

# HP 3000 Computer System



# MPE Segmenter

## Reference Manual



5303 STEVENS CREEK BLVD., SANTA CLARA, CALIFORNIA, 95050

**HP Computer Museum**  
**[www.hpmuseum.net](http://www.hpmuseum.net)**

**For research and education purposes only.**

### **NOTICE**

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another program language without the prior written consent of Hewlett-Packard Company.

# LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the most recent date on which the technical material on any given page was altered. If a page is simply re-arranged due to a technical change on a previous page, it is not listed as a changed page. Within the manual, changes are marked with a vertical bar in the margin.

<b>Pages</b>	<b>Effective Date</b>	<b>Pages</b>	<b>Effective Date</b>
ii to viii . . . . .	Feb 1977	3-1 to 3-24 . . . . .	Feb 1977
1-1 to 1-5. . . . .	Feb 1977	4-1 to 4-5. . . . .	Feb 1977
2-1 to 2-34. . . . .	Feb 1977	I-1 to I-2 . . . . .	Feb 1977

# PRINTING HISTORY

New editions incorporate all update material since the previous edition. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The date on the title page and back cover changes only when a new edition is published. If minor corrections and updates are incorporated, the manual is reprinted but neither the date on the title page and back cover nor the edition change.

First Edition ..... June 1976  
Second Edition ..... Feb 1977

This manual describes the Multiprogramming Executive (MPE) Segmenter Subsystem and is one of the set of manuals that document the MPE Operating System for HP 3000 Computer Systems.

The content of this manual is as follows:

- Section I      Introduction to the MPE Segmenter
- Section II     Command and Intrinsic Specifications
- Section III    Using the Segmenter
- Section IV    MPE Segmenter Error Messages
- Index

# CONVENTIONS USED IN THIS MANUAL

## NOTATION

## DESCRIPTION

[ ]	An element inside brackets is <i>optional</i> . Several elements stacked inside a pair of brackets means the user may select any one or none of these elements.  Example: $\left[ \begin{array}{l} A \\ B \end{array} \right]$ user may select A or B or neither
{ }	When several elements are stacked within braces the user <b>must</b> select one of these elements.  Example: $\left\{ \begin{array}{l} A \\ B \\ C \end{array} \right\}$ user must select A or B or C.
italics	Lowercase italics denote a parameter which must be replaced by a user-supplied variable.  Example: CALL <i>name</i> <i>name</i> one to 15 alphanumeric characters.
underlining	Dialogue: Where it is necessary to distinguish user input from computer output, the input is underlined.  Example: NEW NAME? <u>ALPHA1</u>
superscript C	Control characters are indicated by a superscript C  Example: Y <sup>C</sup>
<i>return</i>	<i>return</i> in italics indicates a carriage return
<i>linefeed</i>	<i>linefeed</i> in italics indicates a linefeed
...	A horizontal ellipsis indicates that a previous bracketed element may be repeated, or that elements have been omitted.

# CONTENTS

Section I	Page	Section III	Page
<b>INTRODUCTION TO THE MPE SEGMENTER</b>		<b>USING THE SEGMENTER</b>	
Relocatable Libraries .....	1-1	Accessing and Exiting the Segmenter .....	3-1
Segmented Libraries .....	1-2	Designating a USL File .....	3-1
Global Variables .....	1-4	Listing RBM's .....	3-8
Commands and Intrinsic .....	1-4	Activating Entry Points .....	3-9
		Creating New USL's .....	3-9
Section II	Page	Transferring RBM's .....	3-10
<b>COMMAND AND INTRINSIC SPECIFICATIONS</b>		Preparing Program Files .....	3-10
Commands .....	2-1	Deleting RBM's .....	3-11
MPE Commands .....	2-1	Deactivating Entry Points .....	3-11
Segmenter Commands .....	2-2	Assigning New Segment Names to RBM's .....	3-11
Intrinsic .....	2-2	Creating and Maintaining Relocatable Libraries (RL's) .....	3-11
RBM Entry Points .....	2-3	Creating a Relocatable Procedure Library (RL) File .....	3-12
ADDRL Segmenter Command .....	2-4	Adding a Procedure to an RL .....	3-12
ADDSL Segmenter Command .....	2-5	Listing Procedures in an RL .....	3-12
ADJUSTUSLF Intrinsic .....	2-6	Designating RL's for Management .....	3-14
AUXUSL Segmenter Command .....	2-8	Deleting an Entry Point or Procedure from an RL .....	3-14
BUILDRL Segmenter Command .....	2-9	Creating and Maintaining Segmented Libraries (SL's) .....	3-17
BUILDSDL Segmenter Command .....	2-10	Creating a Segmented Procedure Library (SL) File .....	3-17
BUILDUSL Segmenter Command .....	2-11	Adding a Procedure Segment to an SL .....	3-17
CEASE Segmenter Command .....	2-12	Listing Procedures in an SL .....	3-17
COPY Segmenter Command .....	2-13	Designating SL's for Management .....	3-20
EXIT Segmenter Command .....	2-14	Deleting an Entry Point from an SL .....	3-20
EXPANDUSLF Intrinsic .....	2-15	Setting RBM Internal Flags .....	3-20
HIDE Segmenter Command .....	2-17	Using MPE Intrinsic to Manipulate USL Files ...	3-23
INITUSLF Intrinsic .....	2-18	Initializing Buffers for USL Files .....	3-24
LISTRL Segmenter Command .....	2-20	Changing the Directory Block/Information Block Size on a USL File .....	3-24
LISTSL Segmenter Command .....	2-21	Changing the Size of a USL File .....	3-24
LISTUSL Segmenter Command .....	2-22		
NEWSEG Segmenter Command .....	2-23	Section IV	Page
PREPARE Segmenter Command .....	2-24	<b>MPE SEGMENTER ERROR MESSAGES</b> .....	4-1
PURGERBM Segmenter Command .....	2-26		
PURGERL Segmenter Command .....	2-27	Index .....	I-1
PURGESL Segmenter Command .....	2-28		
REVEAL Segmenter Command .....	2-29		
RL Segmenter Command .....	2-30		
:SEGMENTER MPE Command .....	2-31		
SL Segmenter Command .....	2-32		
USE Segmenter Command .....	2-33		
USL Segmenter Command .....	2-34		





# ILLUSTRATIONS

Title	Page	Title	Page
Compiling Program Units into a USL .....	3-2	Using the Segmenter Commands RL and PURGERL .....	3-16
Using the Segmenter USL, LISTUSL, USE, BUILDUSL, AUXUSL, COPY, PREPARE, and EXIT Commands .....	3-6	Adding a Function to an SL and Using the Segmenter BUILDSL, ADDSL, and LISTSL Commands .....	3-18
Using the Segmenter PURGERBM, CEASE, and NEWSEG Commands .....	3-7	Referencing and SL Procedure .....	3-21
Using the Segmenter Commands BUILDRL, ADDRL, and LISTRL .....	3-13	Using the Segmenter Commands SL and PURGESL .....	3-22
Referencing a Relocatable Library Procedure .....	3-15		

# TABLES

Title	Page	Title	Page
Segmenter Commands and Intrinsic .....	1-5	Segmenter Error Messages .....	4-2

# INTRODUCTION TO THE MPE SEGMENTER

SECTION

I

The MPE Segmenter performs an intermediary function between compiling the source language and running the program. The following are the steps in compiling and executing a program, and a description of some of the uses of the Segmenter:

- The source program is *compiled* into a User Subprogram Library (USL). If one or more parts of the program are changed, the changed parts (subroutines) can be recompiled into the USL file and old copies of these subroutines will be deactivated. A subroutine exists in the USL as a Relocatable Binary Module (RBM). All versions of subroutines exist in the USL, with only the most recent version being treated as active.
- The Segmenter is used to reactivate previous versions of RBM's and to deactivate the current RBM. RBM's also can be added or deleted by the user with the Segmenter.
- Next, the USL file is *prepared* into a program file. The Segmenter is used to accomplish this, using the MPE :PREP command (calling the Segmenter implicitly) or explicitly calling the Segmenter with the MPE :SEGMENTER command and using the Segmenter PREPARE command. In preparing a program, you may specify the name of a Relocatable Library (RL) which contains subroutines (internal routines) called by the program being prepared. This effectively binds the routines to the program so that they can be found by the system when the subroutines are called for by the program. These Relocatable Libraries are created and managed by the Segmenter.
- Now the program can be *executed* by the MPE :RUN command. Any subroutines not yet found are presumed to be in the system Segmented Library (SL). However, you also may establish your own SL's. In addition, subroutines can be added and deleted by you with the Segmenter.

## RELOCATABLE LIBRARIES

A Relocatable Library (RL) is a specially-formatted file that is searched at *program preparation* time to satisfy references to external procedures required by a user program. Within such libraries, these procedures are placed in a single segment and linked to the user program in the resulting program file. Since a segment containing procedures from an RL is specifically constructed for a given program, it cannot be shared concurrently by different programs. Instead, different copies of the required procedures are made and segmented for each program requesting them.

A Relocatable Library is always a permanent file, but it need not belong to the user's log-on group, and may have any local file name. RL routines go *into* the program file after which time they are handled just like any other segment in the program. Procedures in an RL may not contain TRACE variables.

The RL to be searched during preparation of the user program is specified by the *filename* parameter in the MPE :PREP and :PREPRUN commands or the Segmenter PREPARE command. For example, to specify that a Relocatable Library named RL in the group GROUP2 of the user's log-on account be searched during preparation of a program from the USL ALPHA1 to the program file ALPHA2, the following :FILE and :PREP MPE commands are used:

```
:FILE MYRL = RL.GROUP2
:PREP ALPHA1, ALPHA2; RL=*MYRL
```

See the *MPE Commands Reference Manual* for a discussion of the :FILE and :PREP commands.

## SEGMENTED LIBRARIES

Segmented Libraries (SL's) are specially-formatted files that are searched at *program-allocation* time to satisfy references to external procedures. These libraries, like program files, contain procedures in segmented (prepared) form. An individual procedure may be the only procedure in its segment, or may exist in a segment containing many other procedures. When a procedure is referenced, the segment containing it is loaded along with other segments referenced by that segment. Since the segmentation is not altered when different programs reference procedures in an SL, these segments may be shared concurrently by many programs.

From one to three SL's may be searched during allocation of the user's program. These are

1. The Group Library SL, which is actually the library of the group under which the program file is stored. This library is readable by any user who can access this group.
2. The account's Public Library SL, which is the library of the public group of the account under which the program file is stored. This library is readable by any user who can access this account.
3. The System Library SL, which is the library of the public group of the system account. This library can be accessed by all users of the system.

The order in which the libraries are searched is specified by the LIB parameter in the MPE :RUN and :PREPRUN commands that request program allocation/execution, as follows:

### LIB Parameter

LIB = G

### Specification

The libraries are searched in this order:

- Group Library
- Public Library
- System Library

LIB = P

The libraries are searched in this order:

- Public Library
- System Library

LIB = S

Only the System Library is searched.

When no LIB parameter is specified, the default is LIB = S.

For example, to specify that the public library and then the system library be searched during allocation, the following command is used when executing the program ALPHA2:

```
:RUN ALPHA2; LIB=P
```

The name of the System Account is always SYS. Within an account, the name of any public group is always PUB; the name of the segmented library is always SL. Thus, when referencing a library file in an MPE command or intrinsic, the following rules apply.

1. *If the user's program file is a permanent file, with the name X.Y.Z (in the filereference format), then:*

Group Library SL is referenced as SL.Y.Z

Public Library SL is referenced as SL.PUB.Z

System Library SL is referenced as SL.PUB.SYS

If Y.Z = PUB.SYS, then only LIB = S is a legal entry in the :PREPRUN or :RUN command.

If Y.Z = PUB.Z, then LIB = P or LIB = S are legal entries in the :PREPRUN or :RUN command.

During allocation, the proper libraries are determined by the position of the program file in the file system directory.

2. *If the user's program file is not a permanent file, but is a job/session temporary or passed file with the name X.Y.Z, where Y is the log-on group and Z is the log-on account, or is nameless (such as the file \$OLDPASS) then:*

Group Library SL = SL.Y.Z

Public Library SL = SL.PUB.Z

System Library SL = SL.PUB.SYS

In the above cases, LIB = G, LIB = P, or LIB = S are all legal entries in the :PREPRUN or :RUN command.

During allocation, the proper libraries are determined by the log-on group and account.

As an example of how SL's are used, procedures called frequently by many users throughout the system are stored in the System SL. Those used frequently throughout an account can be stored in a Public SL. Those used by only a few users in a single group, less frequently, might be stored in a Group SL.

## **GLOBAL VARIABLES**

Because the global area of the user's stack is already formed at the time a user's program references an external procedure in an SL, the library procedures cannot contain global variables. (The addresses of global variables might be different for each program referencing them, thus the SL procedure would no longer be sharable.) From the user's standpoint, this means that SPL procedures in a library cannot contain TRACE, EXTERNAL or OWN variables. Also, FORTRAN procedures in a library cannot contain: DATA, COMMON, LABELED COMMON, or TRACE variables. FORTRAN procedures may contain FORMAT statements except for those containing an H specification (Hollerith string) which also are referred to by READ statements. They may contain READ and WRITE statements provided the unit number is specified in an integer variable (i.e., not a constant). They may not contain ACCEPT or DISPLAY statements.

## **COMMANDS AND INTRINSICS**

MPE commands and intrinsics used with the Segmenter and Segmenter commands are summarized in table 1-1.

**Table 1-1. Segmenter Commands and Intrinsic**

<b>NAME</b>	<b>CATEGORY</b>	<b>FUNCTION</b>
ADDRL	Segmenter command	Adds a procedure to the currently-managed RL.
ADDSL	Segmenter command	Adds a segment to an SL.
ADJUSTUSLF	Intrinsic	Adjusts directory space in a USL file.
AUXUSL	Segmenter command	Identifies an auxiliary USL file.
BUILDRL	Segmenter command	Creates a permanent, formatted RL file.
BUILDSDL	Segmenter command	Creates a permanent, formatted SL file.
BUILDUSL	Segmenter command	Creates a new (job/session temporary) USL file.
CEASE	Segmenter command	Deactivates one or more entry points in the currently-managed USL.
COPY	Segmenter command	Copies one or more RBM's from another USL to the currently-managed USL.
EXIT	Segmenter command	Terminates Segmenter operation.
EXPANDUSLF	Intrinsic	Changes length of a USL file.
HIDE	Segmenter command	Sets an internal flag on in an RBM.
INITUSLF	Intrinsic	Initializes buffer for a USL to the empty state.
LISTRL	Segmenter command	Lists all procedure entry points and external references in the currently-managed RL.
LISTSL	Segmenter command	Lists the procedures in the currently-managed SL.
LISTUSL	Segmenter command	Lists the names of the RBM's in the currently-managed USL.
NEWSEG	Segmenter command	Changes the segment name associated with an RBM.
PREPARE	Segmenter command	Prepares the active RBM's in a USL into a program file.
PURGERBM	Segmenter command	Deletes one or more RBM's from a USL.
PURGERL	Segmenter command	Deletes an entry point of a procedure, or the entire procedure, from an RL.
PURGESL	Segmenter command	Deletes a segment entry point, or the entire segment, from an SL.
REVEAL	Segmenter command	Sets an internal flag off in an RBM.
RL	Segmenter command	Identifies an RL.
:SEGMENTER	MPE command	Accesses the Segmenter.
SL	Segmenter command.	Identifies an SL.
USE	Segmenter command	Activates one or more entry points in the currently-managed USL.
USL	Segmenter command	Identifies a USL file.



# COMMAND AND INTRINSIC SPECIFICATIONS

SECTION

II

Specifications for commands and intrinsics used with the Segmenter are presented in this section in alphabetical order.

## COMMANDS

The command specifications contain the following information:

- The command name.
- The type of command (Segmenter or MPE command). This information is shown in italics directly under the command name.
- A brief summary of the command's function.
- Syntax. The command syntax is highlighted by being shown in a shaded box.
- Parameter definitions, including meaning, constraints, and defaults.
- Examples.
- Text discussion. (The page in this manual where usage of the command is discussed.)

## MPE COMMANDS

MPE commands consist of the following elements:

- A colon, required in all cases as an MPE command identifier. In an interactive session, MPE displays the colon on the terminal when it is ready to accept a command. In a batch job, you must enter the colon in the first column of the command record.
- The command name, which must follow immediately after the colon. MPE prohibits embedded blanks within the command name, or between the colon and the command name. A blank signifies the end of the command name.



- Parameters, if any, follow the command name. You must separate the parameter list from the command name by one or more blanks. When several parameters are used in a list, they must be separated by commas (delimiters). Any delimiter can be surrounded by any number of blanks; however, blanks may not be embedded within parameters. The end of the parameter list is indicated by a carriage return in a session or the end of the record in a job. *Positional* parameters have significance due to their position in the parameter list. For example, in the following instance

:COMMAND *parameter1, parameter2, parameter3*

*parameter1* must always be specified before *parameter2* or *parameter3*. If a positional parameter is omitted from a parameter list, commas are used to denote this as follows:

:COMMAND *parameter1,,parameter3* (From middle of list)

:COMMAND *,parameter2,parameter3* (From beginning of list)

:COMMAND *parameter1* (From end of list. Commas are not required.)

## SEGMENTER COMMANDS

Segmenter commands consist of the following elements:

- The command name. In an interactive session, the Segmenter displays a dash (–) as a prompt when it is ready to accept a command. Note that the prompt character is not allowed when entering Segmenter commands in a batch job. The command name is shown in this section with no preceding prompt character.
- Parameters, if any, follow the command name. A blank character must be used between the end of the command name and the first parameter.

## INTRINSICS

The intrinsic specifications contain the following information:

- The intrinsic name. The word *intrinsic*, in italics, directly under the intrinsic name identifies it as an intrinsic.
- A brief summary of the function of the intrinsic.
- The complete intrinsic call description, highlighted by being enclosed in a shaded box. The format is as follows:

I IV IA  
errnum:= INTUSLF(*uslfnm,rec0*);

- For those intrinsics which return a value to the calling process (type procedures), the return is described. For example, INITUSLF is an integer procedure which returns an error number to *errnum* if an error occurs. The superscript, I, over *errnum* identifies INITUSLF as an integer procedure.
- All parameters are described. Required parameters, such as *uslfnm* above, are shown in ***bold face italics***. Superscripts (such as IV) are used to denote the types of parameters and whether they must be passed by *value*, instead of by *reference* (the default case). See Section I of the *MPE Intrinsics Reference Manual* for a discussion of passing parameters by value and by reference.

Superscripts have the following meanings:

- I Integer
- IA Integer array
- IV Integer by value

- Condition codes are included for each intrinsic.
- Text discussion. (The page in this manual where usage of the intrinsic is discussed.)

## RBM ENTRY POINTS

Each RBM is identified by the symbolic name (label) of the primary entry point of the program unit stored in the RBM; the RBM may be associated with a segment name, determined during compilation, that identifies the segment to which it will belong. An activity bit is associated with each entry point (the primary entry point and any secondary entry points). This activity bit determines whether the program unit currently can be entered at the corresponding entry point. When a compiler writes an RBM on a USL, all entry points in the RBM are set to the *active* (entry allowed) state. Through the MPE Segmenter, they can then be switched from that state to the *inactive* (entry disallowed) state, and back again, by the user.

When a program file is prepared from the USL, all RBM's having at least one active entry point are extracted from the USL for segmentation on the program file. Those having identical segment names are placed in the same segment. To permit the creation of a program file that can be executed properly, only one outer-block or main-program RBM can have active entry points, along with the RBM's for the subprograms or procedures on the USL that it references.

Entry points having the same name are permitted in a USL. Each of these is uniquely identified by its name used in conjunction with an index integer; this integer denotes the particular definition of the entry point in chronological terms, beginning with one to indicate the most recent definition. Since the name of an RBM is equivalent to that of its primary entry-point, the name/index identification method allows RBM's containing different versions of the same program unit to be referenced by the same name used with different index integers. Thus, to identify the most recent version of a program unit in an RBM named START, the user would specify START (1); to indicate the next most recent version, he would specify START (2). A special convention (index = 0) is used to indicate the most recent *active* definition; for example, START (0).

# ADDRL

*Segmenter Command*

Adds a procedure to the currently-managed Relocatable Library (RL).

## SYNTAX

```
ADDRL name [(index)]
```

## PARAMETERS

- name*                    The name (primary entry point) of the Relocatable Binary Module (RBM) containing the procedure to be added to the RL. (Required parameter.)
- index*                    An integer further identifying the RBM. The index may be used when the currently-managed USL contains more than one active RBM of this name. If *index* is omitted, a value of 0 is assigned by default. (Optional parameter.)

## EXAMPLE

ADDRL XBM                Adds an RBM containing one procedure named XBM.

## TEXT DISCUSSION

Page 3-12.

Adds a procedure to a Segmented Library (SL).

## SYNTAX

```
ADDSL name [,PMAP]
```

## PARAMETERS

*name*                    The name of the segment to be added to the SL. (Required parameter.)

*PMAP*                    An indication that a listing describing the prepared segment will be produced on the file specified in the MPE :SEGMENTER command parameter *listfile*, which defaults to \$STDLIST. If this parameter is omitted, the prepared segment is not listed. (Optional parameter.)

## EXAMPLE

ADDSL SL1,PMAP           Adds segment SL1 to the currently-managed SL and produces a listing.

## TEXT DISCUSSION

Page 3-17.

# ADJUSTUSLF

*Intrinsic*

Adjusts directory space in a User Subprogram Library (USL) file.

## SYNTAX

```
I           IV      IV  
errnum := ADJUSTUSLF(uslfnm, records);
```

## FUNCTIONAL RETURN

This intrinsic returns either an error number (if an error occurs) or zero (no error).

## PARAMETERS

*uslfnm*            *integer by value (required)*

A word supplying the file number of the USL file (as returned by FOPEN).

*records*           *integer by value (required)*

A word supplying a signed record count. If *records* is greater than zero, the information block is moved toward the end-of-file in the USL file, increasing the space available for the directory block and decreasing the space available for the information block. If *records* is less than zero, the information block is moved toward the start of the USL file, decreasing the directory-block space and increasing the information-block space.

## CONDITION CODES

CCE                Request granted.

CCG                Not returned by this intrinsic.

CCL                Request denied. One of the following error numbers is returned.

# ADJUSTUSLF

*Intrinsic*

<b>Error Number</b>	<b>Meaning</b>
0	The file specified by <i>uslfnm</i> was empty, or an unexpected end-of-file was encountered when reading the old <i>uslfnm</i> , or an unexpected end-of-file was encountered when writing on the new <i>uslfnm</i> .
1	Unexpected input/output error occurred. This can occur on the old <i>uslfnm</i> or the new <i>uslfnm</i> to which the intrinsic is copying the information.
3	Your request attempted to exceed the maximum file directory size (32,768 words).
6	Insufficient space was available in the USL file information block.
7	The intrinsic was unable to open the new USL file.
8	The intrinsic was unable to close (purge) the old USL file.
9	The intrinsic was unable to close (purge) the new USL file.
10	The intrinsic was unable to close \$NEWPASS.
11	The intrinsic was unable to open \$OLDPASS.

## TEXT DISCUSSION

Page 3-24.

# AUXUSL

*Segmenter Command*

Identifies an auxiliary USL file..

## SYNTAX



AUXUSL *filereference*

## PARAMETERS

*filereference*            The name (and optional group and account name) of the auxiliary USL file from which RBM's may be transferred to the current USL file. (Required parameter.)

## EXAMPLE

AUXUSL FILE1            Identifies the auxiliary USL file FILE1.

## TEXT DISCUSSION

Page 3-10.

Creates a permanent, formatted Relocatable Library (RL) file.

## SYNTAX

```
BUILDRL filereference, records, extents
```

## PARAMETERS

<i>filereference</i>	The filename of the new RL, in the <i>filereference</i> format, (optionally including group and account identifiers). (Required parameter.)
<i>records</i>	The total maximum file capacity, specified in terms of 128-word binary logical records. (Required parameter.)
<i>extents</i>	The total number of disc extents that can be dynamically allocated to the file as logical records are written to it. The size of each extent is determined by the <i>records</i> parameter value divided by the <i>extents</i> parameter value. The <i>extents</i> value must be an integer from 1 to 16 (for Series I Computer Systems) or 1 to 32 (for Series II Computer Systems).

## EXAMPLE

BUILDRL R1.G1.A1, 500, 1      Creates an RL file named R1.G1.A1. The RL file can contain 500 records, and is allocated one disc extent.

## TEXT DISCUSSION

Page 3-12.



# BUILDSL

*Segmenter Command*

Creates a permanent, formatted Segmented Library (SL) file.

## SYNTAX

```
BUILDSL filereference, records, extents
```

## PARAMETERS

- filereference* Any file whose local name is SL. You can create an SL file with a local name other than SL, but such a file cannot be searched by the MPE :RUN command or the CREATE or LOADPROC intrinsics unless it is renamed SL. (Required parameter.)
- records* The total maximum file capacity, specified in terms of 128-word binary logical records. (Required parameter.)
- extents* The total number of disc extents that can be dynamically allocated to the file as logical records are written to it. The size of each extent is determined by the *records* parameter value divided by the *extents* parameter value. The *extents* value must be an integer from 1 to 16 (for Series I Computer Systems) or 1 to 32 (for Series II Computer Systems).

## EXAMPLE

BUILDSL SEARCH, 300, 1      Creates an SL file named SEARCH with 300 records and one disc extent.

## TEXT DISCUSSION

Page 3-17.

Creates a new (job/session temporary) User Subprogram Library (USL) file.

## SYNTAX

**BUILDUSL** *filereference, records, extents*



## PARAMETERS

- filereference*      The name (file designator) assigned to the new USL file. This is a fully-qualified file reference that may include file name, group name, and account name, plus a lockword. (Required parameter.)
- records*            The length of this file, specified in terms of 128-word binary logical records. (Required parameter.)
- extents*            The total number of disc extents that can be dynamically allocated to the file as logical records are written to it. The size of each extent is determined by the *records* parameter value divided by the *extents* parameter value. The *extents* value must be an integer from 1 to 16 (for Series I Computer Systems) or 1 to 32 (for Series II Computer Systems).

## EXAMPLE

BUILDUSL FILE2, 200, 1    Creates a new USL file, FILE2, with 200 records and one disc extent.

## TEXT DISCUSSION

Page 3-9.

# CEASE

*Segmenter Command*

Deactivates one or more entry points in the currently-managed User Subprogram Library (USL).

## SYNTAX

```
CEASE [ ENTRY,  
        UNIT,  
        SEGMENT, ] name[(index)]
```

## PARAMETERS

ENTRY	Deactivates the entry point indicated by <i>name</i> ( <i>index</i> ).
UNIT	Deactivates all entry points in the Relocatable Binary Module (RBM) indicated by <i>name</i> ( <i>index</i> ).
SEGMENT	Deactivates all entry points in all RBM's associated with segment <i>name</i> .  The default is ENTRY. (Optional parameter.)
<i>name</i>	The name of the entry point, RBM, or segment referenced. (Required parameter.)
<i>index</i>	An integer further specifying the entry point <i>name</i> . The index may be used when the USL contains more than one entry point or RBM of this name. If <i>index</i> is omitted, a default value of 0 is assigned. If SEGMENT is specified, the <i>index</i> parameter does not apply. (Optional parameter.)

## EXAMPLE

CEASE ENTRY, BEGIN2            Deactivates ENTRY point BEGIN2.

## TEXT DISCUSSION

Page 3-11.

Copies one or more Relocatable Binary Modules (RBM's) from another User Subprogram Library (USL) to the currently-managed USL.

## SYNTAX

```
COPY [ UNIT,  
      SEGMENT, ] name[(index)]
```

## PARAMETERS

UNIT	Transfers the RBM identified by <i>name</i> ( <i>index</i> ) from the source USL.
SEGMENT	Transfers <i>all</i> RBM's associated with the segment <i>name</i> from the source USL. (The source USL is the one specified by an AUXUSL command.)  The default parameter is UNIT. (Optional parameter.)
<i>name</i>	Identifies the RBM to be transferred if UNIT is specified. If SEGMENT is specified, <i>name</i> identifies the segment from which all RBM's are transferred. (Required parameter.)
<i>index</i>	An integer further specifying the RBM name. The <i>index</i> may be used when more than one RBM of this name exists in the USL. If <i>index</i> is omitted, a default value of 0 is assigned. The <i>index</i> parameter is ignored if SEGMENT is specified. (Optional parameter.)

## EXAMPLE

COPY SEGMENT, SEGNAME      Copies all RBM's associated with the segment name SEGNAME.

## TEXT DISCUSSION

Page 3-10.

# EXIT

*Segmenter Command*

Terminates Segmenter operation

## SYNTAX



## TEXT DISCUSSION

Page 3-1.

Changes length of a User Subprogram Library (USL) file.

```
      I           IV      IV  
filenum := EXPANDUSLF(uslfnm,records);
```

## FUNCTIONAL RETURN

This intrinsic returns the new file number. If an error occurs, the error number is returned instead of the new file number. The condition code therefore *must* be tested immediately on return from this intrinsic. If an error number were to be used as a file number, unpredictable results would occur.

## PARAMETERS

*uslfnm*            *integer by value (required).*

A word identifier supplying the file number of the file.

*records*           *integer by value (required).*

A signed integer specifying the number of records by which the length of the USL file is to be changed. If *records* is positive, the new USL file is longer than the old USL. If *records* is negative, the new USL file is shorter than the old USL.

## CONDITION CODES

CCE                Request granted. The new file number is returned.

CCG                Not returned by this intrinsic.

CCL                Request denied. One of the following error numbers is returned.

# EXPANDUSLF

*Intrinsic*

<b>Error Number</b>	<b>Meaning</b>
0	The file specified by <i>uslfnm</i> was empty, or an unexpected end-of-file was encountered when reading the old <i>uslfnm</i> , or an unexpected end-of-file was encountered when writing on the new <i>uslfnm</i> .
1	Unexpected input/output error occurred. This can occur on the old <i>uslfnm</i> or the new <i>uslfnm</i> to which the intrinsic is copying the information.
3	Your request attempted to exceed the maximum file directory size (32,768 words).
6	Insufficient space was available in the USL file information block.
7	The intrinsic was unable to open the new USL file.
8	The intrinsic was unable to close (purge) the old USL file.
9	The intrinsic was unable to close (purge) the new USL file.
10	The intrinsic was unable to close \$NEWPASS.
11	The intrinsic was unable to open \$OLDPASS.

## TEXT DISCUSSION

Page 3-24.

Sets an internal flag on in a Relocatable Binary Module (RBM).

## SYNTAX

```
HIDE name [(index)]
```

## PARAMETERS

- name*                    The name of the RBM entry point in the currently-managed USL whose internal flag is to be set. (Required parameter.)
- index*                    An integer further specifying the entry-point *name*. The index may be used when the RBM contains more than one entry point of this name. If *index* is omitted, 0 is assigned by default. (Optional parameter.)

## EXAMPLE

HIDE XBM                Sets the internal flag on in the RBM named XBM.

## TEXT DISCUSSION

Page 3-23.



# INITUSLF

*Intrinsic*

Initializes a buffer corresponding to record 0 for a User Subprogram Library (USL) to the empty state.

```
I           IV  IA
errnum := INITUSLF(uslfnm,rec0);
```

## FUNCTIONAL RETURN

This intrinsic returns an error number if an error occurs. If no error occurs, no value is returned.

## PARAMETERS

*uslfnm*            *integer by value (required)*

A word identifier supplying the file number of the USL file.

*rec0*            *integer array (required)*

A 128-word buffer, corresponding to the first record of the USL file (record 0), to be initialized to the empty state. This buffer should be set to all zeros. The intrinsic will set certain values in record 0 before returning to the calling program. See the *MPE Ininsics Reference Manual* for record 0 format.

## CONDITION CODES

CCE            Request granted.

CCG            Not returned by this intrinsic.

CCL            Request denied. One of the error numbers listed below is returned.

<b>Error Number</b>	<b>Meaning</b>
0	The file specified by <i>uslfnm</i> was empty, or an unexpected end-of-file was encountered when reading the old <i>uslfnm</i> , or an unexpected end-of-file was encountered when writing on the new <i>uslfnm</i> .
1	Unexpected input/output error occurred. This can occur on the old <i>uslfnm</i> or the new <i>uslfnm</i> to which the intrinsic is copying the information.
3	Your request attempted to exceed the maximum file directory size (32,768 words).
6	Insufficient space was available in the USL file information block.
7	The intrinsic was unable to open the new USL file.
8	The intrinsic was unable to close (purge) the old USL file.
9	The intrinsic was unable to close (purge) the new USL file.
10	The intrinsic was unable to close \$NEWPASS.
11	The intrinsic was unable to open \$OLDPASS.

## TEXT DISCUSSION

Page 3-24.

# **LISTRL**

*Segmenter Command*

Lists all procedure entry points and external references in the currently-managed Relocatable Library (RL).

## **SYNTAX**



## **TEXT DISCUSSION**

Page 3-12.

Lists the procedures in the currently-managed Segmented Library (SL).

## SYNTAX



## TEXT DISCUSSION

Page 3-17.

# LISTUSL

*Segmenter Command*

Lists the names of the Relocatable Binary Modules (RBM's) in the currently-managed User Subprogram Library (USL).

## SYNTAX



## TEXT DISCUSSION

Page 3-8.

Changes the segment name associated with a Relocatable Binary Module (RBM).

## SYNTAX

```
NEWSEG newsegname, rbmname [(index)]
```

## PARAMETERS

<i>newsegname</i>	The new segment name to be associated with the RBM. (Required parameter.)
<i>rbmname</i>	The name of the RBM whose segment name is to be changed. (Required parameter.)
<i>index</i>	An integer further specifying the RBM name. The index may be used when more than one RBM of the same name exists in the User Subprogram Library (USL). If <i>index</i> is omitted, a default value of 0 is assigned. (Optional parameter.)

## EXAMPLE

NEWSEG NEWNAME, RB3	Changes the segment name associated with RBM RB3 to NEWNAME.
---------------------	--

## TEXT DISCUSSION

Page 3-11.

# PREPARE

*Segmenter Command*

Prepares the active Relocatable Binary Modules (RBM's) in a User Subprogram Library (USL) into a program file.

## SYNTAX

```
PREPARE progfile

[;ZERODB]
[;PMAP]
[;MAXDATA = segsz]
[;STACK = stacksz]
[;DL = dlsz]
[;CAP = caplist]
[;RL = filename]
```

## PARAMETERS

*progfile*

The name of the program file onto which the prepared program segments are to be written. If the file named does not exist, the Segmenter will build a job temporary file for you. (Required parameter.)

### NOTE

Code segments in a program file cannot lie across disc extent boundaries. Thus, all segments in such files must be constructed within one extent. See the *MPE Intrinsic Reference Manual* for a discussion of disc extents.

ZERODB

An indication that the initially-defined DL-DB area, and uninitialized portions of the DB-Q (initial) area will be initialized to zero. If this parameter is omitted, these areas are not affected. (Optional parameter.)

*stacksize*

The size of the user's initial local data area (Q (initial) to Z) in the stack, in words. This overrides the *stacksize* estimated by the Segmenter, which applies if the *stacksize* parameter is omitted. (The default is a function of estimated stack requirements for each program unit in the program.) Since it is difficult for the system to predict the behavior of the stack at run time, you may want to override the default by supplying your own estimate with *stacksize*. (Optional parameter.)

*dlsz*

The DL-DB area to be initially assigned to the stack. If the *dlsz* parameter is omitted, a value of zero is used. (Optional parameter.)

# PREPARE

*Segmenter Command*

**PMAP** An indication that a listing describing the prepared program will be produced on the device specified by the :SEGMENTER command parameter *listfile*. If this parameter is omitted, no listing is produced. (Optional parameter.)

*segsiz* Maximum stack area (Z-DL) size permitted, in *words*. This parameter is included if you expect to change the size of the DL-DB or DB-Z areas during process execution. If omitted, MPE assumes that these areas will not be changed. (Optional parameter.)

*caplist* The *capability-class attributes* associated with the user's program; specified as two-character mnemonics. If more than one mnemonic is specified, each must be separated from its neighbor by a comma.

The mnemonics are

IA = Interactive access

BA = Local batch access

PH = Process handling

DS = Data segment management

MR = Multiple resource management

PM = Privileged-mode operation



The user who issues the PREPARE command can only specify capabilities that he himself possesses (through assignment by the Account Manager). If the user does not specify any capabilities, only IA and BA (if the user possesses them) will be assigned to this program. (Optional parameter.)

*filename* The name of a Relocatable Library (RL) to be searched to satisfy external references during preparation. This can be any permanent file of type RL. It need not belong to the log-on group, nor does it have a reserved, local name. This file yields a single segment that is incorporated into the segments of the program file prepared. If *filename* is omitted, no library is searched. (Optional parameter.)

For the *stacksize*, *dlsiz*, and *maxdata* parameters, a value of -1 denotes the default (equivalent to omitting the parameter).

## EXAMPLE

PREPARE PFILE       Prepares the program file named PFILE.

## TEXT DISCUSSION



# PURGERBM

*Segmenter Command*

Deletes one or more Relocatable Binary Modules (RBM's) from a User Subprogram Library (USL).

## SYNTAX

```
PURGERBM [ UNIT,  
          SEGMENT, ] name [(index)]
```

## PARAMETERS

UNIT	Deletes the RBM identified by <i>name</i> ( <i>index</i> ).
SEGMENT	Deletes <i>all</i> RBM's associated with the segment <i>name</i> . The default parameter is UNIT. (Optional parameter.)
<i>name</i>	The name of the RBM to be deleted if UNIT is specified. If SEGMENT is specified, this is the name of the segment in which all RBM's are to be deleted. (Required parameter.)
<i>index</i>	An integer further specifying the RBM name. The index may be used when more than one RBM of this name exists in the USL. If <i>index</i> is omitted, a default value of 0 is assigned. If SEGMENT is specified, this parameter is ignored. (Optional parameter.)

## EXAMPLE

PURGERBM UNIT, XPOINT      Deletes the RBM named XPOINT.

## TEXT DISCUSSION

Page 3-11.

# PURGERL

*Segmenter Command*

Deletes an entry point of a procedure, or the entire procedure, from a Relocatable Library (RL).

## SYNTAX

```
PURGERL [ UNIT,  
         ENTRY, ] name
```

## PARAMETERS

UNIT	Deletes the entire procedure identified by <i>name</i> .
ENTRY	Deletes the entry point identified by <i>name</i> . The default parameter is ENTRY. (Optional parameter.)
<i>name</i>	The name of the procedure to be deleted if UNIT is specified, or the entry point to be deleted if ENTRY is specified. (Required parameter.)

## EXAMPLE

PURGE UNIT, PROC1      Deletes the entire procedure named PROC1.

## TEXT DISCUSSION

Page 3-14.

# PURGESL

*Segmenter Command*

Deletes a segment entry point, or the entire segment, from a Segmented Library (SL).

## SYNTAX

```
PURGESL [ ENTRY,  
          SEGMENT, ] name
```

## PARAMETERS

ENTRY	Deletes the entry point identified by <i>name</i> .
SEGMENT	Deletes the entire segment identified by <i>name</i> . The default parameter is ENTRY. (Optional parameter.)
<i>name</i>	The name of the entry point to be deleted if ENTRY is specified or the name of the segment to be deleted if SEGMENT is specified. (Required parameter.)

## EXAMPLE

PURGESL ENTRY,ENT1           Deletes the entry point named ENT1.

## TEXT DISCUSSION

Page 3-20.

Sets an internal flag off in a Relocatable Binary Module (RBM).

## SYNTAX

```
REVEAL name [(index)]
```

## PARAMETERS

- name*                    The name of the RBM entry point in the User Subprogram Library (USL) whose internal flag is to be set off. (Required parameter.)
- index*                    An integer further specifying the entry point name. The index may be used when the RBM contains more than one entry point of this name. If *index* is omitted, a value of 0 is assigned by default. (Optional parameter.)

## EXAMPLE

REVEAL RBM1            Sets the internal flag off in the RBM entry point named RBM1.

## TEXT DISCUSSION

Page 3-23.

# RL

*Segmenter Command*

Identifies a Relocatable Library (RL).

## SYNTAX



*RL filereference*

## PARAMETERS

*filereference*      The name, and optional group and account name, of the RL. (Required parameter.)

## EXAMPLE

RL R2.G1.A1      Designates the RL file R2.G1.A1 for management.

## TEXT DISCUSSION

Page 3-14.

# :SEGMENTER

*MPE Command*

Accesses the Segmenter.

## SYNTAX

```
:SEGMENTER [listfile]
```

## PARAMETERS

*listfile* An ASCII file (formal designator SEGLIST) to which is written any listable output generated by Segmenter commands. The designator SEGLIST should *not* be used as the *actual* file designator. If *listfile* is omitted, the standard job/session list device (\$STDLIST) is assigned. The only Segmenter commands which use the *listfile* parameter are LISTUSL, LISTSL, LISTRL, PREPARE, and ADDSL. (Optional parameter.)

## EXAMPLE

:FILE OUTPUT;DEV=LP	Equates the file OUTPUT to the line printer.
:SEGMENTER *OUTPUT	Back references the file OUTPUT, sending the listing to the line printer.

## TEXT DISCUSSION

Page 3-1.

# SL

*Segmenter Command*

Identifies a Segmented Library (SL).

## SYNTAX



SL *filereference*

## PARAMETERS

*filereference*      The name of the SL. (Required parameter.)

## EXAMPLE

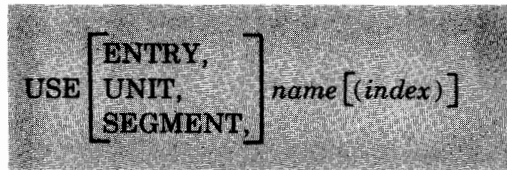
SL SL1      Designates the SL named SL1.

## TEXT DISCUSSION

Page 3-20.

Activates one or more program unit (Relocatable Binary Module) entry points in the User Subprogram Library currently designated for management.

## SYNTAX



```
USE [ ENTRY,  
      UNIT,  
      SEGMENT, ] name [(index)]
```

## PARAMETERS

ENTRY	Activates the entry point indicated by <i>name(index)</i> .
UNIT	Activates all entry points in the RBM indicated by <i>name(index)</i> .
SEGMENT	Activates all entry points in all RBM's associated with the segment <i>name</i> .  The default value is ENTRY. (Optional parameter.)
<i>name</i>	The name of the entry point if ENTRY is specified, or the name of the RBM if UNIT is specified, or the name of the segment if SEGMENT is specified. (Required parameter.)
<i>index</i>	An integer further identifying the entry point name. The index may be used when the USL contains more than one entry point of the same name. If <i>index</i> is omitted, a default value of 0 is assigned. If SEGMENT is specified, the <i>index</i> parameter is ignored. (Optional parameter.)

## EXAMPLE

USE UNIT, RB20            Activates all entry points in the RBM named RB20.

## TEXT DISCUSSION

Page 3-9.



# USL

*Segmenter Command*

Identifies a User Subprogram Library (USL) file.

## SYNTAX

```
USL filereference
```

## PARAMETERS

*filereference*      The name, and optional group and account name, of the USL file.  
(Required parameter.)

## EXAMPLE

USL FILE1      Identifies the USL file FILE1 (in the user's log-on group and account).

## TEXT DISCUSSION

Page 3-1.

# USING THE SEGMENTER

SECTION

III

## ACCESSING AND EXITING THE SEGMENTER

The MPE :SEGMENTER command is used to access the Segmenter. In the following example, the :SEGMENTER command is entered in an interactive session. The Segmenter responds with the following message and displays a dash prompt character.

```
: SEGMENTER  
SEGMENTER SUBSYSTEM (C.0)  
-
```

Segmenter commands can now be entered. Note that the prompt character is not allowed when entering Segmenter commands in a batch job.

The EXIT command is used to terminate Segmenter operation:

```
-EXIT  
END OF SUBSYSTEM
```

### NOTE

All examples in this manual were run in an interactive session and therefore will include the dash prompt character. Note also that user input is underlined.

Figure 3-1 shows three FORTRAN subroutines and a short main program which were compiled into the User Subprogram Library USL1. Note that the subroutines all have the same name and that only lines 5 and 6 are different.

Figures 3-2 and 3-3 show this USL being manipulated with various Segmenter commands.

## DESIGNATING A USL FILE

To modify or manage an existing USL with the Segmenter, you first must designate this USL with the Segmenter USL command.

Item 1 in figure 3-2 shows USL1 being designated for management. All subsequent USL-modification commands will refer implicitly to this USL until a new USL command is entered, superseding the previous USL command, or until a BUILDUSL command creates a new USL, thus effectively superseding the USL command.

```
:BUILD USL1;CODE=USL;DISC=,1
:FORTRAN XMPL2,USL1
```

```
PAGE 0001 HP32102B.00.05 (C) HEWLETT-PACKARD CO. 1976
```

```
00001000      PROGRAM SQUARE
00002000      10  FORMAT('0',T2,"NUMBER",T12,"SQUARE ROOT")
00003000      20  FORMAT(T2,F4.1,T14,F7.4)
00004000      WRITE(6,10)
00005000      A=1.0
00006000      DO 30 I=1,10
00007000      ROOT=SQRT(A)
00008000      WRITE(6,20)A,ROOT
00009000      30  A=A+1.0
00010000      STOP
00011000      END
```

```
****      GLOBAL STATISTICS      ****
****      NO ERRORS, NO WARNINGS  ****
TOTAL COMPILATION TIME  0:00:01
TOTAL ELAPSED TIME     0:00:49
```

```
END OF COMPILE
```

Figure 3-1. Compiling Program Units into a USL (Sheet 1 of 4)

:FORTRAN ST1,USL1

PAGE 0001 HP32102B.00.05 (C) HEWLETT-PACKARD CO. 1976

```
00001000      SUBROUTINE START
00002000      CHARACTER*5 CHEX(8,8),BL,IN,CP,PP,FRAME*41,E*1
00003000      BL="!  "
00004000      IN="! ** "
00005000      CP="! BL "
00006000      PP="! WH "
00007000      DO 100 I=1,3,2
00008000          DO 90 J=2,8,2
00009000      90  CHEX(I,J)=CP
00010000      100 CONTINUE
00011000          DO 110 J=1,7,2
00012000      110 CHEX(2,J)=CP
00013000          DO 130 I=6,8,2
00014000          DO 120 J=1,7,2
00015000      120 CHEX(I,J)=PP
00016000      130 CONTINUE
00017000          DO 140 J=2,8,2
00018000      140 CHEX(7,J)=PP
00019000          DO 160 I=1,7,2
00020000          DO 150 J=1,7,2
00021000      150 CHEX(I,J)=IN
00022000      160 CONTINUE
00023000          DO 180 I=2,8,2
00024000          DO 170 J=2,8,2
00025000      170 CHEX(I,J)=IN
00026000      180 CONTINUE
00027000          DO 190 J=1,7,2
00028000      190 CHEX(4,J)=BL
00029000          DO 200 J=2,8,2
00030000      200 CHEX(5,J)=BL
00031000      E="!"
00032000      300 FORMAT(T12,"1",4X,"2",4X,"3",4X,"4",4X,"5",4X,"6"
00033000      #,4X,"7",4X,"8")
00034000      400 FORMAT(T10,S)
00035000      500 FORMAT(T6,I0,T10,S0,T50,I5)
00036000      FRAME="!----!----!----!----!----!----!----!"
00037000      WRITE(6,300)
00038000      DO 600 I=1,8
00039000          WRITE(6,400)FRAME
00040000      600 WRITE(6,500)I,(CHEX(I,J),J=1,8),E
00041000          WRITE(6,400)FRAME
00042000      RETURN
00043000      END
```



```
****      GLOBAL STATISTICS      ****
****      NO ERRORS,      NO WARNINGS      ****
TOTAL COMPILATION TIME 0:00:02
TOTAL ELAPSED TIME      0:02:35
```

END OF COMPILE

Figure 3-1. Compiling Program Units into a USL (Sheet 2 of 4)

:FORTRAN ST2,USL1

PAGE 0001 HP32102B.00.05 (C) HEWLETT-PACKARD CO. 1976

```
00001000      SUBROUTINE START
00002000      CHARACTER*5 CHEX(8,8),BL,IN,CP,PP,FRAME*41,E*1
00003000      BL="!  "
00004000      IN="! ** "
00005000      CP="! WH "
00006000      PP="! BL "
00007000      DO 100 I=1,3,2
00008000      DO 90 J=2,8,2
00009000  90  CHEX(I,J)=CP
00010000 100  CONTINUE
00011000      DO 110 J=1,7,2
00012000 110  CHEX(2,J)=CP
00013000      DO 130 I=6,8,2
00014000      DO 120 J=1,7,2
00015000 120  CHEX(I,J)=PP
00016000 130  CONTINUE
00017000      DO 140 J=2,8,2
00018000 140  CHEX(7,J)=PP
00019000      DO 160 I=1,7,2
00020000      DO 150 J=1,7,2
00021000 150  CHEX(I,J)=IN
00022000 160  CONTINUE
00023000      DO 180 I=2,8,2
00024000      DO 170 J=2,8,2
00025000 170  CHEX(I,J)=IN
00026000 180  CONTINUE
00027000      DO 190 J=1,7,2
00028000 190  CHEX(4,J)=BL
00029000      DO 200 J=2,8,2
00030000 200  CHEX(5,J)=BL
00031000      E="!"
00032000 300  FORMAT(T12,"1",4X,"2",4X,"3",4X,"4",4X,"5",4X,"6"
00033000      #,4X,"7",4X,"8")
00034000 400  FORMAT(T10,S)
00035000 500  FORMAT(T6,I2,T10,8S,T50,I5)
00036000      FRAME="!----!----!----!----!----!----!----!"
00037000      WRITE(6,300)
00038000      DO 600 I=1,8
00039000      WRITE(6,400)FRAME
00040000 600  WRITE(6,500)I,(CHEX(I,J),J=1,8),E
00041000      WRITE(6,400)FRAME
00042000      RETURN
00043000      END
```

```
****      GLOBAL STATISTICS      ****
****      NO ERRORS, NO WARNINGS  ****
TOTAL COMPILATION TIME  0:00:02
TOTAL ELAPSED TIME     0:02:32
```

END OF COMPILE

Figure 3-1. Compiling Program Units into a USL (Sheet 3 of 4)

:FORTRAN ST3,USL1

PAGE 0001 HP32102B.00.05 (C) HEWLETT-PACKARD CO. 1976

```
00001000      SUBROUTINE START
00002000      CHARACTER*5 CHEX(8,8),BL,IN,CP,PP,FRAME*41,E*1
00003000      BL="!  "
00004000      IN="! ** "
00005000      CP="! BK "
00006000      PP="! WK "
00007000      DO 100 I=1,3,2
00008000      DO 90 J=2,8,2
00009000  90    CHEX(I,J)=CP
00010000  100   CONTINUE
00011000      DO 110 J=1,7,2
00012000  110   CHEX(2,J)=CP
00013000      DO 130 I=6,8,2
00014000      DO 120 J=1,7,2
00015000  120   CHEX(I,J)=PP
00016000  130   CONTINUE
00017000      DO 140 J=2,8,2
00018000  140   CHEX(7,J)=PP
00019000      DO 160 I=1,7,2
00020000      DO 150 J=1,7,2
00021000  150   CHEX(I,J)=IN
00022000  160   CONTINUE
00023000      DO 180 I=2,8,2
00024000      DO 170 J=2,8,2
00025000  170   CHEX(I,J)=IN
00026000  180   CONTINUE
00027000      DO 190 J=1,7,2
00028000  190   CHEX(4,J)=BL
00029000      DO 200 J=2,8,2
00030000  200   CHEX(5,J)=BL
00031000      E="!"
00032000  300   FORMAT(T12,"1",4X,"2",4X,"3",4X,"4",4X,"5",4X,"6"
00033000      #,4X,"7",4X,"8")
00034000  400   FORMAT(T10,S)
00035000  500   FORMAT(T6,I2,T10,S5,T50,I5)
00036000      FRAME="!-----!-----!-----!-----!-----!-----!"
00037000      WRITE(6,300)
00038000      DO 600 I=1,8
00039000      WRITE(6,400)FRAME
00040000  600   WRITE(6,500)I,(CHEX(I,J),J=1,8),E
00041000      WRITE(6,400)FRAME
00042000      RETURN
00043000      END
```

```
****      GLOBAL STATISTICS      ****
****      NO ERRORS, NO WARNINGS  ****
TOTAL COMPILATION TIME 0:00:02
TOTAL ELAPSED TIME    0:02:38
```

END OF COMPILE

Figure 3-1. Compiling Program Units into a USL (Sheet 4 of 4)

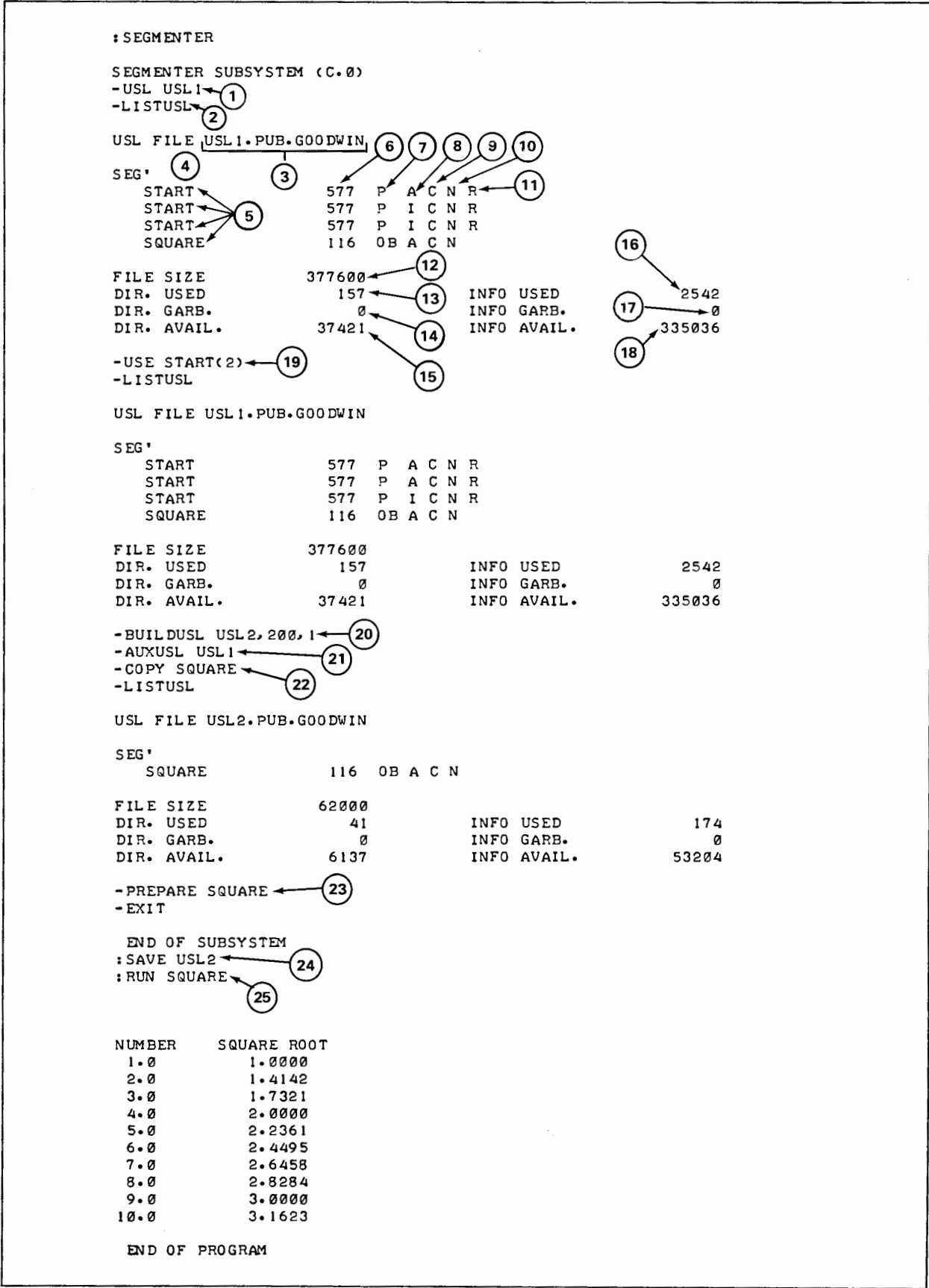


Figure 3-2. Using the Segmenter USL, LISTUSL, USE, BUILDUSL, AUXUSL, COPY, PREPARE and EXIT Commands

```

SEGMENTER SUBSYSTEM (C.0)
-USL USL1
-LISTUSL

USL FILE USL1.PUB.GOODWIN

SEG'
  START          577 P A C N R
  START          577 P A C N R
  START          577 P I C N R
  SQUARE        116 O B A C N

FILE SIZE      377600
DIR. USED      157          INFO USED      2542
DIR. GARB.     0          INFO GARB.     0
DIR. AVAIL.    37421       INFO AVAIL.    335036

-PURGERBM SQUARE ← (1)
-LISTUSL

USL FILE USL1.PUB.GOODWIN

SEG'
  START          577 P A C N R
  START          577 P A C N R
  START          577 P I C N R

FILE SIZE      377600
DIR. USED      157          INFO USED      2542
DIR. GARB.     32          INFO GARB.     174
DIR. AVAIL.    37421       INFO AVAIL.    335036

-CEASE START(2) ← (2)
-LISTUSL

USL FILE USL1.PUB.GOODWIN

SEG'
  START          577 P A C N R
  START          577 P I C N R
  START          577 P I C N R

FILE SIZE      377600
DIR. USED      157          INFO USED      2542
DIR. GARB.     32          INFO GARB.     174
DIR. AVAIL.    37421       INFO AVAIL.    335036

-NEWSEG SEG1.START(1) ← (3)
-LISTUSL

USL FILE USL1.PUB.GOODWIN

SEG1
  START          577 P A C N R
SEG'
  START          577 P I C N R
  START          577 P I C N R

FILE SIZE      377600
DIR. USED      166          INFO USED      2542
DIR. GARB.     32          INFO GARB.     174
DIR. AVAIL.    37412       INFO AVAIL.    335036

-PURGERBM SEGMENT, SEG' ← (4)
-LISTUSL

USL FILE USL1.PUB.GOODWIN

SEG1
  START          577 P A C N R

FILE SIZE      377600
DIR. USED      166          INFO USED      2542
DIR. GARB.     125         INFO GARB.     1700
DIR. AVAIL.    37412       INFO AVAIL.    335036

-EXIT

END OF SUBSYSTEM

```

Figure 3-3. Using the Segmenter PURGERBM, CEASE, and NEWSEG Commands



## LISTING RBM'S

You can list the names of the RBM's in the currently-managed USL, their segment names, entry points, and other related information, with the Segmenter LISTUSL command.

Item 2 in figure 3-2 shows the LISTUSL command being used. The output is written on the file designated in the *listfile* parameter of the MPE :SEGMENTER command or on \$STDLIST if the *listfile* parameter is omitted (as in figure 3-2). The listing shows all program unit entry points. Those entry points related to procedures are listed under the segment in which they are contained. Interrupt procedures and block data program units appear in separate lists.

In figure 3-2, significant entries are indicated with arrows, keyed to the discussion below. All numbers appearing on this listing are octal values.

Item No.	Meaning
3	The name of the USL file ( <i>filename.groupname.accountname</i> ).
4	The segment name. (SEG' is the default segment name.)
5	The RBM (entry point or program unit) names. Note that there are three RBM's with the name START and that only the top one is active (see item 8). This is the one compiled into the USL <i>last</i> and is index number 1, or index number 0. (Remember that index 0 is the first <i>active</i> RBM if more than one RBM of the same name exists.)
6	The number of words (octal) in the RBM code module.
7	The type of entry-point named, where: P = Procedure primary entry point. SP = Secondary entry point for procedure. OB = Outer block primary entry point. SO = Secondary entry point for outer block. BD = Block data entry point In = Interrupt procedure entry point (n is the interrupt procedure type number, ranging from 0 through 2). CP = COBOL secondary entry point.
8	The active/inactive indicator, where: A = Active I = Inactive
9	The callability indicator, where: C = The entry point is directly callable by the user. U = The entry point is not directly callable by the user.
10	The privileged/not-privileged indicator: P = Privileged. N = Not privileged.

Item No.	Meaning
11	<p>The internal flag status:</p> <p style="padding-left: 40px;">H = Internal flag <i>on</i> R = Internal flag <i>off</i>.</p> <p>(The internal flag and its purpose are explained later in this section.)</p>
12	The total size of the USL file, in words (octal).
13	The number of words (in octal) used in the directory portion of the USL file.
14	The number of words (in octal) in the directory part of the USL file, filled with meaningless information, as a result of various RBM manipulations.
15	The number of unused words (in octal) in the directory portion of the USL file.
16	The number of words (in octal) used in the RBM-information portion of the USL file.
17	The number of words (in octal) in the RBM-information portion of the USL file, filled with meaningless information, as a result of various RBM manipulations.
18	The number of unused words (in octal) in the RBM-information portion of the USL file.

## ACTIVATING ENTRY POINTS

The Segmenter USE command activates one or more program unit (RBM) entry points in the USL currently designated for management.

Item 19 in figure 3-2 shows the RBM named START being activated. Note that index (2) is specified, thus activating the second RBM named START in the USL.

## CREATING NEW USL'S

To create a new (job/session temporary) USL onto which RBM's containing user program units can be compiled or copied, the Segmenter BUILDUSL command is used.

### NOTE

See the *MPE Intrinsic Reference Manual* for a discussion of job/session temporary files.

Item 20 in figure 3-2 creates a new USL named USL2. The new USL will contain 200 records and will be allocated one extent.

Once a USL file has been created, it can be managed by MPE commands and intrinsics. For example, it can be purged (:PURGE command), renamed (:RENAME command), or read (FREAD intrinsic). Note that USL's created with the Segmenter BUILDUSL command are job/session temporary and this means that the USL is deleted at log-off time. If it is necessary to save a USL created with the BUILDUSL command as a permanent file, the MPE :SAVE command can be used (see item 24 in figure 3-2).

## TRANSFERRING RBM'S

You can transfer one or more RBM's from another USL to the currently-managed USL. First, you must designate the USL source from which the RBM's are transferred by entering the Segmenter AUXUSL command.

Item 21 in figure 3-2 shows the AUXUSL command being used to designate USL1 as the source.

Next, you transfer (copy) the RBM's to the destination USL by entering the Segmenter COPY command. All subsequent COPY commands will refer implicitly to the source USL file designated by the current AUXUSL command, until a new AUXUSL command is encountered.

Item 22 in figure 3-2 shows the RBM named SQUARE being copied to USL2 with the COPY command. The LISTUSL command verifies that USL2 now contains SQUARE.

## PREPARING PROGRAM FILES

You can prepare the active RBM's residing in a USL into a program file. This is done by entering the MPE :PREP command, described in the *MPE Commands Reference Manual*, or by entering the Segmenter PREPARE command. The currently-managed USL is used as the RBM source.

Item 23 in figure 3-2 shows the RBM named SQUARE being prepared into a program file. After the Segmenter operation is terminated, the program SQUARE is run with the MPE :RUN command (see item 25).

## **DELETING RBM'S**

To delete one or more Relocatable Binary Modules from a USL, the Segmenter PURGERBM command is used.

Item 1 in figure 3-3 uses the PURGERBM command to delete the RBM named SQUARE. The LISTUSL command verifies that SQUARE has been deleted.

Item 4 in figure 3-3 deletes all RBM's from the segment named SEG'. Again, the LISTUSL command verifies that the RBM's in SEG' and the segment itself have been deleted.

## **DEACTIVATING ENTRY POINTS**

The Segmenter CEASE command is used to deactivate one or more entry points in the USL currently designated for management.

Item 2 in figure 3-3 shows the RBM named START being activated. The second entry named START is deactivated because index (2) is specified.

## **ASSIGNING NEW SEGMENT NAMES TO RBM'S**

The Segmenter NEWSEG command is used to change the segment name associated with an RBM, thus assigning the RBM to a different code segment the next time it is prepared onto a program file.

Item 3 in figure 3-3 shows the RBM named START in the segment named SEG' being associated with the segment SEG1. The LISTUSL command verifies that START now is in segment SEG1.

## **CREATING AND MAINTAINING RELOCATABLE LIBRARIES (RL'S)**

To enter commands to create and maintain relocatable libraries (RL's), you first access the Segmenter by entering the MPE :SEGMENTER command.

As with USL's, Segmenter commands are required to create and change RL's because these files are written in a special format. Once an RL is created, however, it can be referenced by other MPE commands and intrinsics. For example, it is purged with the :PURGE command, renamed with the :RENAME command, or read with the FREAD intrinsic.

## CREATING A RELOCATABLE PROCEDURE LIBRARY (RL) FILE

To create a permanent, formatted RL file, the Segmenter BUILDRL command is used.

### NOTE

Do not use the MPE command

```
:BUILD X; CODE = RL
```

This file cannot be used by the Segmenter.

Item 1 in figure 3-4 shows the BUILDRL command being used to create the RL file RLPROC. The file can contain 100 records and is allocated one disc extent.

## ADDING A PROCEDURE TO AN RL

To add a procedure to the currently-managed RL, you copy the RBM containing that procedure from the currently-managed USL to the RL with the Segmenter ADDRL command. You must have write access to the designated RL file.

Item 2 in figure 3-4 adds the procedure START to the RL file RLPROC. The LISTRL command verifies that START has been added.

## LISTING PROCEDURES IN AN RL

The segmenter LISTRL command is used to list all procedure entry points and external references in the currently-managed RL.

Item 3 in figure 3-4 shows the LISTRL command and an example of the output produced. On this listing, significant entries are indicated with item numbers, keyed to the discussion below. All numbers appearing on the listing are octal values.

Item No.	Meaning
4	The name of the RL file ( <i>filename.groupname.accountname</i> ).
5	The procedure entry-point name.
6	The parameter checking level for the entry point.
7	The entry-point address (relative displacement within the code module).

: SEGMENTER

SEGMENTER SUBSYSTEM (C.0)

-USL USL1

-LISTUSL

USL FILE USL1.PUB.GOODWIN

SEG1

START 577 P A C N R

FILE SIZE 377600

DIR. USED 166

INFO USED 2542

DIR. GARB. 125

INFO GARB. 1700

DIR. AVAIL. 37412

INFO AVAIL. 335036

-BUILDRL RLPROC,100,1 ← ①

-ADDRL START ← ②

-LISTRL ← ③

RL FILE RLPROC.PUB.GOODWIN

\* ENTRY POINTS \* ← ④

START ← ⑤

3 ← ⑥

54 ← ⑦

400 ← ⑧

1 ← ⑨

577 ← ⑩

674 ← ⑪

\* EXTERNALS \*

TFORM' ]

FMTINIT' ] ← ⑫

IIO' ]

SIO' ]

0 ← ⑬

0

0

0

USED

1500 ← ⑭

AVAILABLE

27300 ← ⑮

-EXIT

END OF SUBSYSTEM

Figure 3-4. Using the Segmenter Commands BUILDRL, ADDRL, and LISTRL

- 8                   The RL file address of the procedure information block.
- 9                   The number of entry points in the procedure. (If this field is blank, the entry-point name field (item 5) names a secondary entry point; if this field contains a number, the entry-point name field contains a primary entry-point name.)
- 10                  The length of the code module, in words. (If this field is blank, the entry-point name field names a secondary entry-point; if this field contains a number, the entry-point name field contains a primary entry-point name.)
- 11                  The length of the procedure information block, in words. (If this field is blank, the entry-point name field names a secondary entry point; if this field contains a number, the entry-point name field contains a primary entry-point name.)
- 12                  The names of the external references.
- 13                  The parameter checking level for the external reference.
- 14                  The number of words presently used in the RL file.
- 15                  The number of words presently available in the RL file.

Figure 3-5 shows a short program which references the external procedure START.

First, the program is compiled, then prepared. Note that the RL file named RLPROC is referenced in the :PREP command. When the program runs and calls the subroutine START, the RL file named RLPROC is searched for this subroutine.

## **DESIGNATING RL'S FOR MANAGEMENT**

To enter commands that modify an existing RL in any way, you first must identify the RL with the Segmenter RL command which then FOPEN's it.

All subsequent RL-modification commands will refer implicitly to this RL until a new RL command is entered, superseding the previous RL command, or until a BUILDRL command creates a new RL, thus effectively superseding the RL command.

Item 1 in figure 3-6 shows the RL named RLPROC being referenced by the RL command.

## **DELETING AN ENTRY POINT OR PROCEDURE FROM AN RL**

The Segmenter PURGERL command is used to delete an entry point of a procedure, or the entire procedure, from an RL

:FORTRAN XMPL1

PAGE 0001 HP32102B.00.05 (C) HEWLETT-PACKARD CO. 1976

```
00001000      PROGRAM MAIN
00002000      CALL START
00003000      STOP
00004000      END
```

```
****          GLOBAL STATISTICS          ****
**** NO ERRORS, NO WARNINGS          ****
TOTAL COMPILATION TIME  0:00:01
TOTAL ELAPSED TIME      0:00:25
```

```
END OF COMPILE
:PREP $OLDPASS,RLPROG;RL=RLPROC
```

```
END OF PREPARE
:SAVE RLPROG
:RUN RLPROG
```

	1	2	3	4	5	6	7	8
1	! ** !	BK !	** !	BK !	** !	BK !	** !	BK !
2	! BK !	** !	BK !	** !	BK !	** !	BK !	** !
3	! ** !	BK !	** !	BK !	** !	BK !	** !	BK !
4	!   !	** !	!	** !	!	** !	!	** !
5	! ** !	!	** !	!	** !	!	** !	!
6	! WK !	** !	WK !	** !	WK !	** !	WK !	** !
7	! ** !	WK !	** !	WK !	** !	WK !	** !	WK !
8	! WK !	** !	WK !	** !	WK !	** !	WK !	** !

END OF PROGRAM

Figure 3-5. Referencing a Relocatable Library Procedure



:SEGMENTER

SEGMENTER SUBSYSTEM (C.0)

-PL PLPROC  
-LISTRL (1)

PL FILE RLPROC.PUB.TECHPUBS

\* ENTRY POINTS \*

START	3	50	400	1	554	651
-------	---	----	-----	---	-----	-----

\* EXTERNALS \*

TFORM'	0
FMTINIT'	0
IIO'	0
SIO'	0

USED	1500	AVAILABLE	27300
------	------	-----------	-------

-PURGERL UNIT, START  
-LISTPL (2)

PL FILE RLPROC.PUB.TECHPUBS

\* ENTRY POINTS \*

\* EXTERNALS \*

USED	400	AVAILABLE	30400
------	-----	-----------	-------

-EXIT

END OF SUBSYSTEM

Figure 3-6. Using the Segmenter Commands RL and PURGERL

Item 2 in figure 3-6 deletes the procedure named START from an RL. The LISTRL command verifies that START has been deleted. When the last entry point in a procedure is deleted, the entire body of code representing that procedure is deleted.

Additionally, the entire RL can be deleted by the MPE :PURGE command.

## **CREATING AND MAINTAINING SEGMENTED LIBRARIES (SL'S)**

To enter commands to create and maintain segmented libraries (SL's), you first access the Segmenter with the MPE :SEGMENTER command. As with USL's and RL's, Segmenter commands are required to create SL's because these files are written in a special format. Once an SL is created, however, it can be referenced by other MPE commands and intrinsics.

### **CREATING A SEGMENTED PROCEDURE LIBRARY (SL) FILE**

To create a permanent, formatted SL file, the Segmenter BUILDSL command is used. You must have save access to the group to which you assign the SL. The SL will be created with default file-access security. See the *MPE Intrinsics Reference Manual* for a discussion of file-access security.

Item 1 in figure 3-7 shows the BUILDSL command being used to create the SL file DPSL. The file can contain 100 records and is allocated one disc extent.

### **ADDING A PROCEDURE SEGMENT TO AN SL**

The Segmenter ADDSL command is used to add a procedure to an SL. In order to add a procedure to the currently-managed SL, you actually select the RBM containing this procedure, and all other RBM's sharing the same segment name, from the currently-managed USL, format these into a segment, and place this segment in the SL. You must have write access to the designated SL file.

Item 2 in figure 3-7 adds the segment SEG' to the SL currently being managed (DPSL).

### **LISTING PROCEDURES IN AN SL**

The Segmenter LISTSL command is used to list the procedures in the currently-managed SL.

#### **WARNING**

**An SL is locked while being accessed. If you are accessing SL.PUB.SYS, other users may be prevented from normal operation during that time.**

```

:BUILD USL1;CODE=USL;DISC=,1
:FORTRAN SLXMPL1,USL1

PAGE 0001   HP32102B.00.0

00001000      FUNCTION DISP(C,R,H,W,D,T)
00002000      DISP=C*(3.14159*(R**2)*H)
00003000      T=W+D*.0019
00004000      RETURN
00005000      END

****          GLOBAL STATISTICS          ****
**** NO ERRORS, NO WARNINGS ****
TOTAL COMPILATION TIME  0:00:01
TOTAL ELAPSED TIME      0:00:26

END OF COMPILE
:SEGMENTER

SEGMENTER SUBSYSTEM (C.0)
-USL USL1
-LISTUSL

USL FILE USL1.PUB.TECHPUBS

SEG'
DISP                24 P A C N R

FILE SIZE           377600
DIR. USED           42          INFO USED           24
DIR. GARB.          0          INFO GARB.          0
DIR. AVAIL.         37536      INFO AVAIL.         337554
-BUILDSL DPSL,100,1 ← (1)
-ADDSL SEG' ← (2)
-LISTSL ← (3)

SL FILE DPSL.PUB.TECHPUBS

SEGMENT (5) ← 0 SEG' (4) ← 6          LENGTH 30 ← (7)
ENTRY POINTS (8) ← 3 CHECK CAL STT ADR (12) ← 0
DISP (8) ← 3 (9) ← C (10) ← 1 (11) ← 0
EXTERNALS (13) ← 1 CHECK STT SEG (14) ← 1 (15) ← 1 (16) ← 0
1 ← (17)

USED 1600 ← (18) AVAILABLE (19) ← 27200
-EXIT

END OF SUBSYSTEM

```

Figure 3-7. Adding a Function to an SL and Using the Segmenter BUILDSL, ADDSL, and LISTSL Commands.

Item 3 in figure 3-7 illustrates the LISTSL command. On the listing, significant entries are indicated with item numbers, keyed to the discussion below. (All numbers appearing on this listing are octal values.)

Item No.	Meaning
4	The name of the SL file ( <i>filename.groupname.accountname</i> ).
5	The logical segment number (relative to this SL).
6	The segment name.
7	The segment length, in words.
8	The entry-point name in the segment.
9	The parameter checking-level of the entry-point.
10	The callability of the entry point, where C = Callable U = Uncallable
11	The Segment Transfer Table (STT) number of the entry point.
12	The entry-point address (relative displacement within the segment).
13	External references (blank in this case).
14	The parameter checking level of external references (blank).
15	The STT number of the external references (blank).
16	If a number appears in this field, it is a (logical) segment number and indicates that this reference has been bound to an entry point within the SL.
17	A bit list of the segments referenced within the SL (with the left-most bit corresponding to the first (0) logical segment number): For each bit, 1 = Segment referenced. 0 = Segment not referenced.
18	The number of words presently used in the SL file.
19	The number of words not used, at present, in the SL file.



Following the line containing the segment number, name, and length, any of the following segment attributes may appear. (These attributes can only be specified during system configuration. All attributes except PRIVILEGED pertain only to the system SL.)

<b>Attribute</b>	<b>Meaning</b>
RESIDENT	Segment permanently resides in main memory.
ALLOCATED	Segment is permanently allocated in virtual memory.
PRIVILEGED	Segment is executed in privileged mode.
SYSTEM	Segment is part of MPE.

Figure 3-8 contains a program that references the function DISP in the SL file DPSL. Note that DPSL must be renamed to SL with the MPE :RENAME command before the Group Library can be searched for the procedure.

## **DESIGNATING SL'S FOR MANAGEMENT**

To enter commands that modify an existing SL, you first must identify the SL with the Segmenter SL command.

All subsequent SL-modification commands will refer implicitly to this SL until a new SL command is entered, superseding the previous SL command, or until a BUILDSDL command creates a new SL, thus effectively superseding the old SL command.

Item 1 in figure 3-9 illustrates the SL command.

## **DELETING AN ENTRY POINT FROM AN SL**

The Segmenter PURGESL command is used to delete an entry point from a segment in an SL, or to delete an entire segment from an SL. The segment code actually remains in the SL, but the entry-point name is withdrawn from the entry-point directory. When the last entry point in a segment is deleted, the entire segment is automatically removed from the SL.

Item 2 in figure 3-9 deletes the entry DISP from the segment SEG'. Note that the entire segment is deleted because DISP was the last entry point in the segment.

## **SETTING RBM INTERNAL FLAGS**

In addition to the activity bit associated with every entry-point on an RBM, there is also an *internal flag* that determines whether or not that entry-point is to be placed in the library directory—and thus made available to users and segments within that SL — when the RBM is segmented and added to an

:FORTRAN SLXMP2

PAGE 0001 HP32102B.00.0

```
00001000      PROGRAM SL
00002000  100  FORMAT(T5,"DISPLACEMENT OF THIS VEHICLE IS ",F14.5)
00003000  200  FORMAT(T5,"TOTAL COST IS ",M12.2)
00004000  300  FORMAT(T5,"REGISTRATION COST IS ",M12.2//)
00005000      DISPLAY "WEIGHT?"
00006000      ACCEPT A
00007000      DISPLAY "DISTANCE?"
00008000      ACCEPT B
00009000      DISPLAY "COST?"
00010000      ACCEPT COST
00011000      DISPLAY "NO. CYLINDERS?"
00012000      ACCEPT C
00013000      DISPLAY "BORE?"
00014000      ACCEPT D
00015000      DISPLAY "STROKE?"
00016000      ACCEPT E
00017000      D=D/2
00018000      DISPL=DISP(C,D,E,A,B,T)
00019000      WHOLECOST=COST+T
00020000      TAX=1.5*DISPL
00021000      REGISTRATION=SQRT(DISPL)*6.5
00022000      WRITE(6,100)DISPL
00023000      WRITE(6,200)WHOLECOST
00024000      WRITE(6,300)REGISTRATION
00025000      STOP
00026000      END
```

```
****      GLOBAL STATISTICS      ****
****      NO ERRORS, NO WARNINGS  ****
TOTAL COMPILATION TIME  0:00:01
TOTAL ELAPSED TIME     0:01:38
```

```
END OF COMPILE
:PREP $OLDPASS,SLPROG
```

```
END OF PREPARE
:RENAME DPSL,SL
:RUN SLPROG;LIB=G
```

WEIGHT?

?

3423

DISTANCE? ?

3576

COST? ?

17987

NO. CYLINDERS? ?

12

BORE? ?

3.124

STROKE? ?

2.873

```
DISPLACEMENT OF THIS VEHICLE IS      264.25824
TOTAL COST IS      $21,416.79
REGISTRATION COST IS      $105.66
```

END OF PROGRAM

:SAVE SLPROG

Figure 3-8. Referencing an SL Procedure

:SEGMENTER

SEGMENTER SUBSYSTEM (C.0)

-SL DPSL

-LISTSL

SL FILE DPSL.PUB.TECHPUBS

SEGMENT    0    SEG'                    LENGTH    30

ENTRY POINTS    CHECK CAL STT    ADR  
DISP                    3    C    1    0

EXTERNALS            CHECK STT SEG

1

USED                            1600                    AVAILABLE                    27200

-PURGESL ENTRY,DISP

-LISTSL

SL FILE DPSL.PUB.TECHPUBS

USED                            1000                    AVAILABLE                    30000

-EXIT

END OF SUBSYSTEM

Figure 3-9. Using the Segmenter Commands SL and PURGESL

SL. (The internal flag is only *interrogated* when the segment is actually added to an SL.) As an example of how this facility can be used, suppose that a programmer is writing a complicated utility routine that he wants to add to an SL. He writes this routine as a set of procedures to be placed in the same code segment, but wants only the entry-points of certain procedures made available to other users; he does not want them to use the entry-points of the support procedures for the main routine. He can effectively hide these private entry-points from users by using the OPTION INTERNAL statement (SPL/3000 only) in the declarations of the applicable procedures, or by setting the internal flag *on* with the HIDE command described below. When these procedures are prepared onto a segment and added to an SL, the entry-points designated as internal are not entered in the directory of the library.

To set an internal flag *on*, enter:

```
HIDE name[(index)]
```

where

*name* is the name of the entry-point in the currently-managed USL whose internal flag is to be set *on*. (Required parameter.)

*index* is the index integer further specifying the entry-point *name*; the index may be used when the RBM contains more than one entry-point of this name. If *index* is omitted, 0 is assigned by default. (Optional parameter.)

To set an internal flag to *off*, enter the following command:

```
REVEAL name[(index)]
```

where

*name* is the name of the entry-point in the currently-managed USL whose internal flag is to be set *off*. (Required parameter.)

*index* is the index integer further specifying the entry-point *name*; the index may be used when the RBM contains more than one entry-point of this name. If *index* is omitted, a value of 0 is assigned. (Optional parameter.)

## USING MPE INTRINSICS TO MANIPULATE USL FILES

By using MPE intrinsics, you can perform the following functions programmatically:

- Initialize a buffer for a USL file to the empty state with the INITUSLF intrinsic.
- Move the information block in a USL file with the ADJUSTUSL intrinsic.
- Change the length of a USL file with the EXPANDUSLF intrinsic.

The above intrinsics are most useful to programmers writing compilers.



## **INITIALIZING BUFFERS FOR USL FILES**

To initialize the first record (record 0) of a USL file to the empty state, the INITUSLF intrinsic is used.

For example, the INITUSLF intrinsic call could be written as follows:

```
ERRNUM:=INITUSLF(USLF,BUF);
```

## **CHANGING THE DIRECTORY BLOCK/INFORMATION BLOCK SIZE ON A USL FILE**

The information block can be moved forward or backward on a USL file, thereby increasing or decreasing, respectively, the space available for the file directory block, with the ADJUSTUSLF intrinsic.

For example, the ADJUSTUSLF intrinsic call could be written as follows:

```
ERRNUM:=ADJUSTUSLF(USLF,-80);
```

In the above example, the information block is moved backward in the USL file, decreasing the directory block space by 80 records and increasing the information block space by 80 records.

## **CHANGING THE SIZE OF A USL FILE**

The length of a USL file can be increased or decreased with the EXPANDUSLF intrinsic.

For example, the EXPANDUSLF intrinsic call could be written as follows:

```
NEWFNUM:=EXPANDUSLF(USLF,80);
```

When the above intrinsic is executed, a new USL file is created whose length is 80 records longer than the USL file specified by the parameter USLF (the file number of the old USL file to be changed). The old USL file is copied to the new file with the same file name and the old file is deleted. The file number of the new USL file is returned to NEWFNUM.

# MPE SEGMENTER ERROR MESSAGES

SECTION

IV

Segmenter error messages are in the following format:

$$\left\{ \begin{array}{l} \text{*** ERROR ***} \\ \text{*** FILE ERROR} \\ \text{*** WARNING ***} \end{array} \right\} [numericparm] [stringparm]$$

ERROR # *msgnum* *message*

The following is an example of a message containing all optional parameters:

*numericparm*  
FCHECK value      *stringparm*  
file name

\*\*\*FILE ERROR 40 AUX. USL FILE  
ERROR # 84 UNEXPECTED I/O ERROR

*msgnum*      *message*

The diagram illustrates the structure of an error message. It shows two lines of text: "\*\*\*FILE ERROR 40 AUX. USL FILE" and "ERROR # 84 UNEXPECTED I/O ERROR". Annotations include: "numericparm" pointing to "40" (with "FCHECK value" below it); "stringparm" pointing to "AUX. USL FILE" (with "file name" below it); "msgnum" pointing to "# 84"; and "message" pointing to "UNEXPECTED I/O ERROR". Brackets are used to group "40" and "AUX. USL FILE" together, and "ERROR # 84" and "UNEXPECTED I/O ERROR" together.

Table 4-1 contains a list of the error numbers and messages for all Segmenter process errors.

Table 4-1. Segmenter Error Messages

NUMBER	ERROR MESSAGE	COMMENTS
0	ILLEGAL ENTRY	Bad USL generated by compiler, <i>numericparm</i> is the decimal address of the entry in the USL.
1	ILLEGAL HEADER	Bad USL generated by compiler. <i>numericparm</i> is the decimal address of the header in the USL.
2	ATTEMPT TO EXCEED MAXIMUM DIRECTORY SPACE	USL file maximum directory size is %77777.
3	AVAILABLE DIRECTORY SPACE EXHAUSTED	USL file is too small or compiler error.
4	AVAILABLE INFO SPACE EXHAUSTED	USL file too small or compiler error.
5	USL FILE NOT DESIGNATED	Tried to -USE, -PREPARE, -NEWSEG, -LISTUSL, -COPY, -ADDRL, -ADDSL, -HIDE, -REVEAL or -PURGERBM without specifying -USL.
6	ILLEGAL USL FILE SPECIFICATION	USL file length is less than 5 records or greater than 32767 records.
7	UNABLE TO OPEN USL FILE	FOPEN failure in -AUXUSL, -BUILDUSL, or -USL. <i>numericparm</i> is FCHECK value.
8	INVALID USL FILE	File code is not USL or USL file is bad.
9	UNABLE TO CLOSE USL	FCLOSE failure during -EXIT or other USL specification. <i>numericparm</i> is FCHECK value.
10	UNABLE TO CLOSE SL FILE	FCLOSE failure during -EXIT or on previous SL during -SL or BUILDSDL. <i>numericparm</i> is FCHECK value.
11	AVAILABLE FILE SPACE EXHAUSTED	RL or SL file is full.
12	ENTRY POINT ALREADY DEFINED	<i>stringparm</i> is entry name.
13	SEGMENT CONTAINS PROGRAM UNITS OTHER THAN PROCEDURES	Outer block is present. For SPL, recompile subprogram, or -CEASE outer block.
14	SEGMENT REQUIRES GLOBAL STRING	FORTTRAN DATA, or SPL GLOBAL or OWN variables were specified. <i>stringparm</i> is entry name.
15	SEGMENT ALREADY DEFINED	-ADDSL <i>segname</i> and <i>segname</i> already exists. <i>stringparm</i> is <i>segname</i> .
16	SL FILE NOT DESIGNATED	Tried to -PURGESL, -ADDSL, or -LISTSL with no SL file open.
17	ILLEGAL SL FILE SPECIFICATION	SL file length is less than 4 records or greater than %77777 records.

Table 4-1. Segmenter Error Messages (Continued)

NUMBER	ERROR MESSAGE	COMMENTS
18	UNABLE TO OPEN SL FILE	FOPEN failure during -BUILDSL or -SL. <i>numericparm</i> is FCHECK value.
19	INVALID SL FILE	File code is not SL or SL file is bad.
20	ILLEGAL RL FILE SPECIFICATION	RL file length is less than 4 records or greater than %77777 records.
21	RL FILE NOT DESIGNATED	Tried to -ADDRL, -LISTRL, or -PURGERL without opening RL.
22	INVALID RL FILE	File code is not RL or RL file is bad.
23	UNABLE TO CLOSE RL FILE	FCLOSE failure on RL file. <i>numericparm</i> is FCHECK value.
28	PROCEDURE HAS NO USABLE ENTRY POINT	
30	UNABLE TO OPEN RL FILE	FOPEN failure. <i>numericparm</i> if FCHECK value.
32	INVALID PROGRAM FILE	File code is not PROG.
33	ILLEGAL CAPABILITY SPECIFICATION	Program capability specification is greater than the user's.
34	MORE THAN ONE EXTENT USED	Not all code is in one extent. Program will not run unless code is contiguous on disc. Build a new program file.
35	NO PROGRAM TO PREPARE	No program in USL.
36	UNABLE TO CLOSE PROGRAM FILE	FCLOSE failure. <i>numericparm</i> is FCHECK value.
37	UNABLE TO OPEN PROGRAM FILE	FOPEN failure. <i>numericparm</i> is FCHECK value.
38	DATA SEGMENT OVERFLOW	Data is greater than %37777 words.
39	TOO MANY CODE SEGMENTS	More than 255 segments.
40	CODE SEGMENT OVERFLOW	Code segment is greater than %37774.
41	STT OVERFLOW	Too many PCAL instructions. <i>stringparm</i> is entry name where overflow occurred.
42	SEGMENT HAS NO USABLE ENTRY POINT	
43	UNABLE TO ACCESS PROCEDURE	Illegal P-Label or params not matching.
44	REQUIRES PRIVILEGED MODE CAPABILITY	User needs privileged mode capability to add privileged segment.
45	ACTUAL PARAMETERS INCOMPATIBLE WITH FORMAL PARAMETERS	(FORTRAN program) <i>stringparm</i> is procedure name.

Table 4-1. Segmenter Error Messages

NUMBER	ERROR MESSAGE	COMMENTS
46	PROGRAM UNIT CONTAINS FATAL ERROR	<i>stringparm</i> is unit name.
47	PROGRAM UNIT CONTAINS NON-FATAL ERROR	<i>stringparm</i> is unit name.
48	CODE SEGMENT MAY BE TOO LARGE	The length of the code segment exceeds the configured system maximum size.
60	NO OUTER BLOCK IS ACTIVE	The prepared program would have no outer block.
61	MORE THAN ONE OUTER BLOCK IS ACTIVE	The prepared program would have more than one outer block.
62	MORE THAN ONE OUTER BLOCK HAS ACTIVE ENTRY POINTS	The prepared program would have more than one outer block.
63	EXTERNAL VARIABLE NOT DECLARED GLOBAL	<i>stringparm</i> is variable name.
64	EXTERNAL VARIABLE INCOMPATIBLE WITH GLOBAL VARIABLE	<i>stringparm</i> is variable name.
66	TOO MANY COMMON DATA LABELS	(BASICOMP or FORTRAN program) program must be re-coded.
67	COMMON DECLARED WITH DIFFERENT SIZE	<i>stringparm</i> is common name.
68	ATTEMPT TO USE BLOCK DATA ON NON-EXISTENT COMMON	(FORTRAN program)
69	ATTEMPT TO USE BLOCK DATA ON INCOMPATIBLE COMMON	(FORTRAN program)
70	ILLEGAL STACK SIZE	Stack size less than 0.
71	ILLEGAL DL SIZE	DL size less than 0.
72	ILLEGAL MAXDATA SIZE	MAXDATA less than 0.
80	INSUFFICIENT STORAGE	Could not obtain enough DL area (system problem). Program is too large for the segmenter. SEGPROC must be prepared with larger DL.
81	ILLEGAL PATCH	Bad header generated by a compiler.
82	UNABLE TO OPEN SCRATCH FILE	Scratch area used to prepare code from USL before moving to SL file. <i>numericparm</i> is FCHECK value.
83	UNABLE TO OPEN LIST FILE	FOPEN failure. <i>numericparm</i> is FCHECK value.

Table 4-1. Segmenter Error Messages (Continued)

NUMBER	ERROR MESSAGE	COMMENTS
84	UNEXPECTED I/O ERROR	<i>numericparm</i> is FCHECK value and <i>stringparm</i> is file type. Files opened are: USL, AUX USL, SL, RL, RL LIBRARY, PROGRAM, LIST and SCRATCH files.
86	ITEM DIFFERENT FROM CLASS SPECIFICATION	One of the following occurred: 1. Tried to -COPY an item of a different class than specified. 2. Tried to -PURGERBM an item that differs in class specified. 3. Tried to -USE item with a class that differs from that specified.
87	ITEM NOT PRIMARY ENTRY POINT	Tried to -ADDRL an item that is not a primary entry point.
88	INCOMPATIBLE ITEM TYPE	One of the following occurred: 1. Tried to -HIDE an item that is not an entry point. 2. Tried to -NEWSEG an item that is not a procedure. 3. Tried to -PURGERBM an item that is not a UNIT.
89	INVALID CLASS SPECIFICATION	One of the following occurred: 1. Tried to -COPY an ENTRY. 2. Tried to -PURGERL a SEGMENT. 3. Tried to -PURGESL a UNIT.
93	UNABLE TO LOCATE ITEM	One of the following occurred: 1. Item not in AUXUSL for -COPY. 2. Item not in USL for -ADDSL. 3. Item not in USL for -ADDRL. 4. Item not in USL for -HIDE & -REVEAL. 5. Item not in USL for -NEWSEG. 6. Item not in RL for -PURGERL. 7. Item not in USL for -USE.
110	SEGMENT CURRENTLY LOADED	Segment referenced is in use and has been loaded by the loader.
111	SEGMENT CONTAINS EXTERNAL VARIABLE	<i>stringparm</i> is entry name.
112	SEGMENT CONTAINS COMMON	<i>stringparm</i> is entry name.
113	SEGMENT CONTAINS LOGICAL UNITS	FORTRAN program contains reference to logical units.
120	AUX USL FILE NOT DESIGNATED	Tried to -COPY with no AUXUSL open.



## A

ADDRL segmenter command, 1-5, 2-4  
 ADDSL segmenter command, 1-5, 2-5  
 ADJUSTUSL intrinsic, 1-5, 2-6, 2-7  
 AUXUSL segmenter command, 1-5, 2-8

## B

BUILDRL segmenter command, 1-5, 2-9  
 BUILDSL segmenter command, 1-5, 2-10  
 BUILDUSL segmenter command, 1-5, 2-11

## C

CEASE segmenter command, 1-5, 2-12  
 Colon, 2-1  
 Command  
   ADDRL, 1-5, 2-4  
   ADDSL, 1-5, 2-5  
   AUXUSL, 1-5, 2-8  
   BUILDRL, 1-5, 2-9  
   BUILDSL, 1-5, 2-10  
   BUILDUSL, 1-5, 2-11  
   CEASE, 1-5, 2-12  
   COPY, 1-5, 2-13  
   EXIT, 1-5, 2-14  
   HIDE, 1-5, 2-17  
   LISTRL, 1-5, 2-20  
   LISTSL, 1-5, 2-21  
   LISTUSL, 1-5, 2-22  
   NEWSEG, 1-5, 2-23  
   PREPARE, 1-5, 2-24, 2-25  
   PURGERBM, 1-5, 2-26  
   PURGERL, 1-5, 2-27  
   PURGESL, 1-5, 2-28  
   REVEAL, 1-5, 2-29  
   RL, 1-5, 2-30  
   :SEGMENTER, 1-5, 2-31  
   SL, 1-5, 2-32  
   USE, 1-5, 2-33  
   USL, 1-5, 2-34  
 Commands  
   MPE, 1-5, 2-1, 2-31  
   Segmenter, 1-5, 2-2, 2-4 - 2-34  
 COPY segmenter command, 1-5, 2-13

## D

Directory Block Size, changing on a  
 USL file, 3-23

## E

Entry points  
   activating, 3-8  
   deactivating, 3-10  
   deleting from an RL, 3-13, 3-15, 3-16  
   deleting from an SL, 3-19  
 Error Messages, Segmenter, 4-1 - 4-5  
 EXIT segmenter command, 1-5, 2-14  
 EXPANDUSLF intrinsic, 1-5, 2-15, 2-16

## G

Global Variables, 1-4



## H

HIDE segmenter command, 1-5, 2-17

## I

Information Block Size, changing on  
 a USL file, 3-23  
 INITUSLF intrinsic, 1-5, 2-18, 2-19  
 Internal Flags, setting RBM, 3-19, 3-22  
 Intrinsic  
   ADJUSTUSLF, 1-5, 2-6, 2-7  
   EXPANDUSLF, 1-5, 2-15, 2-16  
   INITUSLF, 1-5, 2-18, 2-19  
 Intrinsic, 1-5, 2-2, 2-6, 2-7, 2-15, 2-16  
 2-18, 2-19

## L

LIB parameter, 1-3  
 Library  
   Group, 1-2, 1-3  
   Public, 1-2, 1-3  
   Relocatable, 1-1, 3-10 - 3-16  
   Segmented, 1-2, 3-16 - 3-19  
   System, 1-2, 1-3  
 LISTRL segmenter command, 1-5, 2-20  
 LISTSL segmenter command, 1-5, 2-21  
 LISTUSL segmenter command, 1-5, 2-22

## M

Messages, Segmenter Error, 4-1 - 4-5  
 MPE commands, 1-5, 2-1, 2-31



## N

Name  
  Intrinsic, 2-2  
  MPE command, 2-1  
  Segmenter command, 2-2  
New segment names, assigning  
  to RBM's, 3-10  
NEWSEG segmenter command, 2-23

## P

Parameters  
  Intrinsic, 2-3  
  MPE command, 2-2  
  Segmenter command, 2-2  
PREPARE segmenter command, 1-5, 2-24, 2-25  
Program Files, preparing, 3-9  
PURGERBM segmenter command, 1-5, 2-26  
PURGERL segmenter command, 1-5, 2-27  
PURGESL segmenter command, 1-5, 2-28

## R

RBM Entry Points, 2-3  
RBM Internal Flags, setting, 3-19, 3-22  
RBM's  
  deleting, 3-10  
  entry points, 2-3  
  listing of, 3-7  
  transferring, 3-9  
Relocatable Libraries (RL's)  
  adding a procedure to, 3-11, 3-12  
  creating, 3-10, 3-12  
  definition of, 1-1  
  deleting an entry point or  
    procedure from, 3-13, 3-15, 3-16  
  designating for management of, 3-13

## R (Continued)

  listing of, 3-11, 3-12  
REVEAL segmenter command, 1-5, 2-30

## S

Segmented Libraries  
  adding a procedure to, 3-16, 3-18  
  creating, 3-16, 3-17  
  definition of, 1-2  
  deleting an entry point from, 3-19  
  designating for management of, 3-19  
  listing of, 3-16, 3-18  
Segment names, New  
  assigning to RBM's, 3-10  
Segmenter  
  accessing and exiting, 3-1  
  Commands, 1-5, 2-2, 2-4 - 2-34  
  Error Messages, 4-1 - 4-5  
  :SEGMENTER command, 1-5, 2-31  
  SL segmenter command, 1-5, 2-32

## U

USE segmenter command, 1-5, 2-33  
USL File  
  changing size of, 3-23  
  changing the directory  
    block size on, 3-23  
  changing the information  
    block size on, 3-23  
  creating, 3-8  
  designating, 3-1  
  initializing buffers for, 3-23  
  using intrinsics to manipulate, 3-22

Part No. 30000-90011  
Printed in U.S.A. 2/77

HEWLETT  PACKARD

Sales and service from 172 offices in 65 countries.  
5303 Stevens Creek Blvd., Santa Clara, California 95050