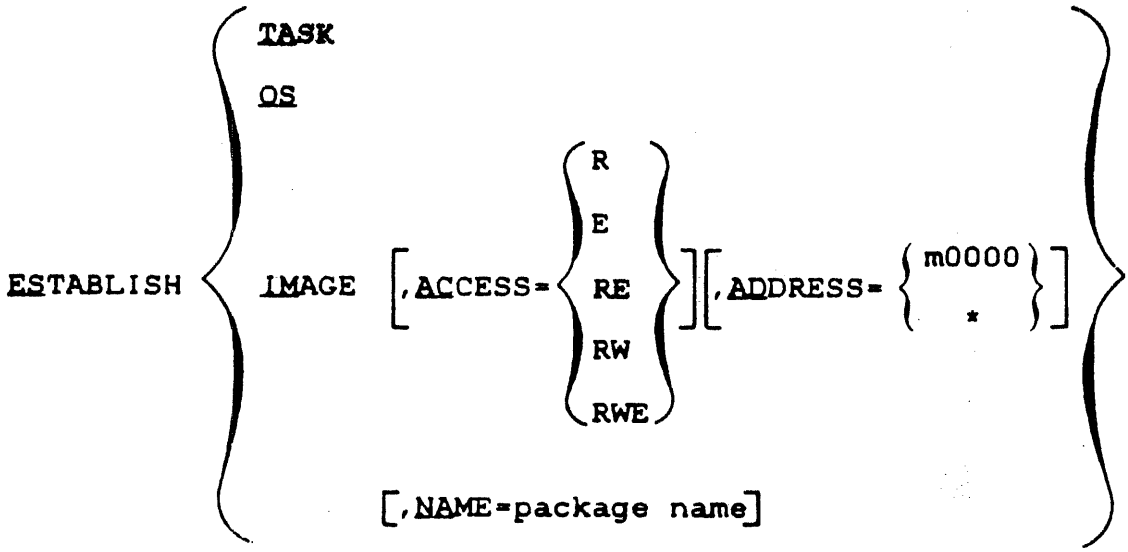
  The BUILD command fd has a default extension of .IMG in the Link R01 version, and an R02 default extension of .SEG, documented on Pages 3-5 and A-1 of the R02 version.

- **ESTABLISH Command**

The ESTABLISH command in the R01 manual has a SHARED option. The R01 version of the ESTABLISH command is:

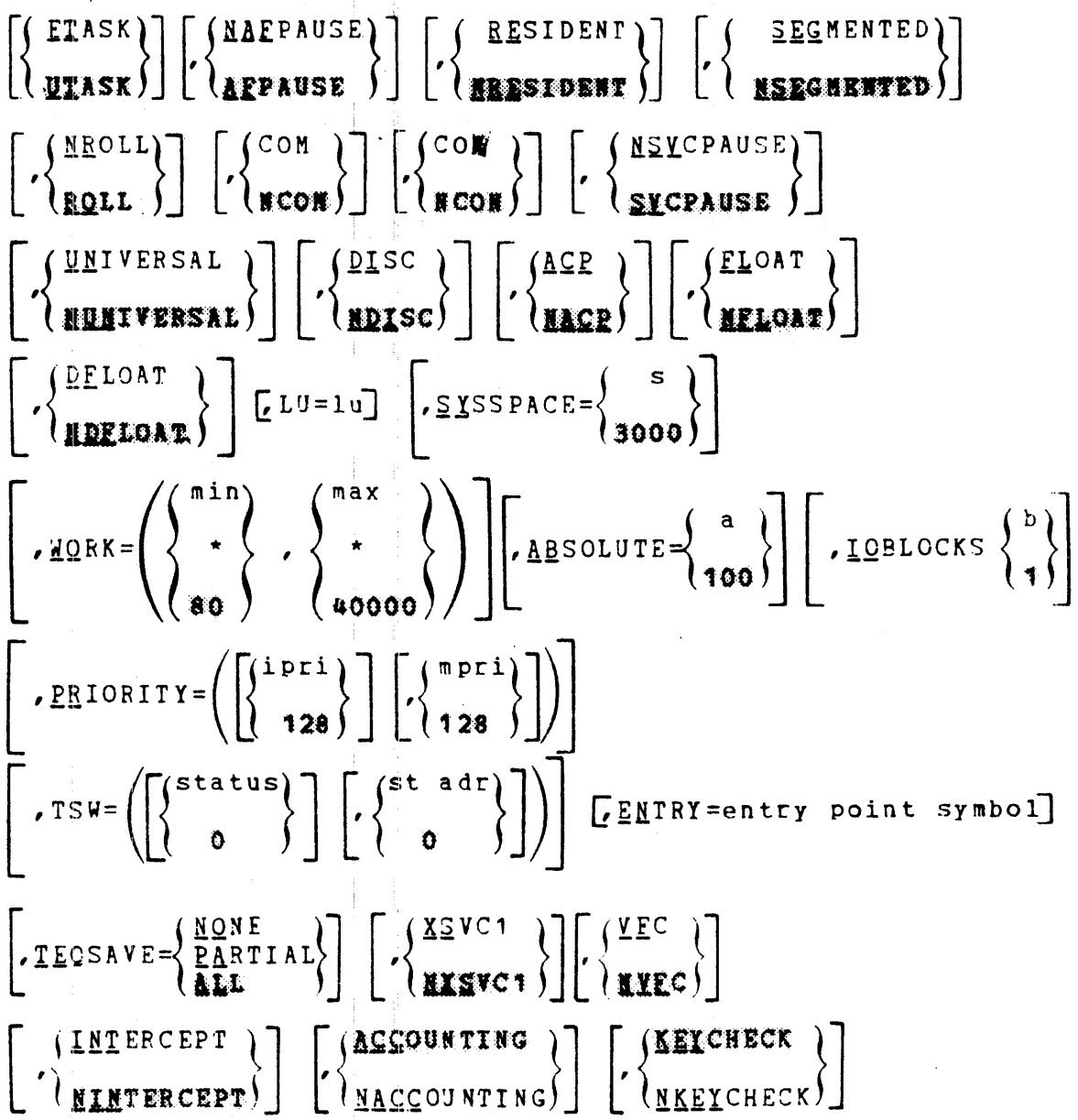


On Pages 3-12 and A-1 of the R02 manual, the ESTABLISH command has an IMAGE option instead of the SHARED option in the R01 version. The R02 version is:

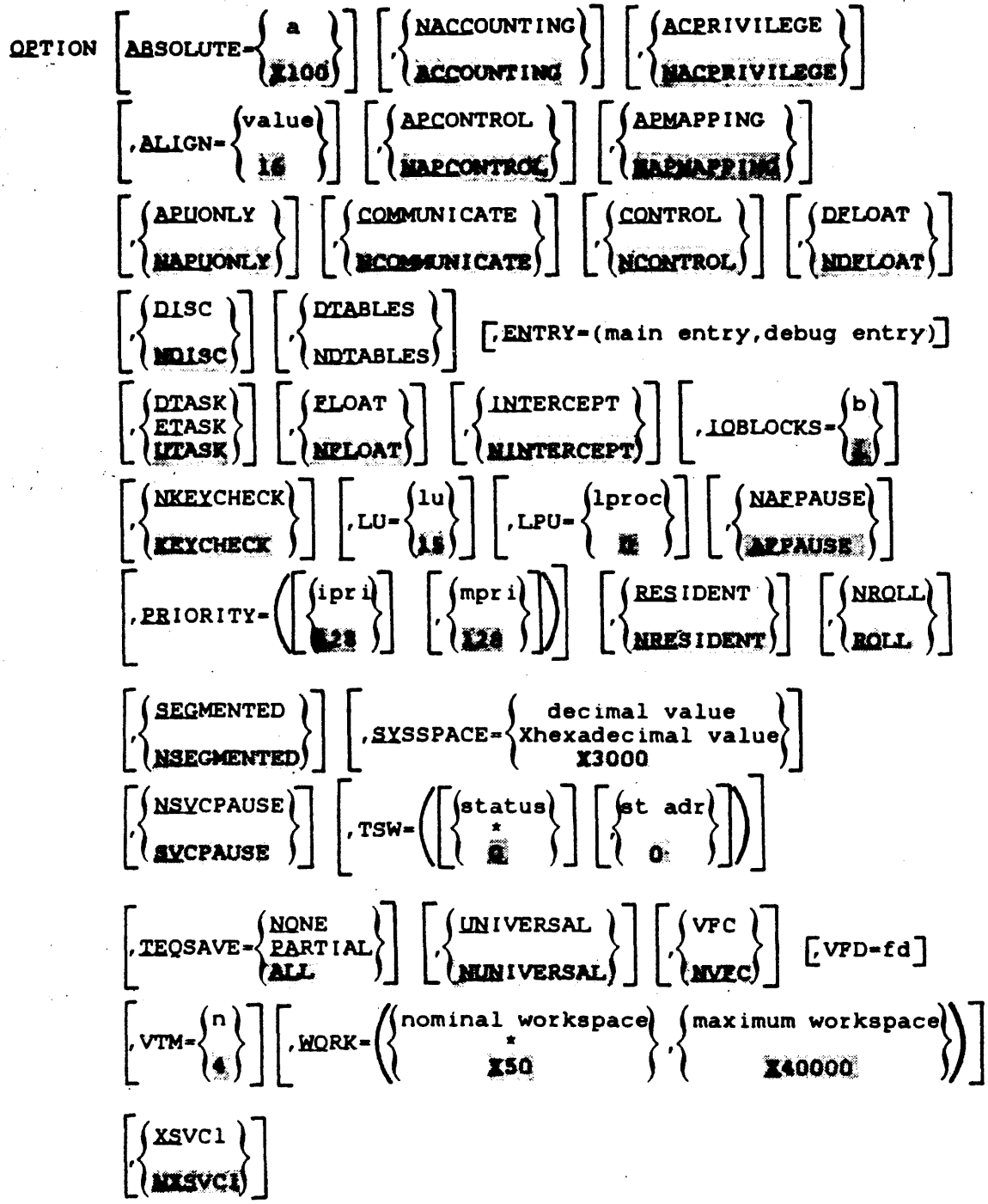


- OPTION Command

The OPTION command has different values for the ENTRY, WORK, and SYSSPACE options. The R01 version of the OPTION command is:



The R02 version of the OPTION command, that appears on Pages 3-33 and A-2, is:



- SHARED Command

The SHARED command in the R01 manual is replaced by the RESOLVE command on Pages 3-52 and A-3 in the R02 manual. The SHARED command syntax is:

```
SHARED    [fd]  [,NAME=segname]
                                  [ R   ]
                                  [ E   ]
,ACCESS=                          [ RE  ]  [,ADDRESS= { m0000 }]
                                  [ RW  ]             { *     }
                                  [ RWE ]
[,STRUCTURE=(name1 [/size]  [,...,namen [/sizen]])]
[,SIZE=([min  [,max]])]
```

The R02 RESOLVE command syntax is:

```
RESOLVE    [fd] [,NAME=package name]
                                  [ R   ]
                                  [ E   ]
,ACCESS=                          [ RE  ]  [,ADDRESS=m0000]
                                  [ RW  ]
                                  [ RWE ]
[,STRUCTURE=(name1 [/size1] [,...,namen [/sizen]])]
[,SIZE= [min ,max]]
```

The rest of this DCN refers to errors that must be corrected in the R02 version of the Link Manual. This portion of the DCN is not related to the Compatible Link Utility.

- Page iv

  Please delete reference to Table 5-1, and add the following reference after B-2:

  C-1 VIRTUAL TASK MANAGEMENT (VTM) MEMORY FAULT CODES

  with a page reference of C-1.

- Page 5-3

  In the last sentence, please change:

  Absolute-original code... to:

  Absolute-origined code...

- Page 5-4

  In the last paragraph, please change:

  or one of the end of task codes explained in Table 5-1.

  to:

  or one of the memory fault codes explained in Table C-1.

- Page 5-5

  Please delete Table 5-1 from Page 5-5. This table will appear on Page C-1.

- Page C-1

  After the second message, please insert the table from Page 5-5, with the following changes:

  TABLE C-1 VIRTUAL TASK MANAGEMENT (VTM) MEMORY FAULT CODES

Please change the heading for the first column of this table from:

END OF TASK CODES

to:

MEMORY FAULT CODES

- Page C-1

After the second message (MEM FAULT AT ...), please delete the sentence that reads:

xx is the SVC 7 error status.

and replace it with the following sentence:

xx specifies the code that describes the type of memory error fault that occurred. These codes are defined in Table C-1.

- Page C-1

In the explanation for the fifth message (VIRT FD ASGN-ERR...), please change:

xxx is the SVC...

to:

xx is the SVC...

- Page IND-3

  Under the alphabetical heading V, in the 6th line, please change:

  end of task codes

  to:

  memory fault codes

  with a page reference of C-1.