LUND
PERFORMANCE SOLUTIONS

WHAT YOU NEED TO KNOW. **WHEN** YOU NEED TO KNOW IT.

Developer's Toolbox

User Guide

Legal Notices

Lund Performance Solutions makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Lund Performance Solutions shall not be held liable for errors contained herein or direct, indirect, special, incidental, or consequential damages in connection with the furnishing, performance, or use of this material.

Restricted Rights Legend

All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Lund Performance Solutions. The information contained in this document is subject to change without notice.

LUND PERFORMANCE SOLUTIONS

240 2nd Avenue SW
Albany, OR 97321
USA

Use of this manual and flexible disk(s), tape cartridge(s), or CD-ROM(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations is expressly prohibited.

Copyright Notices

Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under the copyright laws.

Printed in the United States of America.

Trademark Notices

De-Frag/X Disk Manager, **Developer's Toolbox** and System Manager's Toolbox, Forecast Capacity Planner, Intact Dynamic Rollback (Intact D/R), Performance Gallery, Performance Gallery Gold, SOS/3000 Performance Advisor, SOS/9000 Performance Advisor, SOS/Linux Performance Advisor, SOS/Solaris Performance Advisor, Q-Xcelerator Resource Manager, and Shadow Data Replicator (Shadow D/R), are trademarks owned by Lund Performance Solutions in the USA and other countries.

Other brands and their products are trademarks or registered trademarks of their respective holders and should be noted as such.

Developer's Toolbox version A.09

Whitney Olsen, Laura Bryngelson, Rodica Popa 03152002

TABLE OF CONTENTS

LIST OF FIGURES

1

DEVELOPER'S TOOLBOX

Welcome to Developer's Toolbox

Welcome to the Developer's Toolbox™ software package by Lund Performance Solutions. Developer's Toolbox is the industry-standard performance monitoring and management application, devised to streamline, increase performance, and help day-to-day operations and repetitive tasks on the HP3000 easier and more efficient.

This software consists of eight unique utilities that are designed to help with programming tasks, including optimized replacements for frequently called intrinsics and program modification assistance. All of the utilities that comprise the toolbox were designed by HP e3000 professionals with years of experience. Further, this toolbox was developed with the idea of improving existing MPE utilities and providing solutions that simply have not existed.

Product Support

Lund Performance Solutions Main Offices

When you purchase support from Lund Performance Solutions, you benefit from the knowledge and experience of our technical support team. We are glad to help you interpret data and resolve performance issues. Our contracted product support entitles you to receive timely updates, bug fixes, documentation and direct technical support.

Postal Address

Lund Performance Solutions
240 2nd Avenue SW
Albany OR 97321 USA

Internet URL

Visit the Lund Performance Solutions website at **http://www.lund.com/**.

Telephone Number

For customer and technical support, call **(541) 812-7600**, Monday through Friday during the hours of 8:00 A.M., to 5:00 P.M., Pacific time, excluding holidays.

Fax Number

Transmit fax messages to **(541) 812-7611**.

E-mail Addresses

Send e-mail messages to:

• Sales Team	**info@lund.com**
• Technical Support Team	**support@lund.com**
• Documentation Team	**documentation@lund.com**
• Certified Trainers	**lti@lund.com**
• Consulting Team	**lcs@lund.com**

Lund Performance Solutions Sales Team

Lund Performance Solutions' professional sales team is available to answer your sales and customer support questions Monday through Friday during the hours 8:00 A.M., to 5:00 P.M., Pacific time, excluding major holidays.

Please contact your sales representative for information about the latest Lund Performance Solutions products, the Lund Software Subscription Plan, upgrade options and prices, and more.

Lund Performance Solutions Technical Support Team

At Lund Performance Solutions, we are working hard to provide you with intuitive software products. Additionally, we try to provide superior online and printed documentation. However, should you find yourself with a technical question that you cannot answer with the tools provided, please contact our technical support team.

NOTE You must be a registered user to access Lund Performance Solutions' support services. Lund Performance Solutions' support services are subject to Lund Performance Solutions' prices, terms, and conditions in place at the time the services are used.

E-mail Tech Support

Ask questions and receive detailed answers from the technical support team by sending an e-mail message to **support@lund.com**. Please include the product serial number with your question. You will receive a reply by e-mail.

Telephone Tech Support

You can reach the technical support team by phone at **(541) 812-7600**, Monday through Friday during the hours 8:00 A.M., to 5:00 P.M., Pacific time, excluding major holidays. Emergency technical support is also available after hours, seven days a week.

When you call, please be at your computer, have the product documentation in hand, and be prepared to provide the following information:

- Product name and version number.
- Type of computer hardware you are using.
- Software version number of your operating system(s).
- Exact wording of any messages that appear on your screen.
- What you were doing when the problem occurred.
- How you tried to solve the problem.

Lund Performance Solutions Documentation Team

Lund Performance Solutions makes every effort to produce the highest quality documentation for our products, and we welcome your feedback. If you have comments or suggestions about our online Help or printed guides, send an e-mail message to **documentation@lund.com** or contact your account manager.

Lund Training Institute Certified Trainers

Lund Training Institute presents system performance training courses at their corporate training center in Oregon and at various locations across the United States and Canada throughout the year. The Certified Trainer Program is designed for trainers from all educational areas, including academia, consulting, and business.

For information about Lund Training Institute or to receive an application, please review our website, send an e-mail message to **lti@lund.com**, or contact your account manager.

Lund Consulting Services IT Consultants

Lund Consulting Services, a division of Lund Performance Solutions, offers strategic IT solutions and expert support to a wide range of businesses. Our team of experienced IT professionals provides onsite consulting, training, and project management services to help businesses optimize their computer resources and achieve long-lasting success.

For information about Lund Consulting Services, please review our website, send an e-mail message to **lcs@lund.com**, or contact your account manager.

Product Documentation

User's Guide

This document accompanies the Developer's Toolbox software as a guide for the new user and as a quick reference for experienced users. This guide assumes that you have a working knowledge of the MPE/iX operating environment.

Online Help System

In the online Help system, you will find explanations of the many features of Developer's Toolbox, as well as tips to guide you through the program's basic functionality.

2

GETTING STARTED

If you have received an application update tape, please install all files shipped in the LPSTOOLS account. During installation, several account-level UDCs are set so that each tool can be run by typing its name. The UDCs are operable by anyone using the MGR logon. If the UDCs are not used, then the user will need to issue a run statement for the tool. All of the tools in each toolbox run out of the LPSTOOLS account.

To familiarize yourself with the on-line edit facility and available function keys for each tool, refer to Appendix E, “Standard Function Keys” on page 203, and Appendix F, “The MODIFY Editor” on page 205. For information on the standard setting you would use for each tool, please see Appendix G, “Setting Options” on page 213.

Viewing Program Version Information

To find out which version of a Tool you are using without running the Tool, issue a RUN statement in the following form:

```
RUN toolname.PUB.LPSTOOLS, VERSION
```

To view the on-line help for a Tool without running the Tool, issue a RUN statement like the one above but replace the word "version" with the word "help" as in the following:

```
RUN toolname.PUB.LPSTOOLS, HELP
```

Conventions

When showing syntax for statement entry, what you type is indented, bold and uppercase (in most cases). Commands or computer statements that are included within the text are in double quotes and bolded or in uppercase.

In the example sections illustrating computer output, ellipsis (...) indicate that lines have been removed in cases where that particular output was judged to be superfluous.

Words in angle brackets (< >) denote user-specified inputs (usually a filename).

Words in square brackets ([ ]) denote optional parameters.

Organization of this Manual

This manual is divided into 8 chapters and 8 appendices. There is a chapter devoted to each tool, and each chapter is organized alphabetically within the toolbox.

Each chapter includes full information for the particular tool, including operations, syntax, commands, examples, and any background topics that may assist you in using the tools.

3

THE AVATAR TOOL

AVATAR's decompiler capabilities include the ability to find, view, and modify the contents of any Native Mode program file, object file, executable library, or relocatable library. The AVATAR command set includes a variety of commands that simplify tasks like disassembling and modifying program files. Other features are geared towards deciphering header information in executable libraries and extracting portions of code into assembly language source.

Warning: AVATAR was designed to be used by experienced software engineers. In terms of how it is used, AVATAR is very similar in feel to Hewlett-Packard's DEBUG. Therefore, if you are not comfortable using DEBUG you will not be comfortable using AVATAR. Proceed at your own risk, exercising appropriate caution.

AVATAR is more effectively used if you understand the following concepts:

1. HPPA assembly language
2. Procedure calling
3. Parameter passing conventions

Operation

The primary use of AVATAR is to perform operations on SOMs. A SOM is a file that conforms to HP's Standard Object Module conventions. There are four classes of files with which AVATAR is particularly familiar. Each of these four classes is easily identified by its filecode:

NMPRG	Native mode program files
NMXL	Native mode executable libraries
NMRL	Native mode relocatable libraries
NMOBJ	Native mode object files

In addition to working on the file classes listed above, AVATAR can also be used as a binary editor to display and modify most other MPE files.

When AVATAR is used as a decompiler, its output is displayed as assembly language and hexadecimal constants. To add symbolic information about register usage to the disassembled display, use AVATAR's SYN command.

A complete description of the assembly language can be found in HP's *Precision Architecture and Instruction Reference Manual*. Another useful manual is HP's *Procedure Calling Convention Reference Manual*, which describes how the general registers and stack frame are set up for procedure calls. Use the CSEQ tool to display the calling sequences for MPE intrinsics.

After starting AVATAR, the **AVATAR:** prompt will be displayed. The next step is usually to OPEN a file. At that point, commands are entered to accomplish the task at hand. The general form for entering commands is:

```
AVATAR: <command> [<expression>]
```

The sections that follows describe the syntax and usage for all of AVATAR's commands as well as the structure of an expression.

Capabilities

Program capabilities required include IA, BA, PM, DS, and PH. PM is required to run DEBUG.

Usage

AVATAR can be started from the supplied UDC or from a RUN statement. AVATAR does not use the INFO string or PARM.

To start AVATAR, use one of the following methods:

- UDC

  ```
  :AVATAR
  ```

- RUN

  ```
  :RUN AVATAR.PUB.LPSTOOLS
  ```

Expression Structure

Expressions are used in many of the commands.

Syntax:

```
<expression> ::= <term> [ + | - <term> ]

<term> ::= <factor> [ * | / <factor> ]

<factor> ::= [ + | - ] <primary>

<primary> ::= [ <expression> ]

              [` <assembler instruction> `]

              [ <number> ]
```

```
[ SOM_HEADER ]
[ LST_HEADER ]
[ AUX_HEADER ]
[ SPACE_DICT ]
[ SUBSPACE_DICT ]
[ LOADER_FIXUP ]
[ SPACE_STRINGS ]
[ INIT_ARRAY ]
[ COMPILER_DICT ]
[ SYMBOL_DICT ]
[ FIXUP ]
[ SYMBOL_STRINGS ]
[ UNL_SPACE ]
[ PROCTIME ]
[ <symbol> ]
[ " <symbol> " ]
```

<assembler instruction> Is a valid assembler instruction. The instruction is enclosed in back-quotes.

<number> ::= [$ <hexadecimal digits>]

[% <octal digits>]

[# <decimal digits>]

[<digits in current radix>]

<symbol> Is the value of any symbol defined in the current SOM. If the symbol is not enclosed in quotes, then it can not be one of the previously defined words (i.e. PROCTIME) and it can only contain characters from the set 'A'..'Z', 'a'..'z', '0'..'9', '_', '$', '#', '%'.

If the name of the symbol is preceded with a ? then the value of a stub with that name is used.

Strings are also used in many commands. Strings can be given as a simple string or as a compound string. A simple string is 'zero or more characters enclosed in double-quotes'. A compound string is a list of substrings, enclosed in braces ({}). A substring can be a string enclosed in double-quotes or a number representing the value of one byte. Example: "This is a string", while {"This is a string with a new-line character" $a}.

Foundation Topic Discussions

This section discusses concepts and terminology that you may find helpful in understanding the information presented about AVATAR. First, a brief background section introduces Standard Object Modules (SOMs), and then assembly language and mapped files are discussed in relation to how they are used in AVATAR.

Standard Object Modules

Standard Object Modules are the smallest unit which may be generated by a compiler. They correspond to a given order, regardless of the file type. For instance, the architecture of an NMPRG begins with header and procedural information that is important to the operating system. After this, data and code segments follow.

A set of SOMs is defined as a library which may be either executable (NMXL) or relocatable (NMRL). Each library will contain library symbol table (LST) that describes its contents in terms of SOMs.

Relocatable libraries contain one or more SOMs that must be linked (using LINKEDIT) with the SOM that references it. Executable libraries contain one or more SOMs that have already been linked and are ready to execute. The SOMs in an executable library are dynamically loaded by MPE/iX when referenced.

Multiple SOMs can be stored in an object file, an executable library or a relocatable library. Once procedures are bound into a single SOM, they cannot be separated. AVATAR provides the capability to patch the assembler code of your compiled program. This means you now have the ability to support discontinued programs that may be important to your business or patch those almost-perfect programs when your vendor's bug priority list doesn't quite coincide with yours.

A SOM can contain many procedures that have been combined into a single SOM. Normally, once a set of procedures has been combined by a compiler into a SOM, they are not easily separated from the SOM. AVATAR's EXTRACT command breaks the SOM out into a separate ASCII file in assembler format that can be edited and assembled.

Assembly Language

Hewlett-Packard's Precessions Architecture Assembly Language is a symbolic, more approachable, representation of MPE/iX machine language. Familiarity with assembly language may prove helpful in understanding AVATAR's output, capabilities, and features.

Mapped Files

"Mapped Files" refers to the virtual address space used by files. This gives the operating system direct reference to all types of information in a manner that is reminiscent of disk-caching. Every byte of every opened file ha a unique virtual address. Portions of files are brought into real memory on demand, leaving behind other portions that are not yet required.

NOTE Use the KLONDIKE tool from the ***System Manager's Toolbox*** to view how much of a file is in real memory.

MPE/iX's treatment of virtual memory brings efficiency and flexibility to memory management that was non-existent with MPE V.

Command Summary

The following list provides a simple description of AVATAR commands that you can use to quickly locate the command that suits the task at hand. Detailed information on each command is provided in the next section.

NOTE Portions of the Command Codes are printed in uppercase to denote the part of the command that AVATAR requires in order to distinguish one command from another. The commands themselves are not case-sensitive.

Table 3.1 *AVATAR Commands*

Command Code	Description
=	Calculates a value from an expression
ASM	Shows the machine code for an assembler instruction
AUX	Prints the auxiliary headers
CALCulate	Evaluates an expression and displays the result
CALLee	Lists all calls to a given object from a code range

Command Code	Description
CALLS	Lists all call objects from a code range
CHecksum	Computes a new SOM checksum value
CLose	Closes a SOM file
COmpiler	Displays compiler information
COUnt	Counts all symbol types
DC	Displays data at a code address
DD	Displays data at a data address
Debug	Enters the system debugger
DIsasm	Shows the assembler instruction for a binary machine code
DP	Displays data starting at a procedure
DR	Displays real memory, use with extreme caution
DV	Displays data at a file offset
Exit	Terminates AVATAR
EXtract	Extracts a portion of code into an assembler source file
Find	Finds a symbol in the current SOM
FINDAll	Finds a symbol in all SOMs in the current SOM file
FIXup	Displays fixup information
FORMAT	Format data at a file offset
HELP	Invokes AVATAR help
Init	Displays initialization record information
Look	Looks at a symbols attributes
LSt	Lists all module names in SOM
MC	Modifies data at a code address
MD	Modifies data at a data address
MV	Modifies data at a file offset

Command Code	Description
Next	Displays more data, after a DC, DD, DP or DV command
Open	Opens a SOM file for processing
Quit	Exits the program
Radix	Changes the default radix
Search	Search for a value in the SOM file
SPace	Displays space header information
STatistics	Displays SOM file statistics
STRIP	Remove symbolic information from SOM
SUbspace	Displays subspace header information
SYMFormat	Format options for SYMOS information display
SYMOpen	Opens a SYMOS file for examination
SYn	Sets up synonyms for registers
UNCALLED	Displays entry points that are never called
UNWIND	Displays unwind descriptors

Most commands may be abbreviated somewhat.

Although most of the AVATAR commands require that a SOM file be open, the DV and MV commands can be used after OPENing any kind of file.

Command Definitions

This section describes AVATAR commands in detail.

=

This command has the following syntax:

```
= <expression>
```

The equal sign (=) operator when followed by an expression can be used to calculate the value of the expression.

Example 1: = `5+3`

Example 2: = `'nop'`

ASM

This command has the following syntax:

```
ASM <assembler instruction>
```

The ASM (assemble) command shows the binary machine code for an assembler instruction.

<assembler instruction> = a valid assembler instruction.

Example1: `ASM ldo 1(0),31`

Example 2: `ASM bl $$lr_unk_unk,31`

AUX

The AUX command prints all the auxiliary headers from the current SOM. The format will depend on the actual header type of each header. Each auxiliary header is constructed of 6 fields:

MANDATORY The MANDATORY field is used to indicate if this SOM contains information that the linker must understand.

COPY The COPY field is used to indicate that this auxiliary header should be copied without change to any new SOM created from this SOM.

APPEND The APPEND field is used to indicate entries with the same TYPE and APPEND fields should be merged together.

IGNORE The IGNORE field is used to indicate this auxiliary header should be ignored if its TYPE field is unknown.

TYPE The TYPE field is a numeric field that is used to describe the contents of this auxiliary header. The list of known values are provided next.

LENGTH See the following table.

Known values for the Type field are shown in the following table:

Table 3.2 *TYPE field values*

Value	Meaning	Associated Auxiliary Header
0	NULL	
1	LINK information	LINK aux header
2,7	HP Program	HP Program aux header
3	DEBUG	DEBUG aux header
4	HP-UX aux header	HP-UX aux header

Value	Meaning	Associated Auxiliary Header
5	IPL aux header	IPL aux header
6	User string aux header	User string aux header
8	SOM	HP SOM aux header

The LENGTH field contains the number of bytes in the auxiliary header less 4 bytes.

Auxiliary Header Definitions

The various headers that can be used with the AUX command are described below:

LINK This auxiliary header is used to record the last time the linker modified the SOM. The four elements in this header include:

aux header id **linker product id**

linker version id **link time**

HP Program This auxiliary header contains information that is used by the operating system to load an executable. The seven elements in this header include:

aux header id **entry name**

unsat names **search list**

capabilities **max stacksize**

max heapsize

DEBUG This auxiliary header is used to record the last time that the debugger modified the SOM. The four elements in this header include:

aux header id **debugger product id**

debugger version id **debug time**

HP-UX This auxiliary header contains information that is used by the UX loader. The eleven elements in this header include:

aux header id

execute code size

execute code offset memory

execute code offset SOM

execute data size

execute data offset memory

execute data offset SOM

execute uninitialized data size

execute start entry

execute initialized data

execute loader flags

IPL This auxiliary header contains information that is used for loading bootable utilities. The six elements in this header include:

aux header id

physical address destination

bbs size

file length

entry offset

checksum

User String This auxiliary header is used to store user definable strings. Typically the user-definable strings are defined through compiler directives like VERSION and COPYRIGHT. The three elements in this header are:

aux header id

string length

string

HP SOM This auxiliary header contains information necessary to load executable SOMs. The seven elements of this header are:

aux header id

num of XRTs

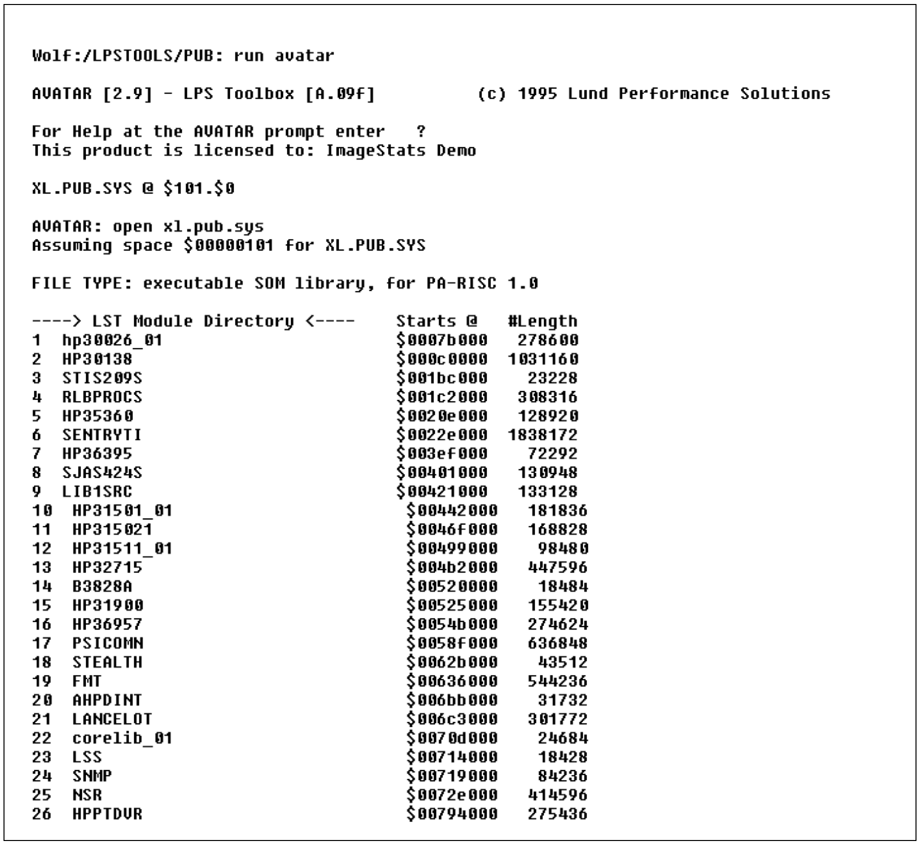
unwind end

recover end

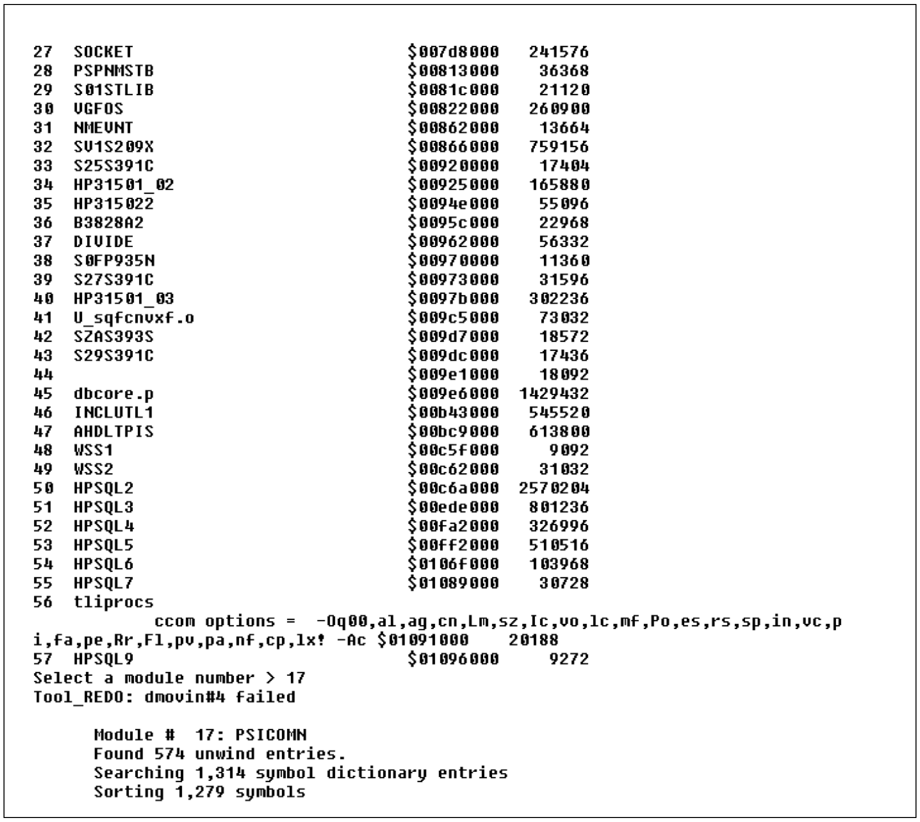
SOM flag

unwind start

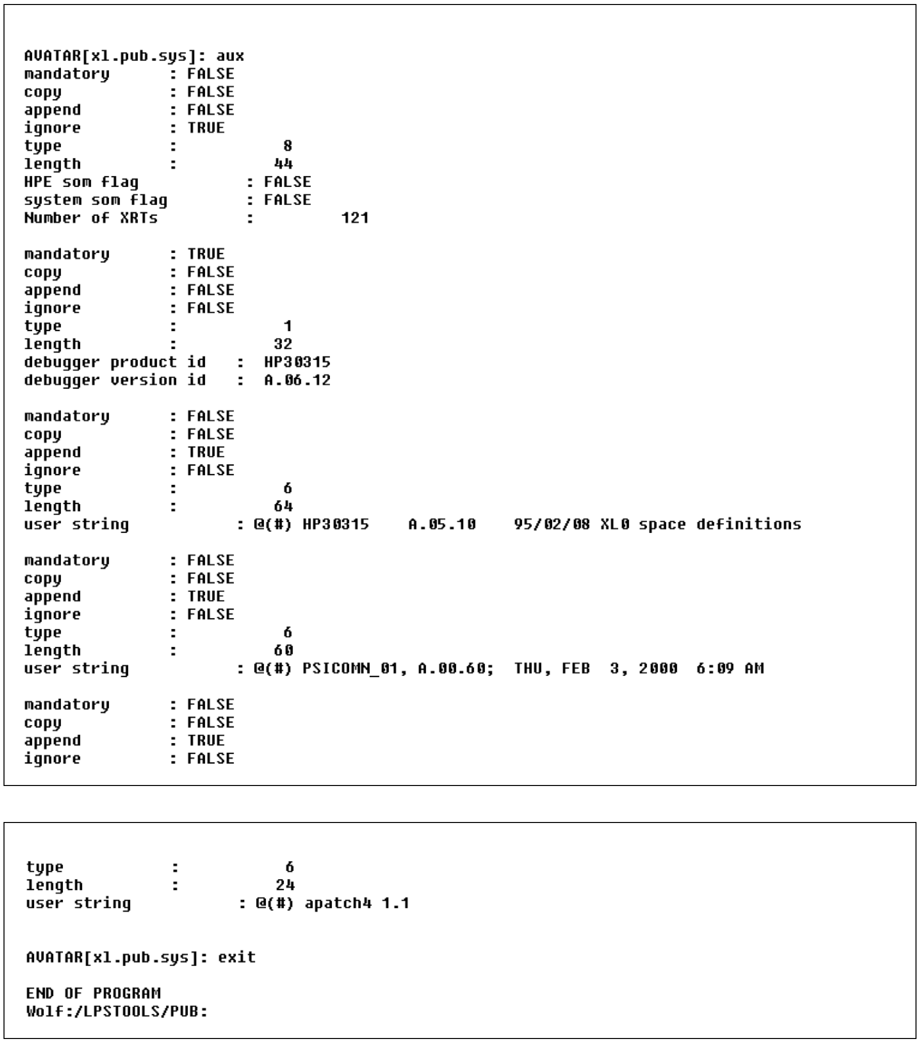
recover start



```
Wolf:/LPSTOOLS/PUB: run avatar

AVATAR [2.9] - LPS Toolbox [A.09f]          (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?
This product is licensed to: ImageStats Demo

XL.PUB.SYS @ $101.$0

AVATAR: open xl.pub.sys
Assuming space $00000101 for XL.PUB.SYS

FILE TYPE: executable SOM library, for PA-RISC 1.0

----> LST Module Directory <----    Starts @   #Length
1  hp30026_01                       $0007b000   278600
2  HP30138                          $000c0000  1031160
3  STIS209S                         $001bc000    23228
4  RLBPROCS                         $001c2000   308316
5  HP35360                          $0020e000   128920
6  SENTRYTI                         $0022e000  1838172
7  HP36395                          $003ef000    72292
8  SJAS424S                         $00401000   130948
9  LIB1SRC                          $00421000   133128
10  HP31501_01                       $00442000   181836
11  HP315021                         $0046f000   168828
12  HP31511_01                       $00499000    98480
13  HP32715                          $004b2000   447596
14  B3828A                           $00520000    18484
15  HP31900                          $00525000   155420
16  HP36957                          $0054b000   274624
17  PSICOMN                          $0058f000   636848
18  STEALTH                          $0062b000    43512
19  FMT                              $00636000   544236
20  AHPDINT                          $006bb000    31732
21  LANCELOT                         $006c3000   301772
22  corelib_01                       $0070d000    24684
23  LSS                              $00714000    18428
24  SNMP                             $00719000    84236
25  NSR                              $0072e000   414596
26  HPPTDVR                          $00794000   275436
```



```
27  SOCKET                           $007d8000    241576
28  PSPNMSTB                         $00813000     36368
29  S01STLIB                         $0081c000     21120
30  VGFOS                            $00822000    260900
31  NMEVNT                           $00862000     13664
32  SV1S209X                         $00866000    759156
33  S25S391C                         $00920000     17404
34  HP31501_02                       $00925000    165880
35  HP315022                         $0094e000     55096
36  B3828A2                          $0095c000     22968
37  DIVIDE                           $00962000     56332
38  S0FP935N                         $00970000     11360
39  S27S391C                         $00973000     31596
40  HP31501_03                       $0097b000    302236
41  V_sqfcnvxf.o                     $009c5000     73032
42  SZAS393S                         $009d7000     18572
43  S29S391C                         $009dc000     17436
44                                   $009e1000     18092
45  dbcore.p                         $009e6000   1429432
46  INCLUTL1                         $00b43000    545520
47  AHDLTPIS                         $00bc9000    613800
48  WSS1                             $00c5f000      9092
49  WSS2                             $00c62000     31032
50  HPSQL2                           $00c6a000   2570204
51  HPSQL3                           $00ede000    801236
52  HPSQL4                           $00fa2000    326996
53  HPSQL5                           $00ff2000    510516
54  HPSQL6                           $0106f000    103968
55  HPSQL7                           $01089000     30728
56  tliprocs
            ccom options =  -Oq00,al,ag,cn,Lm,sz,Ic,vo,lc,mf,Po,es,rs,sp,in,vc,p
i,fa,pe,Rr,Fl,pv,pa,nf,cp,lx! -Ac $01091000    20188
57  HPSQL9                           $01096000      9272
Select a module number > 17
Tool_REDO: dmovin#4 failed

      Module #  17: PSICOMN
      Found 574 unwind entries.
      Searching 1,314 symbol dictionary entries
      Sorting 1,279 symbols
```



```
AVATAR[xl.pub.sys]: aux
mandatory      : FALSE
copy           : FALSE
append         : FALSE
ignore         : TRUE
type           :           8
length         :          44
HPE som flag           : FALSE
system som flag        : FALSE
Number of XRTs         :         121

mandatory      : TRUE
copy           : FALSE
append         : FALSE
ignore         : FALSE
type           :           1
length         :          32
debugger product id   :  HP30315
debugger version id   :  A.06.12

mandatory      : FALSE
copy           : FALSE
append         : TRUE
ignore         : FALSE
type           :           6
length         :          64
user string           : @(#) HP30315    A.05.10    95/02/08 XL0 space definitions

mandatory      : FALSE
copy           : FALSE
append         : TRUE
ignore         : FALSE
type           :           6
length         :          60
user string           : @(#) PSICOMN_01, A.00.60;  THU, FEB  3, 2000  6:09 AM

mandatory      : FALSE
copy           : FALSE
append         : TRUE
ignore         : FALSE
```

```
type           :           6
length         :          24
user string           : @(#) apatch4 1.1


AVATAR[xl.pub.sys]: exit

END OF PROGRAM
Wolf:/LPSTOOLS/PUB:
```

Figure 3.1 *AUX Example*

CALCulate

This command has the following syntax:

```
CALCulate <expression>
```

The CALCULATE command will evaluate will evaluate an expression and display the resultant value in hexadecimal and decimal.

<expression>	An arithmetic expression

All calculations are done using 32-bit integer arithmetic.

Example1: `=5+20-$15`

Example 2: `=FOPEN`

CALLee

This command has the following syntaxes:

```
CALLee <calleename> <procedurename>
```

or

```
CALLee <calleename> <startoffset> <endoffset>
```

<calleename>	The name of the callee to be searched.
<procedurename>	The procedure in which the calls must occur. If neither a procedure name nor a range is given, then the whole SOM will be searched.
<startoffset>	The staring point from where searching begins.
<endoffset>	The ending point of the search.

This command is used to locate all calls to a given procedure **calleename** over the specified range. The **calleename** can be any symbol found in the currently selected SOM. The **calleename** cannot contain any wildcards and is case-sensitive. Ranges are specified in one of the three ways: by **procedurename**, by an explicit offset range, or not specified. An unspecified range forces a search of the entire SOM. The range specifiers can be constructed using any valid expression.

Example 1: `CALLEE fwrite`

Example 2: `CALLEE fwrite myprocedure`

Example 3: `CALLEE fwrite myprocedure+$b myprocedure+1000`

CALLS

This command has the following syntaxes:

```
CALLS <procedurename>
```

or

```
CALLS <startoffset> <endoffset>
```

<procedurename> The procedure in which the calls must occur.

If neither a procedure name nor a range is given, then the whole SOM will be searched.

<startoffset> The staring point from where searching begins.

<endoffset> The ending point of the search.

This command is used to locate calls to a given procedure over the specified range. Ranges are specified in one of the three ways: by **procedurename**, by an explicit offset range, or not specified. The latter forces a search of the entire SOM. The range specifiers can be constructed using any valid expression.

Example 1: `CALLS`

Example 2: `CALLS myprocedure`

Example 3: `CALLS myprocedure+$b myprocedure+1000`

CHecksum

This command has no parameters. It works with the currently opened SOM. The SOM header is constructed of 124 bytes. The CHECKSUM entry is the last word in the header and is not included in its calculation. Anytime AVATAR changes part of the SOM header the CHECKSUM command should be used to recalculate the CHECKSUM. The CHECKSUM word is created by exclusive "OR-ing" all of the words in the SOM header. This provides a simple, quick way of determining if the SOM contains a valid header. The CHECKSUM command will calculate the checksum for the currently opened SOM file and replace the existing value with the new one. This is necessary if you have modified the first 124 bytes of the SOM file.

CLose

This command will close the currently opened SOM file. This really is not necessary because the OPEN command will close any opened SOM file.

COmpiler

This command has no parameters. Output from this command provides general information about the SOM, as well as information specific to each module within the SOM. For example, a compiled C module could contain the following information. For a NMOBJ file on a Spectrum machine, general information items (numbers in hexadecimal) are:

system id = 20B	Spectrum architecture
magic# = 106	Relocatable SOM
version id = 5124000	The ending point of the search.

SOM-specific information:

source file name	**= C2SPL**
language name	**= HP-C**
product id	**= HP31506**
version id	**= C/XL Compiler Version A.01.22**

Other magic numbers that might be generated for a NMOBJ file on a Spectrum machine include the following:

These two entries have Library Symbol Table headers

$104 = executable library (NMXL)

$619 = relocatable library (NMRL)

These entries have SOM headers.

$106 = relocatable SOM

$107 = non-shareable, executable SOM

$108 = shareable, executable SOM

$10B = shareable, demand-loaded executable SOM

COUnt

This command is used to display the symbol type and corresponding scopes for all symbols in the current SOM in tabular format. Provided in the following list is a complete listing and short definition of each symbol type for a SOM. Following the symbol types list is a similar list for symbol scopes.

Table 3.3 *SOM Symbol Types*

Symbol	Description
NULL	Invalid symbol record
ABSOLUTE	Absolute constant
DATA	Normal initialized data
CODE	Unspecified code, resolved at link time
PRI_PROG	Primary program entry point
SEC_PROG	Secondary program entry point
ENTRY	Code entry point symbols

Symbol	Description
STORAGE	Storage requirement; known length, unknown value
STUB	External call stub, or relocation stub
MODULE	Source module name
SYM_EXT	Extension record of the current entry
ARG_EXT	Extension record of the current entry
MILLICODE	Name of a millicode subroutine
PLABEL	Procedure label
OCT_DIS	Used by OCT (Object Code Translator)
MILLI_EXT	Address of an external millicode subroutine

Table 3.4 *SOM Symbol Scopes*

Symbol	Description
UNSAT	Unsatisfied import request
EXTRN	Import request to a symbol in another SOM
LOCAL	Private symbol
UNIVERSAL	Symbol to be exported outside the SOM

	SCOPE			
TYPE	**UNSAT**	**LOCAL**	**EXTERNAL**	**UNIVERSAL**
NULL				
ABSOLUTE	X	X		X
DATA	X	X		X
CODE				X
PRI_PROG				X
SEC_PROG				X
ENTRY		X		X

STORAGE	X	X		X
STUB		X	X	
MODULE		X		X
SYM_EXT				
ARG_EXT				
MILLICODE	X	X		X
PLABEL		X		
OCT_DIS		X		X
MILLI_EXT				X

DC

This command has the following syntax:

```
DC <code offset> [<display format>] [<numlines>]
```

The DC command will display data at a given code address. The data will be displayed in assembler format by default, but other formats may be specified.

<code offset> An expression giving the offset to the start of the current code module, where data to display starts.

<display format> Either C or D. Default is C. If D is specified then data is displayed in hex and ascii format. If C is specified then data is displayed as disassembled code.

<numlines> A decimal number, indicating the number of lines to show.

Example 1: `DC my_procedure_name`

Example 2: `DC proc + $15 D 12`

DD

This command has the following syntax:

```
DD <data offset> [<display format>] [<numlines>]
```

The DD command will display values (data) from an initialized block within the current module of the opened SOM file. Data will be displayed in hexadecimal and ASCII formats by default.

<data offset> An expression giving the start within an initialized block, where display starts.

<display format> Either C or D. Default is D. If D is specified then data is displayed in hex and ascii format. If C is specified then data is displayed as disassembled code.

<numlines> A decimal number, indicating the number of lines to show.

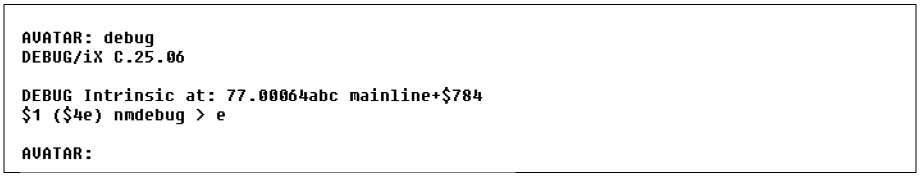
The default value is to keep on displaying until the next control-y or '/' reply.

Example 1: `DD $40000008`

Example 2: `DD dl_area 20`

Debug

This command invokes the system debug program, DEBUG. There are no parameters for this command.



```
AVATAR: debug
DEBUG/iX C.25.06

DEBUG Intrinsic at: 77.00064abc mainline+$784
$1 ($4e) nmdebug > e

AVATAR:
```

Figure 3.2 *DEBUG Command*

See the *System Debug Reference Manual* for details on using DEBUG.

DIsasm

This command has the following syntax:

```
DIsasm <expression>
```

The DISASM command shows the assembler instruction corresponding to a machine code defined by expression.

Example 1: `DISASM $8000240`

Example 2: `DISASM 'nop'`

NOTE The assembler instruction is enclosed in back-quotes.

DP

This command has the following syntax:

```
DP <procedure name>
```

The DP command will display code for a given procedure. The data will be displayed in assembler format.

<procedure name>	Any procedure that has been defined in the current module of the opened SOM file.

Example: `DP my_procedure_name`

DR

This command has the following syntax:

```
DR <expression>
```

The DR command will display real memory starting at the physical address given by the **expression**.

Example: `AVATAR DR $4000` (see Figure 3.22)

DV

This command has the following syntax:

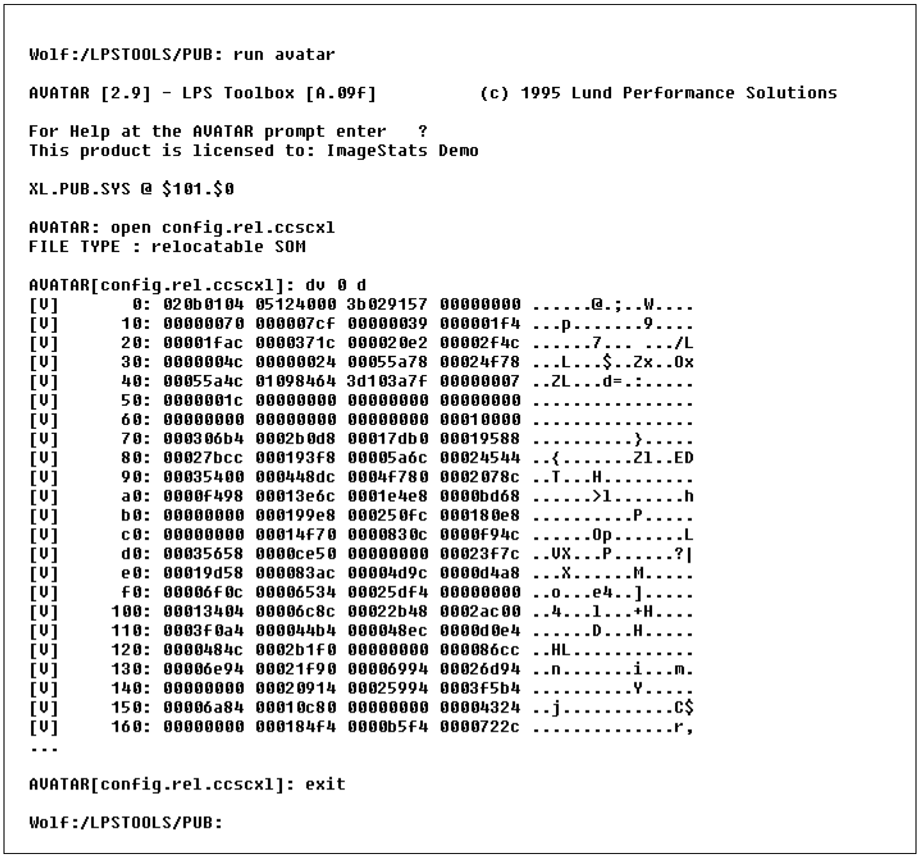
```
DV <data offset> [<display format>] [<numlines>]
```

The DV command will display values (data) from within the current SOM file. Data will be displayed in hexadecimal and ASCII formats by default.

<data offset>	An expression giving the offset from the start of the file, where display starts.
<display format>	Either C or D. Default is D. If D is specified then data is displayed in hex and ascii format. If C is specified then data is displayed as disassembled code.
<numlines>	A decimal number, indicating the number of lines to show. The default value is to keep on displaying until the next control-y or '/' reply.

Example 1: `DV $100`	Displays data starting at address hex 100
Example 2: `DV 5+20-$15 C`	Display lines at address 4, disassembled as code

To dump the SOM header for a NMOBJ, try the following:



```
Wolf:/LPSTOOLS/PUB: run avatar

AVATAR [2.9] - LPS Toolbox [A.09f]          (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?
This product is licensed to: ImageStats Demo

XL.PUB.SYS @ $101.$0

AVATAR: open config.rel.ccscxl
FILE TYPE : relocatable SOM

AVATAR[config.rel.ccscxl]: dv 0 d
[V]       0: 020b0104 05124000 3b029157 00000000 ......@.;..W....
[V]      10: 00000070 000007cf 00000039 000001f4 ...p.......9....
[V]      20: 00001fac 0000371c 000020e2 00002f4c ......7... .../L
[V]      30: 0000004c 00000024 00055a78 00024f78 ...L...$..Zx..Ox
[V]      40: 00055a4c 01098464 3d103a7f 00000007 ..ZL...d=.:.....
[V]      50: 0000001c 00000000 00000000 00000000 ................
[V]      60: 00000000 00000000 00000000 00010000 ................
[V]      70: 000306b4 0002b0d8 00017db0 00019588 ..........}.....
[V]      80: 00027bcc 000193f8 00005a6c 00024544 ..{.......Zl..ED
[V]      90: 00035400 000448dc 0004f780 0002078c ..T...H.........
[V]      a0: 0000f498 00013e6c 0001e4e8 0000bd68 ......>l.......h
[V]      b0: 00000000 000199e8 000250fc 000180e8 ..........P.....
[V]      c0: 00000000 00014f70 0000830c 0000f94c ......Op.......L
[V]      d0: 00035658 0000ce50 00000000 00023f7c ..VX...P......?|
[V]      e0: 00019d58 000083ac 00004d9c 0000d4a8 ...X.......M.....
[V]      f0: 00006f0c 00006534 00025df4 00000000 ..o...e4..].....
[V]     100: 00013404 00006c8c 00022b48 0002ac00 ..4...l...+H....
[V]     110: 0003f0a4 000044b4 000048ec 0000d0e4 ......D...H.....
[V]     120: 0000484c 0002b1f0 00000000 000086cc ..HL............
[V]     130: 00006e94 00021f90 00006994 00026d94 ..n.......i...m.
[V]     140: 00000000 00020914 00025994 0003f5b4 ..........Y.....
[V]     150: 00006a84 00010c80 00000000 00004324 ..j...........C$
[V]     160: 00000000 000184f4 0000b5f4 0000722c ..............r,
...

AVATAR[config.rel.ccscxl]: exit

Wolf:/LPSTOOLS/PUB:
```

Figure 3.3 *Dumping the SOM*

Exit

The EXIT command terminates AVATAR.

EXtract

This command has the following syntax:

```
EXtract <file name> <start> [<end>]
```

The EXTRACT command extracts a portion of code into an ASCII file. This file can be used as input to the ASSEMBLER.

<file name> The name of a file to be created. The file may not exist.

<start> The starting point from which extraction begins.

<end> The last instruction to be extracted. If omitted, AVATAR tries to extract to the end of the procedure referred to in <start>.

Example 1: `EXTRACT file1 $1000 $2000`

Example 2: `EXTRACT file2 my_proc my_proc+1000`

Example 3: `EXTRACT foo fopen_nm`

Find

This command has the following syntax:

```
Find <string> <filter> | <symbol> <filter>
```

The FIND command will find all entries in the symbol dictionary that have a name that matches or partially matches the provided string. Entries in the symbol dictionary include all procedures, global data items, intrinsics, etc. The search is limited to the current module of the opened SOM file. The FINDALL command searches through all modules.

<string> Any string of ASCII characters enclosed in double quotes.

<symbol> Any string of valid symbol characters.

<filter> A symbol type to be filtered out. The default is that all symbols are listed. The possible filter values are listed below:

Filter Types

ABSOLUTE
ARG_EXT
CODE
DATA
ENTRY
EXTERNAL
LOCAL
MILLICODE
MILLI_EXT
MODULE
NULL
OCT_DIS
PLABEL
PRI_PROG
SEC_PROG
STORAGE
STUB
SYM_EXT
UNIVERSAL
UNSAT

NOTE To display all symbols, pass a Null string to the FIND command.

Example 1: `FIND time`	Find any entries that includes the strings time in the name.
Example 2: `FIND $$divoI millicode`	

FINDAll

This command has the following syntax:

```
FINDAll <string> <filter> | <symbol> <filter>
```

The FINDALL command will find all entries in the symbol dictionary that have a name that matches or partially matches the provided string. Entries in the symbol dictionary include all procedures, global data items, intrinsics, etc. The search will include all modules in the currently opened SOM file, unlike the FIND command (which searches only the current module).

This is extremely helpful when looking through an XL or even NL.PUB.SYS!

<string>	Any string of ASCII characters.
<symbol>	Any string of valid symbol characters.
<filter>	A symbol type to be filtered out. The default is that all symbols are listed.

The possible filter values are noted in the Filter Types list for the FIND command. See the section on “Filter Types” on page 28 for a complete list.

FIXup

The FIXUP command displays the fixup records from an object file. Currently only fixups generated by the ASSEMBLER are correctly recognized.

Key to FIXUP Output

Symbol_type

ABSOLUTE	Absolute constant
ARG_EXT	Extension record of the current entry
CODE	Unspecified code, resolved at link time
DATA	Normal initialized data
ENTRY	Code entry point symbols
MILLICODE	Name of a millicode subroutine
MILLI_EXT	Address of an external millicode subroutine
MODULE	Source module name

NULL	Invalid symbol record
OCT_DIS	Used by OCT (Object Code Translator)
PLABEL	Procedure label
PRI_PROG	Primary program entry point
SEC_PROG	Secondary program entry point
STORAGE	Storage requirement with known length & unknown value
STUB	External call stub, or relocation stub
SYM_EXT	Extension record of the current entry

Symbol_scope

EXTERNAL	Import request to a symbol in another SOM
LOCAL	Private symbol
UNIVERSAL	Symbol to be exported outside the SOM
UNSAT	Unsatisfied import request

Check_level

Determines the type checking error level that the linker uses while binding external references to procedures and global variables. All object modules indicate a checking level for each reference and each definition of a procedure or a global variable. When binding an external reference to a definition, the linker compares the type information at the lower of the two checking levels specified by the reference and the definition. If a type mismatch is found, the linker reports it as either a warning or an error.

The values for **check_level** are:

0 All type mismatches are warnings.

1 Mismatches of the procedures, function, or variable type are errors; all other mismatches are warnings.

2 Mismatches of the procedures, function, or variable type, and mismatches of the number of arguments for procedures or functions are errors; all other mismatches (i.e., parameter types) are warnings.

3 All type mismatches are errors.

Must_qualify

Used to indicate if more than one entry has the same symbol name.

0	Multiple symbol not present
1	Multiple symbol present

Initially_frozen

Code using this symbol will be locked into physical memory when the operating system is booted.

0	Not frozen
1	Frozen

Memory_resident

Indicates that the code that will use the symbol is frozen in memory. This provides a way for frozen procedures to communicate.

0	Not in memory
1	In memory

Is_common

Used to indicate if a symbol is in an initialized common data area.

0	Not in common
1	In common

Duplicate_common

Used to indicate if the source language allows duplicate initialization.

0	No allowed
1	Allowed

Xleast

Execution level that is required to call this entry point.

0:

1:

2:

3:

Argument Relocation

This fixup information is used to communicate the locations of the first four parameters, and the function return parameter. The linker used this information to match up exported symbol information with fixup references. The four possible values for this field are:

0	"Do not relocate"
1	"Argument register"
2	"Floating point coprocessor register, low"
3	"Floating point coprocessor register, high"

Code offset

Offset into the SOM where "symbol name" code begins.

Privilege_level

Determines the privilege level used by the executable program file. This parameter changes the privilege level of all procedures in the symbol and export tables (of the relocatable object file) that were set during compilation. The values for this field are:

0	System level access
1	Unused
2	Privileged level access
3	User level access

FORMAT

This command has the following syntax:

```
FORMAT <data offset> <format> <count>
```

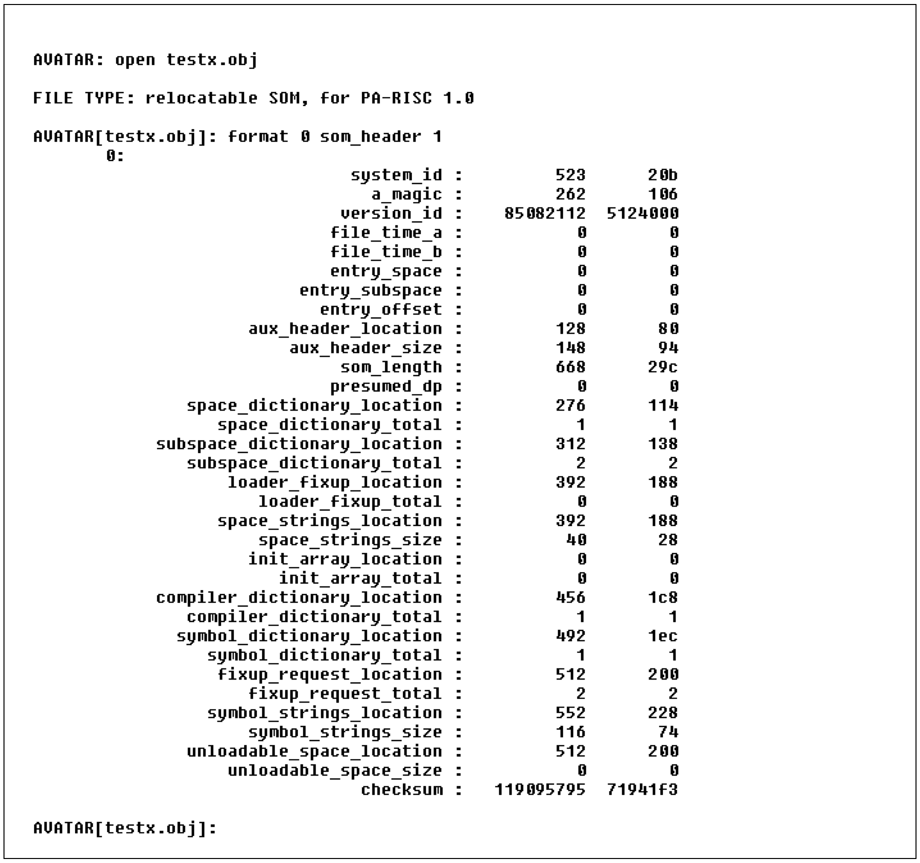
The FORMAT command allows to display data relative to the start of the SOM file in one of many different formats.

<data offset>	An expression giving the offset from the start of the file, where formatting starts.
<format>	The format specifier. The valid format specifiers are listed below. The composition of each format is detailed in the “Format Specifier Definitions” on page 34.
<count>	An integer giving the number of elements to format.

Format Specifier List

AUX_HEADER_ID
ARG_DESCRIPTOR
COMPILER_REC
FIXUP_BITS
FIXUP_REC
INIT_REC
LST_BITS
LST_HEADER
LST_SYMBOL
SOM_HEADER
SPACE_REC
SUBSPACE_BITS
SUBSPACE_REC
SYMBOL_DICT_BITS
SYMBOL_DICT_REC
SYMDICT_ARG_REC
SYMDICT_EXT_REC
UNWIND_DESCRIPT
UNWIND_ENTRY

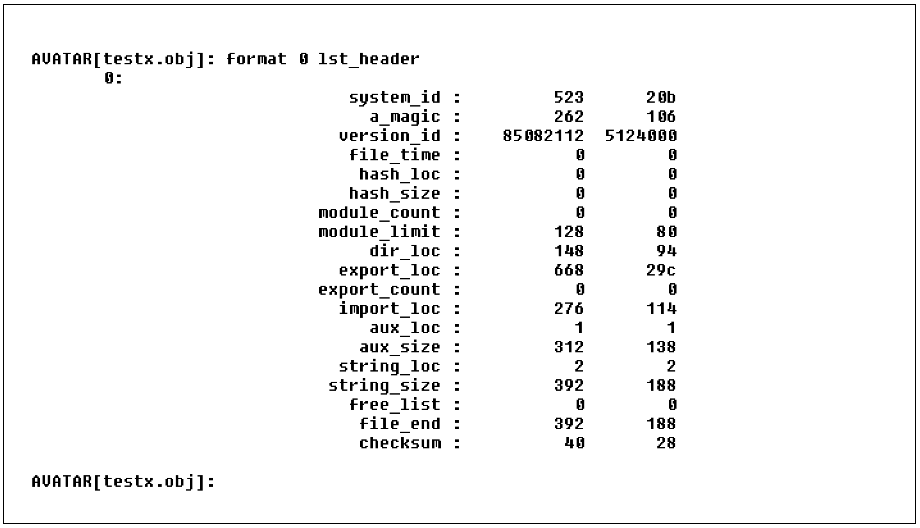
Sample output using the format specifier SOM_HEADER:



```
AVATAR: open testx.obj

FILE TYPE: relocatable SOM, for PA-RISC 1.0

AVATAR[testx.obj]: format 0 som_header 1
       0:
                               system_id :         523      20b
                                 a_magic :         262      106
                              version_id :    85082112  5124000
                             file_time_a :           0        0
                             file_time_b :           0        0
                             entry_space :           0        0
                          entry_subspace :           0        0
                            entry_offset :           0        0
                     aux_header_location :         128       80
                         aux_header_size :         148       94
                              som_length :         668      29c
                             presumed_dp :           0        0
               space_dictionary_location :         276      114
                  space_dictionary_total :           1        1
            subspace_dictionary_location :         312      138
               subspace_dictionary_total :           2        2
                   loader_fixup_location :         392      188
                      loader_fixup_total :           0        0
                  space_strings_location :         392      188
                      space_strings_size :          40       28
                     init_array_location :           0        0
                        init_array_total :           0        0
            compiler_dictionary_location :         456      1c8
               compiler_dictionary_total :           1        1
              symbol_dictionary_location :         492      1ec
                 symbol_dictionary_total :           1        1
                  fixup_request_location :         512      200
                     fixup_request_total :           2        2
                 symbol_strings_location :         552      228
                     symbol_strings_size :         116       74
               unloadable_space_location :         512      200
                   unloadable_space_size :           0        0
                                checksum :   119095795  71941f3

AVATAR[testx.obj]:
```

Figure 3.4 *SOM_HEADER Format*

Sample output using the format specifier LST_HEADER:



```
AVATAR[testx.obj]: format 0 lst_header
       0:
                             system_id :        523      20b
                               a_magic :        262      106
                            version_id :   85082112  5124000
                             file_time :          0        0
                              hash_loc :          0        0
                             hash_size :          0        0
                          module_count :          0        0
                          module_limit :        128       80
                               dir_loc :        148       94
                            export_loc :        668      29c
                          export_count :          0        0
                            import_loc :        276      114
                               aux_loc :          1        1
                              aux_size :        312      138
                            string_loc :          2        2
                           string_size :        392      188
                             free_list :          0        0
                              file_end :        392      188
                              checksum :         40       28

AVATAR[testx.obj]:
```

Figure 3.5 *LST_HEADER Format*

Format Specifier Definitions

The LST_HEADER is composed of the following elements:

system_id

a_magic

version_id

file_time

hash_loc

hash_size

module_count

module_limit

dir_loc

export_loc

export_count

import_loc

aux_loc

aux_size

string_loc

string_size

free_list

file_end

checksum

The LST_BITS is composed of the following elements:

hidden

symbol_type

symbol_scope

check_level

must_qualify

initially_frozen

memory_resident

is_common

duplicate_common

xleast

arg_relocation

The LST_SYMBOL is composed of the following elements:

ls_bits
symbol_name_ptr
qualifier_name_ptr
symbol_info
symbol_descriptor
max_num_args
min_num_args
num_args
som_index
symbol_key
next_entry

The SOM_HEADER is composed of the following elements:

system_id
a_magic
version_id
file_time_a
file_time_b
entry_space
entry_subspace
entry_offset
aux_header_location
aux_header_size
som_length
presumed_dp
space_dictionary_location
space_dictionary_total
subspace_dictionary_location
subspace_dictionary_total
loader_fixup_location
loader_fixup_total
space_strings_location
space_strings_size
init_array_location
init_array_total
compiler_dictionary_location
compiler_dictionary_total
symbol_dictionary_location
symbol_dictionary_total
fixup_request_location
fixup_request_total
symbol_strings_location
symbol_strings_size
unloadable_space_location
unloadable_space_size
checksum

The AUX_HEADER_ID is composed of the following elements:

mandatory
copy
append
ignore
type
length

The SPACE_REC is composed of the following elements:

name_pointer
access_bits
sort_key
number
subspace_index
subspace_quantity
loader_fixup_index
loader_fixup_quantity
init_pointer_index
init_pointer_quantity

The SUBSPACE_BITS is composed of the following elements:

acb
is_loadable
sort_key

memory_resident

duplicate_common

is_common

quadrant

initially_frozen

code_only

replicate_init

continuation

The SUBSPACE_REC is composed of the following elements:

space_index

bits

file_loc

init_length

startfixup_request_qty

length

reserved

alignment

name_pointer

fixup_request_index

The COMPILER_REC is composed of the following elements:

name

language_name

product_id

version_id

compile_time

source_time

The FIXUP_BITS is composed of the following elements:

need_data_reference

arg_relocation

expression_type

execution_level

fixup_format

fixup_field

The FIXUP_REC is composed of the following elements:

bits

space_offset

symbol_index_one

symbol_index_two

fixup_constant

The INIT_REC is composed of the following elements:

space_index

file_location

memory_resident

access_control

has_data

offset

new_locality

length

initially_frozen

The SYMBOL_DICT_BITS is composed of the following elements:

bits

symbol_info

symbol_name_ptr

symbol_value

qualifier_name_ptr

The ARG_DESCRIPTOR is composed of the following elements:

packing
structure
alignment
hash
mode
data_type

The SYMBOL_DICT_REC is composed of the following elements:

bits
symbol_info
symbol_name_ptr
symbol_value
qualifier_name_ptr

The SYMDICT_EXT_REC is composed of the following elements:

type
max_num_args
min_num_args
num_args
symbol_descriptor
arg_descriptor[1]
arg_descriptor[2]
arg_descriptor[3]

The SYMDICT_ARG_REC is composed of the following elements:

type
arg_descriptor[1]
arg_descriptor[2]
arg_descriptor[3]
arg_descriptor[4]

The UNWIND_DESCRIPT is composed of the following elements:

cannot_unwind
millicode
millicode_save_sr0
region_description
save_srs
entry_fr
entry_gr
args_stored
call_fr
call_gr
save_sp
save_rp
save_mrp_in_frame
cleanup_defined
hpe_interrupt_marker
hpux_interrupt_marker
large_frame_r3
total_framesize

The UNWIND_ENTRY is composed of the following elements:

starting_offset
ending_offset

Example: `FORMAT LST_HEADER LST_HEADER`

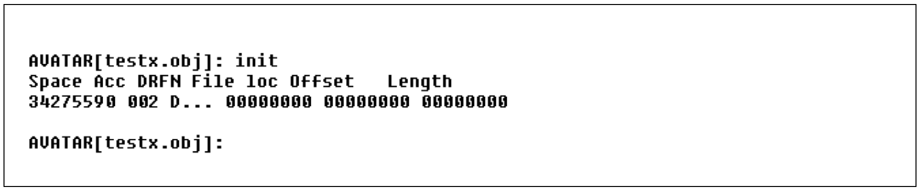
The first LST_HEADER is an expression, giving an offset in the file to where the LST HEADER starts, While the second one is a format specifier.

HELP

The HELP command invokes AVATAR Help.

Init

The INIT command will display all the compiler init entries for the current module of the opened SOM file. This information comes from the INIT_REC portion of the SOM.



```
AVATAR[testx.obj]: init
Space Acc DRFN File loc Offset   Length
34275590 002 D... 00000000 00000000 00000000

AVATAR[testx.obj]:
```

Figure 3.6 *INIT Command*

Space = index into the space dictionary

Acc = access control bits, valid value from 0-7

0: read only data page

1: normal data page

2: normal code page

3: dynamic code page

4: gateway to PL0

5: gateway to PL1

6: gateway to PL2

7: gateway to PL3

DRFN = bit encoded flag

D: data defined in SOM for this space

R: this subspace is locked into physical memory once the subspace goes into execution

F: this subspace is locked into physical memory when the operating system is booted

N: initialization pointers starts a new locality set

File-loc	=	if ’D’ (data defined in SOM), then this entry points at the data to initialize one or more subspaces. Otherwise this field contains a 32-bit value used to initialize one or more subspaces.
Length	=	if ’D’, then this field contains the byte length of the area to initialize. Otherwise it is undefined.
Offset	=	byte offset where initialization is to start, relative to the start of the space.

Look

This command has the following syntax:

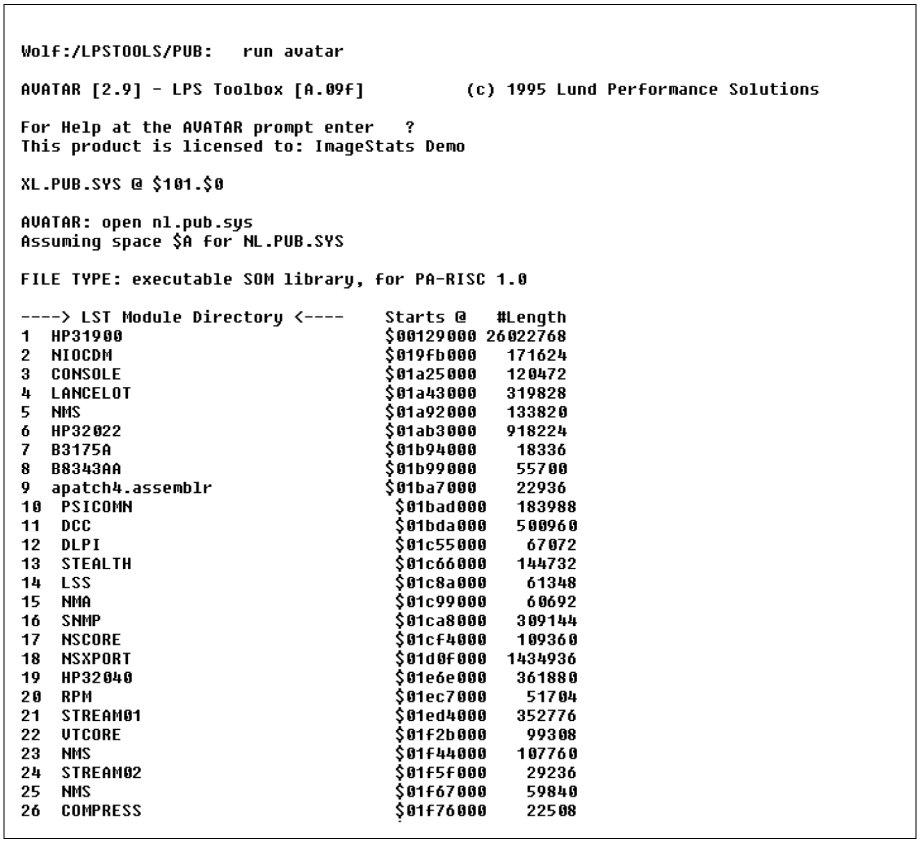
```
Look <string>
     <symbol>
```

The LOOK command will display all the available information about a given entry in the symbol dictionary. See the FIND command. For example, this command can be used to find the argument types and order for any procedure, anywhere. Parameters for this command are case-sensitive.

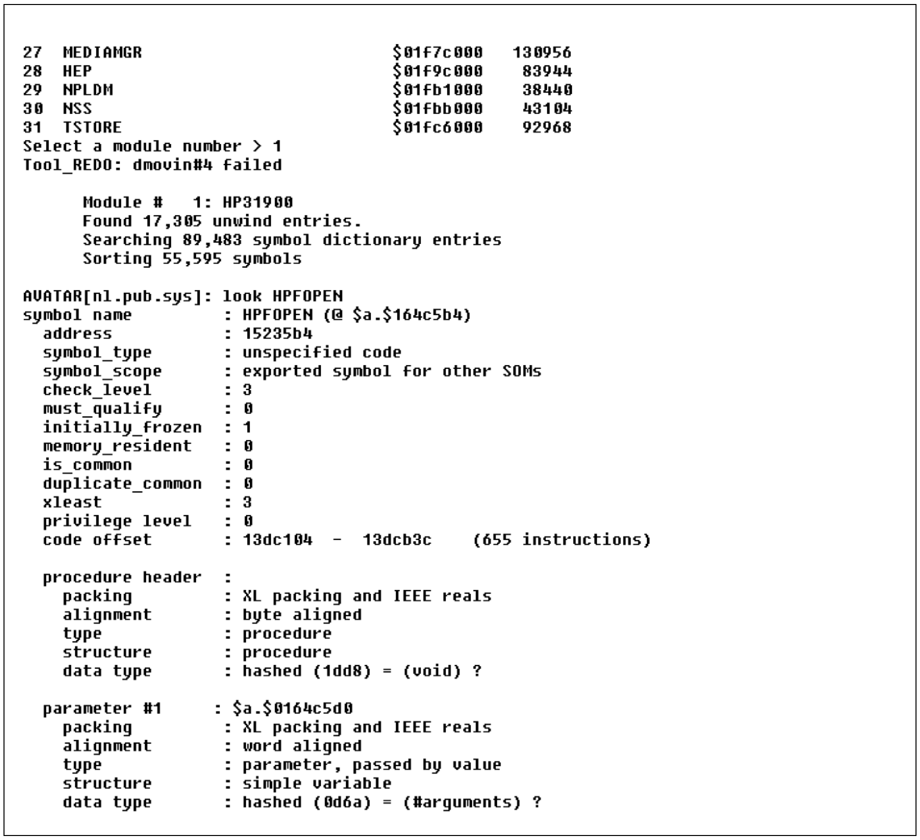
<string>	Any string of ASCII characters.
<symbol>	Any string of valid symbol characters.

Example: `LOOK HPFOPEN`

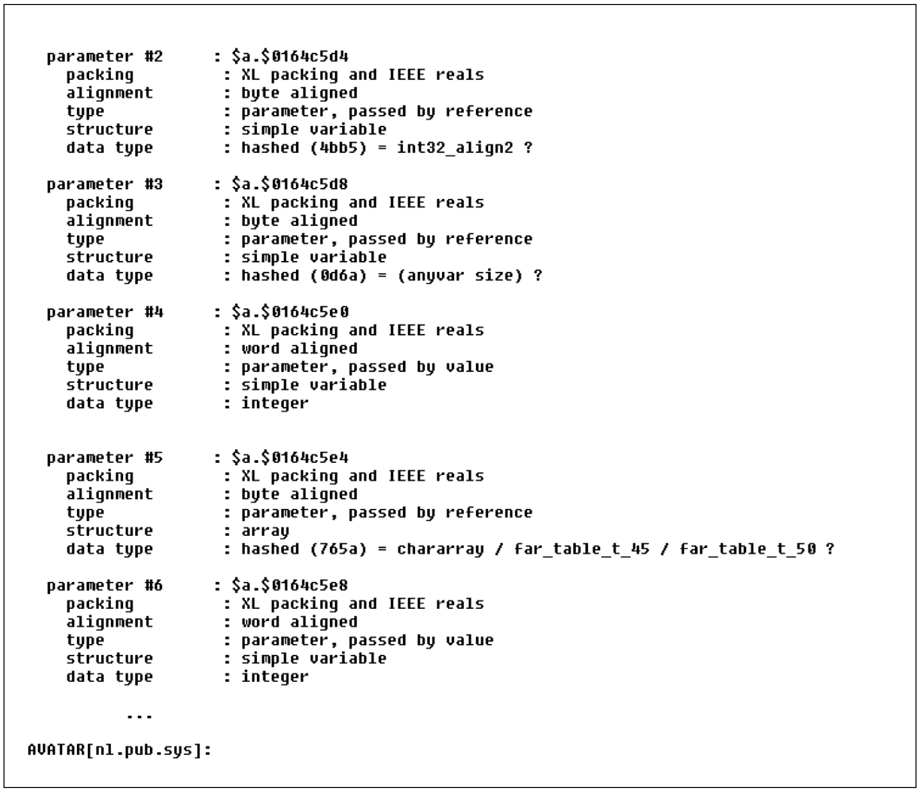
Displays all information for the intrinsic HPFOPEN (See Figure 3.7)



```
Wolf:/LPSTOOLS/PUB:   run avatar

AVATAR [2.9] - LPS Toolbox [A.09f]          (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?
This product is licensed to: ImageStats Demo

XL.PUB.SYS @ $101.$0

AVATAR: open nl.pub.sys
Assuming space $A for NL.PUB.SYS

FILE TYPE: executable SOM library, for PA-RISC 1.0

----> LST Module Directory <----    Starts @   #Length
1  HP31900                          $00129000 26022768
2  NIOCDM                           $019fb000   171624
3  CONSOLE                          $01a25000   120472
4  LANCELOT                         $01a43000   319828
5  NMS                              $01a92000   133820
6  HP32022                          $01ab3000   918224
7  B3175A                           $01b94000    18336
8  B8343AA                          $01b99000    55700
9  apatch4.assemblr                 $01ba7000    22936
10  PSICOMN                          $01bad000   183988
11  DCC                              $01bda000   500960
12  DLPI                             $01c55000    67072
13  STEALTH                          $01c66000   144732
14  LSS                              $01c8a000    61348
15  NMA                              $01c99000    60692
16  SNMP                             $01ca8000   309144
17  NSCORE                           $01cf4000   109360
18  NSXPORT                          $01d0f000  1434936
19  HP32040                          $01e6e000   361880
20  RPM                              $01ec7000    51704
21  STREAM01                         $01ed4000   352776
22  VTCORE                           $01f2b000    99308
23  NMS                              $01f44000   107760
24  STREAM02                         $01f5f000    29236
25  NMS                              $01f67000    59840
26  COMPRESS                         $01f76000    22508
```

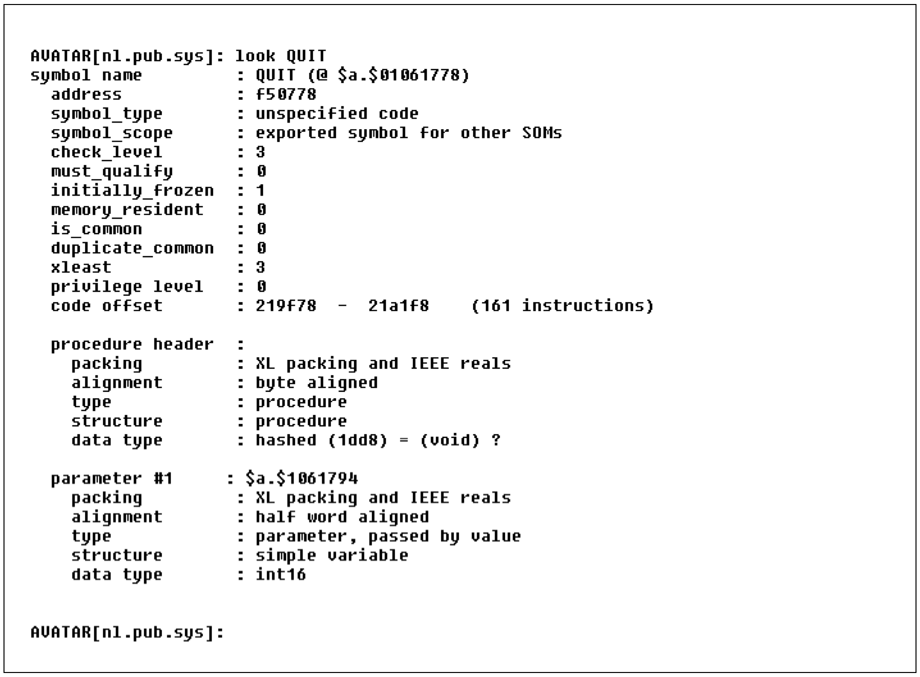


```
27  MEDIAMGR                         $01f7c000   130956
28  HEP                              $01f9c000    83944
29  NPLDM                            $01fb1000    38440
30  NSS                              $01fbb000    43104
31  TSTORE                           $01fc6000    92968
Select a module number > 1
Tool_REDO: dmovin#4 failed

      Module #   1: HP31900
      Found 17,305 unwind entries.
      Searching 89,483 symbol dictionary entries
      Sorting 55,595 symbols

AVATAR[nl.pub.sys]: look HPFOPEN
symbol name         : HPFOPEN (@ $a.$164c5b4)
  address           : 15235b4
  symbol_type       : unspecified code
  symbol_scope      : exported symbol for other SOMs
  check_level       : 3
  must_qualify      : 0
  initially_frozen  : 1
  memory_resident   : 0
  is_common         : 0
  duplicate_common  : 0
  xleast            : 3
  privilege level   : 0
  code offset       : 13dc104  -  13dcb3c    (655 instructions)

  procedure header  :
    packing         : XL packing and IEEE reals
    alignment       : byte aligned
    type            : procedure
    structure       : procedure
    data type       : hashed (1dd8) = (void) ?

  parameter #1     : $a.$0164c5d0
    packing         : XL packing and IEEE reals
    alignment       : word aligned
    type            : parameter, passed by value
    structure       : simple variable
    data type       : hashed (0d6a) = (#arguments) ?
```



```
  parameter #2      : $a.$0164c5d4
    packing          : XL packing and IEEE reals
    alignment        : byte aligned
    type             : parameter, passed by reference
    structure        : simple variable
    data type        : hashed (4bb5) = int32_align2 ?

  parameter #3      : $a.$0164c5d8
    packing          : XL packing and IEEE reals
    alignment        : byte aligned
    type             : parameter, passed by reference
    structure        : simple variable
    data type        : hashed (0d6a) = (anyvar size) ?

  parameter #4      : $a.$0164c5e0
    packing          : XL packing and IEEE reals
    alignment        : word aligned
    type             : parameter, passed by value
    structure        : simple variable
    data type        : integer


  parameter #5      : $a.$0164c5e4
    packing          : XL packing and IEEE reals
    alignment        : byte aligned
    type             : parameter, passed by reference
    structure        : array
    data type        : hashed (765a) = chararray / far_table_t_45 / far_table_t_50 ?

  parameter #6      : $a.$0164c5e8
    packing          : XL packing and IEEE reals
    alignment        : word aligned
    type             : parameter, passed by value
    structure        : simple variable
    data type        : integer

          ...

AVATAR[nl.pub.sys]:
```

Figure 3.7 *LOOK HPFOPEN*



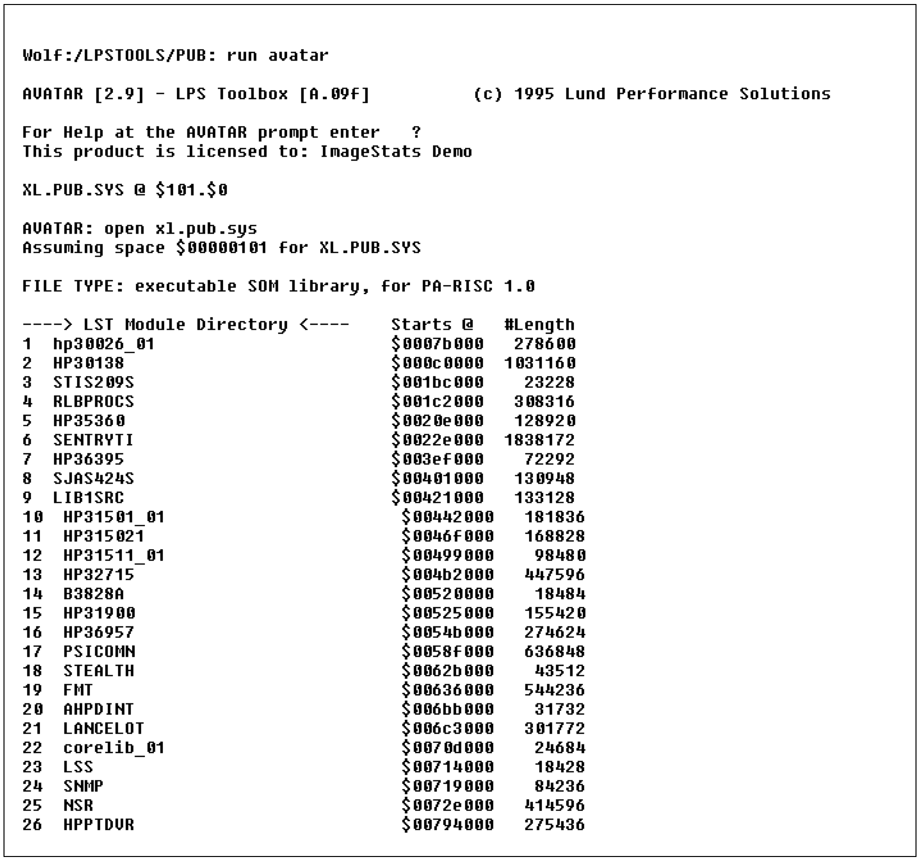
```
AVATAR[nl.pub.sys]: look QUIT
symbol name          : QUIT (@ $a.$01061778)
  address            : f50778
  symbol_type        : unspecified code
  symbol_scope       : exported symbol for other SOMs
  check_level        : 3
  must_qualify       : 0
  initially_frozen   : 1
  memory_resident    : 0
  is_common          : 0
  duplicate_common   : 0
  xleast             : 3
  privilege level    : 0
  code offset        : 219f78  -  21a1f8    (161 instructions)

  procedure header   :
    packing          : XL packing and IEEE reals
    alignment        : byte aligned
    type             : procedure
    structure        : procedure
    data type        : hashed (1dd8) = (void) ?

  parameter #1      : $a.$1061794
    packing          : XL packing and IEEE reals
    alignment        : half word aligned
    type             : parameter, passed by value
    structure        : simple variable
    data type        : int16


AVATAR[nl.pub.sys]:
```

Figure 3.8 *LOOK QUIT*

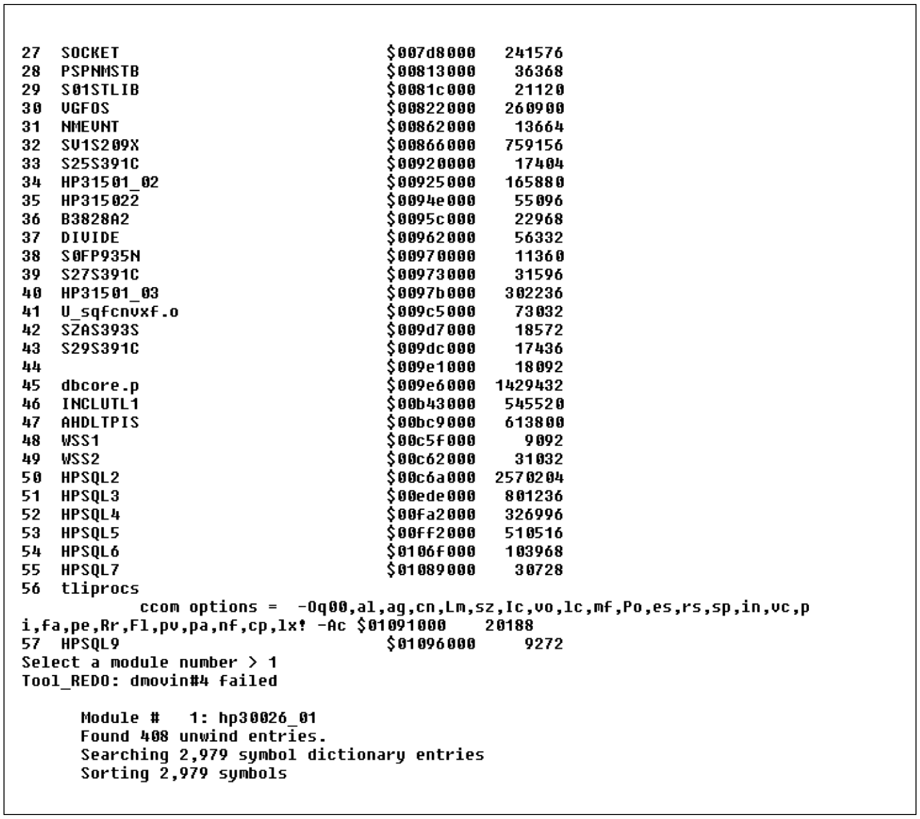
LSt

The LST command will display all the available modules in the currently opened SOM file and allow you to select a module. When a SOM with multiple entries is OPENed, the LST command is automatically invoked.

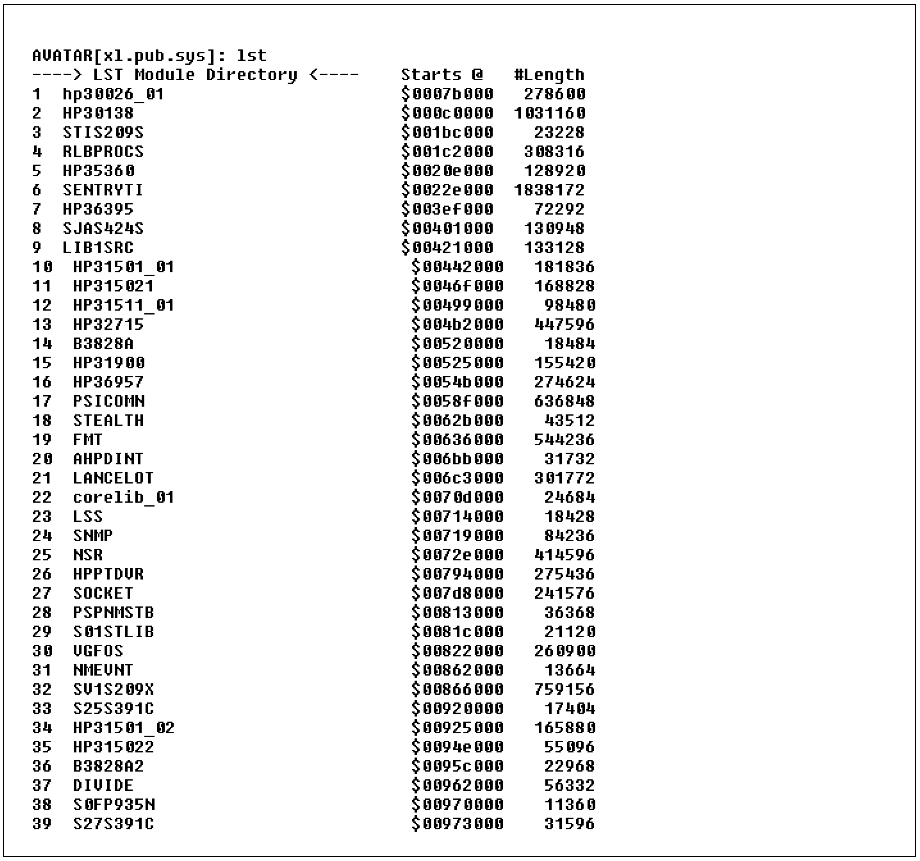
Example: Only SOMs with multiple entries require the use of the LST command.



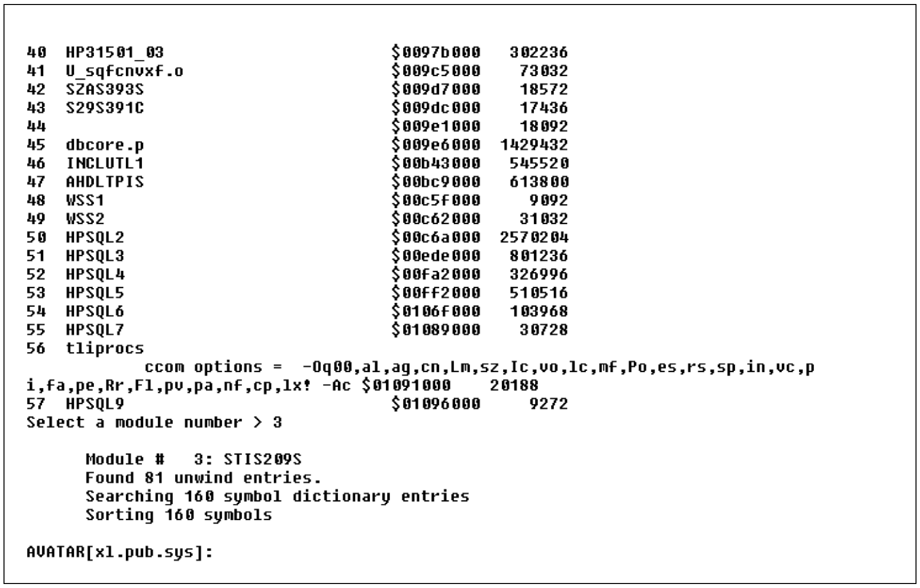
```
Wolf:/LPSTOOLS/PUB: run avatar

AVATAR [2.9] - LPS Toolbox [A.09f]          (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?
This product is licensed to: ImageStats Demo

XL.PUB.SYS @ $101.$0

AVATAR: open xl.pub.sys
Assuming space $00000101 for XL.PUB.SYS

FILE TYPE: executable SOM library, for PA-RISC 1.0

----> LST Module Directory <----    Starts @   #Length
1  hp30026_01                       $0007b000   278600
2  HP30138                          $000c0000  1031160
3  STIS209S                         $001bc000    23228
4  RLBPROCS                         $001c2000   308316
5  HP35360                          $0020e000   128920
6  SENTRYTI                         $0022e000  1838172
7  HP36395                          $003ef000    72292
8  SJAS424S                         $00401000   130948
9  LIB1SRC                          $00421000   133128
10  HP31501_01                       $00442000   181836
11  HP315021                         $0046f000   168828
12  HP31511_01                       $00499000    98480
13  HP32715                          $004b2000   447596
14  B3828A                           $00520000    18484
15  HP31900                          $00525000   155420
16  HP36957                          $0054b000   274624
17  PSICOMN                          $0058f000   636848
18  STEALTH                          $0062b000    43512
19  FMT                              $00636000   544236
20  AHPDINT                          $006bb000    31732
21  LANCELOT                         $006c3000   301772
22  corelib_01                       $0070d000    24684
23  LSS                              $00714000    18428
24  SNMP                             $00719000    84236
25  NSR                              $0072e000   414596
26  HPPTDVR                          $00794000   275436
```



```
27  SOCKET                           $007d8000   241576
28  PSPNMSTB                         $00813000    36368
29  S01STLIB                         $0081c000    21120
30  VGFOS                            $00822000   260900
31  NMEVNT                           $00862000    13664
32  SV1S209X                         $00866000   759156
33  S25S391C                         $00920000    17404
34  HP31501_02                       $00925000   165880
35  HP315022                         $0094e000    55096
36  B3828A2                          $0095c000    22968
37  DIVIDE                           $00962000    56332
38  S0FP935N                         $00970000    11360
39  S27S391C                         $00973000    31596
40  HP31501_03                       $0097b000   302236
41  U_sqfcnvxf.o                     $009c5000    73032
42  SZAS393S                         $009d7000    18572
43  S29S391C                         $009dc000    17436
44                                   $009e1000    18092
45  dbcore.p                         $009e6000  1429432
46  INCLUTL1                         $00b43000   545520
47  AHDLTPIS                         $00bc9000   613800
48  WSS1                             $00c5f000     9092
49  WSS2                             $00c62000    31032
50  HPSQL2                           $00c6a000  2570204
51  HPSQL3                           $00ede000   801236
52  HPSQL4                           $00fa2000   326996
53  HPSQL5                           $00ff2000   510516
54  HPSQL6                           $0106f000   103968
55  HPSQL7                           $01089000    30728
56  tliprocs
            ccom options =  -Oq00,al,ag,cn,Lm,sz,Ic,vo,lc,mf,Po,es,rs,sp,in,vc,p
i,fa,pe,Rr,Fl,pv,pa,nf,cp,lx! -Ac $01091000    20188
57  HPSQL9                           $01096000     9272
Select a module number > 1
Tool_REDO: dmovin#4 failed

      Module #   1: hp30026_01
      Found 408 unwind entries.
      Searching 2,979 symbol dictionary entries
      Sorting 2,979 symbols
```



```
AVATAR[xl.pub.sys]: lst
----> LST Module Directory <----     Starts @   #Length
1  hp30026_01                        $0007b000   278600
2  HP30138                           $000c0000  1031160
3  STIS209S                          $001bc000    23228
4  RLBPROCS                          $001c2000   308316
5  HP35360                           $0020e000   128920
6  SENTRYTI                          $0022e000  1838172
7  HP36395                           $003ef000    72292
8  SJAS424S                          $00401000   130948
9  LIB1SRC                           $00421000   133128
10  HP31501_01                        $00442000   181836
11  HP315021                          $0046f000   168828
12  HP31511_01                        $00499000    98480
13  HP32715                           $004b2000   447596
14  B3828A                            $00520000    18484
15  HP31900                           $00525000   155420
16  HP36957                           $0054b000   274624
17  PSICOMN                           $0058f000   636848
18  STEALTH                           $0062b000    43512
19  FMT                               $00636000   544236
20  AHPDINT                           $006bb000    31732
21  LANCELOT                          $006c3000   301772
22  corelib_01                        $0070d000    24684
23  LSS                               $00714000    18428
24  SNMP                              $00719000    84236
25  NSR                               $0072e000   414596
26  HPPTDVR                           $00794000   275436
27  SOCKET                            $007d8000   241576
28  PSPNMSTB                          $00813000    36368
29  S01STLIB                          $0081c000    21120
30  VGFOS                             $00822000   260900
31  NMEVNT                            $00862000    13664
32  SV1S209X                          $00866000   759156
33  S25S391C                          $00920000    17404
34  HP31501_02                        $00925000   165880
35  HP315022                          $0094e000    55096
36  B3828A2                           $0095c000    22968
37  DIVIDE                            $00962000    56332
38  S0FP935N                          $00970000    11360
39  S27S391C                          $00973000    31596
```



```
40  HP31501_03                       $0097b000   302236
41  U_sqfcnvxf.o                     $009c5000    73032
42  SZAS393S                         $009d7000    18572
43  S29S391C                         $009dc000    17436
44                                   $009e1000    18092
45  dbcore.p                         $009e6000  1429432
46  INCLUTL1                         $00b43000   545520
47  AHDLTPIS                         $00bc9000   613800
48  WSS1                             $00c5f000     9092
49  WSS2                             $00c62000    31032
50  HPSQL2                           $00c6a000  2570204
51  HPSQL3                           $00ede000   801236
52  HPSQL4                           $00fa2000   326996
53  HPSQL5                           $00ff2000   510516
54  HPSQL6                           $0106f000   103968
55  HPSQL7                           $01089000    30728
56  tliprocs
            ccom options =  -Oq00,al,ag,cn,Lm,sz,Ic,vo,lc,mf,Po,es,rs,sp,in,vc,p
i,fa,pe,Rr,Fl,pv,pa,nf,cp,lx! -Ac $01091000    20188
57  HPSQL9                           $01096000     9272
Select a module number > 3

      Module #   3: STIS209S
      Found 81 unwind entries.
      Searching 160 symbol dictionary entries
      Sorting 160 symbols

AVATAR[xl.pub.sys]:
```

Figure 3.9 *LST Command*

MC

This command has the following syntax:

```
MC <expression> <value>
```

The MC command will modify data at a given code address. The address can be specified as any valid expression for the current SOM. The new value can be any valid value for the code space, including assembler instructions.

<expression> Is an arithmetic expression that represents an offset to the start of the current module.

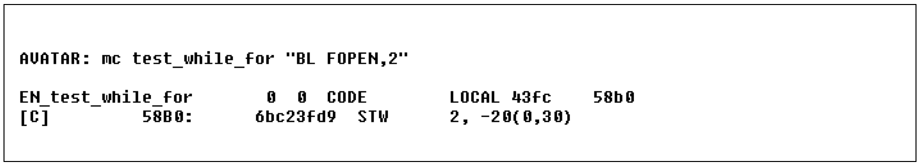
<value> Is an expression representing the new data.

Example 1: `MC my_proc+$100 `LDW -4(0,30),2``

Example 2: `MC my_proc + $200 `BL FOPEN,2``

Example 3: `MC $1200 8004000`

Example 4: `mc test_while_for "BL FOPEN,2"` (See Figure 3.10)



```
AVATAR: mc test_while_for "BL FOPEN,2"

EN_test_while_for       0  0  CODE        LOCAL 43fc    58b0
[C]         58B0:      6bc23fd9  STW      2, -20(0,30)
```

Figure 3.10 *MC Command*

MD

This command has the following syntax:

```
MD <expression> <value>
```

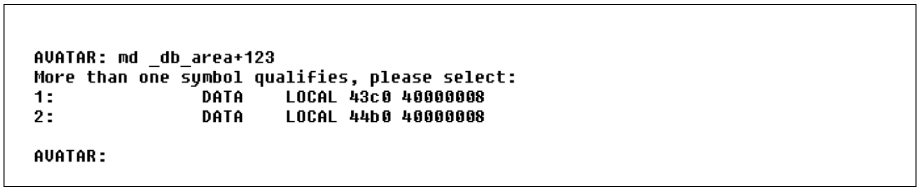
The MD command will modify values from an initialized block within the current module of the opened SOM file.

`<expression>` Is an arithmetic expression that represents an offset into the initialized data area in the current SOM.

`<value>` Is an expression representing the new data.

Example 1: `MD db_area+#100 $5078`

Example 2: `md _db_area+123` (See Figure 3.11)



```
AVATAR: md _db_area+123
More than one symbol qualifies, please select:
1:               DATA    LOCAL 43c0 40000008
2:               DATA    LOCAL 44b0 40000008

AVATAR:
```

Figure 3.11 *MD Command*

MV

This command has the following syntax:

```
MV <expression> <value>
```

The MV command will modify values (data) within the current SOM file.

`<expression>` Is an arithmetic expression that represents an offset to the start of the current file.

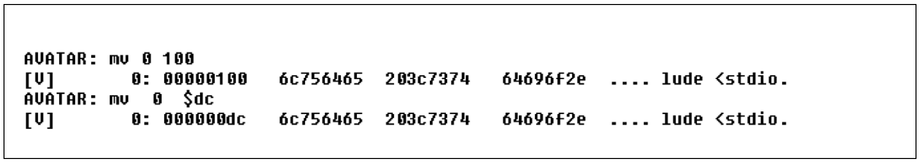
`<value>` Is an expression representing the new data.

Example 1: `MV $100 $5078`

Change the data at address hex 100 to hex 5078.

Example 2: `mv 0 100` (See Figure 3.12)

Example 3: `mv 0 $dc` (See Figure 3.12)



```
AVATAR: mv 0 100
[V]        0: 00000100   6c756465  203c7374   64696f2e  .... lude <stdio.
AVATAR: mv  0  $dc
[V]        0: 000000dc   6c756465  203c7374   64696f2e  .... lude <stdio.
```

Figure 3.12 *MV Command*

Next

The NEXT command repeats the last DV, DC or DD command starting from the point where it left off.

NOTE MC, MV or MD command will reposition the pointer as well. The NEXT command is optional, a mere <cr> will do.

Open

This command has the following syntax:

```
OPEN filename [READ]
```

The OPEN command opens an NMPRG or an NMXL.

filename must be a valid MPE/iX file descriptor (no wildcards). If the specified file is a Native Mode executable file (SOM) and it contains more than one LST entry, then the user is asked to select which LST entry to analyze. If the specified file is nor an SOM, then the file is mapped into virtual memory where the user is restricted to basic operations, such as displaying and modifying memory. If READ is specified, then the SOM file is opened for Read-Access only. The MC, MV and MD commands are then disabled. The default is read/write access.

Example 1: `OPEN myprog`

Example 2: `OPEN myprog READ`

Quit

The QUIT command exits the program.

Radix

This command has the following syntax:

```
Radix <mnemonic>
```

The RADIX command will change the default radix, which is the base that all input is assumed to be.

```
<mnemonic>=  [DECIMAL]
             [HEX]
             [OCTAL]
```

Example 1: `RADIX HEX`

Example 2: `RADIX DECIMAL`

Example 3: `RADIX OCTAL`

Search

This command has the following syntax:

```
SEARCH <expression>
       <string>
```

The SEARCH command will search the entire opened SOM file an look for a given value. If the value is found, a line of data is displayed.

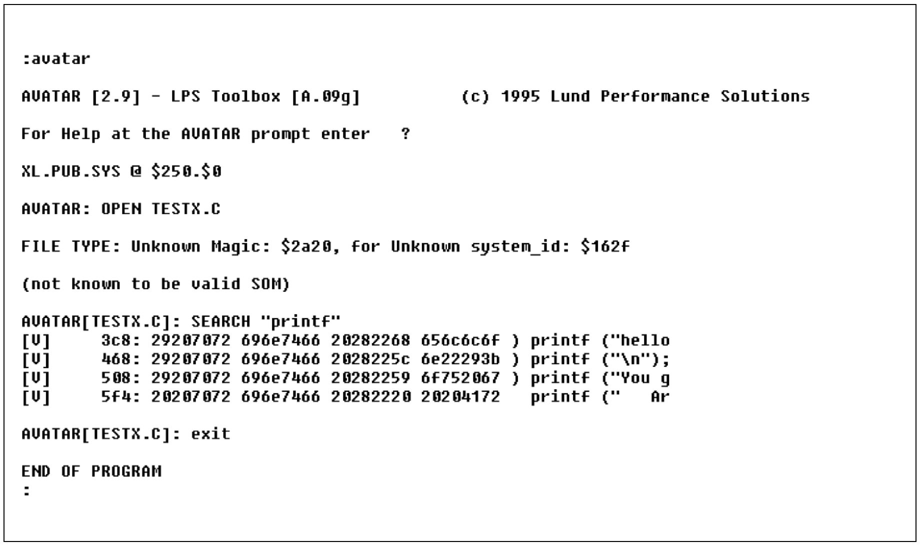
<expression>	An expression that result in an integer value to be searched in the file.
<string>	A quoted string to be searched in the file.

Example 1: `SEARCH $8861`

Example 2: `SEARCH "A string"`

Example 3: `` SEARCH `LDIL $1000,0` ``

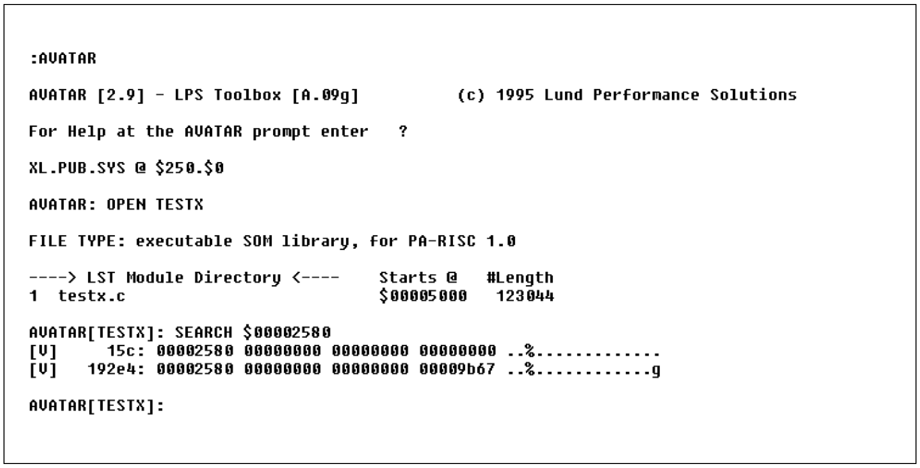
Searching an ASCII file:



```
:avatar

AVATAR [2.9] - LPS Toolbox [A.09g]          (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?

XL.PUB.SYS @ $250.$0

AVATAR: OPEN TESTX.C

FILE TYPE: Unknown Magic: $2a20, for Unknown system_id: $162f

(not known to be valid SOM)

AVATAR[TESTX.C]: SEARCH "printf"
[V]      3c8: 29207072 696e7466 20282268 656c6c6f ) printf ("hello
[V]      468: 29207072 696e7466 2028225c 6e22293b ) printf ("\n");
[V]      508: 29207072 696e7466 20282259 6f752067 ) printf ("You g
[V]      5f4: 20207072 696e7466 20282220 20204172   printf ("   Ar

AVATAR[TESTX.C]: exit

END OF PROGRAM
:
```

Figure 3.13 *SEARCH Command*

Searching a Native Mode program file:

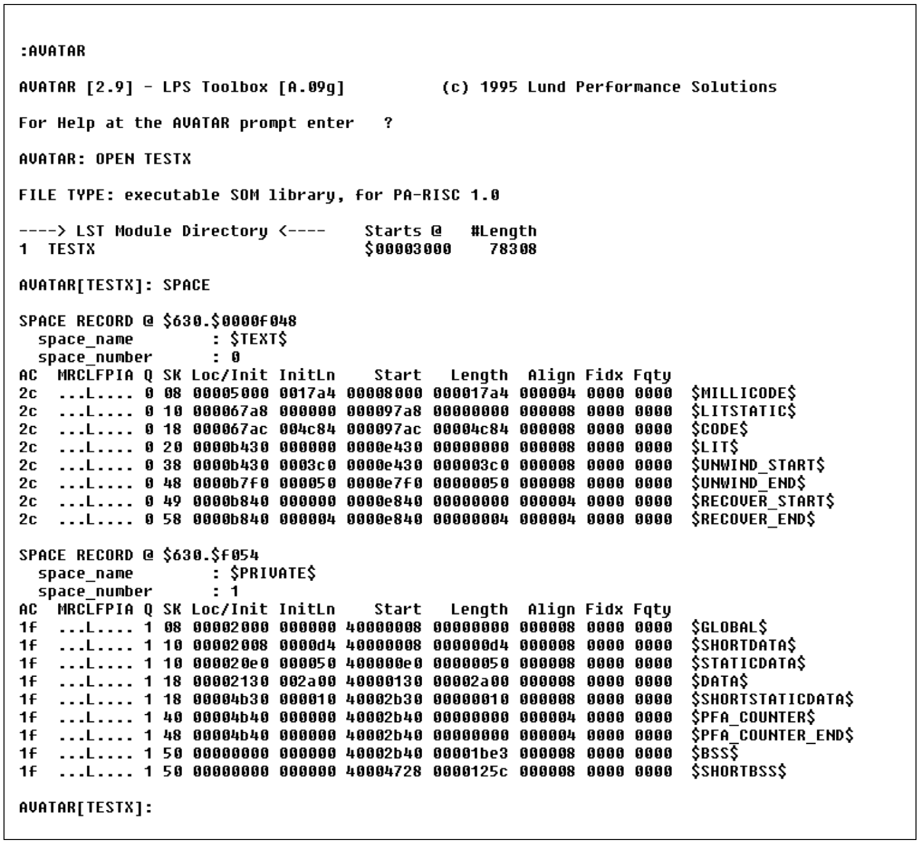


```
:AVATAR

AVATAR [2.9] - LPS Toolbox [A.09g]          (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?

XL.PUB.SYS @ $250.$0

AVATAR: OPEN TESTX

FILE TYPE: executable SOM library, for PA-RISC 1.0

----> LST Module Directory <----    Starts @   #Length
1  testx.c                          $00005000   123044

AVATAR[TESTX]: SEARCH $00002580
[V]     15c: 00002580 00000000 00000000 00000000 ..%.............
[V]   192e4: 00002580 00000000 00000000 00009b67 ..%............g

AVATAR[TESTX]:
```

Figure 3.14 *SEARCH a Native Mode Program*

SPace

The SPACE command will display the space and subspace information about the current module of the opened SOM file.

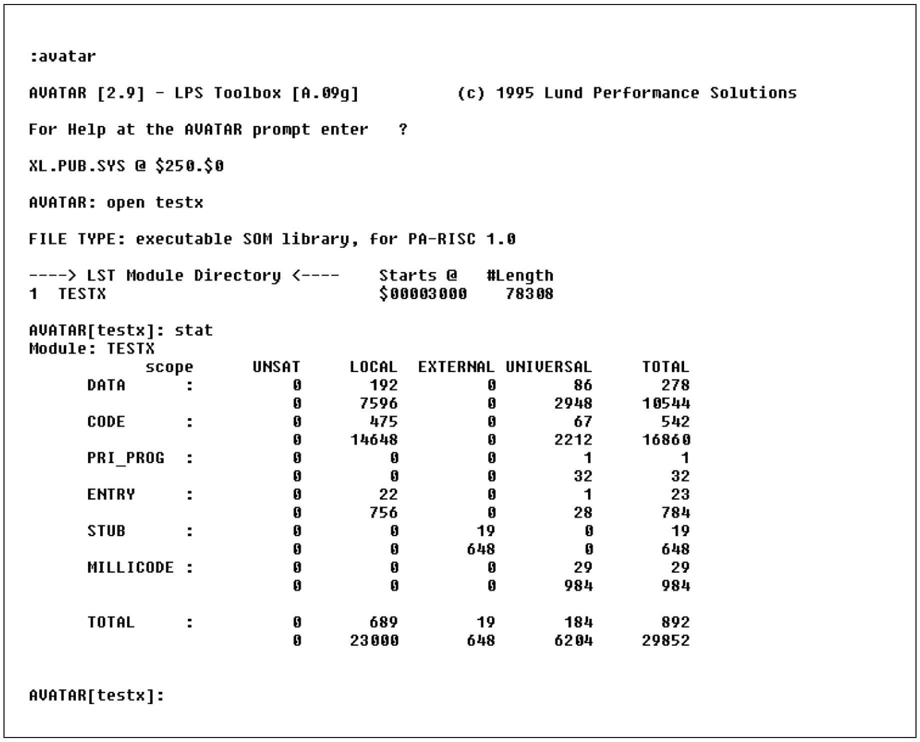


```
:AVATAR

AVATAR [2.9] - LPS Toolbox [A.09g]          (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?

AVATAR: OPEN TESTX

FILE TYPE: executable SOM library, for PA-RISC 1.0

----> LST Module Directory <----    Starts @   #Length
1  TESTX                           $00003000    78308

AVATAR[TESTX]: SPACE

SPACE RECORD @ $630.$0000f048
  space_name         : $TEXT$
  space_number       : 0
AC  MRCLFPIA Q SK Loc/Init InitLn    Start   Length  Align Fidx Fqty
2c  ...L.... 0 08 00005000 0017a4 00008000 000017a4 000004 0000 0000  $MILLICODE$
2c  ...L.... 0 10 000067a8 000000 000097a8 00000000 000008 0000 0000  $LITSTATIC$
2c  ...L.... 0 18 000067ac 004c84 000097ac 00004c84 000008 0000 0000  $CODE$
2c  ...L.... 0 20 0000b430 000000 0000e430 00000000 000008 0000 0000  $LIT$
2c  ...L.... 0 38 0000b430 0003c0 0000e430 000003c0 000008 0000 0000  $UNWIND_START$
2c  ...L.... 0 48 0000b7f0 000050 0000e7f0 00000050 000008 0000 0000  $UNWIND_END$
2c  ...L.... 0 49 0000b840 000000 0000e840 00000000 000004 0000 0000  $RECOVER_START$
2c  ...L.... 0 58 0000b840 000004 0000e840 00000004 000004 0000 0000  $RECOVER_END$

SPACE RECORD @ $630.$f054
  space_name         : $PRIVATE$
  space_number       : 1
AC  MRCLFPIA Q SK Loc/Init InitLn    Start   Length  Align Fidx Fqty
1f  ...L.... 1 08 00002000 000000 40000008 00000000 000008 0000 0000  $GLOBAL$
1f  ...L.... 1 10 00002008 0000d4 40000008 000000d4 000008 0000 0000  $SHORTDATA$
1f  ...L.... 1 10 000020e0 000050 400000e0 00000050 000008 0000 0000  $STATICDATA$
1f  ...L.... 1 18 00002130 002a00 40000130 00002a00 000008 0000 0000  $DATA$
1f  ...L.... 1 18 00004b30 000010 40002b30 00000010 000008 0000 0000  $SHORTSTATICDATA$
1f  ...L.... 1 40 00004b40 000000 40002b40 00000000 000004 0000 0000  $PFA_COUNTER$
1f  ...L.... 1 48 00004b40 000000 40002b40 00000000 000004 0000 0000  $PFA_COUNTER_END$
1f  ...L.... 1 50 00000000 000000 40002b40 00001be3 000008 0000 0000  $BSS$
1f  ...L.... 1 50 00000000 000000 40004728 0000125c 000008 0000 0000  $SHORTBSS$

AVATAR[TESTX]:
```

Figure 3.15 *SPACE Command*

STatistics

The STATISTICS command displays SOM file statistics.

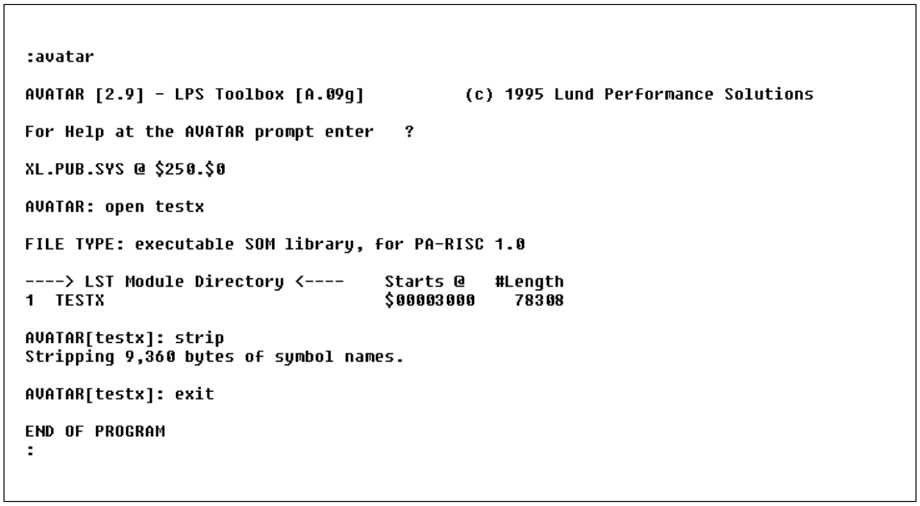


```
:avatar

AVATAR [2.9] - LPS Toolbox [A.09g]          (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?

XL.PUB.SYS @ $250.$0

AVATAR: open testx

FILE TYPE: executable SOM library, for PA-RISC 1.0

----> LST Module Directory <----    Starts @   #Length
1  TESTX                            $00003000    78308

AVATAR[testx]: stat
Module: TESTX
            scope      UNSAT     LOCAL  EXTERNAL UNIVERSAL     TOTAL
      DATA      :          0       192         0        86       278
                           0      7596         0      2948     10544
      CODE      :          0       475         0        67       542
                           0     14648         0      2212     16860
      PRI_PROG  :          0         0         0         1         1
                           0         0         0        32        32
      ENTRY     :          0        22         0         1        23
                           0       756         0        28       784
      STUB      :          0         0        19         0        19
                           0         0       648         0       648
      MILLICODE :          0         0         0        29        29
                           0         0         0       984       984

      TOTAL     :          0       689        19       184       892
                           0     23000       648      6204     29852


AVATAR[testx]:
```

Figure 3.16 *STATISTICS Command*

STRIP

The STRIP command will remove symbolic information from SOM.

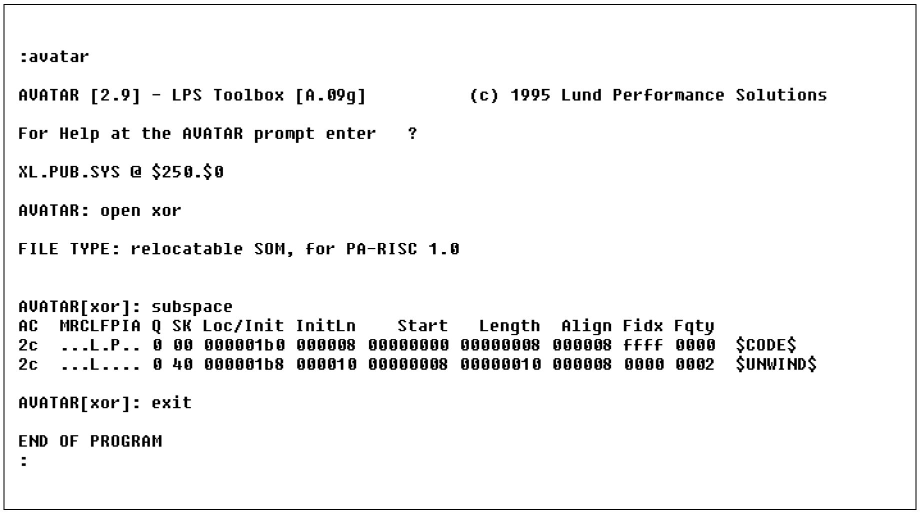


```
:avatar

AVATAR [2.9] - LPS Toolbox [A.09g]          (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?

XL.PUB.SYS @ $250.$0

AVATAR: open testx

FILE TYPE: executable SOM library, for PA-RISC 1.0

----> LST Module Directory <----    Starts @   #Length
1  TESTX                            $00003000    78308

AVATAR[testx]: strip
Stripping 9,360 bytes of symbol names.

AVATAR[testx]: exit

END OF PROGRAM
:
```

Figure 3.17 *STRIP Command*

SUbspace

The SUBSPACE command will display the subspace information about the current module of the opened SOM file.



```
:avatar

AVATAR [2.9] - LPS Toolbox [A.09g]          (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?

XL.PUB.SYS @ $250.$0

AVATAR: open xor

FILE TYPE: relocatable SOM, for PA-RISC 1.0


AVATAR[xor]: subspace
AC  MRCLFPIA Q SK Loc/Init InitLn    Start   Length  Align Fidx Fqty
2c  ...L.P.. 0 00 000001b0 000008 00000000 00000008 000008 ffff 0000  $CODE$
2c  ...L.... 0 40 000001b8 000010 00000008 00000010 000008 0000 0002  $UNWIND$

AVATAR[xor]: exit

END OF PROGRAM
:
```

Figure 3.18 *SUBSPACE Command*

SYMformat

This command has the following syntax:

```
SYMformat <HEADER | GNTT | LNTT | SLT | VT>
```

The SYMFORMAT command is used to format and display various portions of a SYMOS file. Using this command requires an expert-level knowledge of the MPE/iX operating system.

HEADER is used to format and display header information.

GNTT not implemented

LNTT is used to format and display LNTT information.

SLT is used to format and display SLT information.

VT is used to format and display VT information.

SYMOpen

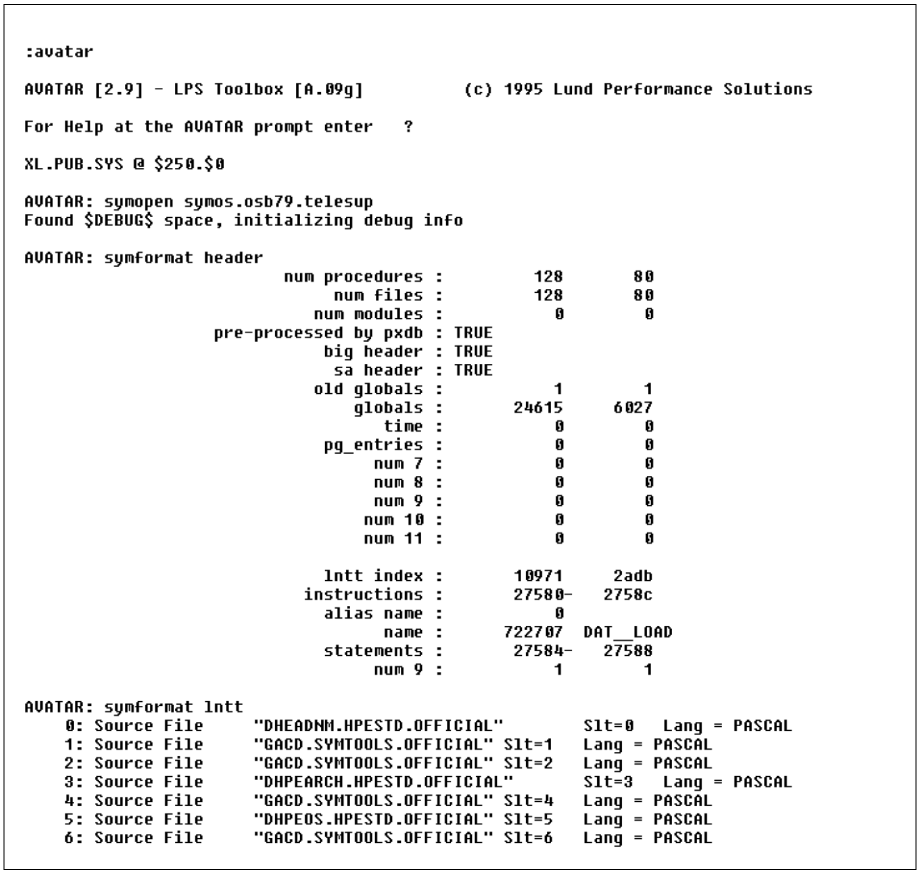
This command has the following syntax:

```
SYMOpen <symos filename>
```

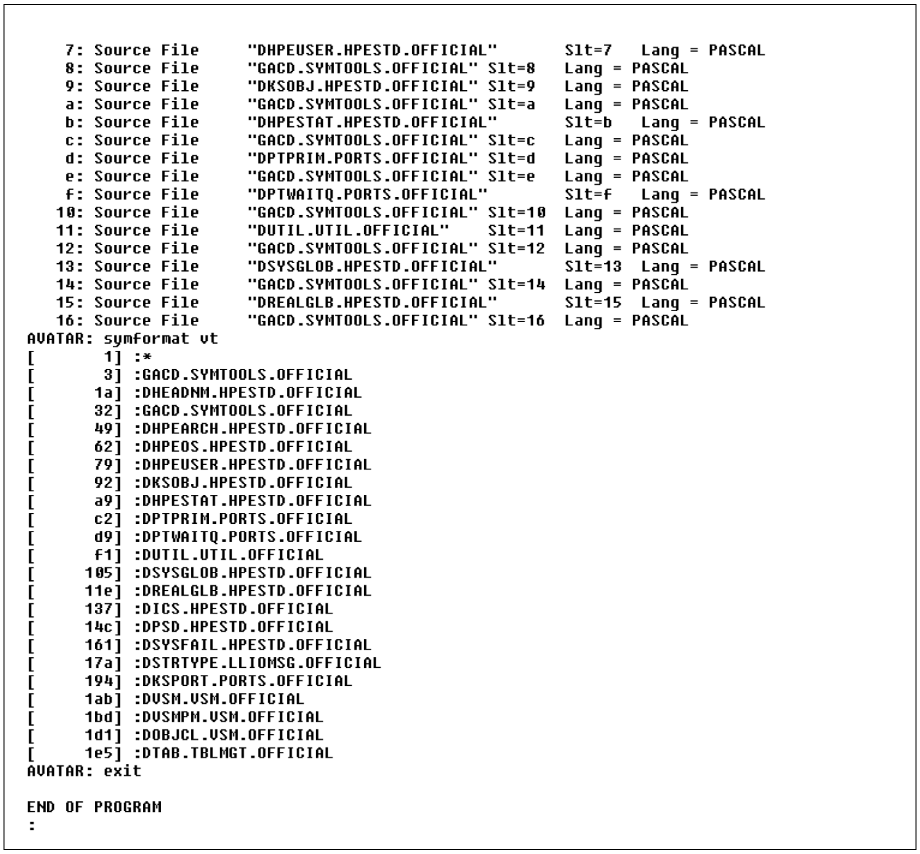
The SYMOPEN command is used to open a SYMOS file for examination. Using this command requires an expert-level knowledge of the MPE/iX operating system.

SYM-based Examples

These Examples illustrate usage for SYMOPEN, SYMFORMAT HEADER, SYMFORMAT VT, and SYMFORMAT LNTT.



```
:avatar

AVATAR [2.9] - LPS Toolbox [A.09g]          (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?

XL.PUB.SYS @ $250.$0

AVATAR: symopen symos.osb79.telesup
Found $DEBUG$ space, initializing debug info

AVATAR: symformat header
                          num procedures :          128       80
                               num files :          128       80
                             num modules :            0        0
                   pre-processed by pxdb : TRUE
                              big header : TRUE
                               sa header : TRUE
                             old globals :            1        1
                                 globals :        24615     6027
                                    time :            0        0
                              pg_entries :            0        0
                                   num 7 :            0        0
                                   num 8 :            0        0
                                   num 9 :            0        0
                                  num 10 :            0        0
                                  num 11 :            0        0

                              lntt index :        10971     2adb
                            instructions :        27580-   2758c
                              alias name :            0
                                    name :       722707  DAT__LOAD
                              statements :        27584-   27588
                                   num 9 :            1        1

AVATAR: symformat lntt
    0: Source File     "DHEADNM.HPESTD.OFFICIAL"        Slt=0   Lang = PASCAL
    1: Source File     "GACD.SYMTOOLS.OFFICIAL" Slt=1   Lang = PASCAL
    2: Source File     "GACD.SYMTOOLS.OFFICIAL" Slt=2   Lang = PASCAL
    3: Source File     "DHPEARCH.HPESTD.OFFICIAL"       Slt=3   Lang = PASCAL
    4: Source File     "GACD.SYMTOOLS.OFFICIAL" Slt=4   Lang = PASCAL
    5: Source File     "DHPEOS.HPESTD.OFFICIAL" Slt=5   Lang = PASCAL
    6: Source File     "GACD.SYMTOOLS.OFFICIAL" Slt=6   Lang = PASCAL
```



```
    7: Source File     "DHPEUSER.HPESTD.OFFICIAL"       Slt=7   Lang = PASCAL
    8: Source File     "GACD.SYMTOOLS.OFFICIAL" Slt=8   Lang = PASCAL
    9: Source File     "DKSOBJ.HPESTD.OFFICIAL" Slt=9   Lang = PASCAL
    a: Source File     "GACD.SYMTOOLS.OFFICIAL" Slt=a   Lang = PASCAL
    b: Source File     "DHPESTAT.HPESTD.OFFICIAL"       Slt=b   Lang = PASCAL
    c: Source File     "GACD.SYMTOOLS.OFFICIAL" Slt=c   Lang = PASCAL
    d: Source File     "DPTPRIM.PORTS.OFFICIAL" Slt=d   Lang = PASCAL
    e: Source File     "GACD.SYMTOOLS.OFFICIAL" Slt=e   Lang = PASCAL
    f: Source File     "DPTWAITQ.PORTS.OFFICIAL"        Slt=f   Lang = PASCAL
   10: Source File     "GACD.SYMTOOLS.OFFICIAL" Slt=10  Lang = PASCAL
   11: Source File     "DUTIL.UTIL.OFFICIAL"    Slt=11  Lang = PASCAL
   12: Source File     "GACD.SYMTOOLS.OFFICIAL" Slt=12  Lang = PASCAL
   13: Source File     "DSYSGLOB.HPESTD.OFFICIAL"       Slt=13  Lang = PASCAL
   14: Source File     "GACD.SYMTOOLS.OFFICIAL" Slt=14  Lang = PASCAL
   15: Source File     "DREALGLB.HPESTD.OFFICIAL"       Slt=15  Lang = PASCAL
   16: Source File     "GACD.SYMTOOLS.OFFICIAL" Slt=16  Lang = PASCAL
AVATAR: symformat vt
[        1] :*
[        3] :GACD.SYMTOOLS.OFFICIAL
[       1a] :DHEADNM.HPESTD.OFFICIAL
[       32] :GACD.SYMTOOLS.OFFICIAL
[       49] :DHPEARCH.HPESTD.OFFICIAL
[       62] :DHPEOS.HPESTD.OFFICIAL
[       79] :DHPEUSER.HPESTD.OFFICIAL
[       92] :DKSOBJ.HPESTD.OFFICIAL
[       a9] :DHPESTAT.HPESTD.OFFICIAL
[       c2] :DPTPRIM.PORTS.OFFICIAL
[       d9] :DPTWAITQ.PORTS.OFFICIAL
[       f1] :DUTIL.UTIL.OFFICIAL
[      105] :DSYSGLOB.HPESTD.OFFICIAL
[      11e] :DREALGLB.HPESTD.OFFICIAL
[      137] :DICS.HPESTD.OFFICIAL
[      14c] :DPSD.HPESTD.OFFICIAL
[      161] :DSYSFAIL.HPESTD.OFFICIAL
[      17a] :DSTRTYPE.LLIOMSG.OFFICIAL
[      194] :DKSPORT.PORTS.OFFICIAL
[      1ab] :DVSM.VSM.OFFICIAL
[      1bd] :DVSMPM.VSM.OFFICIAL
[      1d1] :DOBJCL.VSM.OFFICIAL
[      1e5] :DTAB.TBLMGT.OFFICIAL
AVATAR: exit

END OF PROGRAM
:
```

Figure 3.19 *SYMOPEN/SYMFORMAT: LNTT, VT*

SYn

This command has the following syntax:

```
SYn SPLASH
    SYSTEM
    NONE
    R
    REG <general register number> <synonym>
    CR <control register number> <synonym>
```

The SYN command sets up synonyms for the general registers and the special registers. These synonyms will be shown in the disassembled format of any instruction.

SPLASH	Sets synonyms to reflect the normal SPLash! usage for synonyms.
SYSTEM	Sets synonyms to reflect the normal system usage for registers.
NONE	Resets all synonyms.
R	Precedes every register number by the letter 'r' in generated assembly language.
<general register number>	Is a number between 0 and 31 inclusive, designating the general register for which a synonym is set up.
<special register number>	Is a number between 0 and 31 inclusive, designating the special register for which a synonym is set up.
<synonym>	Is any name, to be used as a synonym.

UNCALLED

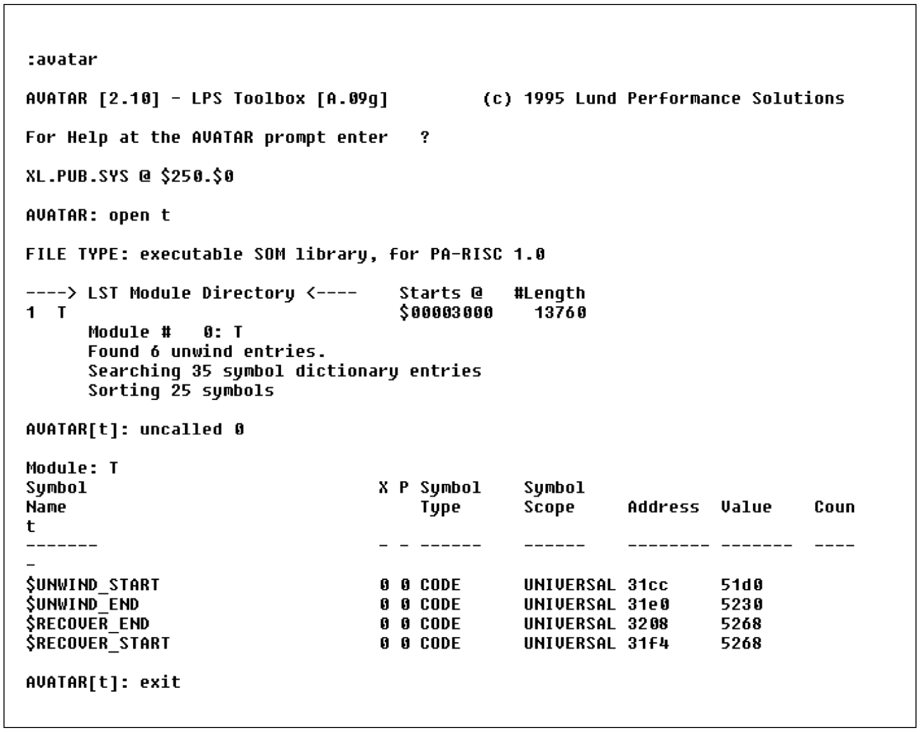
This command has the following syntax:

```
UNCALLED [<expression>]
```

The UNCALLED command lists all code entry points that are called less than a specified number of times from within the same SOM.

<expression>	The maximum number of times that an entry point may be called to be listed. Default is 0.

Example: `UNCALLED 0`

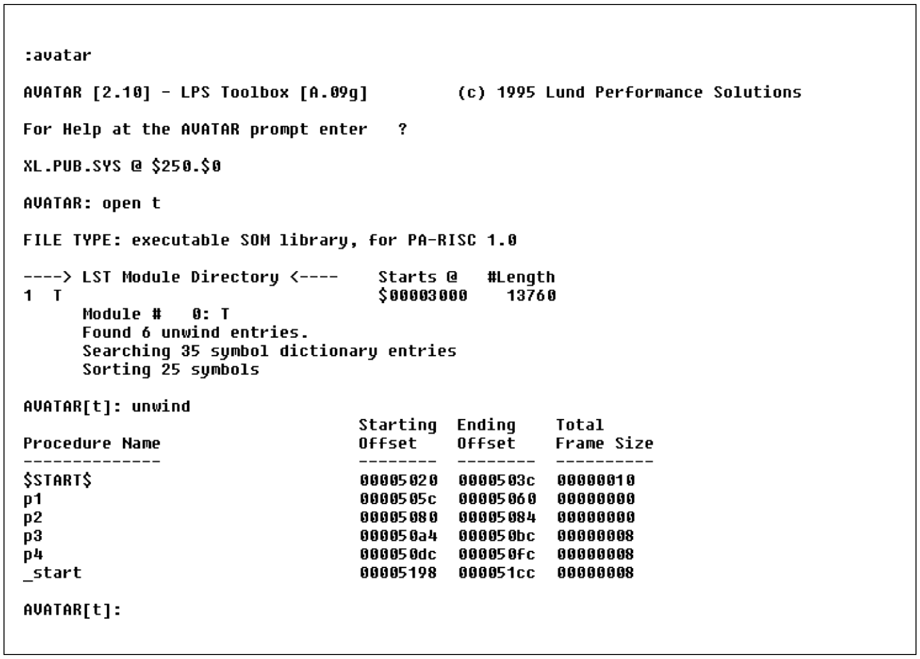


```
:avatar

AVATAR [2.10] - LPS Toolbox [A.09g]         (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?

XL.PUB.SYS @ $250.$0

AVATAR: open t

FILE TYPE: executable SOM library, for PA-RISC 1.0

----> LST Module Directory <----    Starts @   #Length
1  T                                $00003000    13760
      Module #   0: T
      Found 6 unwind entries.
      Searching 35 symbol dictionary entries
      Sorting 25 symbols

AVATAR[t]: uncalled 0

Module: T
Symbol                            X P Symbol    Symbol
Name                                  Type      Scope     Address  Value    Coun
t
-------                           - - ------    ------    -------- -------  ----
-
$UNWIND_START                     0 0 CODE      UNIVERSAL 31cc     51d0
$UNWIND_END                       0 0 CODE      UNIVERSAL 31e0     5230
$RECOVER_END                      0 0 CODE      UNIVERSAL 3208     5268
$RECOVER_START                    0 0 CODE      UNIVERSAL 31f4     5268

AVATAR[t]: exit
```

Figure 3.20 *UNCALLED Command*

UNWind

The UNWIND commands produces a formatted listing of the unwind descriptor entries from the current SOM.



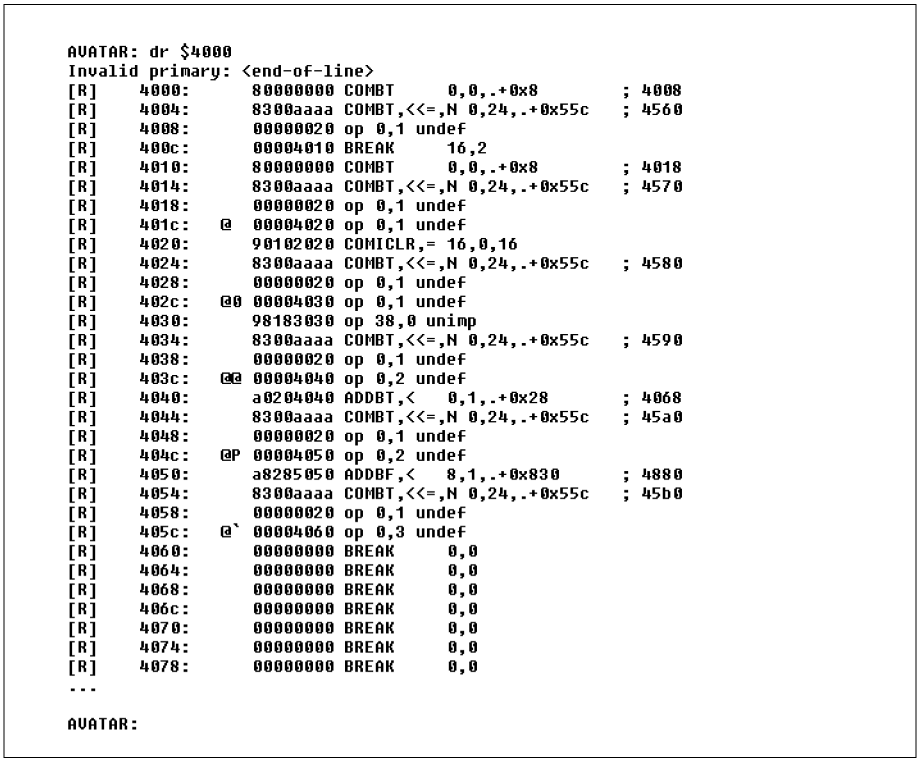
```
:avatar

AVATAR [2.10] - LPS Toolbox [A.09g]         (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?

XL.PUB.SYS @ $250.$0

AVATAR: open t

FILE TYPE: executable SOM library, for PA-RISC 1.0

----> LST Module Directory <----    Starts @   #Length
1  T                                $00003000    13760
      Module #   0: T
      Found 6 unwind entries.
      Searching 35 symbol dictionary entries
      Sorting 25 symbols

AVATAR[t]: unwind
                                  Starting  Ending    Total
Procedure Name                    Offset    Offset    Frame Size
--------------                    --------  --------  ----------
$START$                           00005020  0000503c  00000010
p1                                0000505c  00005060  00000000
p2                                00005080  00005084  00000000
p3                                000050a4  000050bc  00000008
p4                                000050dc  000050fc  00000008
_start                            00005198  000051cc  00000008

AVATAR[t]:
```

Figure 3.21 *UNWIND Command*

AVATAR Examples

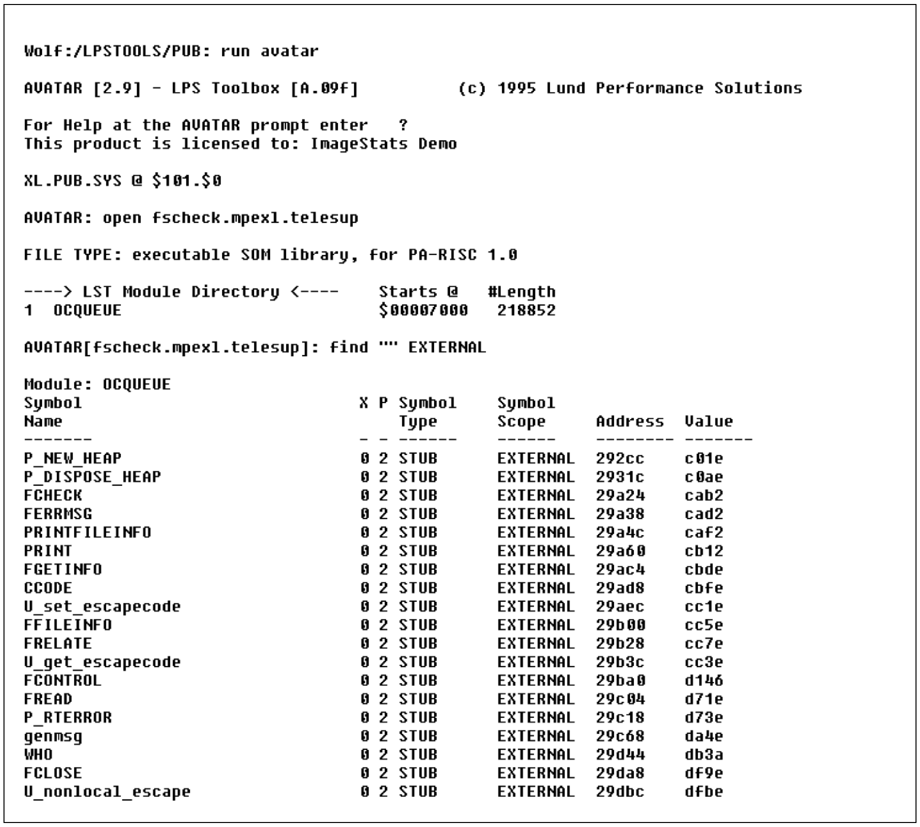
Figure 3.22 uses the DR command.

Figure 3.23 uses the FIND command to locate external procedure calls. Remember, most symbol parameters are case-sensitive.



```
AVATAR: dr $4000
Invalid primary: <end-of-line>
[R]    4000:      80000000 COMBT      0,0,.+0x8        ; 4008
[R]    4004:      8300aaaa COMBT,<<=,N 0,24,.+0x55c    ; 4560
[R]    4008:      00000020 op 0,1 undef
[R]    400c:      00004010 BREAK      16,2
[R]    4010:      80000000 COMBT      0,0,.+0x8        ; 4018
[R]    4014:      8300aaaa COMBT,<<=,N 0,24,.+0x55c    ; 4570
[R]    4018:      00000020 op 0,1 undef
[R]    401c:   @  00004020 op 0,1 undef
[R]    4020:      90102020 COMICLR,= 16,0,16
[R]    4024:      8300aaaa COMBT,<<=,N 0,24,.+0x55c    ; 4580
[R]    4028:      00000020 op 0,1 undef
[R]    402c:   @0 00004030 op 0,1 undef
[R]    4030:      98183030 op 38,0 unimp
[R]    4034:      8300aaaa COMBT,<<=,N 0,24,.+0x55c    ; 4590
[R]    4038:      00000020 op 0,1 undef
[R]    403c:   @@ 00004040 op 0,2 undef
[R]    4040:      a0204040 ADDBT,<    0,1,.+0x28       ; 4068
[R]    4044:      8300aaaa COMBT,<<=,N 0,24,.+0x55c    ; 45a0
[R]    4048:      00000020 op 0,1 undef
[R]    404c:   @P 00004050 op 0,2 undef
[R]    4050:      a8285050 ADDBF,<    8,1,.+0x830      ; 4880
[R]    4054:      8300aaaa COMBT,<<=,N 0,24,.+0x55c    ; 45b0
[R]    4058:      00000020 op 0,1 undef
[R]    405c:   @` 00004060 op 0,3 undef
[R]    4060:      00000000 BREAK      0,0
[R]    4064:      00000000 BREAK      0,0
[R]    4068:      00000000 BREAK      0,0
[R]    406c:      00000000 BREAK      0,0
[R]    4070:      00000000 BREAK      0,0
[R]    4074:      00000000 BREAK      0,0
[R]    4078:      00000000 BREAK      0,0
...

AVATAR:
```

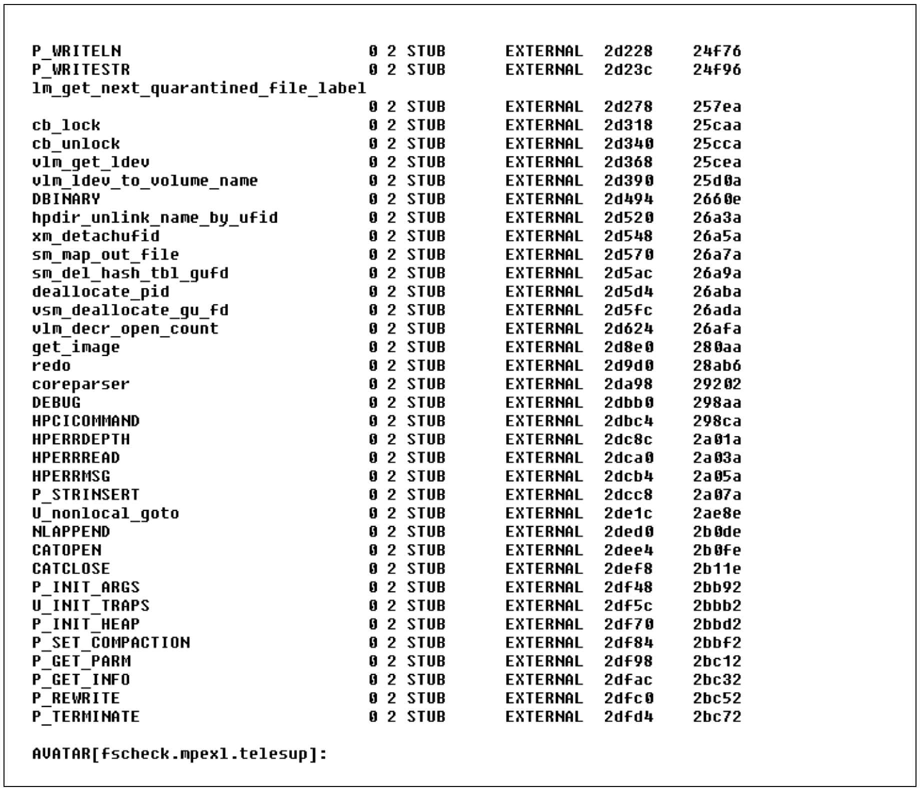
Figure 3.22 *DR Command*



```
Wolf:/LPSTOOLS/PUB: run avatar

AVATAR [2.9] - LPS Toolbox [A.09f]          (c) 1995 Lund Performance Solutions

For Help at the AVATAR prompt enter   ?
This product is licensed to: ImageStats Demo

XL.PUB.SYS @ $101.$0

AVATAR: open fscheck.mpexl.telesup

FILE TYPE: executable SOM library, for PA-RISC 1.0

----> LST Module Directory <----    Starts @   #Length
1  OCQUEUE                          $00007000   218852

AVATAR[fscheck.mpexl.telesup]: find "" EXTERNAL

Module: OCQUEUE
Symbol                            X P Symbol    Symbol
Name                                  Type      Scope     Address  Value
-------                           - - ------    -------   -------- -------
P_NEW_HEAP                        0 2 STUB      EXTERNAL  292cc    c01e
P_DISPOSE_HEAP                    0 2 STUB      EXTERNAL  2931c    c0ae
FCHECK                            0 2 STUB      EXTERNAL  29a24    cab2
FERRMSG                           0 2 STUB      EXTERNAL  29a38    cad2
PRINTFILEINFO                     0 2 STUB      EXTERNAL  29a4c    caf2
PRINT                             0 2 STUB      EXTERNAL  29a60    cb12
FGETINFO                          0 2 STUB      EXTERNAL  29ac4    cbde
CCODE                             0 2 STUB      EXTERNAL  29ad8    cbfe
U_set_escapecode                  0 2 STUB      EXTERNAL  29aec    cc1e
FFILEINFO                         0 2 STUB      EXTERNAL  29b00    cc5e
FRELATE                           0 2 STUB      EXTERNAL  29b28    cc7e
U_get_escapecode                  0 2 STUB      EXTERNAL  29b3c    cc3e
FCONTROL                          0 2 STUB      EXTERNAL  29ba0    d146
FREAD                             0 2 STUB      EXTERNAL  29c04    d71e
P_RTERROR                         0 2 STUB      EXTERNAL  29c18    d73e
genmsg                            0 2 STUB      EXTERNAL  29c68    da4e
WHO                               0 2 STUB      EXTERNAL  29d44    db3a
FCLOSE                            0 2 STUB      EXTERNAL  29da8    df9e
U_nonlocal_escape                 0 2 STUB      EXTERNAL  29dbc    dfbe
```

```
P_STRWRITECHR                     0 2 STUB      EXTERNAL  29e34    e0c2
P_STRWRITESTR                     0 2 STUB      EXTERNAL  29e48    e0e2
HPFOPEN                           0 2 STUB      EXTERNAL  29e5c    e102
P_STRAPPEND                       0 2 STUB      EXTERNAL  2a03c    eafa
FWRITE                            0 2 STUB      EXTERNAL  2a050    eb1a
PRINTOP                           0 2 STUB      EXTERNAL  2a140    fa72
P_STRDELETE                       0 2 STUB      EXTERNAL  2a1b8    1007e
genfparse_nm                      0 2 STUB      EXTERNAL  2a5f0    122da
genfxlate                         0 2 STUB      EXTERNAL  2a62c    122fa
vlm_get_vol_set_info              0 2 STUB      EXTERNAL  2a744    12686
vlm_get_next_vol_in_set           0 2 STUB      EXTERNAL  2a780    126c6
chkparm                           0 2 STUB      EXTERNAL  2a820    12a0a
gword                             0 2 STUB      EXTERNAL  2a848    12a2a
CATREAD                           0 2 STUB      EXTERNAL  2a8ac    12b0a
vlm_get_mvt_lab_tbl_ptr           0 2 STUB      EXTERNAL  2a9ec    1302e
P_STRREADINT                      0 2 STUB      EXTERNAL  2aac8    13266
io_device_class                   0 2 STUB      EXTERNAL  2aadc    13286
vlm_ldev_to_mv_id                 0 2 STUB      EXTERNAL  2ab04    132a6
vlm_obtain_mvt_entry              0 2 STUB      EXTERNAL  2ab2c    132c6
get_device_list                   0 2 STUB      EXTERNAL  2ab68    13306
vlm_get_vol_set_id                0 2 STUB      EXTERNAL  2aba4    13326
vlm_volume_id_to_mv_id            0 2 STUB      EXTERNAL  2abe0    13346
lm_get_next_extent_blk_ptr        0 2 STUB      EXTERNAL  2ac58    139ca
gparmcol                          0 2 STUB      EXTERNAL  2ad0c    13d32
vlm_obtain_mvt_ptr                0 2 STUB      EXTERNAL  2ad34    13d52
chkopt                            0 2 STUB      EXTERNAL  2ad5c    13d72
enter_system_code                 0 2 STUB      EXTERNAL  2adc0    14436
set_critical                      0 2 STUB      EXTERNAL  2adfc    14456
dirchk01                          0 2 STUB      EXTERNAL  2ae24    14496
PUTJCW                            0 2 STUB      EXTERNAL  2ae60    144b6
reset_critical                    0 2 STUB      EXTERNAL  2ae74    144d6
leave_system_code                 0 2 STUB      EXTERNAL  2ae9c    144f6
lm_dealloc_label                  0 2 STUB      EXTERNAL  2b068    14e6e
direcinsertfile_nm                0 2 STUB      EXTERNAL  2b0e0    15336
direcfind                         0 2 STUB      EXTERNAL  2b194    159d2
lm_get_next_file_label            0 2 STUB      EXTERNAL  2b284    1617a
vlm_vol_id_to_vol_set_name        0 2 STUB      EXTERNAL  2b2c0    1619a
lm_label_diagnostic_              0 2 STUB      EXTERNAL  2b2e8    161ba
hpdir_get_path_from_ufid          0 2 STUB      EXTERNAL  2b324    161da
P_STRWRITEINT                     0 2 STUB      EXTERNAL  2b360    161fa
```

```
lm_check_free_list                 0 2 STUB      EXTERNAL  2b3b0    16dea
lm_enumerate_table                 0 2 STUB      EXTERNAL  2b608    17cfa
FPOINT                             0 2 STUB      EXTERNAL  2b680    17ea6
HPSORTINIT                         0 2 STUB      EXTERNAL  2b6e4    184b2
HPSORTEND                          0 2 STUB      EXTERNAL  2b6f8    184d2
vlm_get_master_vol_id              0 2 STUB      EXTERNAL  2b84c    189b2
P_STRPOS                           0 2 STUB      EXTERNAL  2b888    189d2
vlm_get_volume_id                  0 2 STUB      EXTERNAL  2b89c    189f2
vlm_get_slough_ufid                0 2 STUB      EXTERNAL  2b914    1966a
lm_forget_lost_extents             0 2 STUB      EXTERNAL  2b93c    1968a
P_STRREADPAC                       0 2 STUB      EXTERNAL  2baa4    19ce6
direcopen                          0 2 STUB      EXTERNAL  2bacc    19d26
direcscan_files                    0 2 STUB      EXTERNAL  2bb08    19d46
dirc_unlock_shr                    0 2 STUB      EXTERNAL  2bb58    19d66
request_service                    0 2 STUB      EXTERNAL  2bb80    19d86
pmatcher                           0 2 STUB      EXTERNAL  2bba8    19da6
FLABELINFO                         0 2 STUB      EXTERNAL  2bbf8    19dc6
lm_label_diagnostic                0 2 STUB      EXTERNAL  2bc0c    19de6
direcclose                         0 2 STUB      EXTERNAL  2bc34    19e06
dismember_name                     0 2 STUB      EXTERNAL  2bcd4    1ba5e
hpdir_find_path                    0 2 STUB      EXTERNAL  2bd24    1ba7e
sm_get_gufd                        0 2 STUB      EXTERNAL  2bd9c    1beca
direcpurgefile                     0 2 STUB      EXTERNAL  2be28    1bfa6
lm_unlock_file                     0 2 STUB      EXTERNAL  2bf90    1c6fe
lf_get_flab_eof_offset_64          0 2 STUB      EXTERNAL  2c5f8    1f0ea
lm_get_next_extent_map_ptr         0 2 STUB      EXTERNAL  2c864    2005e
lm_get_file_label_ptr              0 2 STUB      EXTERNAL  2c8a0    2007e
sm_open_fd                         0 2 STUB      EXTERNAL  2c918    2030a
sm_open_object                     0 2 STUB      EXTERNAL  2c954    2032a
cb_shr_lock                        0 2 STUB      EXTERNAL  2c9a4    2034a
cb_shr_unlock                      0 2 STUB      EXTERNAL  2c9cc    2036a
sm_close                           0 2 STUB      EXTERNAL  2c9f4    2038a
P_STRWRITELONGINT                  0 2 STUB      EXTERNAL  2cb20    2113a
disc_sm_start_write                0 2 STUB      EXTERNAL  2cb34    2115a
disc_sm_finish_write               0 2 STUB      EXTERNAL  2cb70    2117a
vlm_vs_name_to_root_ufid           0 2 STUB      EXTERNAL  2cbfc    224fa
abortgoldenxid                     0 2 STUB      EXTERNAL  2cc38    2251a
lm_get_next_locked_file_label      0 2 STUB      EXTERNAL  2ce40    238ea
msec_to_cal_clk                    0 2 STUB      EXTERNAL  2d1c4    24eaa
FMTDATE                            0 2 STUB      EXTERNAL  2d1ec    24eca
```



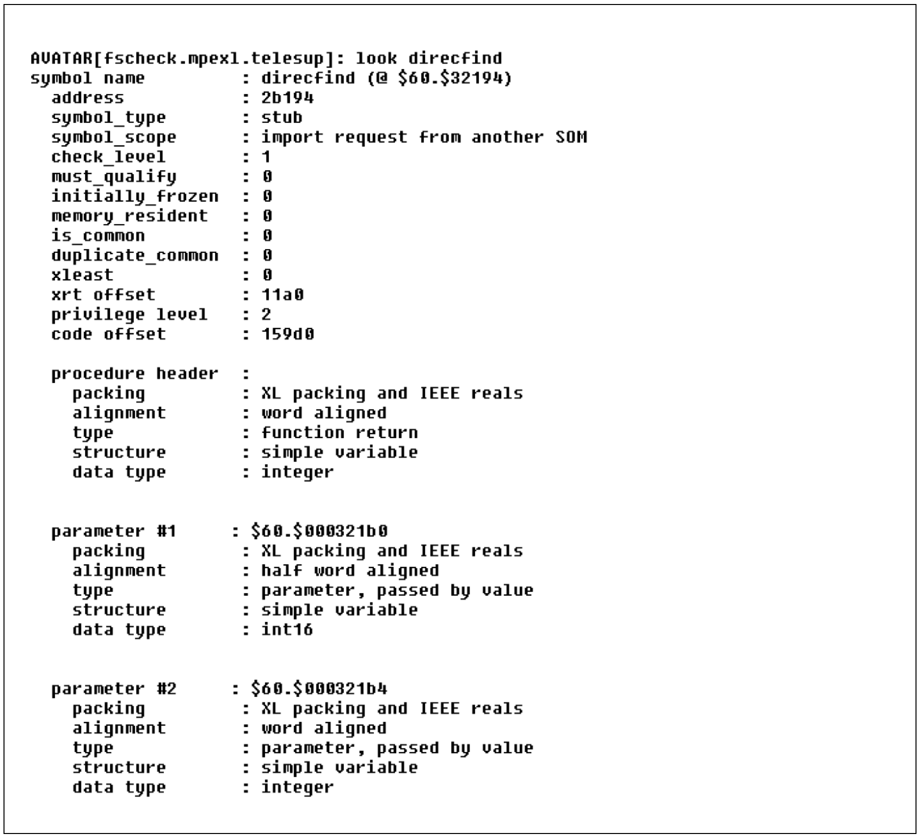
```
P_WRITELN                          0 2 STUB      EXTERNAL  2d228    24f76
P_WRITESTR                         0 2 STUB      EXTERNAL  2d23c    24f96
lm_get_next_quarantined_file_label
                                   0 2 STUB      EXTERNAL  2d278    257ea
cb_lock                            0 2 STUB      EXTERNAL  2d318    25caa
cb_unlock                          0 2 STUB      EXTERNAL  2d340    25cca
vlm_get_ldev                       0 2 STUB      EXTERNAL  2d368    25cea
vlm_ldev_to_volume_name            0 2 STUB      EXTERNAL  2d390    25d0a
DBINARY                            0 2 STUB      EXTERNAL  2d494    2660e
hpdir_unlink_name_by_ufid          0 2 STUB      EXTERNAL  2d520    26a3a
xm_detachufid                      0 2 STUB      EXTERNAL  2d548    26a5a
sm_map_out_file                    0 2 STUB      EXTERNAL  2d570    26a7a
sm_del_hash_tbl_gufd               0 2 STUB      EXTERNAL  2d5ac    26a9a
deallocate_pid                     0 2 STUB      EXTERNAL  2d5d4    26aba
vsm_deallocate_gu_fd               0 2 STUB      EXTERNAL  2d5fc    26ada
vlm_decr_open_count                0 2 STUB      EXTERNAL  2d624    26afa
get_image                          0 2 STUB      EXTERNAL  2d8e0    280aa
redo                               0 2 STUB      EXTERNAL  2d9d0    28ab6
coreparser                         0 2 STUB      EXTERNAL  2da98    29202
DEBUG                              0 2 STUB      EXTERNAL  2dbb0    298aa
HPCICOMMAND                        0 2 STUB      EXTERNAL  2dbc4    298ca
HPERRDEPTH                         0 2 STUB      EXTERNAL  2dc8c    2a01a
HPERRREAD                          0 2 STUB      EXTERNAL  2dca0    2a03a
HPERRMSG                           0 2 STUB      EXTERNAL  2dcb4    2a05a
P_STRINSERT                        0 2 STUB      EXTERNAL  2dcc8    2a07a
U_nonlocal_goto                    0 2 STUB      EXTERNAL  2de1c    2ae8e
NLAPPEND                           0 2 STUB      EXTERNAL  2ded0    2b0de
CATOPEN                            0 2 STUB      EXTERNAL  2dee4    2b0fe
CATCLOSE                           0 2 STUB      EXTERNAL  2def8    2b11e
P_INIT_ARGS                        0 2 STUB      EXTERNAL  2df48    2bb92
U_INIT_TRAPS                       0 2 STUB      EXTERNAL  2df5c    2bbb2
P_INIT_HEAP                        0 2 STUB      EXTERNAL  2df70    2bbd2
P_SET_COMPACTION                   0 2 STUB      EXTERNAL  2df84    2bbf2
P_GET_PARM                         0 2 STUB      EXTERNAL  2df98    2bc12
P_GET_INFO                         0 2 STUB      EXTERNAL  2dfac    2bc32
P_REWRITE                          0 2 STUB      EXTERNAL  2dfc0    2bc52
P_TERMINATE                        0 2 STUB      EXTERNAL  2dfd4    2bc72

AVATAR[fscheck.mpexl.telesup]:
```

Figure 3.23 *FIND Command (External)*

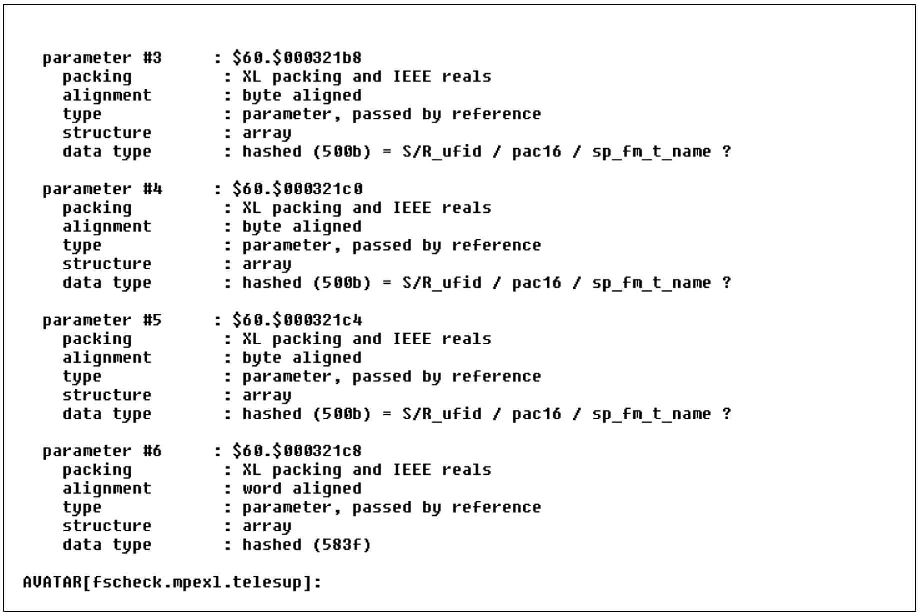
Occasionally, you may want to know what external procedures a program calls. The FIND command can easily locate all external procedure calls.

Figure 3.24 uses the LOOK command to determine parameter types.

Because the parameter to the LOOK command was entered in lower case type, we know immediately that it is not an intrinsic call but rather an external procedure. There are six parameters for this procedure and we can see that the first 3 parameters are simple variables while the last 3 are array (or record) parameters.



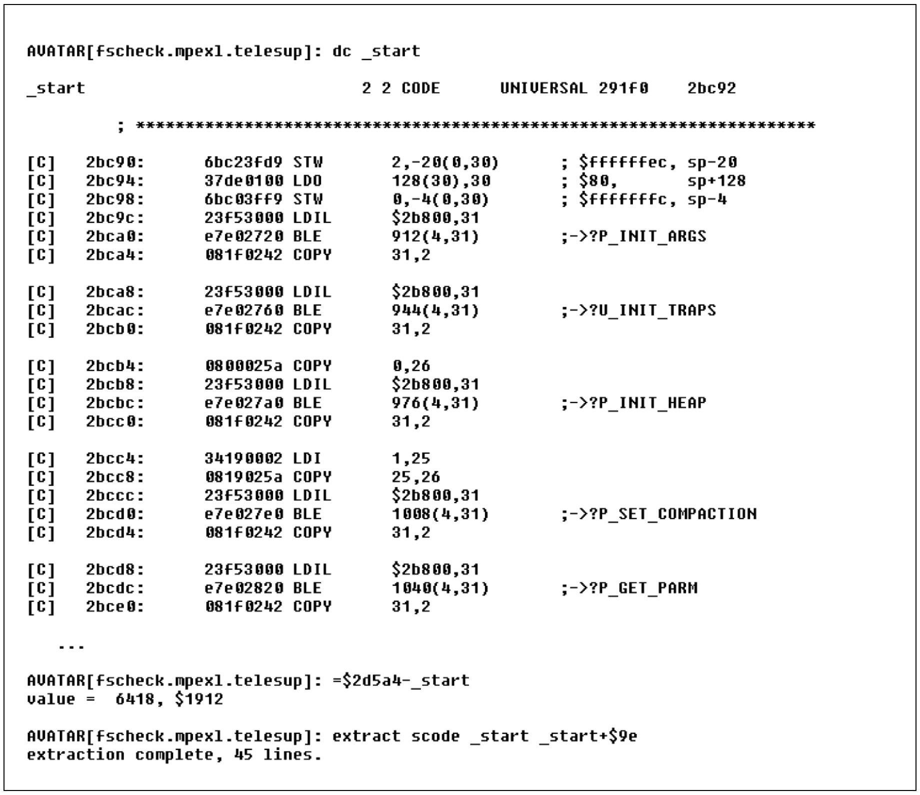
```
AVATAR[fscheck.mpexl.telesup]: look direcfind
symbol name         : direcfind (@ $60.$32194)
  address           : 2b194
  symbol_type       : stub
  symbol_scope      : import request from another SOM
  check_level       : 1
  must_qualify      : 0
  initially_frozen  : 0
  memory_resident   : 0
  is_common         : 0
  duplicate_common  : 0
  xleast            : 0
  xrt offset        : 11a0
  privilege level   : 2
  code offset       : 159d0

  procedure header  :
    packing         : XL packing and IEEE reals
    alignment       : word aligned
    type            : function return
    structure       : simple variable
    data type       : integer


  parameter #1     : $60.$000321b0
    packing         : XL packing and IEEE reals
    alignment       : half word aligned
    type            : parameter, passed by value
    structure       : simple variable
    data type       : int16


  parameter #2     : $60.$000321b4
    packing         : XL packing and IEEE reals
    alignment       : word aligned
    type            : parameter, passed by value
    structure       : simple variable
    data type       : integer
```



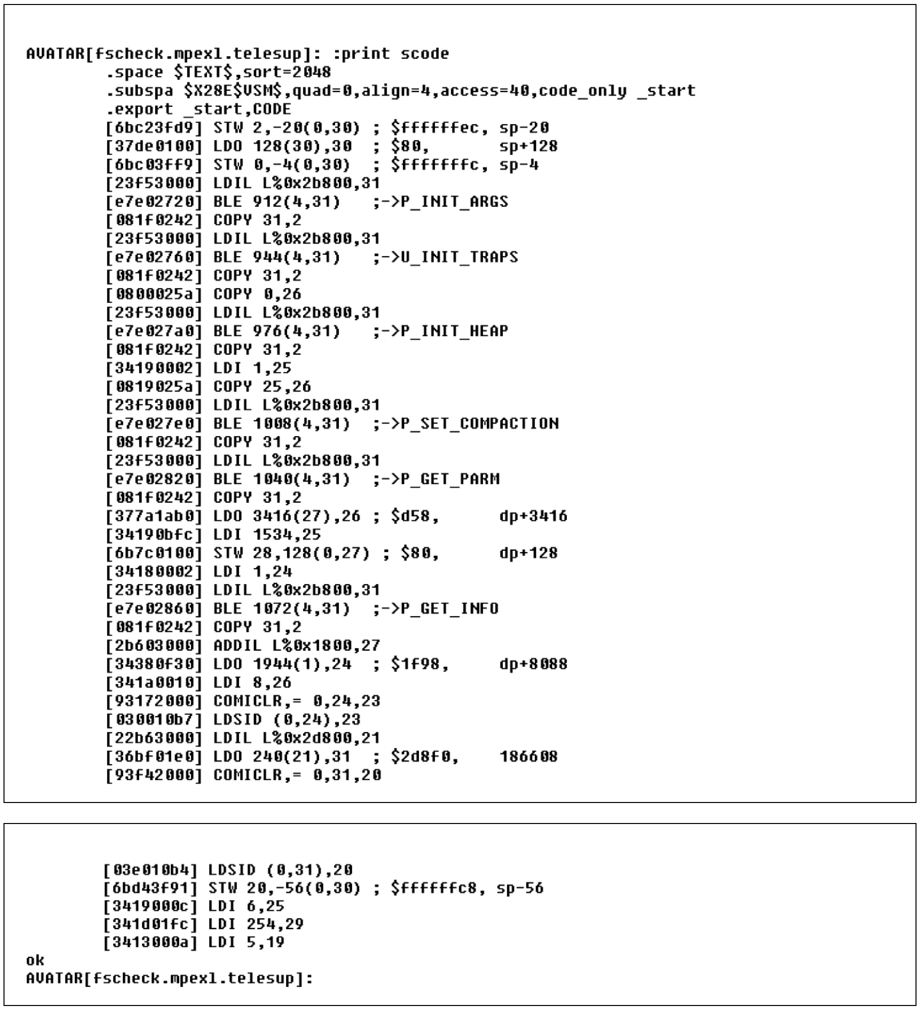
```
  parameter #3      : $60.$000321b8
    packing          : XL packing and IEEE reals
    alignment        : byte aligned
    type             : parameter, passed by reference
    structure        : array
    data type        : hashed (500b) = S/R_ufid / pac16 / sp_fm_t_name ?

  parameter #4      : $60.$000321c0
    packing          : XL packing and IEEE reals
    alignment        : byte aligned
    type             : parameter, passed by reference
    structure        : array
    data type        : hashed (500b) = S/R_ufid / pac16 / sp_fm_t_name ?

  parameter #5      : $60.$000321c4
    packing          : XL packing and IEEE reals
    alignment        : byte aligned
    type             : parameter, passed by reference
    structure        : array
    data type        : hashed (500b) = S/R_ufid / pac16 / sp_fm_t_name ?

  parameter #6      : $60.$000321c8
    packing          : XL packing and IEEE reals
    alignment        : word aligned
    type             : parameter, passed by reference
    structure        : array
    data type        : hashed (583f)

AVATAR[fscheck.mpexl.telesup]:
```

Figure 3.24 *LOOK Command (direcfind)*

Figure 3.25 shows how the EXTRACT command is used.



```
AVATAR[fscheck.mpexl.telesup]: dc _start

_start                            2 2 CODE      UNIVERSAL 291f0    2bc92

         ; ***********************************************************************

[C]   2bc90:      6bc23fd9 STW       2,-20(0,30)      ; $ffffffec, sp-20
[C]   2bc94:      37de0100 LDO       128(30),30       ; $80,       sp+128
[C]   2bc98:      6bc03ff9 STW       0,-4(0,30)       ; $fffffffc, sp-4
[C]   2bc9c:      23f53000 LDIL      $2b800,31
[C]   2bca0:      e7e02720 BLE       912(4,31)        ;->?P_INIT_ARGS
[C]   2bca4:      081f0242 COPY      31,2

[C]   2bca8:      23f53000 LDIL      $2b800,31
[C]   2bcac:      e7e02760 BLE       944(4,31)        ;->?U_INIT_TRAPS
[C]   2bcb0:      081f0242 COPY      31,2

[C]   2bcb4:      0800025a COPY      0,26
[C]   2bcb8:      23f53000 LDIL      $2b800,31
[C]   2bcbc:      e7e027a0 BLE       976(4,31)        ;->?P_INIT_HEAP
[C]   2bcc0:      081f0242 COPY      31,2

[C]   2bcc4:      34190002 LDI       1,25
[C]   2bcc8:      0819025a COPY      25,26
[C]   2bccc:      23f53000 LDIL      $2b800,31
[C]   2bcd0:      e7e027e0 BLE       1008(4,31)       ;->?P_SET_COMPACTION
[C]   2bcd4:      081f0242 COPY      31,2

[C]   2bcd8:      23f53000 LDIL      $2b800,31
[C]   2bcdc:      e7e02820 BLE       1040(4,31)       ;->?P_GET_PARM
[C]   2bce0:      081f0242 COPY      31,2

   ...

AVATAR[fscheck.mpexl.telesup]: =$2d5a4-_start
value =  6418, $1912

AVATAR[fscheck.mpexl.telesup]: extract scode _start _start+$9e
extraction complete, 45 lines.
```



```
AVATAR[fscheck.mpexl.telesup]: :print scode
        .space $TEXT$,sort=2048
        .subspa $X28E$VSM$,quad=0,align=4,access=40,code_only _start
        .export _start,CODE
        [6bc23fd9] STW 2,-20(0,30) ; $ffffffec, sp-20
        [37de0100] LDO 128(30),30  ; $80,       sp+128
        [6bc03ff9] STW 0,-4(0,30)  ; $fffffffc, sp-4
        [23f53000] LDIL L%0x2b800,31
        [e7e02720] BLE 912(4,31)   ;->P_INIT_ARGS
        [081f0242] COPY 31,2
        [23f53000] LDIL L%0x2b800,31
        [e7e02760] BLE 944(4,31)   ;->U_INIT_TRAPS
        [081f0242] COPY 31,2
        [0800025a] COPY 0,26
        [23f53000] LDIL L%0x2b800,31
        [e7e027a0] BLE 976(4,31)   ;->P_INIT_HEAP
        [081f0242] COPY 31,2
        [34190002] LDI 1,25
        [0819025a] COPY 25,26
        [23f53000] LDIL L%0x2b800,31
        [e7e027e0] BLE 1008(4,31)  ;->P_SET_COMPACTION
        [081f0242] COPY 31,2
        [23f53000] LDIL L%0x2b800,31
        [e7e02820] BLE 1040(4,31)  ;->P_GET_PARM
        [081f0242] COPY 31,2
        [377a1ab0] LDO 3416(27),26 ; $d58,      dp+3416
        [34190bfc] LDI 1534,25
        [6b7c0100] STW 28,128(0,27) ; $80,      dp+128
        [34180002] LDI 1,24
        [23f53000] LDIL L%0x2b800,31
        [e7e02860] BLE 1072(4,31)  ;->P_GET_INFO
        [081f0242] COPY 31,2
        [2b603000] ADDIL L%0x1800,27
        [34380f30] LDO 1944(1),24  ; $1f98,     dp+8088
        [341a0010] LDI 8,26
        [93172000] COMICLR,= 0,24,23
        [030010b7] LDSID (0,24),23
        [22b63000] LDIL L%0x2d800,21
        [36bf01e0] LDO 240(21),31  ; $2d8f0,    186608
        [93f42000] COMICLR,= 0,31,20
```

```
        [03e010b4] LDSID (0,31),20
        [6bd43f91] STW 20,-56(0,30) ; $ffffffc8, sp-56
        [3419000c] LDI 6,25
        [341d01fc] LDI 254,29
        [3413000a] LDI 5,19
ok
AVATAR[fscheck.mpexl.telesup]:
```

Figure 3.25 *EXTRACT Command*

AVATAR Error Messages

Each AVATAR error message is described in the following table.

Table 3.5 *AVATAR Error Messages*

Message	Cause	Action
Expected "D" or "C" as data type	DISPLAY command expects known data type modified.	Use only the "D" (hex/ASCII dump) or "C" (disassemble) data type modifiers.
Invalid primary	User entered unknown AVATAR command or option.	Review command syntax.
Invalid start address	EXTRACT command start address must be within procedure.	Use AVATAR to review the address range for the procedure being extracted.
Invalid type	User entered invalid symbol type for lookup symbol command.	Review symbol types. The valid types are: ABSOLUTE, DATA, CODE, PRI_PROG, SEC_PROG, ENTRY, STORAGE, STUB, MODULE, SYM_EXT, ARG_EXT, MILLICODE, PLABEL, OCT_DIS, MILLI_EXT.
Start must be less then end.	EXTRACT command expects and offset to be larger than beginning offset.	Review EXTRACT command syntax.

4

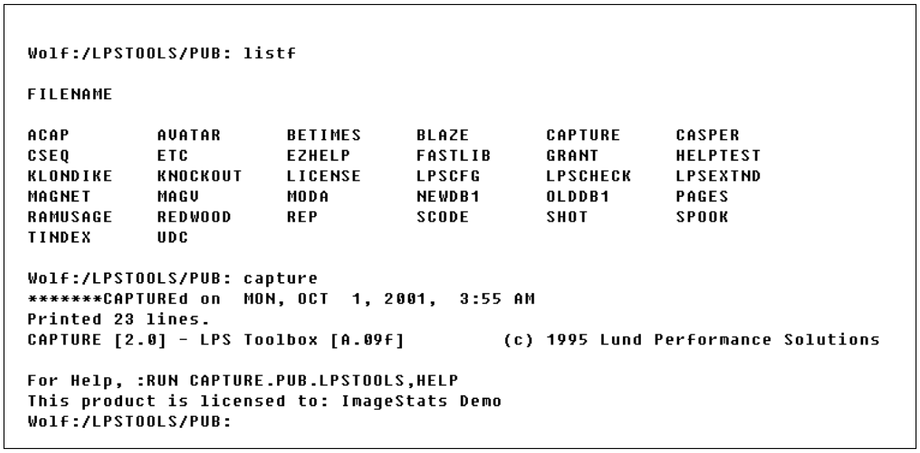
THE CAPTURE TOOL

CAPTURE will copy all (or only part) of the text displayed on a terminal screen (or contained in terminal memory) to a printer or a designated disc file. The terminal must obey standard Hewlett-Packard terminal control codes.

CAPTURE may be executed either as a stand-alone program or as a callable procedure. Several options are available at run-time that permit you to capture only that portion of the screen desired, in the format you require.

Operation

To use CAPTURE as a stand-alone program, simply type: **CAPTURE** at the system prompt. Doing this will cause CAPTURE to copy your entire screen memory to the line printer. Like all ***LPS-Tools***, CAPTURE displays the standard ***LPS-Tools*** banner. In this case, however, the banner display follows the screen dump. See the following screen display for an example of how this works.



```
Wolf:/LPSTOOLS/PUB: listf

FILENAME

ACAP        AVATAR      BETIMES     BLAZE       CAPTURE     CASPER
CSEQ        ETC         EZHELP      FASTLIB     GRANT       HELPTEST
KLONDIKE    KNOCKOUT    LICENSE     LPSCFG      LPSCHECK    LPSEXTND
MAGNET      MAGV        MODA        NEWDB1      OLDDB1      PAGES
RAMUSAGE    REDWOOD     REP         SCODE       SHOT        SPOOK
TINDEX      UDC

Wolf:/LPSTOOLS/PUB: capture
*******CAPTUREd on  MON, OCT  1, 2001,  3:55 AM
Printed 23 lines.
CAPTURE [2.0] - LPS Toolbox [A.09f]         (c) 1995 Lund Performance Solutions

For Help, :RUN CAPTURE.PUB.LPSTOOLS,HELP
This product is licensed to: ImageStats Demo
Wolf:/LPSTOOLS/PUB:
```

Figure 4.1 *CAPTURE Screen*

You are limited to capturing up to 1023 characters per line on the screen. You may suppress portions of the informational displays that usually follow a successful screen capture, and you may also suppress some of the error handling procedures of the program.

CAPTURE permits you to control the handling of escape codes that are often present on a screen, but not always desirable in a copy.

Capabilities

Program capabilities required include IA, BA, DS, and PH. No special user capabilities are required to run CAPTURE.

Usage

The MPE syntax for executing CAPTURE is:

```
    RUN CAPTURE.PUB.LPSTOOLS [;INFO="keywords"][;PARM=parms]
```

The default mode of operation sends a copy of the entire terminal memory to the system line printer, which is designated with the formal file name of LPSLP.

NOTE There are many keywords (see Table 4.1) available for use with CAPTURE using the **INFO="keywords"** option. You must supply the full keyword, or only the portion indicated with capital letters. More than one keyword may be used; the space character is the delimiter.

Option Summary

Unlike other ***LPS-Tools***, CAPTURE is a single-command tool, where the CAPTURE command is the only command that can be specified. Several options, however, can be used to further define the task at hand. These options are listed next.

Table 4.1 *CAPTURE Options*

Option	Description
COMPRESS	Compressed portrait output to LaserJet
CUT firstcol/lastcol	Specify columns to capture
FF	Formfeed line printer
FFL	Formfeed LaserJet
FLAT	Directs output to a disk file

Option	Description
HELP	Starts CAPTURE's help subsystem
Landscape	Landscape output to LaserJet
LEFT column	Specifies left column to capture
[NO]CHEck	Limits error checking activity
[NO]ENHance	Strips display enhancements
[NO]ENHOFfeol	Adds printer escape sequence at end of line
[NO]RESETL	Reset LaserJet
[NO]SETMSG	Controls SETMSG use during capture
[NO]STAmp	Provides for a date and time stamp to be displayed
[NO]SUMmary	Suppresses line & timestamp on the screen
OFFSET	Specifies offset in output column
PARTial	Captures a user-specified range of lines
QUIET	Suppresses error messages
RIGHT column	Specifies right column to capture

CAPTURE Commands

```
CAPTURE (parm=0)

(No parameters)
```

Running CAPTURE without any options causes your entire screen memory to be captured.

CAPTURE starts copying from the top of terminal memory (line number 1) and copies through the line that contains the cursor's original location. CAPTURE does not alter terminal memory. The output from CAPTURE will be sent to the line printer attached to your system. The formal file equation that CAPTURE (as well as all other ***LPS-Tools***) uses is LPSLP. To redirect CAPTURE's output to a line printer other than LP, simply issue an appropriate file equation.

Options Definitions

CAPTURE options may be specified as keywords in the INFO string or as bits in the PARM value.

COMPRESS

Format for LaserJet compressed output. This option can be used in Landscape mode.

CUT firstcol/lastcol

This option is used to specify the column range (where **n** equals the column number) used in the capture.

FF

This option instructs capture to send a formfeed command at the end of the screen capture. It is used for line printer output.

FFL

This option is the same as FF except that it is used for LaserJet output.

FLAT

This option is used to tell CAPTURE that you wish to redirect output to a disk file. The formal file designator for the disk file is also called FLAT. You may redirect output to a file of another name by using an appropriate file equation.

NOTE The file is built with a default record length of 80 bytes, that it must not exist prior to running CAPTURE, and that if a system problem prevents the saving of the file as permanent, an attempt will be made to save it as session temporary. When FLAT is used, only the lines of text in terminal memory are copied to the disc file. There will be no additional information lines appended by CAPTURE for audit purposes, such as information summaries or date and time stamps. CAPTURE will not purge any existing file when the FLAT option is used.

HELP

Starts Capture's help subsystem.

Landscape

Format for LaserJet landscape output.

LEFT column

This option is used to specify the starting (left) column for the capture.

[NO]CHEck

This option inhibits CAPTURE from checking certain error conditions. This can be useful if you have non-HP terminals that are similar to standard, but that would be ignored by CAPTURE if it detected the error conditions. Using NOCHECK also causes CAPTURE to ignore errors that might be generated when running in BATCH mode.

Be sure that you understand what you are doing when you use this option. Ports and jobs could be hung if this option is used improperly.

[NO]ENHance

When NOENHANCE is specified, the text read from terminal memory is examined for escape sequences that control the screen enhancements. If any are found, they are removed.

[NO]ENHOFfeol

ENHOFFEOL forces a special escape sequence to be appended to any line that has a screen enhancement. This special sequence manually terminates the enhancement, which is useful when the output is directed to a LaserJet. In normal operation, use of enhancements may cause unexpected LaserJet output if proper enhancement terminators have not been used, ENHOFFEOL solves this problem.

[NO]RESETL

Resets the LaserJet with an <esc> **E** before sending the capture.

[NO]SETMSG

This option controls whether or not capture uses SETMSG ON (default) or SETMSG OFF.

[NO]STAmp

CAPTURE provides for a date and time stamp to be displayed on the screen at the start of the screen capture process, and for this line to be displayed at the end of any LPSLP listing. Use of the NOSTAMP keyword will suppress the display on the listing; the display on the screen will remain.

CAPTURE provides for a date and time stamp to be displayed on the screen at the start of the screen capture process, and for this line to be displayed at the end of any LPSLP listing. Use of the STAMP keyword will ensure that the display appears on both the listing and the screen. This keyword is a default setting.

[NO]SUMMARY

CAPTURE provides for a summary of the number of lines printed and the keywords selected to be displayed on the screen at the start of the screen capture process. Use of this keyword will suppress this display from the screen.

NOSUMMARY also suppresses the display of the date and time stamp on the screen. However, this information line will be displayed on the LPSLP listing.

OFFSET

This option is used to specify a column offset for the capture output.

PARTIAL

Using this option tells CAPTURE to capture only a portion of the lines in terminal memory. CAPTURE will interactively request that you mark the last line an then the first line of text that you want to CAPTURE. Use of this option will suppress the information summary (see SUMMARY) from the display.

QUIET

Suppresses CAPTURE's error messages.

RIGHT column

This option is used to specify the ending (right) column for the capture.

PARMs

You may use PARMs instead of INFO= strings in many circumstances. Table 4.2 indicates the parm value, the INFO= keyword string to which it is equivalent, and the bit location of the keyword when represented as a binary value.

Table 4.2 *CAPTURE's PARM*

Parm	Bit	INFO= String
32768	0	HELP
16384	1	QUIET
8192	2	RESETLaserjet
4096	3	COMPRESSED Laserjet
2048	4	LANDSCAPE Laserjet

Parm	Bit	INFO= String
1024	5	<internal use>
512	6	FFLaserjet
256	7	FF line printer
128	8	NOSETMSG
64	9	NOSTAmp
32	10	ENHOFfeol
16	11	NOENHance
8	12	NOSUMmary
4	13	NOCHEck
2	14	FLAT
1	15	PARTial

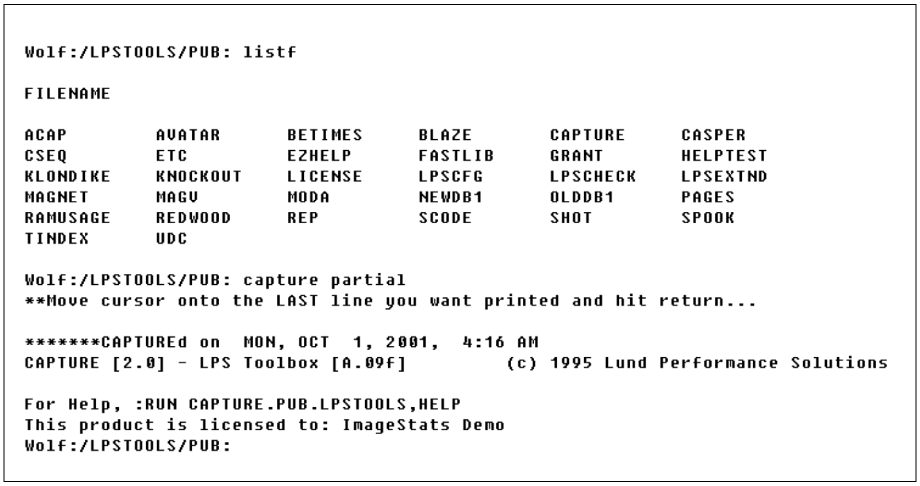
Combinations of keywords may be represented by adding the parm numbers (their decimal values) together. For instance, to combine NOCHECK with NOSUMMARY you need only add 8 and 4 yielding a parm value of 12. This is equivalent to setting bits 12 and 13 to an "on" value in a binary string.

TOOLBOX STANDARDS

The ToolBox collections from Lund Performance Solutions have a uniform user interface. As a result, in addition to the commands specific to each Toolbox tool, most tools allow the commands described in “TOOLBOX STANDARDS” on page 213.

CAPTURE Examples

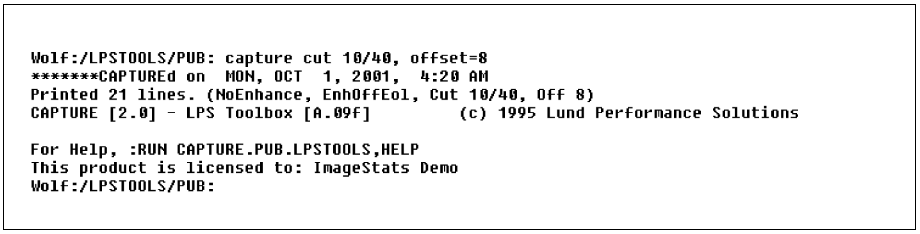
Use CAPTURE to select portions of the terminal screen, send the contents to a flat file, or combine various options. Combine the two (CAPTURE and PARTIAL) of CAPTURE’s options to realize a partial screen capture. See Figure 4.2. Do this with the command CAPTURE PARTIAL.



```
Wolf:/LPSTOOLS/PUB: listf

FILENAME

ACAP        AVATAR      BETIMES     BLAZE       CAPTURE     CASPER
CSEQ        ETC         EZHELP      FASTLIB     GRANT       HELPTEST
KLONDIKE    KNOCKOUT    LICENSE     LPSCFG      LPSCHECK    LPSEXTND
MAGNET      MAGV        MODA        NEWDB1      OLDDB1      PAGES
RAMUSAGE    REDWOOD     REP         SCODE       SHOT        SPOOK
TINDEX      UDC

Wolf:/LPSTOOLS/PUB: capture partial
**Move cursor onto the LAST line you want printed and hit return...

*******CAPTUREd on  MON, OCT  1, 2001,  4:16 AM
CAPTURE [2.0] - LPS Toolbox [A.09f]         (c) 1995 Lund Performance Solutions

For Help, :RUN CAPTURE.PUB.LPSTOOLS,HELP
This product is licensed to: ImageStats Demo
Wolf:/LPSTOOLS/PUB:
```

Figure 4.2 *Capturing a Portion of Screen Memory*

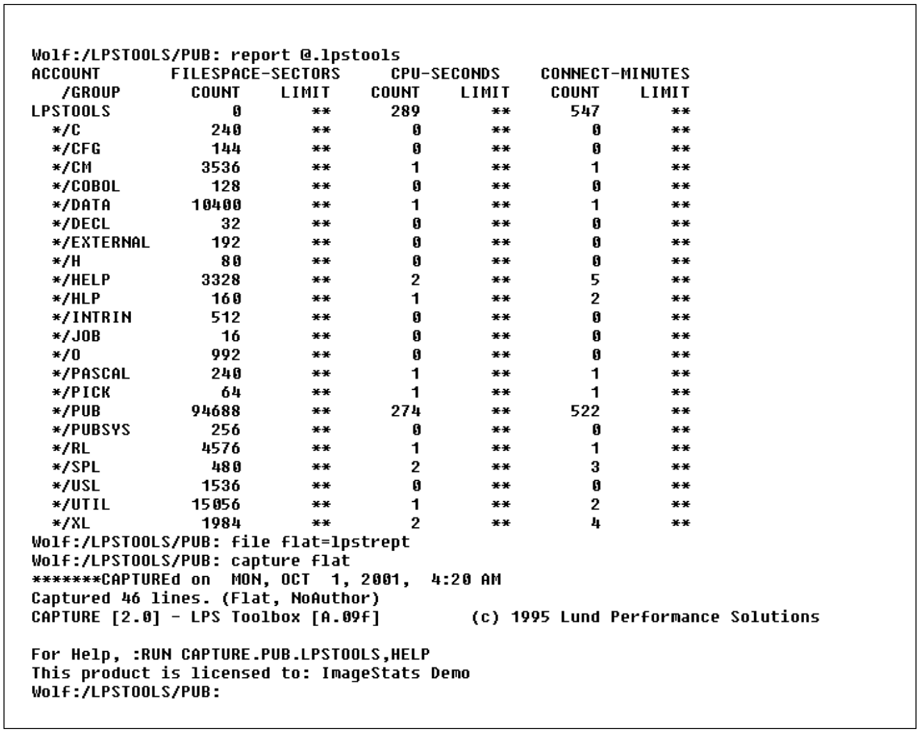
Figure 4.3 shows the Capture of columns 10 through 30 and starting output in the capture file at column 8.



```
Wolf:/LPSTOOLS/PUB: capture cut 10/40, offset=8
*******CAPTUREd on  MON, OCT  1, 2001,  4:20 AM
Printed 21 lines. (NoEnhance, EnhOffEol, Cut 10/40, Off 8)
CAPTURE [2.0] - LPS Toolbox [A.09f]         (c) 1995 Lund Performance Solutions

For Help, :RUN CAPTURE.PUB.LPSTOOLS,HELP
This product is licensed to: ImageStats Demo
Wolf:/LPSTOOLS/PUB:
```

Figure 4.3 *Column CAPTURE*

The following example demonstrates how to set a file equation so that CAPTURE's output goes to the filename of your choice.

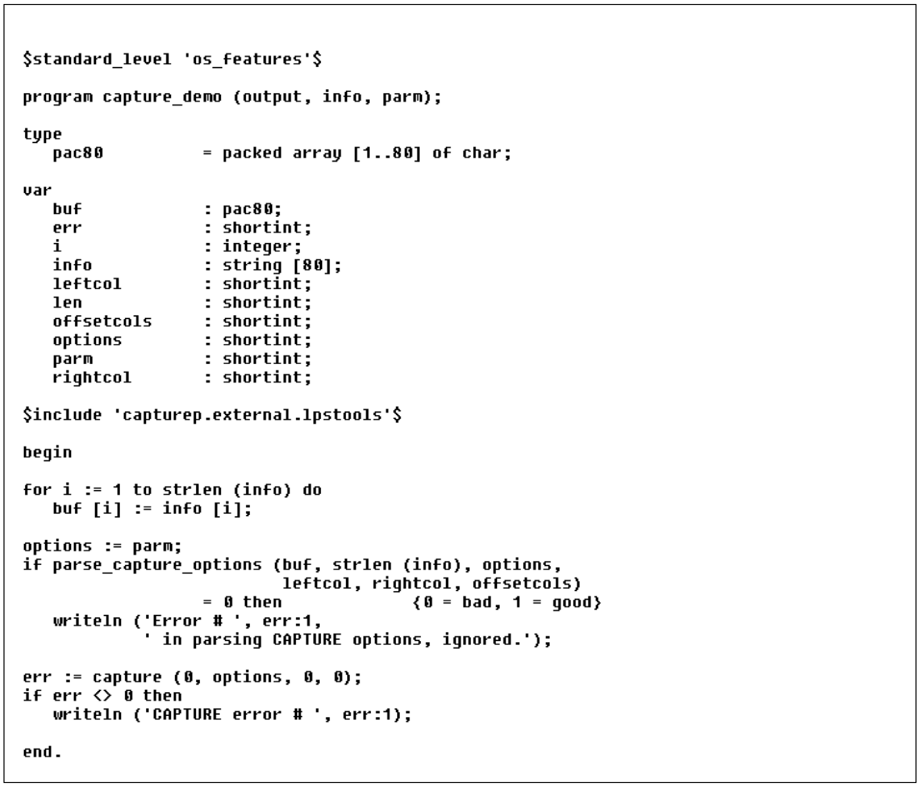


```
Wolf:/LPSTOOLS/PUB: report @.lpstools
ACCOUNT       FILESPACE-SECTORS     CPU-SECONDS    CONNECT-MINUTES
   /GROUP       COUNT    LIMIT    COUNT    LIMIT    COUNT    LIMIT
LPSTOOLS            0       **      289       **      547       **
  */C             240       **        0       **        0       **
  */CFG           144       **        0       **        0       **
  */CM           3536       **        1       **        1       **
  */COBOL         128       **        0       **        0       **
  */DATA        10400       **        1       **        1       **
  */DECL           32       **        0       **        0       **
  */EXTERNAL      192       **        0       **        0       **
  */H              80       **        0       **        0       **
  */HELP         3328       **        2       **        5       **
  */HLP           160       **        1       **        2       **
  */INTRIN        512       **        0       **        0       **
  */JOB            16       **        0       **        0       **
  */O             992       **        0       **        0       **
  */PASCAL        240       **        1       **        1       **
  */PICK           64       **        1       **        1       **
  */PUB         94688       **      274       **      522       **
  */PUBSYS        256       **        0       **        0       **
  */RL           4576       **        1       **        1       **
  */SPL           480       **        2       **        3       **
  */USL          1536       **        0       **        0       **
  */UTIL        15056       **        1       **        2       **
  */XL           1984       **        2       **        4       **
Wolf:/LPSTOOLS/PUB: file flat=lpstrept
Wolf:/LPSTOOLS/PUB: capture flat
*******CAPTUREd on  MON, OCT  1, 2001,  4:20 AM
Captured 46 lines. (Flat, NoAuthor)
CAPTURE [2.0] - LPS Toolbox [A.09f]         (c) 1995 Lund Performance Solutions

For Help, :RUN CAPTURE.PUB.LPSTOOLS,HELP
This product is licensed to: ImageStats Demo
Wolf:/LPSTOOLS/PUB:
```

Figure 4.4 *Sending CAPTURE Output to a File*

Using CAPTURE as a Callable Procedure

CAPTURE may be used as a callable procedure from both CM and NM programs. To include CAPTURE in a CM program, refer to the USL file in the USL group of the LPSTOOLS account. For inclusion in an NM program, take a look at the NMOBJ file in the O group. The following display is from the file CAPTURE.PASCAL.LPSTOOLS. Use it as a reference in implementing CAPTURE in your program for sending screen memory to the line printer. Also provided are examples in C, COBOL, and SPL, as well as RL and XL versions of the executables.

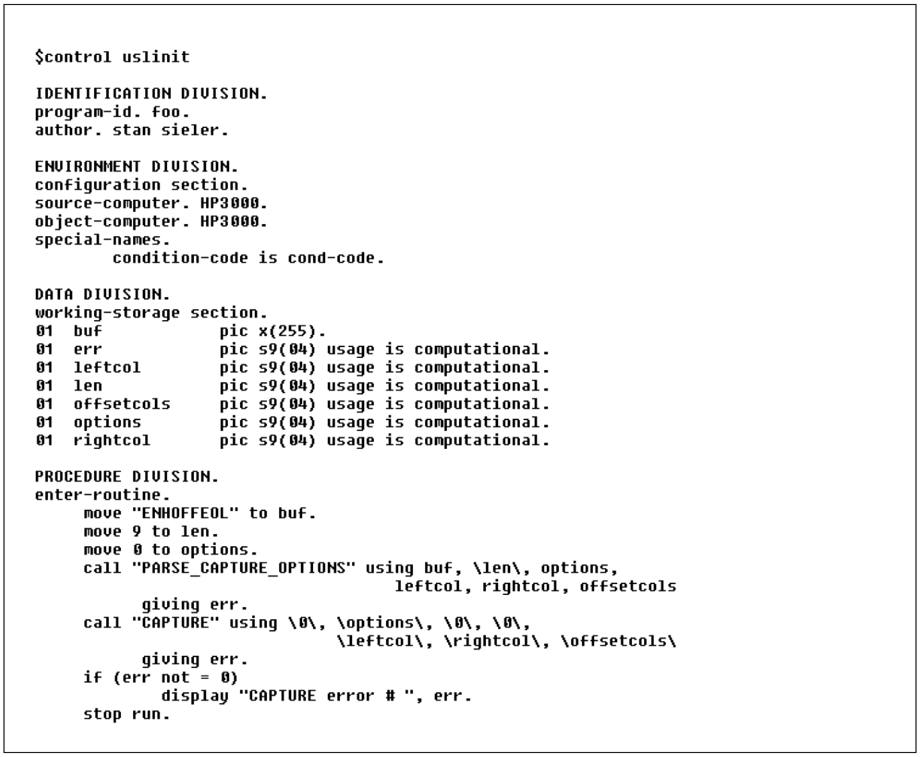


```
$standard_level 'os_features'$

program capture_demo (output, info, parm);

type
   pac80          = packed array [1..80] of char;

var
   buf            : pac80;
   err            : shortint;
   i              : integer;
   info           : string [80];
   leftcol        : shortint;
   len            : shortint;
   offsetcols     : shortint;
   options        : shortint;
   parm           : shortint;
   rightcol       : shortint;

$include 'capturep.external.lpstools'$

begin

for i := 1 to strlen (info) do
   buf [i] := info [i];

options := parm;
if parse_capture_options (buf, strlen (info), options,
                          leftcol, rightcol, offsetcols)
                  = 0 then             {0 = bad, 1 = good}
   writeln ('Error # ', err:1,
            ' in parsing CAPTURE options, ignored.');

err := capture (0, options, 0, 0);
if err <> 0 then
   writeln ('CAPTURE error # ', err:1);

end.
```

Figure 4.5 *CAPTURE as a Callable Procedure*

Using CAPTURE Procedures in COBOL

The following code fragment illustrates the use of CAPTURES in COBOL code where whatever is on the screen is sent to the line printer.



```
$control uslinit

IDENTIFICATION DIVISION.
program-id. foo.
author. stan sieler.

ENVIRONMENT DIVISION.
configuration section.
source-computer. HP3000.
object-computer. HP3000.
special-names.
        condition-code is cond-code.

DATA DIVISION.
working-storage section.
01  buf            pic x(255).
01  err            pic s9(04) usage is computational.
01  leftcol        pic s9(04) usage is computational.
01  len            pic s9(04) usage is computational.
01  offsetcols     pic s9(04) usage is computational.
01  options        pic s9(04) usage is computational.
01  rightcol       pic s9(04) usage is computational.

PROCEDURE DIVISION.
enter-routine.
     move "ENHOFFEOL" to buf.
     move 9 to len.
     move 0 to options.
     call "PARSE_CAPTURE_OPTIONS" using buf, \len\, options,
                                     leftcol, rightcol, offsetcols
           giving err.
     call "CAPTURE" using \0\, \options\, \0\, \0\,
                               \leftcol\, \rightcol\, \offsetcols\
           giving err.
     if (err not = 0)
             display "CAPTURE error # ", err.
     stop run.
```

Figure 4.6 *CAPTURE Procedures in COBOL*

Using CAPTURE Procedures in SLPash!

The following code fragment illustrates the use of CAPTURE in SPLash!, the native mode SPL compiler. Whatever is on the screen is sent to the line printer.

```
integer Procedure capture (quiet, options, printer, recchars,
                           left'col, right'col, offset'cols);
        value quiet, options, printer, recchars, left'col, right'col, offset'cols;
        logical quiet, options;
        integer left'col, right'col, offset'cols;
        integer printer, recchars;
option variable, intrinsic, native, nocc,       ! no PARM mask!!
                external;


logical procedure Parse'capture'options (itemp, ileft, options,
                                  left'col, right'col, offset'cols);
        value itemp, ileft;
        integer left'col, right'col, offset'cols;
        integer ileft;
        logical options;
        byte pointer itemp;
  option external, variable, intrinsic, native, nocc;      ! no mask!

...
logical err, options, quiet;
integer left'col, right'col, offset'cols, printer, rec'chars;
byte array buf' (0:79);
...
err  :=  parse'capture'options(buf',move buf' :="COMPRESSED",options
                               left'col, right'col, offset'cols);
capture (quiet, options, printer, rec'chars,
                left'col, right'col, offset'cols);
```

Figure 4.7 *CAPTURE in SPLash*

CAPTURE Error Messages

Each CAPTURE error message is described in the following table.

Table 4.3 *CAPTURE Error Messages*

Message	Cause	Action
Could not open FLAT file	Possible bad file-equation for FLAT.	Check file equation for FLAT with the HP command LISTEQ.
Could not open LPSLP	Possible bad file-equation for LPSLP.	Check file equation for LPSLP with the HP command LISTEQ.

Message	Cause	Action
Error in attempt to turn off echo.	As part of the screen capture process, CAPTURE needs to disable echo on the terminal - temporarily.	Try running CAPTURE again with PARM=4.
Error writing to LPSLP	Possible bad file-equation for LPSLP.	Check file equation for LPSLP with the HP command LISTEQ.
I/O error in reading terminal status	During an **fcontrol(,4,3)** (a three second read) CAPTURE received an error status.	Try running CAPTURE again with PARM=4.
I/O error on read from screen	CAPTURE failed to read a line of text from the screen via an **ESC d** command.	Terminal may not be compatible with HP26xx commands. Try again with PARM=4.
Not a 26xx terminal	CAPTURE determined that the terminal was not compatible with the HP 26xx command set.	Depending on the terminal, running with PARM=4 may allow correct CAPTURE operation.

Message	Cause	Action
Too many empty lines found (more than 99)	CAPTURE remembers how many empty (consecutive) lines it has read. Currently the maximum allowed is 99. CAPTURE does this so that a "runaway" screen capture will not send (possible thousands) unwanted empty lines to LPSLP.	Make sure that the CAPTURE range (for a partial capture) is valid.
Too many lines found (more than 9000)	CPATURE can only screen capture 9000 lines at a time.	Use CAPTURE's partial option to break the screen capture into smaller pieces.
Unknown CAPTURE option:	An invalid option was input.	Make sure that the option is spelled correctly. It may have captured the screen contents anyway and disregarded the invalid option. Try CAPTURE again, using the correct option.

5

THE CHRONOS TOOL

CHRONOS is a library of procedures that manipulate date and time information in a variety of formats. Information can easily be converted from one form to another, including forms that permit arithmetic calculations. It is also possible to increment or decrement time or date values.

CHRONOS supports a date range from year 0 to 4095, offering an immediate solution to "turn of the century" problems.

Operation

CHRONOS is most typically used to translate a date or time from a "stored" format in a data base (i.e., 990331) to a "display" format on a screen or report (i.e., March 31, 1999), or to reformat a date from a data entry field (i.e., 033199) to a "stored" format (i.e., 990331), or to recalculate the amount of time between two events.

CHRONOS can be called like an MPE intrinsic. This means that the user intrinsic file, CHRONOS.INTRIN.LPSTOOLS (in SYSINTR format), should be specified in your source along with the CHRONOS intrinsic declaration. Parameter specifications are used to further define the operation. Therefore calling CHRONOS boils down to determining what kind of operation you want performed is specified in the CHRONOS **mode** parameter. There are literally hundreds of possible configurations that you can specify. Appendix H, "CHRONOS Modes", provides a comprehensive listing of all modes.

Because CHRONOS provides so many conventions, not all parameters may be required for each call. Parameter omission is language dependent, and you should consult your language documentation for details. *HP C/iX*, *HP Pascal/iX*, and *SPLash*! all use the comma to omit parameters. For ANSI-C compatibility, use the keyword "NULL" to omit parameters.

The examples provided show you how to handle parameter specification. The syntax example show you the ordering sequence of the parameters and the data type for each parameter.

Date and Time Formats

CHRONOS supports several date and time formats:

chronos-stamp

CHRONOS supports an internal format called **chronos-stamp**. The **chronos-stamp** is a 6-byte time field with millisecond precision. For example, the 6-byte **chronos-stamp** for January 28, 1993 at precisely 4:38:00.1269 p.m. is (in hexadecimal) $7C90E4270F5 (see "CHRONOS_STAMP" on page 92, for a bit-level description).

Gregorian (formatted)

The formatted Gregorian date and time uses 8 byes of storage. The field separator for the date defaults to the slash (/). The field separator for the time defaults to the colon (:). The standard US formatting for the date and time for the last day of 1999 at noon would look like 12/31/99 12:00:00. You may choose any symbol as a field separator when a call is made to CHRONOS.

The date can be returned in one of three ways:

```
year-month-day (i.e., 99/12/31)

day-month-year (i.e., 31/12/99)

month-day-year (i.e., 12/31/99)
```

Gregorian (unformatted)

The unformatted Gregorian date and time uses 6 byes of storage. The standard US -style for the unformatted date and time for the last day of 1999 at noon would look like 123199 120000.

The date can be returned in one of three ways:

```
year-month-day (i.e., 991231)

day-month-year (i.e., 311299)

month-day-year (i.e., 123199)
```

Julian

The Julian year is returned in a 2-byte array and is not terminated with any special character. Leading zero digits are padded with ASCII "0"not ASCII spaces. For example, for 1999 the Julian year would return 99. The Julian day of the year is returned in a 3-byte array and is not terminated with any special character. Leading zero digits are padded with ASCII "0" not ASCII spaces. For example, the Julian day for Feb 1 would return 032.

String

CHRONOS provides four ways to format string output:

```
day-month-year

month-day-year

dayname-day-month-year

dayname-month-day-year
```

The length of the string output is always 30 and is not terminated with any special character. Unused characters are set to ASCII spaces:

`day-month-year`	looks like "28 January 1999 "
`month-day-year`	looks like "January 28, 1999 "
`dayname-day-month-year`	looks like "Thursday, 28 January 1999 "
`dayname-month-day-year`	looks like "Thursday, January 28, 1999 "

CHRONOS can convert any of the above formats into any of the other formats. In addition, by specifying an increment, CHRONOS can increment the time or date either forward or backward.

Providing the optional parameter **day_of_week** will cause CHRONOS to return the numerical day of the week, where Sunday=0, Monday=1, and so forth.

Providing the optional parameter **day_of_week** allows the user to change the default century, or to obtain the current century. For example, this parameter returns 1900 currently.

CHRONOS Intrinsic

CHRONOS Intrinsic performs the requested date/time conversion:

```
int chronos (parameter1, parameter2 [,parameter3, ... parameter15])
```

The Parameter Set is listed next where each parameter is either an integer, character array, or byte:

Table 5.1 *CHRONOS Parameter Set*

Parameter	Name	Type	Comment
1	status	integer 32-bit signed	Required
2	mode	integer 32-bit signed	Required
3	chronos_stamp	character array	Optional
4	formatted_date	character array	Optional
5	formatted_time	character array	Optional
6	unformatted_date	character array	Optional
7	unformatted_time	character array	Optional
8	date_symbol	byte	Optional
9	time_symbol	byte	Optional
10	increment	integer 32-bit signed	Optional

Parameter	Name	Type	Comment
11	chronos_string	character array	Optional
12	julian_year	character array	Optional
13	julian_date	character array	Optional
14	day_of_week	integer 32-bit signed	Optional
15	century	integer 32-bit signed	Optional

Return Value

CHRONOS returns a 32-bit integer encoded as follows:

< 0 :*Error*

-1 = bad parameter, check the status variable for more information.

-23 = conversion error, check the status variable for more information.

= 0 :*No error*

> 0 :*Warning* conversion probably worked, the status variable for more information.

Parameters

The following table describes the CHRONOS’s parameters.

Table 5.2 CHRONOS’s Parameters

Parameter	Description
status	Integer by reference (required). Contains the status of the call to CHRONOS. The sign of the return value describes the kind of status where a negative value denotes an error and a positive value denotes a warning. The absolute value of status is the number of the error or warning. A zero value means the call was successful.
mode	Integer by value (required). Contains the bit-encoded directions for the conversion. See “CHRONOS_MODE” on page 90 for complete information.

Parameter	Description
chronos_stamp	Byte array by reference (optional). Contains the 6-byte CHRONOS time and date stamp. See “CHRONOS_STAMP” on page 92 for complete information.
formatted_date	Byte array by references (optional). Contains the 8-byte string that represents the month, day and year in various formats. For example, 03/12/99. The number separator defaults to the slash (/). Use the **date_symbol** parameter to specify an alternate separator symbol.
formatted_time	Byte array by references (optional). Contains the 8-byte string that represents the hour, minute and second formatted as hh:mm:ss where hh is in 24 hour format. The number separator defaults to the colon (:). Use the **time_symbol** parameter to specify an alternate symbol.
unformatted_date	Byte array by references (optional). Contains the same information as **formatted_date**, except that the number separator has been omitted. The length of the array is 6 bytes.
unformatted_time	Byte array by references (optional). Contains the same information as **formatted_time**, except that the number separator has been omitted. The length of the array is 6 bytes.
date_symbol	Byte by value (optional). Contains the single ASCII character that will be used to separate the numbers in the **formatted_date** string.
time_symbol	Byte by value (optional). Contains the single ASCII character that will be used to separate the numbers in the **formatted_time** string.
increment	Byte array by references (optional). Contains the signed value that can be used to add or subtract values from the time or date as specified by the mode parameter.
chronos_string	Byte array by references (optional). The 30-byte array that contains the formatted date string in one of several formats as specified by the mode parameter. For example: Thursday, January 14, 1999.
julian_year	Byte array by references (optional). The 2-byte array that contains the last two digits of the year. For example: "99" for the year 1999.

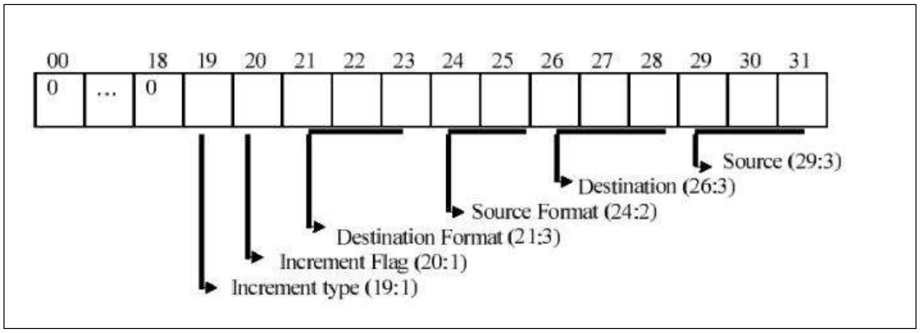
Parameter	Description
julian_date	Byte array by references (optional). The 3-byte array that contains the Julian date of the current year. For example: "312" for the 312th day of the year.
day_of_week	Integer by reference (optional). If provided, this parameter returns the numerical day of the week. The number returned is in the range 0..6 where 0=Sunday, 1=Monday, and so forth.
century	Integer by reference (optional). Can be used to specify the century, or will return the current century if passed in with a value of zero (0).

Operation

This section provides how-to information for two key topics. First, information on how to specify the CHRONOS **mode** parameter is discussed. This section is followed by chronos-stamp specifications.

CHRONOS_MODE

The CHRONOS **mode** parameter is used to specify the type of operation you want performed. The CHRONOS **mode** is a 32-bit integer where bits 0 to 18 should be zero and bits 19 through 31 are encoded as follows:



Each of the encoded bit fields (source, destination, source format, etc.) is discussed next.

Source (29:3) and Destination (26:3) Bit Mapping

000 System Local Time and Date from the CALENDAR intrinsic (Source only)

001	CHRONOS time and date stamp	Required parameter: chronos_stamp
010	Formatted string	Required parameter: formatted_date Optional parameter: formatted_time
011	Unformatted string	Required parameter: unformatted_date Optional parameter: formatted_time
100	Julian date and year	Required parameter: julian_year, julian_date
101	String	Required parameter: chronos_string (Destination only)

Source format (24:2) Bit Mapping

00	MDY	(month, day, year)
01	DMY	(day, month, year)
10	YMD	(year, month, day)

NOTE Only meaningful for formatted string and unformatted string modes.

Destination format (21:3) Bit Mapping

000	MDY	(month, day, year)
001	DMY	(day, month, year)
010	YMD	(year, month, day)

NOTE Only meaningful for formatted string and unformatted string modes.

For example, if the Destination field is **101 (STRING)**, then the Destination format is bit mapped as follows:

000	dayname, monthname, day, year	(i.e., MON, JANUARY 21, 1995)

001	dayname, day, monthname, year	(i.e., MON, 21 JANUARY 1995)
010	monthname, day, year	(i.e., JANUARY 21, 1995)
011	day, monthname, year	(i.e., 21 JANUARY, 1995)

Increment Flag (20:1)

This bit is a flag that is used to determine if a time or date field should be incremented.

0	no increment
1	increment wanted Check bit (19:1) to determine if source time or date increment is desired.

Increment Type (19:1)

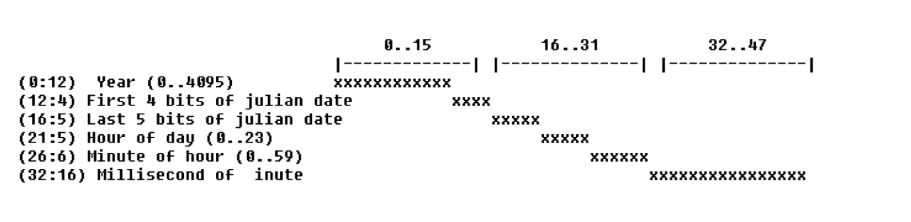
This bit field is used in conjunction with bit field (20:1) and the increment parameter to specify an increment value and type. If the value for this bit field is zero, then the increment parameter contains the number of days to be added or subtracted to the source date. If the value for this bit field is one, then the increment parameter contains the number of minutes to be added or subtracted to the source time.

0	source date increment (in days)
1	source time increment (in minutes)

NOTE Some combination of mode values and parameters can result in superfluous information being passed to CHRONOS. If CHRONOS can detect such a case, a warning will be returned. See Appendix H, "CHRONOS Modes", for a complete list of all mode numbers. Because there are some "don't care" cases, there are several mode numbers that produce the same results.

CHRONOS_STAMP

CHRONOS has a unique format for storing the precise "definition" of a moment in time, including century through millisecond and all components in between. This is accomplished by using a "bit-mapping" technique in a 6-byte field:



```
                                     0..15           16..31           32..47
                                |-------------| |--------------| |--------------|
(0:12)  Year (0..4095)          xxxxxxxxxxxx
(12:4) First 4 bits of julian date          xxxx
(16:5) Last 5 bits of julian date               xxxxx
(21:5) Hour of day (0..23)                           xxxxx
(26:6) Minute of hour (0..59)                             xxxxxx
(32:16) Millisecond of  inute                                   xxxxxxxxxxxxxxxx
```

Figure 5.1 *Defining CHRONOS_STAMP*

When **chronos_stamps** are being stored as data, it may be desirable to zero out all or portions of the time maps. For instance, if the **chronos_stamps** is being used as a Key into a data base record based on date, the time portion would cause multiple entries for the same date to be created.

If Keys are being set up based on the date and time of an entry, for instance in an auditing situation for tracking when data was placed in the data base, the milliseconds might cause multiple entries for the same minute.

CHRONOS Examples

Figure 5.2, Figure 5.3 and Figure 5.4 in this section were compiled with HP's C/iX compiler using the following command statements:

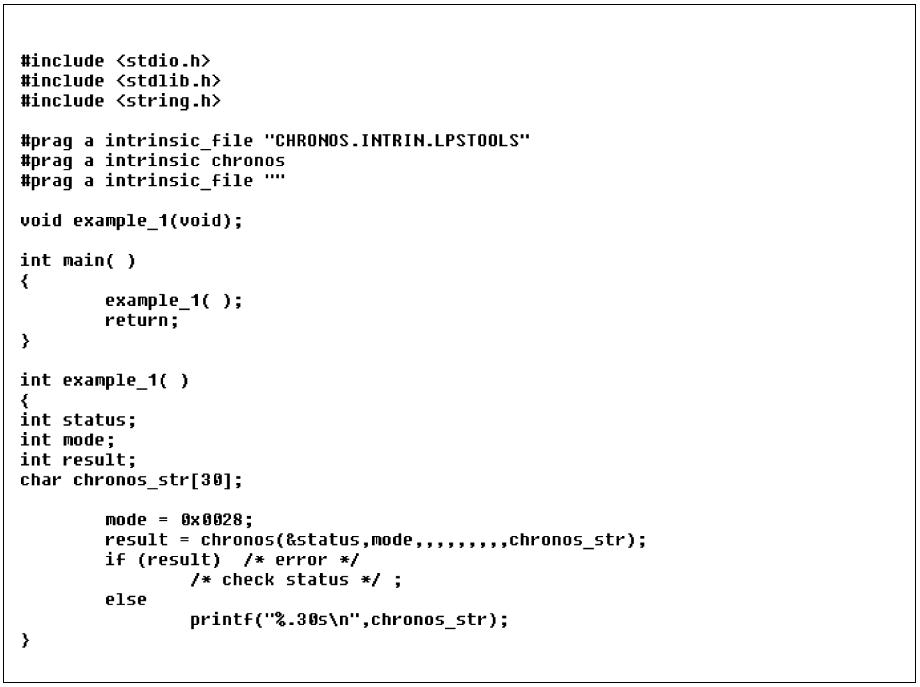
Compile statement:

```
ccxl exam1,,$NULL;info="-Aa -Wc, -e"
```

Link statements (i.e., linking to the RL's **chronos.rl.lpstools** and **libcinit.lib.sys**)

```
:link from=$OLDPASS;TO=exam1.pub;rl=chronos.rl.lpstools,
 libcinit.lib.sys
```

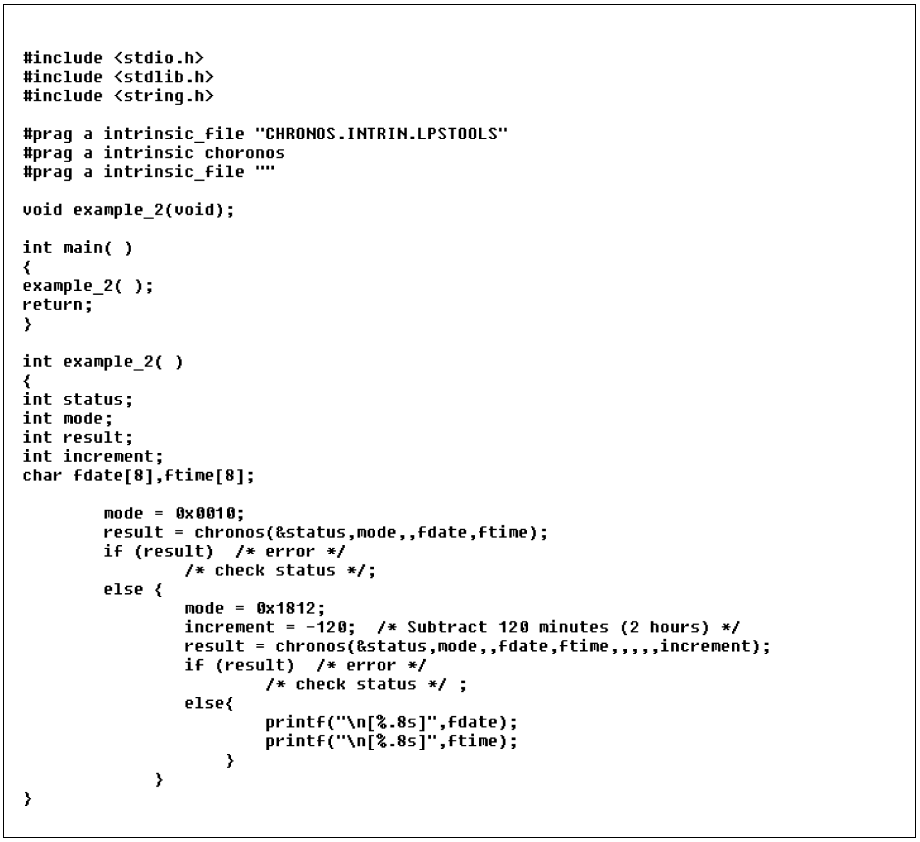
Figure 5.2 shows how CHRONOS will use the system-local date and return the chronos-string in dayname-month-day-year format:



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#prag a intrinsic_file "CHRONOS.INTRIN.LPSTOOLS"
#prag a intrinsic chronos
#prag a intrinsic_file ""

void example_1(void);

int main( )
{
        example_1( );
        return;
}

int example_1( )
{
int status;
int mode;
int result;
char chronos_str[30];

        mode = 0x0028;
        result = chronos(&status,mode,,,,,,,,,chronos_str);
        if (result)  /* error */
                /* check status */ ;
        else
                printf("%.30s\n",chronos_str);
}
```

Figure 5.2 *System-Local Date*

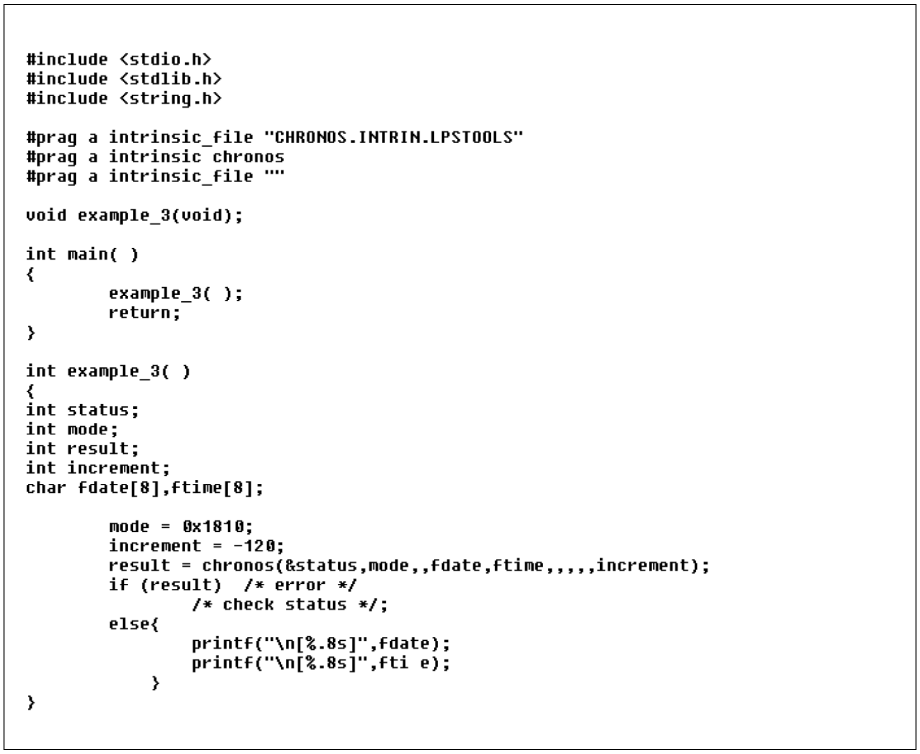
Figure 5.3 is an example of how to call CHRONOS twice, the first time to get the current date and time and return it as formatted date and time. Then call CHRONOS again to subtract 2 hours from the formatted time.



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#prag a intrinsic_file "CHRONOS.INTRIN.LPSTOOLS"
#prag a intrinsic choronos
#prag a intrinsic_file ""

void example_2(void);

int main( )
{
example_2( );
return;
}

int example_2( )
{
int status;
int mode;
int result;
int increment;
char fdate[8],ftime[8];

        mode = 0x0010;
        result = chronos(&status,mode,,fdate,ftime);
        if (result)  /* error */
                /* check status */;
        else {
                mode = 0x1812;
                increment = -120;  /* Subtract 120 minutes (2 hours) */
                result = chronos(&status,mode,,fdate,ftime,,,,,increment);
                if (result)  /* error */
                        /* check status */ ;
                else{
                        printf("\n[%.8s]",fdate);
                        printf("\n[%.8s]",ftime);
                    }
             }
}
```

Figure 5.3 *Calling CHRONOS Twice*

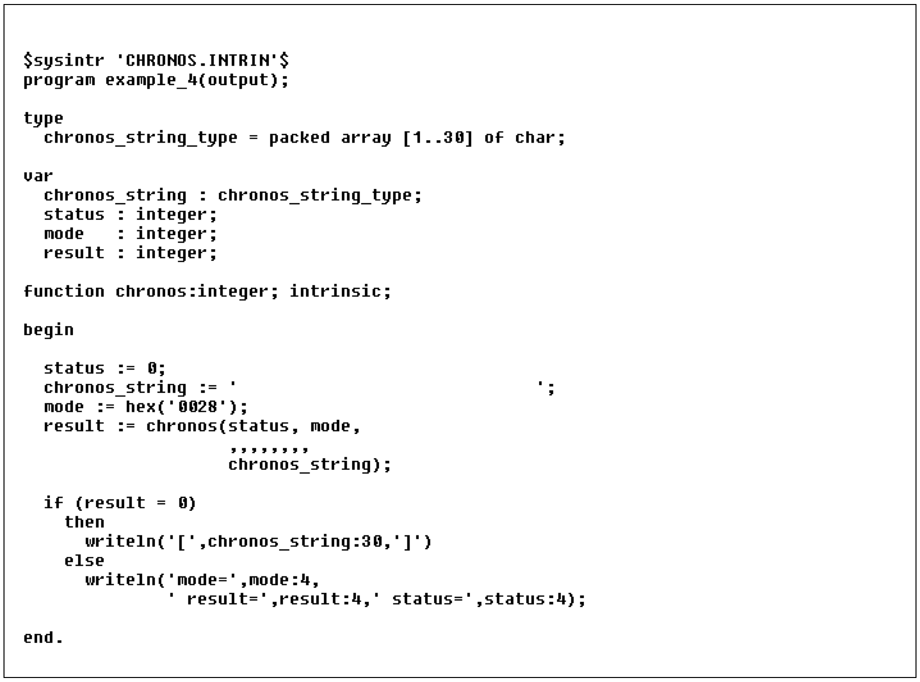
Figure 5.4 is an example of rewriting the previous example to perform the same function with only one call to CHRONOS:



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#prag a intrinsic_file "CHRONOS.INTRIN.LPSTOOLS"
#prag a intrinsic chronos
#prag a intrinsic_file ""

void example_3(void);

int main( )
{
        example_3( );
        return;
}

int example_3( )
{
int status;
int mode;
int result;
int increment;
char fdate[8],ftime[8];

        mode = 0x1810;
        increment = -120;
        result = chronos(&status,mode,,fdate,ftime,,,,,increment);
        if (result)  /* error */
                /* check status */;
        else{
                printf("\n[%.8s]",fdate);
                printf("\n[%.8s]",fti e);
            }
}
```

Figure 5.4 *Calling CHRONOS Once*

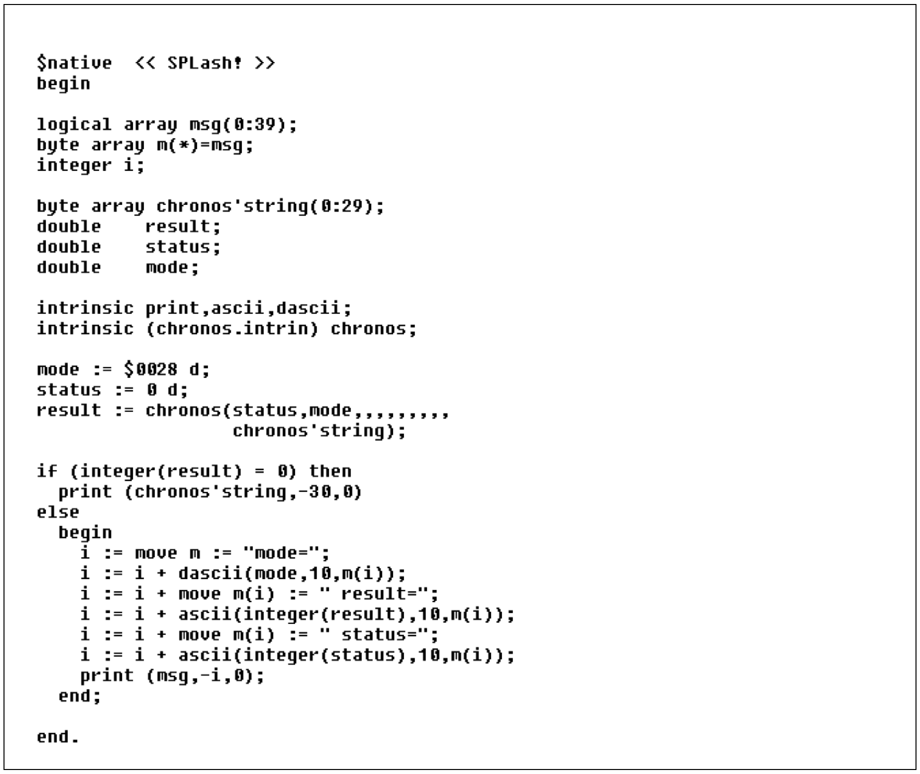
The following figure shows how to call CHRONOS in Pascal (see the **testchro.pascal** file):



```
$sysintr 'CHRONOS.INTRIN'$
program example_4(output);

type
  chronos_string_type = packed array [1..30] of char;

var
  chronos_string : chronos_string_type;
  status : integer;
  mode   : integer;
  result : integer;

function chronos:integer; intrinsic;

begin

  status := 0;
  chronos_string := '                              ';
  mode := hex('0028');
  result := chronos(status, mode,
                    ,,,,,,,,
                    chronos_string);

  if (result = 0)
    then
      writeln('[',chronos_string:30,']')
    else
      writeln('mode=',mode:4,
              ' result=',result:4,' status=',status:4);

end.
```

Figure 5.5 *Pascal Sample Calling CHRONOS*

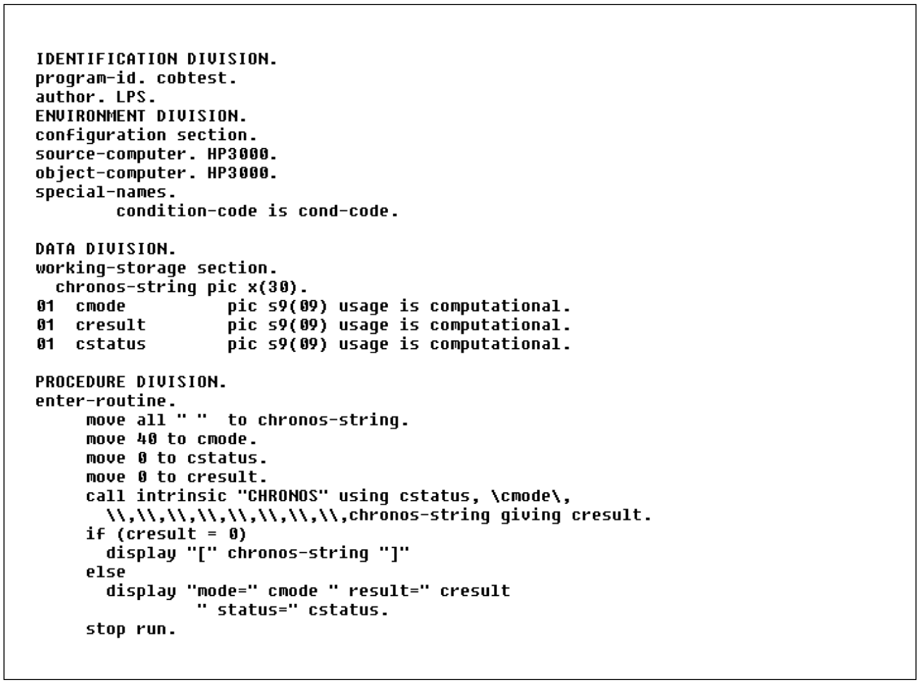
Figure 5.6 shows how to call CHRONOS in SPLash! (See the **testchro.spl** file):



```
$native  << SPLash! >>
begin

logical array msg(0:39);
byte array m(*)=msg;
integer i;

byte array chronos'string(0:29);
double    result;
double    status;
double    mode;

intrinsic print,ascii,dascii;
intrinsic (chronos.intrin) chronos;

mode := $0028 d;
status := 0 d;
result := chronos(status,mode,,,,,,,,,
                  chronos'string);

if (integer(result) = 0) then
  print (chronos'string,-30,0)
else
  begin
    i := move m := "mode=";
    i := i + dascii(mode,10,m(i));
    i := i + move m(i) := " result=";
    i := i + ascii(integer(result),10,m(i));
    i := i + move m(i) := " status=";
    i := i + ascii(integer(status),10,m(i));
    print (msg,-i,0);
  end;

end.
```

Figure 5.6 *SPLash! Sample Calling CHRONOS*

Figure 5.7 shows how to call CHRONOS in COBOL (See the **testchro.cobol** file):



```
IDENTIFICATION DIVISION.
program-id. cobtest.
author. LPS.
ENVIRONMENT DIVISION.
configuration section.
source-computer. HP3000.
object-computer. HP3000.
special-names.
        condition-code is cond-code.

DATA DIVISION.
working-storage section.
  chronos-string pic x(30).
01  cmode          pic s9(09) usage is computational.
01  cresult        pic s9(09) usage is computational.
01  cstatus        pic s9(09) usage is computational.

PROCEDURE DIVISION.
enter-routine.
     move all " "  to chronos-string.
     move 40 to cmode.
     move 0 to cstatus.
     move 0 to cresult.
     call intrinsic "CHRONOS" using cstatus, \cmode\,
       \\,\\,\\,\\,\\,\\,\\,\\,chronos-string giving cresult.
     if (cresult = 0)
       display "[" chronos-string "]"
     else
       display "mode=" cmode " result=" cresult
                " status=" cstatus.
     stop run.
```

Figure 5.7 *COBOL Sample Calling CHRONOS*

CHRONOS Error Messages

In the next table are listed the CHRONOS error message numbers and their respective meaning:

Table 5.3 *CHRONOS Error Messages*

Number	Meaning
2	Source (29:3) not in bit range 000..101
3	Missing source parameter chronos_stamp
4	Missing source parameter formatted_time or formatted_date
5	Missing source parameter unformatted_time or unformatted_date
6	Missing source parameter julian_time or julian_date
7	chronos_string cannot be used as source

Number	Meaning
8	System Local cannot be used as source
9	Missing destination parameter chronos_string
10	Destination (26:3) not in bit range 000..101
11	chronos_string destination format not in bit range 000..011
12	Destination format not in bit range 000..010
13	Source format not in bit range 00..10
14	*NOT USED*
15	Missing destination parameter julian_time or julian_date
16	Missing destination parameter formatted_time or formatted_date
17	Missing destination parameter unformatted_time or unformatted_date
18	Missing destination parameter chronos_stamp
19	Bad source numbers in one or both unformatted parameters
20	Bad source numbers in one or both julian parameters
21	Bad source numbers in one or both formatted parameters
22	Returned when something is wrong with the source or destination parameters (which was initially undetected), causing a conversion error.
Errors 19 - 21	are triggered when the following conditions apply: For unformatted, formatted, or Julian conversions, these errors result when the numbers are not in range or are not formatted correctly. The CHRONOS function will return ASCII zeros in the destination field.
Error 22	is returned by CHRONOS when it finds a source or destination field that it does not understand.

6

THE CSEQ TOOL

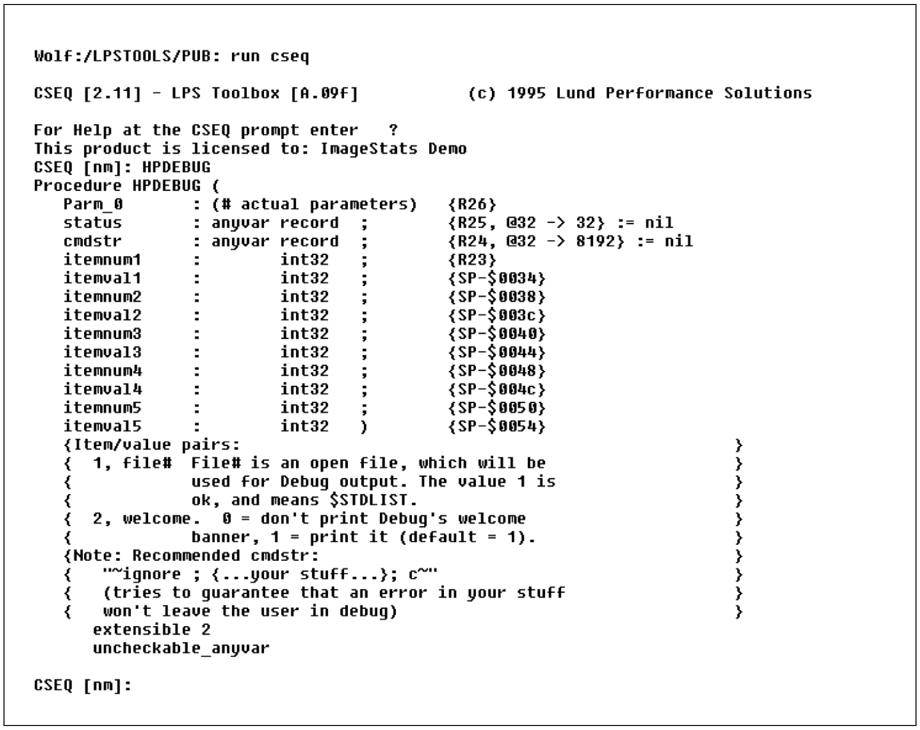
CSEQ reports the calling sequence of intrinsics. The intrinsic may be a Native Mode intrinsic, a Compatibility Mode intrinsic, or both. Also CSEQ can report on user-defined intrinsic files via the SPLINTR or SYSINTR commands.

Operation

CSEQ is used to display Native Mode and Compatibility Mode intrinsic calling sequences as defined by either the SYSINTR or SPLINTR files. The defaults startup condition for CSEQ assumes that the user is interested in reporting on Native Mode intrinsics from SYSINTR.PUB.SYS, AIFINTR.PUB.SYS, PEINTR.PE.SYS, or SPLINTR.PUB.SYS. At that point it is simply a matter of entering the name of the intrinsic for which you are interested. See the sample output provided next for an illustration on how this works.

Native Mode Output

When CSEQ is asked to display the calling sequence of a Native Mode intrinsic, it generates output like following example.



```
Wolf:/LPSTOOLS/PUB: run cseq

CSEQ [2.11] - LPS Toolbox [A.09f]           (c) 1995 Lund Performance Solutions

For Help at the CSEQ prompt enter   ?
This product is licensed to: ImageStats Demo
CSEQ [nm]: HPDEBUG
Procedure HPDEBUG (
   Parm_0       : (# actual parameters)   {R26}
   status       : anyvar record  ;        {R25, @32 -> 32} := nil
   cmdstr       : anyvar record  ;        {R24, @32 -> 8192} := nil
   itemnum1     :        int32   ;        {R23}
   itemval1     :        int32   ;        {SP-$0034}
   itemnum2     :        int32   ;        {SP-$0038}
   itemval2     :        int32   ;        {SP-$003c}
   itemnum3     :        int32   ;        {SP-$0040}
   itemval3     :        int32   ;        {SP-$0044}
   itemnum4     :        int32   ;        {SP-$0048}
   itemval4     :        int32   ;        {SP-$004c}
   itemnum5     :        int32   ;        {SP-$0050}
   itemval5     :        int32   )        {SP-$0054}
   {Item/value pairs:                                                   }
   {  1, file#  File# is an open file, which will be                    }
   {            used for Debug output. The value 1 is                   }
   {            ok, and means $STDLIST.                                 }
   {  2, welcome.  0 = don't print Debug's welcome                      }
   {            banner, 1 = print it (default = 1).                     }
   {Note: Recommended cmdstr:                                           }
   {   "~ignore ; {...your stuff...}; c~"                               }
   {   (tries to guarantee that an error in your stuff                  }
   {   won't leave the user in debug)                                   }
      extensible 2
      uncheckable_anyvar

CSEQ [nm]:
```

Figure 6.1 *Native Mode Intrinsic Calling Sequence*

The first line of output means that the intrinsic HPDEBUG is in the SYSINTR file in UPPERCASE. If the procedure name had been reported in lowercase then that would be the exact name of the procedure. When you enter a procedure name in CSEQ, it first tries uppercase and then lowercase automatically.

For HPDEBUG, CSEQ noticed that it was an untyped-procedure. If it had a type (i.e., integer) then it would report it as a **function... :integer**.

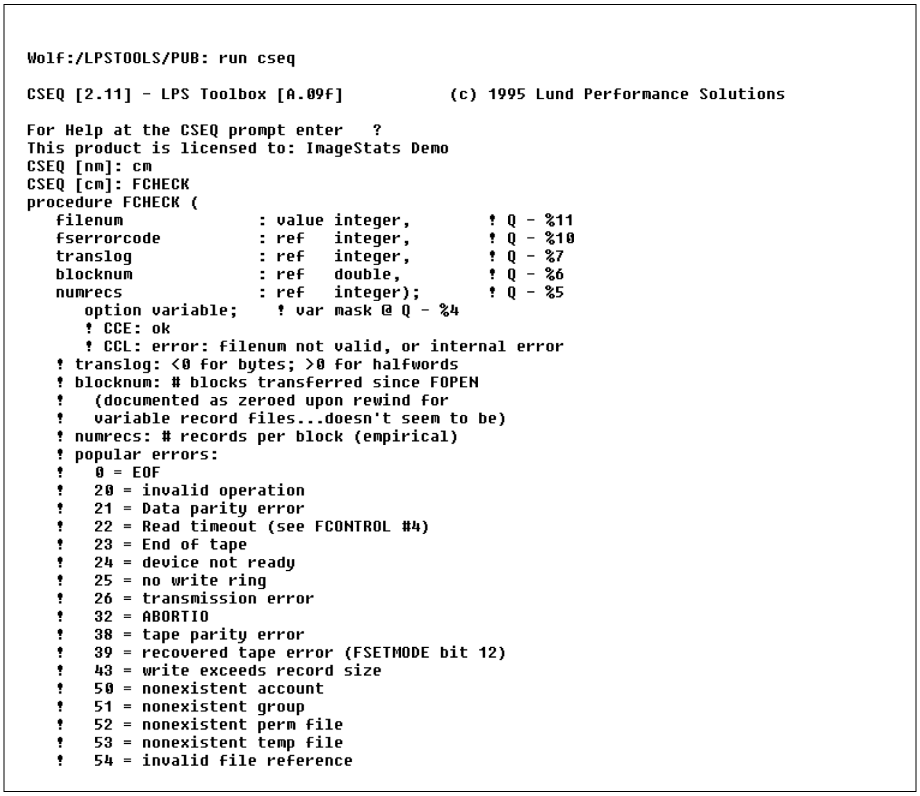
After reporting all of the parameters, CSEQ reports general information about the intrinsic. The intrinsic marked as **extensible 2** and **uncheckable anyvar**. These are explained below:

extensible 2	The intrinsic must be called with at least the first two parameters and the number of actual parameters is passed in as a hidden value in register R26.

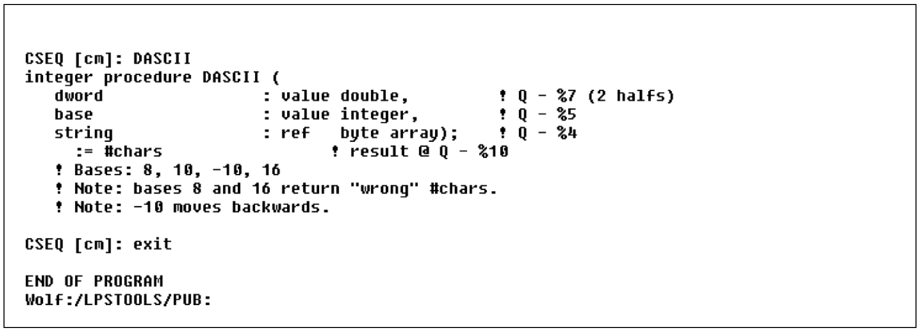
uncheckable anyvar	Any parameters declared as **anyvar** normally have a hidden size parameter passed in just after the actual parameter. **Uncheckable anyvar** means that no hidden size parameters are passed in. If this intrinsic had not been "uncheckable" then it would have reported the location of the hidden size parameters.

Compatibility Mode Output

When CSEQ is asked to display the calling sequence of a compatibility mode intrinsic, it generates output like the following example.



```
Wolf:/LPSTOOLS/PUB: run cseq

CSEQ [2.11] - LPS Toolbox [A.09f]          (c) 1995 Lund Performance Solutions

For Help at the CSEQ prompt enter   ?
This product is licensed to: ImageStats Demo
CSEQ [nm]: cm
CSEQ [cm]: FCHECK
procedure FCHECK (
   filenum              : value integer,        ! Q - %11
   fserrorcode          : ref   integer,        ! Q - %10
   translog             : ref   integer,        ! Q - %7
   blocknum             : ref   double,         ! Q - %6
   numrecs              : ref   integer);       ! Q - %5
      option variable;    ! var mask @ Q - %4
      ! CCE: ok
      ! CCL: error: filenum not valid, or internal error
   ! translog: <0 for bytes; >0 for halfwords
   ! blocknum: # blocks transferred since FOPEN
   !   (documented as zeroed upon rewind for
   !   variable record files...doesn't seem to be)
   ! numrecs: # records per block (empirical)
   ! popular errors:
   !   0 = EOF
   !   20 = invalid operation
   !   21 = Data parity error
   !   22 = Read timeout (see FCONTROL #4)
   !   23 = End of tape
   !   24 = device not ready
   !   25 = no write ring
   !   26 = transmission error
   !   32 = ABORTIO
   !   38 = tape parity error
   !   39 = recovered tape error (FSETMODE bit 12)
   !   43 = write exceeds record size
   !   50 = nonexistent account
   !   51 = nonexistent group
   !   52 = nonexistent perm file
   !   53 = nonexistent temp file
   !   54 = invalid file reference
```



```
CSEQ [cm]: DASCII
integer procedure DASCII (
   dword               : value double,        ! Q - %7 (2 halfs)
   base                : value integer,       ! Q - %5
   string              : ref   byte array);   ! Q - %4
     := #chars                ! result @ Q - %10
   ! Bases: 8, 10, -10, 16
   ! Note: bases 8 and 16 return "wrong" #chars.
   ! Note: -10 moves backwards.

CSEQ [cm]: exit

END OF PROGRAM
Wolf:/LPSTOOLS/PUB:
```

Figure 6.2 *Compatibility Mode Intrinsic Calling Sequence*

All **Q-** addresses are valid as of the start of the intrinsic. Since most parameters are one halfword in size (16-bits), CSEQ doesn't list their size. Instead, only those parameters that are larger than one halfword are flagged with a size, as in Parm 1 in DASCII.

In the above example, FCHECK is an untyped procedure and DASCII is type **integer** (returns a 16-bit value). Since DASCII returns a result, the stack storage location for the result is shown.

After all of the parameters, if any, CSEQ reports general information about the intrinsic. FCHECK was marked as "option variable", which means it has a parameter mask at Q-4. Option variable procedures with more than 16 parameters have a two-halfword parameter mask stored at Q-5 and Q-4.

NOTE For some intrinsics, CSEQ displays detailed parameter information. FFILEINFO, for instance, has an additional 100 lines of itemnum information that can be displayed after the normal parameter list information. If you want the itemnum information only, precede the intrinsic name with a plus (+) sign. For example: **+ffileinfo**

Capabilities

Program capabilities required include IA, BA, DS, and PH. No special user capabilities are required to run CSEQ.

Usage

CSEQ can be run via the supplied UDC or with the MPE RUN statement. CSEQ can accept input through the INFO string parameter or directly from the user in query mode.

- UDC

```
CSEQ [<commands | [+] intrinsics>]
```

- RUN

```
RUN CSEQ.PUB.LPSTOOLS;INFO="[<commands | [+] intrinsics>]"
```

Command Summary

The following table provides a simple description of CSEQ commands that you can use to quickly locate the command that suits the task at hand.

NOTE Portions of command codes are printed in uppercase to denote the part of the command that CSEQ requires in order to distinguish one command from another.

Table 6.1 *CSEQ Commands*

Command Code	Description
ALL	Lists all matching intrinsics for the current mode (CM, NM or BOTH).
ALLCM	Displays all CM intrinsics of a class
ALLNM	Displays all NM intrinsics of a class
BOTH	Displays both NM & CM intrinsics information
CLOSE	Closes a SYSINTR, SPLINTR, or file #
CM	Displays CM intrinsic information only
Exit	Terminates CSEQ
HELP	Invokes CSEQ Help
NM	Displays NM intrinsic information only
SET/REset	Enables and disables options
SPLINTR	Opens an MPE V intrinsics file
STATUS	Displays information about currently opened files
SYSINTR	Opens an MPE/iX intrinsics file
//	Synonym for EXIT
?	Synonym for HELP

Command Definitions

This section describes CSEQ commands in detail.

ALL

The ALL command provides a means for listing all intrinsics or all intrinsics that have a common prefix. ALL will list all matching intrinsics for the current mode (CM, NM, or BOTH).

ALLCM

This command has the following syntax:

```
ALLCM [intrinsic name]
```

This CSEQ command will display parameter information for the specified class of intrinsics. If an **intrinsic name** is not specified, then all of the Compatibility Mode intrinsics will be displayed. Partial names can be specified to display a class of intrinsics.

NOTE The puls (+) option, which displays **itemnum** information only, is not available for this command, but the minus (-) option, which disables extra information displays is available.

For example: **ALLCM MY** could be used to display all of the intrinsics that start with the letters **MY**.

ALLNM

This command has the following syntax:

```
ALLNM [intrinsic name]
```

This CSEQ command will display parameter information for the specified class of intrinsics. If an **intrinsic name** is not specified, then all of the Native Mode intrinsics will be displayed. Partial names can be specified to display a class of intrinsics.

NOTE The puls (+) option, which displays **itemnum** information only, is not available for this command, but the minus (-) option, which disables extra information displays is available.

For example: **ALLNM HP** could be used to display all of the intrinsics that start with the letters **HP**.

BOTH

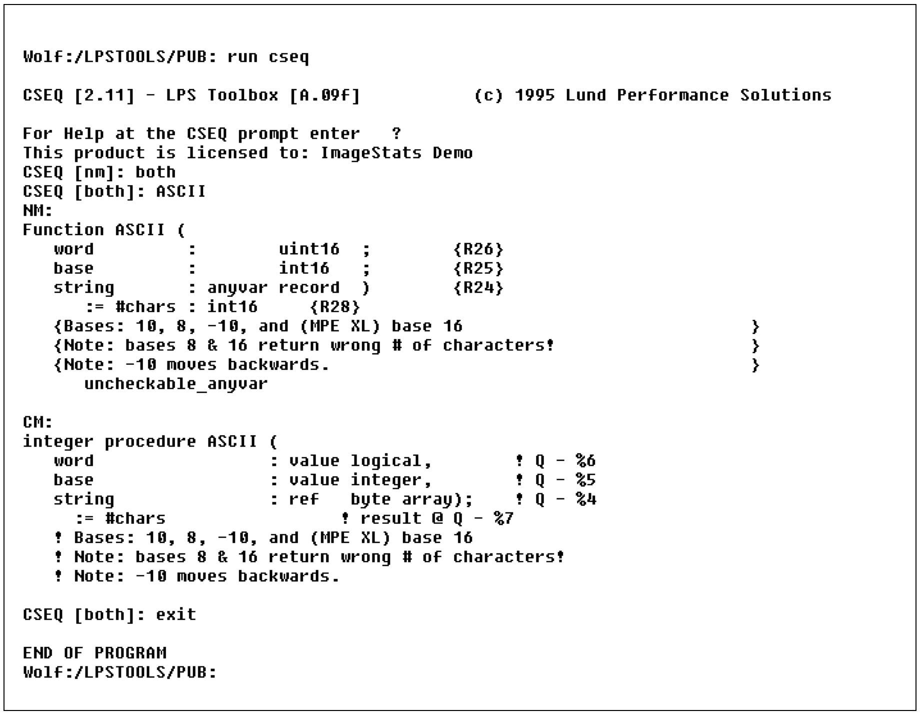
This command has the following syntax:

```
BOTH [intrinsic name]
```

The BOTH command tells CSEQ to display the calling sequence for both Native Mode and Compatibility Mode intrinsics.

NOTE The puls (+) option, which displays **itemnum** information only, is not available for this command, but the minus (-) option, which disables extra information displays is available.

After issuing BOTH, entering the intrinsic name ASCII would result in:



```
Wolf:/LPSTOOLS/PUB: run cseq

CSEQ [2.11] - LPS Toolbox [A.09f]          (c) 1995 Lund Performance Solutions

For Help at the CSEQ prompt enter   ?
This product is licensed to: ImageStats Demo
CSEQ [nm]: both
CSEQ [both]: ASCII
NM:
Function ASCII (
   word         :        uint16  ;        {R26}
   base         :        int16   ;        {R25}
   string       : anyvar record  )        {R24}
      := #chars : int16      {R28}
   {Bases: 10, 8, -10, and (MPE XL) base 16                         }
   {Note: bases 8 & 16 return wrong # of characters!                }
   {Note: -10 moves backwards.                                      }
      uncheckable_anyvar

CM:
integer procedure ASCII (
   word                 : value logical,        ! Q - %6
   base                 : value integer,        ! Q - %5
   string               : ref   byte array);    ! Q - %4
     := #chars                 ! result @ Q - %7
   ! Bases: 10, 8, -10, and (MPE XL) base 16
   ! Note: bases 8 & 16 return wrong # of characters!
   ! Note: -10 moves backwards.

CSEQ [both]: exit

END OF PROGRAM
Wolf:/LPSTOOLS/PUB:
```

Figure 6.3 *BOTH Command Screen*

CLOSE

This command has the following syntax:

```
CLOSE ["sysintr" | "splintr" | <file#>]
```

The BOTH command can be used to limit CSEQ's intrinsic file scan. By default (on NM machines) SYSINTR.PUB.SYS, AIFINTR.PUB.SYS, PEINTR.PE.SYS, and SPLINTR.PUB.SYS are scanned. See the STATUS command to determine file numbers <file#> for use with this command.

CM

The CM command tells CSEQ that you now want to see the calling sequence for Compatibility Mode intrinsics. If CM is followed by an intrinsic name, it is looked up immediately.

Exit (or //)

The Exit command terminates CSEQ.

HELP (or ?)

The HELP command invokes the CSEQ help facility.

Help on a specific command is available by typing:

```
? commandname
```

or

```
HELP commandname
```

NM

The NM command tells CSEQ that you now want to see the calling sequence for Native Mode intrinsics.

SET | RESET

These commands have the following syntaxes:

```
SET option
RESET option
```

The SET/RESET commands are used to turn options on or off.

An option can be set by entering: **SET optionname**

An option can be reset by entering: **RESET optionname** or: **SET NOoptionname**

Options are described in the next table:

Table 6.2 *SET / RESET Options*

Option	Description
ALLSIZES	Tells CSEQ to report the size of every parameter, in bits. Normally, CSEQ reports only the sizes of selected parameter types.
C	Tells CSEQ that you want to see intrinsic headers in C language style.
CASEsensitivity	Tells CSEQ to look for intrinsics in a case sensitive manner. Normally, this should not be necessary.
CM	Tells CSEQ you want to see Compatibility Mode intrinsics, not Native Mode (NM) intrinsics.
CSEQDATA	Tells CSEQ that the ALLNM command should search the CSEQ.DATA file for extra "intrinsics" (i.e., printf). Default: SET CSEQDATA
EXTRAS	Tells CSEQ that you want to see extra comments about intrinsics. SET NOEXTRAS suppresses the extra comments. Default: SET EXTRAS
EXTRASONLY	Tells CSEQ that you want to see only extra comments above intrinsics, and nothing about the actual calling sequence. This is useful for intrinsics with many parameters, like HPFOPEN. Default: RESET EXTRASONLY (SET NOEXTRASONLY)
GCC	Tells CSEQ that you want to see intrinsic headers in gcc language style.
LANGuage	Tells CSEQ to report the language that each NM intrinsic is written in. The language is shown as a number, not as a name. Default: RESET LANG (SET NOLANG) (because all NM intrinsics report that they are written in Pascal/XL as of MPE/iX 4.0)
MACRO	Tells CSEQ to emit a Debug/XL macro to show the parameters of each subsequent intrinsic. Default: RESET MACRO
NM	Tells CSEQ you want to see Native Mode intrinsics, not Compatibility Mode (CM) intrinsics.

Option	Description
PARMS	When reset, tells CSEQ to list only the names of intrinsics, and not any parameters or functional results. This is most useful in conjunction with the ALL command. When set, CSEQ lists the parameters of intrinsics. Default: SET PARMS
PARMTRUNC	If SET, tells CSEQ that if it sees a parameter name beginning with "...", that it should skip the rest of the parameters for the intrinsic. (As delivered, CSEQ.DATA has two instances of such parameters, in HPFOPEN and HPDEVCREATE.) Default: SET PARMTRUNC
PE	Tells CSEQ to report addresses of NM intrinsic parameters are relative to the Procedure Exit parameter data structure. **NOTE** Setting PE implicitly does a RESET MACRO.
PLUSPLUS	Tells CSEQ that intrinsics with extra documentation are interesting. (Obscure)
SORT	Tells CSEQ to sort the list of NM intrinsics found in an ALL command. CM intrinsic names are not sorted. Default: SET SORT
UNNAMED	Tells CSEQ to only list intrinsics that have no names for their parameters. This is usually used with the ALL command, and is intended as an internal debugging tool. Default: RESET UNNAMED

SPLINTR

This command has the following syntax:

```
SPLINTR filename
```

The SPLINTR command tells CSEQ that you now want to look for Compatibility Mode intrinsics in a different intrinsic file. CSEQ opens the specified CM intrinsic file. If the new file cannot be opened, CSEQ will report an error and revert to SPLINTR.PUB.SYS. Also does an implied CM command.

Example: `splintr splintrx.pub.splash` Closes the current SPLINTR file (if any) and opens SPLINTR.PUB.SPLASH. Sets mode to CM.

STATUS

This command is used to display a small report about which intrinsic files are being used.

SYSINTR

This command has the following syntax:

```
SYSINTR filename
```

The SYSINTR command tells CSEQ that you now want to look for Native Mode intrinsics in a different intrinsic file. CSEQ opens the specified NM intrinsic file. If the new file cannot be opened, CSEQ will report an error and revert to SYSINTR.PUB.SYS. Also does an implied NM command.

Example 1: `sysintr aifintr.pub.sys` switches to SYSINTR format file **AIFINTR.PUB.SYS**. Sets mode to NM.

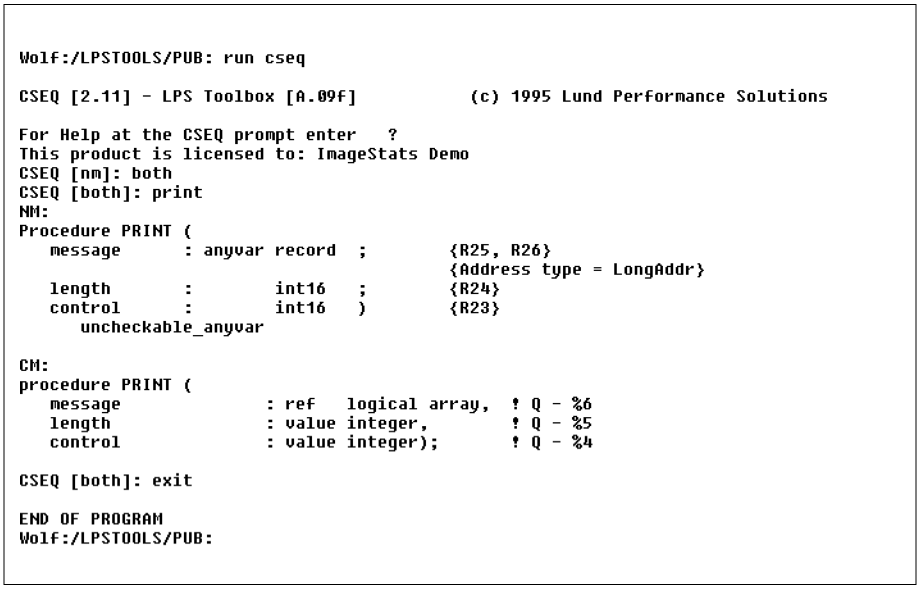
Example 2: `allnm` lists all intrinsics in current sysintr file (**AIFINTR.PUB.SYS**).

TOOLBOX STANDARDS

The ToolBox collections from Lund Performance Solutions have a uniform user interface. As a result, in addition to the commands specific to each Toolbox tool, most tools allow the commands described in “TOOLBOX STANDARDS” on page 213.

CSEQ Examples

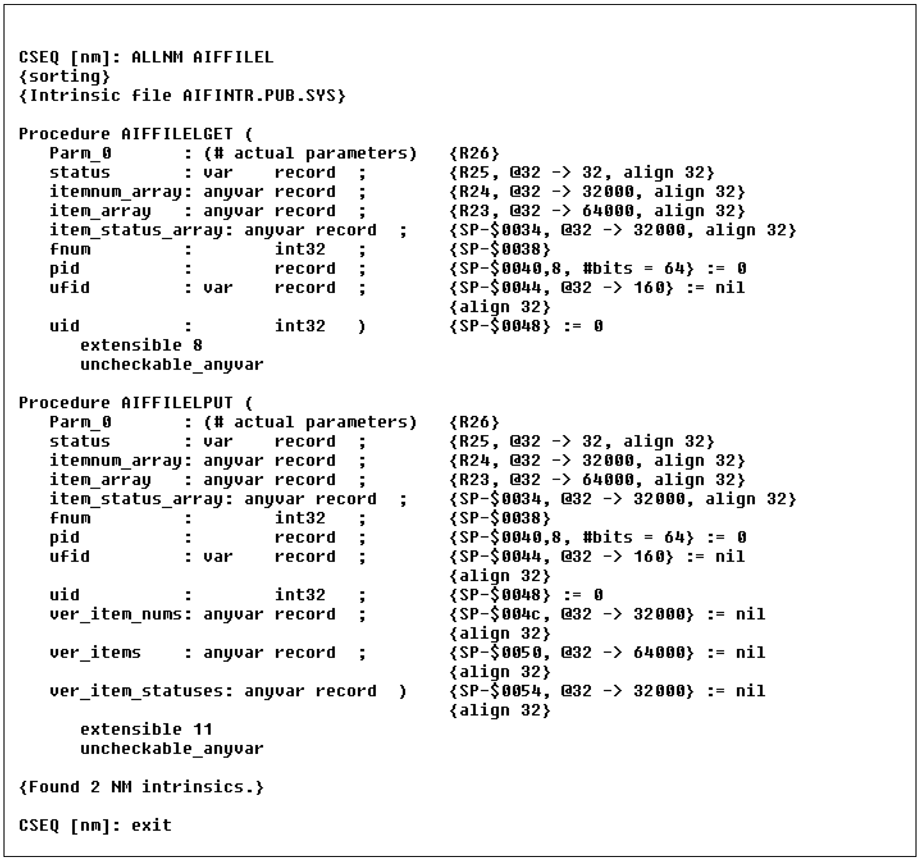
Following are some examples of the information discussed in the previous sections.



```
Wolf:/LPSTOOLS/PUB: run cseq

CSEQ [2.11] - LPS Toolbox [A.09f]           (c) 1995 Lund Performance Solutions

For Help at the CSEQ prompt enter   ?
This product is licensed to: ImageStats Demo
CSEQ [nm]: both
CSEQ [both]: print
NM:
Procedure PRINT (
   message      : anyvar record  ;        {R25, R26}
                                          {Address type = LongAddr}
   length       :        int16   ;        {R24}
   control      :        int16   )        {R23}
      uncheckable_anyvar

CM:
procedure PRINT (
   message              : ref   logical array,  ! Q - %6
   length               : value integer,        ! Q - %5
   control              : value integer);       ! Q - %4

CSEQ [both]: exit

END OF PROGRAM
Wolf:/LPSTOOLS/PUB:
```

Figure 6.4 *CSEQ Output Using the Both Option*

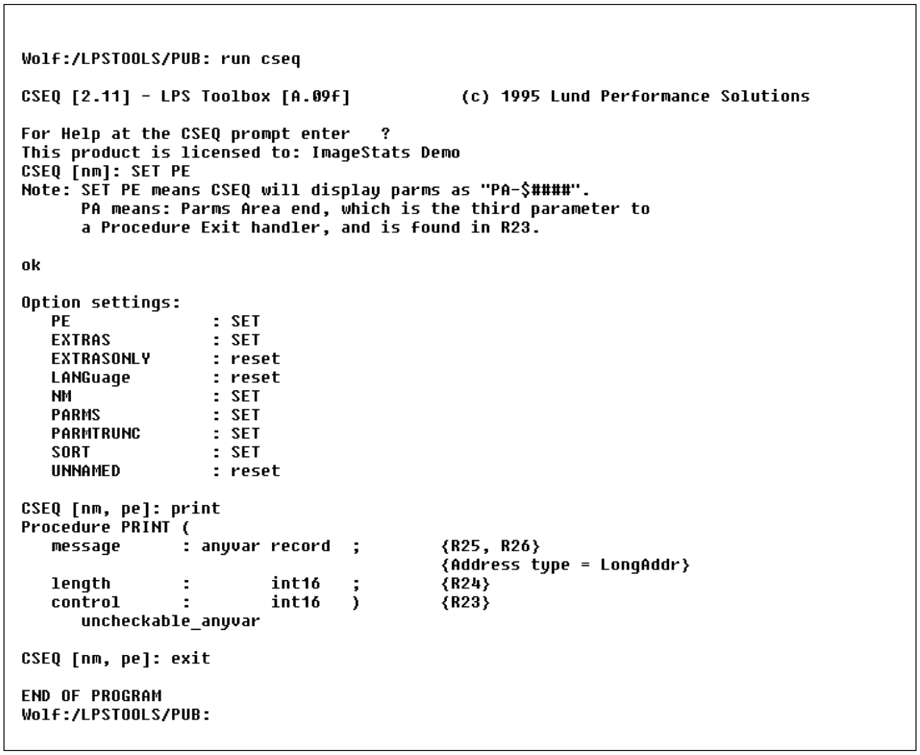
Figure 6.5 shows how CSEQ’s **ALLNM** command works. If a partial intrinsic name is given, then all intrinsics that match that partial description are displayed. In this example, two intrinsics matched the partial description.



```
CSEQ [nm]: ALLNM AIFFILEL
{sorting}
{Intrinsic file AIFINTR.PUB.SYS}

Procedure AIFFILELGET (
   Parm_0       : (# actual parameters)   {R26}
   status       : var    record  ;        {R25, @32 -> 32, align 32}
   itemnum_array: anyvar record  ;        {R24, @32 -> 32000, align 32}
   item_array   : anyvar record  ;        {R23, @32 -> 64000, align 32}
   item_status_array: anyvar record  ;    {SP-$0034, @32 -> 32000, align 32}
   fnum         :        int32   ;        {SP-$0038}
   pid          :        record  ;        {SP-$0040,8, #bits = 64} := 0
   ufid         : var    record  ;        {SP-$0044, @32 -> 160} := nil
                                          {align 32}
   uid          :        int32   )        {SP-$0048} := 0
      extensible 8
      uncheckable_anyvar

Procedure AIFFILELPUT (
   Parm_0       : (# actual parameters)   {R26}
   status       : var    record  ;        {R25, @32 -> 32, align 32}
   itemnum_array: anyvar record  ;        {R24, @32 -> 32000, align 32}
   item_array   : anyvar record  ;        {R23, @32 -> 64000, align 32}
   item_status_array: anyvar record  ;    {SP-$0034, @32 -> 32000, align 32}
   fnum         :        int32   ;        {SP-$0038}
   pid          :        record  ;        {SP-$0040,8, #bits = 64} := 0
   ufid         : var    record  ;        {SP-$0044, @32 -> 160} := nil
                                          {align 32}
   uid          :        int32   ;        {SP-$0048} := 0
   ver_item_nums: anyvar record  ;        {SP-$004c, @32 -> 32000} := nil
                                          {align 32}
   ver_items    : anyvar record  ;        {SP-$0050, @32 -> 64000} := nil
                                          {align 32}
   ver_item_statuses: anyvar record  )    {SP-$0054, @32 -> 32000} := nil
                                          {align 32}
      extensible 11
      uncheckable_anyvar

{Found 2 NM intrinsics.}

CSEQ [nm]: exit
```

Figure 6.5 *ALLNM Command*

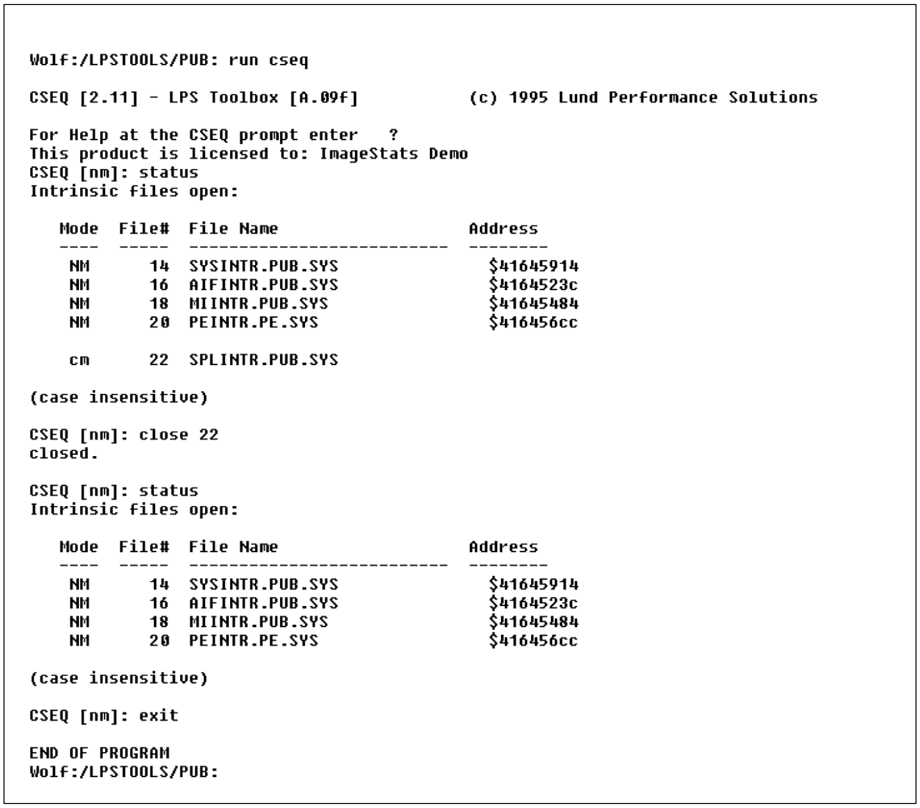
Figure 6.6 shows how the **SET PE** command affects CSEQ's NM output. When enabled, this command is used to display an intrinsic's parameters as offsets from the parameter area of a procedure exit handler.



```
Wolf:/LPSTOOLS/PUB: run cseq

CSEQ [2.11] - LPS Toolbox [A.09f]           (c) 1995 Lund Performance Solutions

For Help at the CSEQ prompt enter   ?
This product is licensed to: ImageStats Demo
CSEQ [nm]: SET PE
Note: SET PE means CSEQ will display parms as "PA-$####".
      PA means: Parms Area end, which is the third parameter to
      a Procedure Exit handler, and is found in R23.

ok

Option settings:
   PE               : SET
   EXTRAS           : SET
   EXTRASONLY       : reset
   LANGuage         : reset
   NM               : SET
   PARMS            : SET
   PARMTRUNC        : SET
   SORT             : SET
   UNNAMED          : reset

CSEQ [nm, pe]: print
Procedure PRINT (
   message      : anyvar record  ;        {R25, R26}
                                          {Address type = LongAddr}
   length       :        int16   ;        {R24}
   control      :        int16   )        {R23}
      uncheckable_anyvar

CSEQ [nm, pe]: exit

END OF PROGRAM
Wolf:/LPSTOOLS/PUB:
```

Figure 6.6 *SET PE Command*

Figure 6.7 shows how to use the status and close commands.



```
Wolf:/LPSTOOLS/PUB: run cseq

CSEQ [2.11] - LPS Toolbox [A.09f]           (c) 1995 Lund Performance Solutions

For Help at the CSEQ prompt enter   ?
This product is licensed to: ImageStats Demo
CSEQ [nm]: status
Intrinsic files open:

   Mode  File#  File Name                   Address
   ----  -----  --------------------------  --------
    NM      14  SYSINTR.PUB.SYS               $41645914
    NM      16  AIFINTR.PUB.SYS               $4164523c
    NM      18  MIINTR.PUB.SYS                $41645484
    NM      20  PEINTR.PE.SYS                 $416456cc

    cm      22  SPLINTR.PUB.SYS

(case insensitive)

CSEQ [nm]: close 22
closed.

CSEQ [nm]: status
Intrinsic files open:

   Mode  File#  File Name                   Address
   ----  -----  --------------------------  --------
    NM      14  SYSINTR.PUB.SYS               $41645914
    NM      16  AIFINTR.PUB.SYS               $4164523c
    NM      18  MIINTR.PUB.SYS                $41645484
    NM      20  PEINTR.PE.SYS                 $416456cc

(case insensitive)

CSEQ [nm]: exit

END OF PROGRAM
Wolf:/LPSTOOLS/PUB:
```

Figure 6.7 *STATUS and CLOSE Commands*

CSEQ Error Messages

Table 6.3 *CSEQ Error Messages*

Message	Cause	Action
CSEQ does not have an open command	User accidentally typed "open"	Use **SYSINTR filename** or **SPLINTR filename**

7

THE EZHELP TOOL

EZHELP is a file-browsing tool for MPE HELP catalogs that brings the advantages of terminal-based windowing to these standard information resources. EZHELP is a dual-purposed application. First, EZHELP functions as a windowed replacement for MPE HELP. Popup, scrollable windows contain lists of topics and items for easy information retrieval. Related information is easily accessed for any MPE command. You can pop up window displays on **Examples**, **Parms**, and **Operations** in just a few keystrokes.

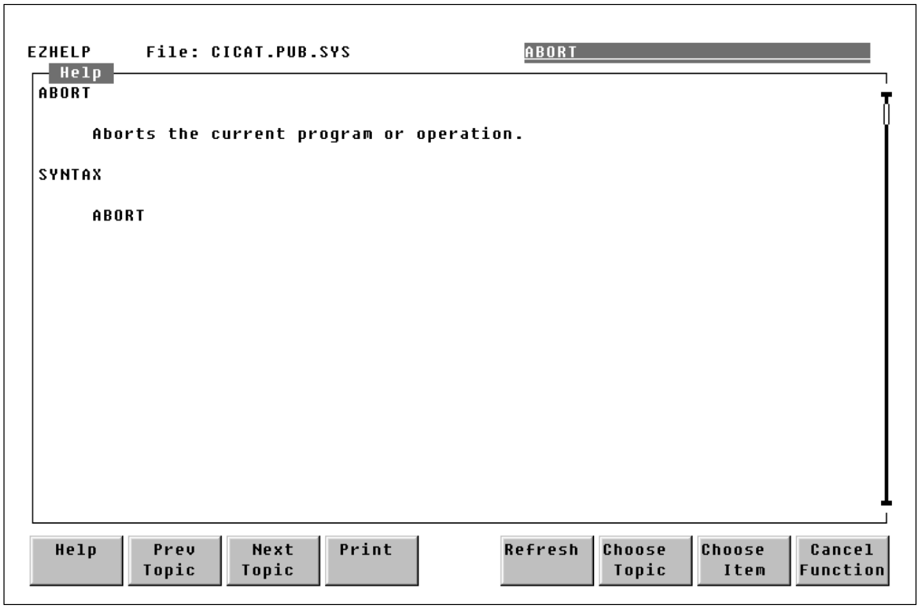
EZHELP also offers a way to interactively view any other help catalog you may have on your system, delivering the same kind of look and feel interface to these information resources that you find with the system help catalog running under EZHELP. Because these files follow a standard structure, EZHELP is able to read the structure and dynamically arrange the information into a format that can be used in a windowing environment.

NOTE In this document, the "system help file" and "HELP catalog" refer to the MPE HELP formatted files. Refer to the HP manual entitled *Message Catalogs Programmer’s Guide* for more detail.

About HELP Catalogs

HELP CATALOGS follow a certain syntax that arranges information according to a set structure. In general, information is grouped according to whether it is an Entry or an Item. Entries are high-level descriptors, including commands, system utilities, and so forth. Items are categories of information that are provided for each Entry. **Examples**, **Operation**, and **Parms** are typical Items. EZHELP organizes the Entry and Item information into window displays that complement each other.

NOTE HELP catalog format statements (STARTHELP, STOPHELP, SUBSET, SUBITEM) cause no specific action in EZHELP. For example, shows what CICAT.PUB.SYS (the file for MPE HELP) contains for the ABORT command.



```
EZHELP     File: CICAT.PUB.SYS                    ABORT

ABORT

     Aborts the current program or operation.

SYNTAX

     ABORT
```

Figure 7.1 *ABORT Command*

Operation

EZHELP uses a PC-style interface which allows you to choose actions from a menu. There are selections for opening up the system help file, as well as for a help catalog file of your choice. When you choose a help catalog file, EZHELP asks you to key in the name of the file.

An easier way to choose a help catalog file is to select it from a popup picklist of file names. To display HELP Catalog Picklist, press F2 [Picklist] key when EZHELP asks you for the file name (see “Changing the HELP Catalog Picklist” on page 119 for more information on what this is about).

If you want to directly access an MPE topic without having to work through the menu system, you can type **ezhelp <MPE COMMAND>** at the colon prompt.

How EZHELP Formats Information

When a help catalog is chosen, EZHELP opens the file and prepares it for display.

First, EZHELP reads the catalog file and creates a sorted list of all the Entry lines (ENTRYs are referred to as Topics in EZHELP). This list placed in a scrollable window called "Topics". From this list, you select the Topic to be displayed.

Second, EZHELP finds the first Entry line in the help catalog and treats it as the first topic. So, when you run a help catalog under EZHELP, the initial topic displayed is the first Entry that EZHELP found in the file.

Displaying other topics is achieved by pressing the **Choose Topic** function key. When you press this key, EZHELP displays the sorted list it prepared when you first selected the catalog file. Use **Pg Dn** and **Pg Up** keys to scroll through the list, and press **Return** key to choose the Topic to display.

Other function key operations are available to assist you with both Topic and Item selections. Refer to “Function Keys” on page 120 for a complete description.

Cross-Reference Navigating

Hypertext-like cross-referencing is the ability to display information on related topics mentioned in the current window. Typically, related topics in the current window are selected by moving the cursor from one topic to the next. Having several topics in a single window, then, means that the user can branch to a new information display for a given topic just by pressing **Return** at a highlighted topic. True hypertext functionality implies that selectable topics in a window are informationally related to the current topic.

Since EZHELP has only a superficial knowledge of the data it is displaying, it would be impossible to provide the kind of true hypertext, relational links for sophisticated cross-reference support. however, it does provide a mechanism for superficial cross-referencing when the "Help" window is active (this is the window that contains text for a given topic).

Cross-referencing in EZHELP, then, is limited to accessing information displays for Topics mentioned in the current "Help" window (remember that Topics are those ENTRYs that are listed in the "Topics" window). EZHELP scans the current window to determine if there are any Topics in it. Any Topic, whether it is related to the current one or not, becomes a cross-reference candidate if it is in the current window.

Selecting a cross-reference Topic is done by pressing the letter **t**. When **t** is pressed, EZHELP scans the text on the screen and then looks for matches against other Entry lines. If a match is found, the cursor will be positioned at the beginning of the match. Pressing **Return** will cause the screen to switch to the new Topic window. Similarly, moving the cursor to a word and then pressing **Return** will tell EZHELP to check the text by the cursor against the Topic list. Again, switching to the new Topic if a match occurs.

To return to a previous screen, press **p**. Remember, the **t** and **p** options are only available in the Help window for a Topic.

Changing the HELP Catalog Picklist

The HELP Catalog Picklist is the picklist of help catalog file names used for selecting a catalog to display. This list is displayed by pressing the F2 [**Picklist**] key when EZHELP prompts for the catalog file name. You can add or remove catalog files from this list on an as-needed basis. For instance, the picklist provided with EZHELP may contain system help catalogs that your system doesn’t have. If this is the case, you may ant to remove these filenames from the list.

The picklist itself is stored in a flat file called EZHPCH.HELP. The first line in this file should not be modified. It contains formatting commands used to define the window. Right below this line are the names of the catalog files. Simply key in the name of the additional file, or delete unneeded file names as required. Because this window was defined with "unlimited" scrolling capabilities, the list can be as long as you need it to be.

NOTE If you own MAGNET (part of ***System Manager’s Toolbox***), you can use it to build a help catalog picklist for you.

For example, to scan the entire file system for help catalogs you could enter:

```
    magnet "-F@.@.@ -m -o mypic ’\ENTRY’ ’\ITEM’"
```

This sends the filelist to the file **mypic** in LISTF,6 format. Then, add the format line to this file and rename it **ezhpck.help.lpstools**. This replaces the file provided with EZHELP.

Capabilities

Program capabilities required include IA, PH, and DS. No special user capabilities are required to run EZHELP.

Function Keys

This section discusses function key operations that are specific to EZHELP.

PREV TOPIC

This function is used to return to the Topic previously displayed. Topics refer to the ENTRY topic, such as a command, tool, error, and so forth. It is similar to the PREVIOUS function that is standard across all windowed-based ***LPS-Tools***.

NEXT TOPIC

This function is used to display the next Topic in the catalog file where a Topic refers to the ENTRY topic (such as a command, tool, error, and so forth).

CHOOSE TOPIC

This function is used to choose a Topic from the Topic List. The **Pg Dn** and **Pg Up** keys are used to navigate the Topic List.

CHOOSE ITEM

This function is used to choose an Item for the currently selected Topic. It is operational only when a Topic is displayed. When you press the **Choose Item** key, EZHELP displays a picklist containing a list of ITEMS. These ITEMS include **Example**, **Operation**, and **Parm**.

Using EZHELP

This section provides step-by-step instruction for using the EZHELP program. It takes a tutorial-like approach that leads you through basic EZHELP operations. When you are finished with this section, you should have a very clear idea of how to use this tool.

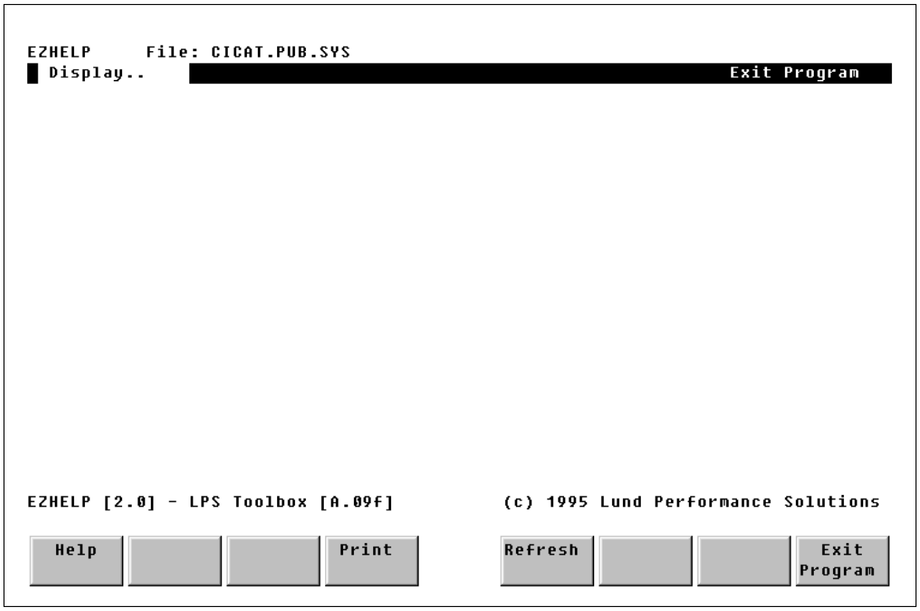
Several screen captures are provided to guide you through each step.

Starting EZHELP

To start EZHELP, type **EZHELP** at the colon prompt and press **Return**.

MPE Help users have the option of directly displaying an MPE topic by typing **ezhelp <mpe command>**. For example, typing **ezhelp getlog** displays the GETLOG Entry for MPE Help. Using EZHELP in this way bypasses the opening screen displays and prompts. For now, however, it is assumed that you will be using the EZHELP menus.

The next display shows the EZHELP main menu bar.



```
EZHELP     File: CICAT.PUB.SYS
```

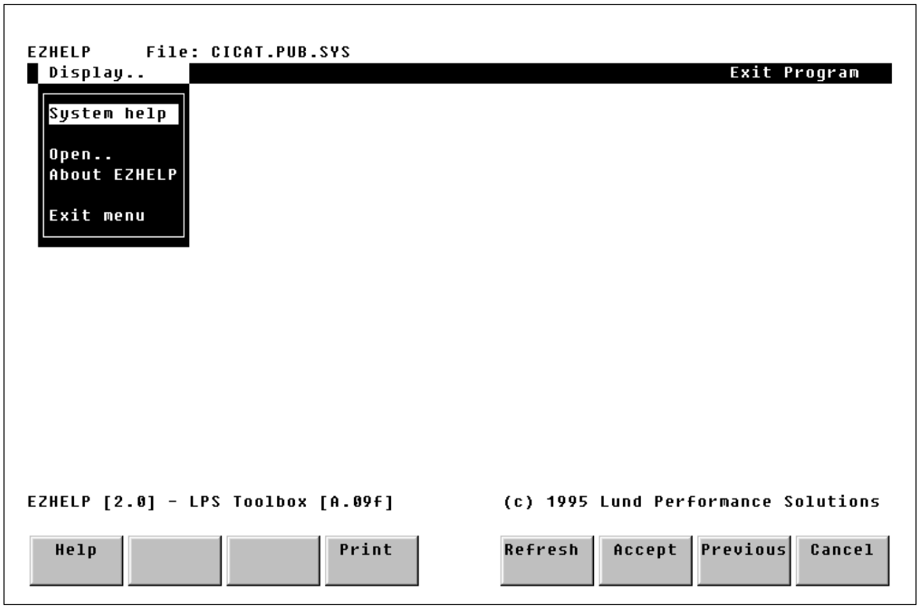
```
EZHELP [2.0] - LPS Toolbox [A.09f]          (c) 1995 Lund Performance Solutions
```

Figure 7.2 *EZHELP Main Menu*

When you first run EZHELP, it displays copyright and banner information at the bottom of the screen. The top row of the screen is an information line that contains the EZHELP name and the name of the currently opened help catalog. The second row is a menu bar that functions as the main menu. The option to the far right, EXIT, terminates EZHELP and returns control to MPE. The other option, **Display**, is discussed next.

Using the DISPLAY Menu

The EZHELP main menu contains the Display pull-down menu option. From this menu, use **System help** to select the system help file (CICAT.PUB.SYS) or **Open** to select a Help catalog of your own choosing.



```
EZHELP     File: CICAT.PUB.SYS
```

```
EZHELP [2.0] - LPS Toolbox [A.09f]         (c) 1995 Lund Performance Solutions
```

Figure 7.3 *DISPLAY's Pull-Down Menu*

To select an item from the menu, use the arrow key to highlight the option and then press **Return**.

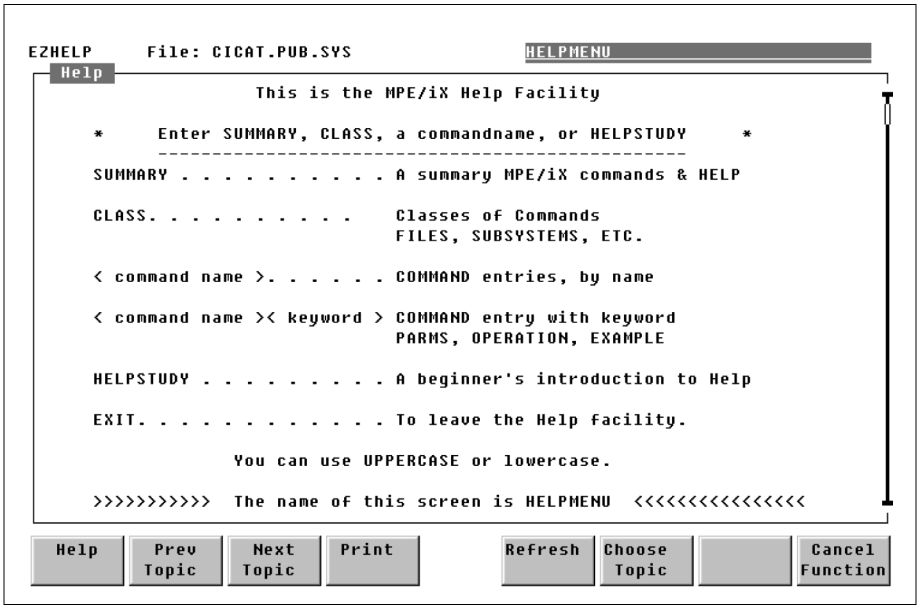
System help is used to navigate the system help facility. EZHELP enhances the use of system help by providing access to the information contained in the CICAT.PUB.SYS file in a windowed environment.

Open is used to select a help catalog of your choosing. This option is discussed in the section entitled "Viewing Other HELP Catalogs".

Using System Help

Running EZHELP's windowed interface for the system help catalog is achieved via the **System help** option. This section leads you through the various selections that are available, using examples to demonstrate the basic operations.

The next figure shows the System Help screen displayed when you select the **System help** option. For this version of CICAT, the first ENTRY is HELPMENU. Thus, HELPMENU becomes the first topic displayed when you select the **System help** option.



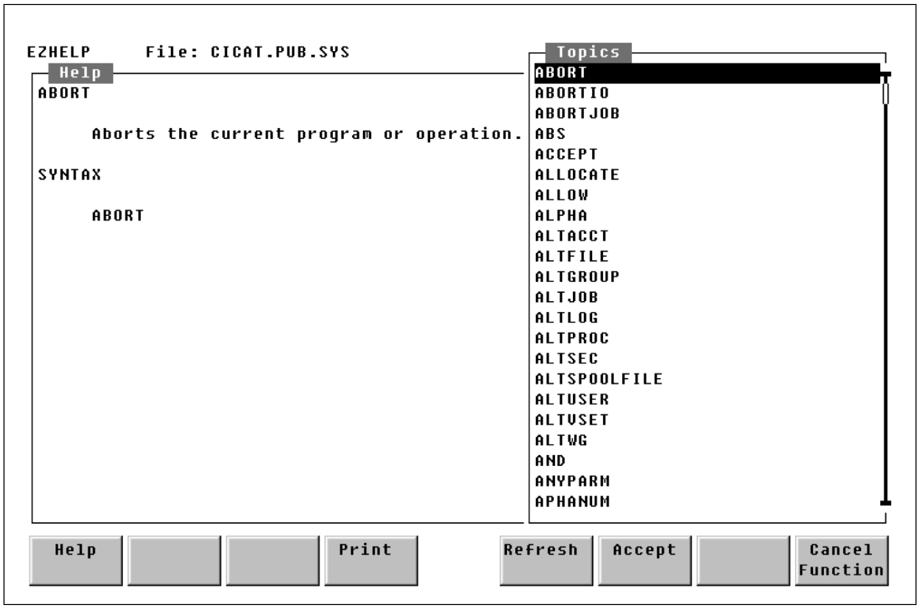
```
EZHELP     File: CICAT.PUB.SYS                  HELPMENU
  Help
               This is the MPE/iX Help Facility

     *     Enter SUMMARY, CLASS, a commandname, or HELPSTUDY     *
           --------------------------------------------------
     SUMMARY . . . . . . . . . . A summary MPE/iX commands & HELP

     CLASS. . . . . . . . . .    Classes of Commands
                                 FILES, SUBSYSTEMS, ETC.

     < command name >. . . . . . COMMAND entries, by name

     < command name >< keyword > COMMAND entry with keyword
                                 PARMS, OPERATION, EXAMPLE

     HELPSTUDY . . . . . . . . . A beginner's introduction to Help

     EXIT. . . . . . . . . . . . To leave the Help facility.

                  You can use UPPERCASE or lowercase.

     >>>>>>>>>>>  The name of this screen is HELPMENU  <<<<<<<<<<<<<<<<
```

Figure 7.4 *The System Help Display: Initial Screen*

Selections of topics is done through the **Choose Topic** function key. When you press this key, EZHELP pops up a scrollable window containing a list of all possible Topics.

Press F6 [**Choose Topic**] to display the pop up window of selectable topics.

The next screen shows the Topics window that is displayed whenever you select **Choose Topic**.



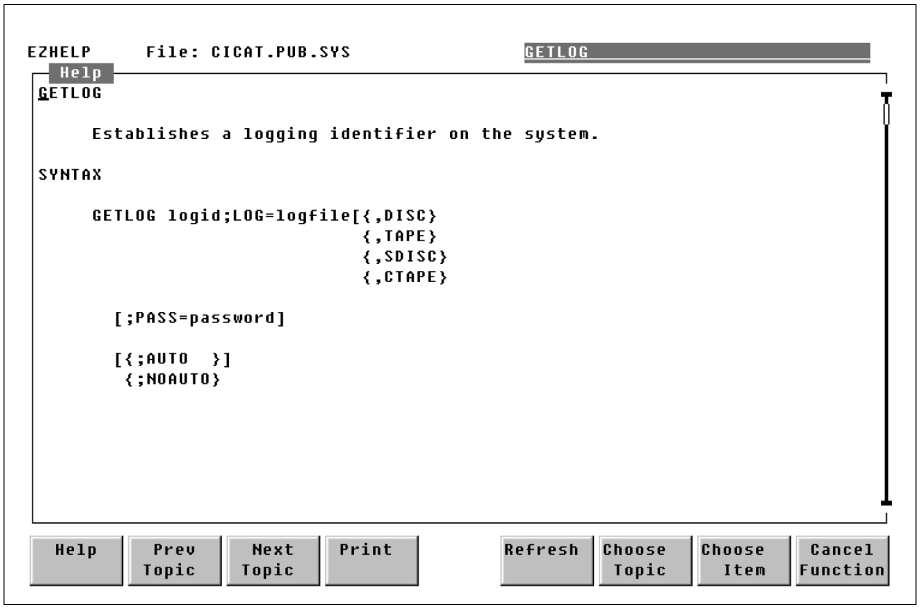
```
EZHELP      File: CICAT.PUB.SYS
Help
ABORT

     Aborts the current program or operation.

SYNTAX

     ABORT
```

Figure 7.5 *The Topic Selection Window*

Use the arrow keys to highlight a topic and then press **Return**.

For example, to find information on GETLOG, use the **Pg Dn** or arrow keys to highlight GETLOG in the Topics window and press **Return**.

The GETLOG entry is displayed next.



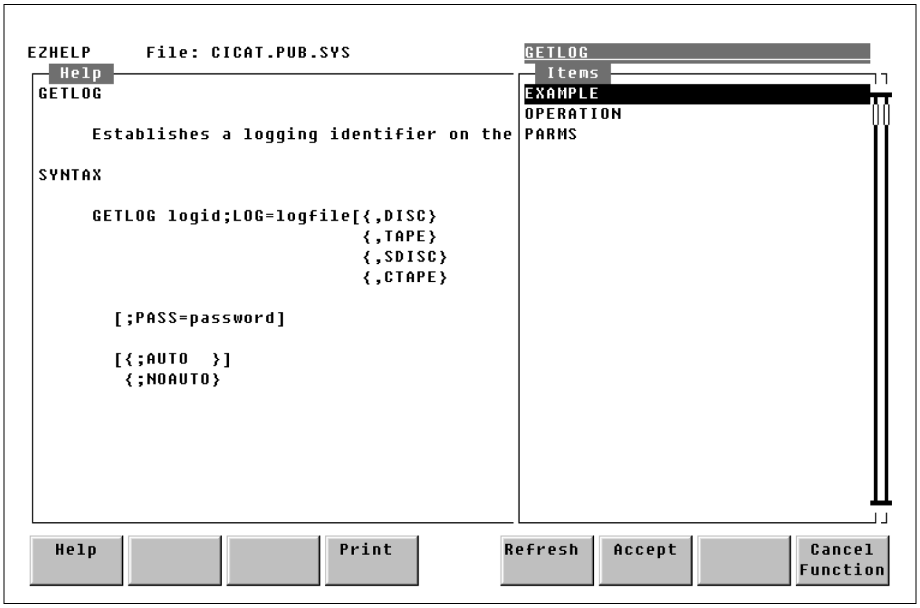
```
EZHELP     File: CICAT.PUB.SYS                 GETLOG
 Help
GETLOG

     Establishes a logging identifier on the system.

SYNTAX

     GETLOG logid;LOG=logfile[{,DISC}
                             {,TAPE}
                             {,SDISC}
                             {,CTAPE}

       [;PASS=password]

       [{;AUTO  }]
        {;NOAUTO}
```

Figure 7.6 *The GETLOG Entry in CICAT*

Notice that several function keys become operational once an entry is displayed.

Use the **Choose Item** function key to display additional information about the GETLOG command. For MPE commands in the CICAT file, the item choices are **Examples**, **Parms**, and **Operation**. Items are displayed in the popup, picklist window. As with any EZHELP picklist, use the arrow keys to select the item of interest and press **Return** to display the information.

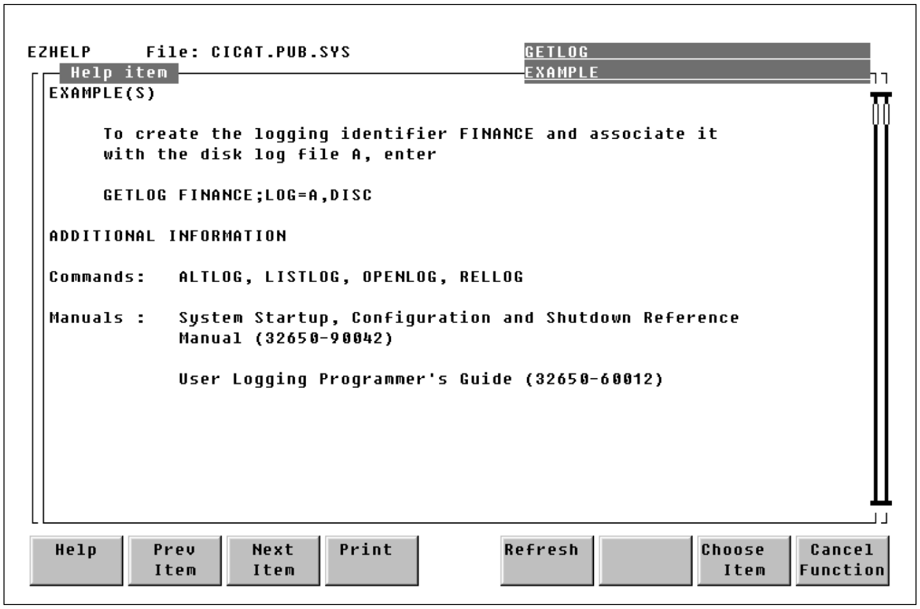
The next screen display shows the Items pop up menu that is displayed when you press the F7 [**Choose Item**] function key.



```
EZHELP      File: CICAT.PUB.SYS
GETLOG

     Establishes a logging identifier on the

SYNTAX

     GETLOG logid;LOG=logfile[{,DISC}
                              {,TAPE}
                              {,SDISC}
                              {,CTAPE}

       [;PASS=password]

       [{;AUTO   }]
        {;NOAUTO}
```

Figure 7.7 *The Item Selection Window*

The screen below shows the **Example** text for GETLOG. Had you selected **Parms** or **Operation**, a screen containing information on those items would be displayed.



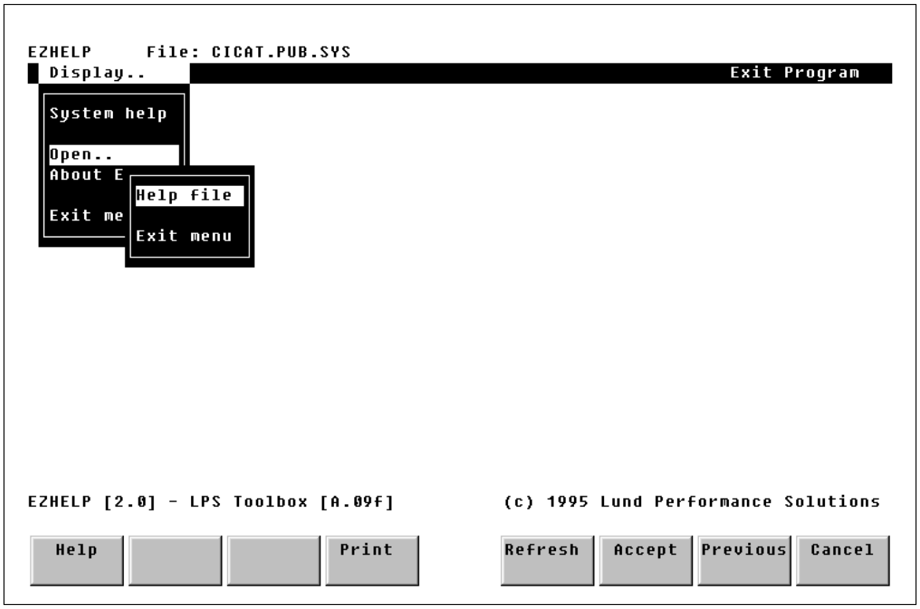
```
EZHELP     File: CICAT.PUB.SYS                         GETLOG
  Help item                                            EXAMPLE
EXAMPLE(S)

     To create the logging identifier FINANCE and associate it
     with the disk log file A, enter

     GETLOG FINANCE;LOG=A,DISC

ADDITIONAL INFORMATION

Commands:   ALTLOG, LISTLOG, OPENLOG, RELLOG

Manuals :   System Startup, Configuration and Shutdown Reference
            Manual (32650-90042)

            User Logging Programmer's Guide (32650-60012)
```

Figure 7.8 *The GETLOG Example*

Viewing Other HELP Catalogs

EZHELP can be used to view any help catalog that follows the structure used by the MPE HELP catalog. To view one of these catalogs, choose the Open command in the Display pull-down menu.

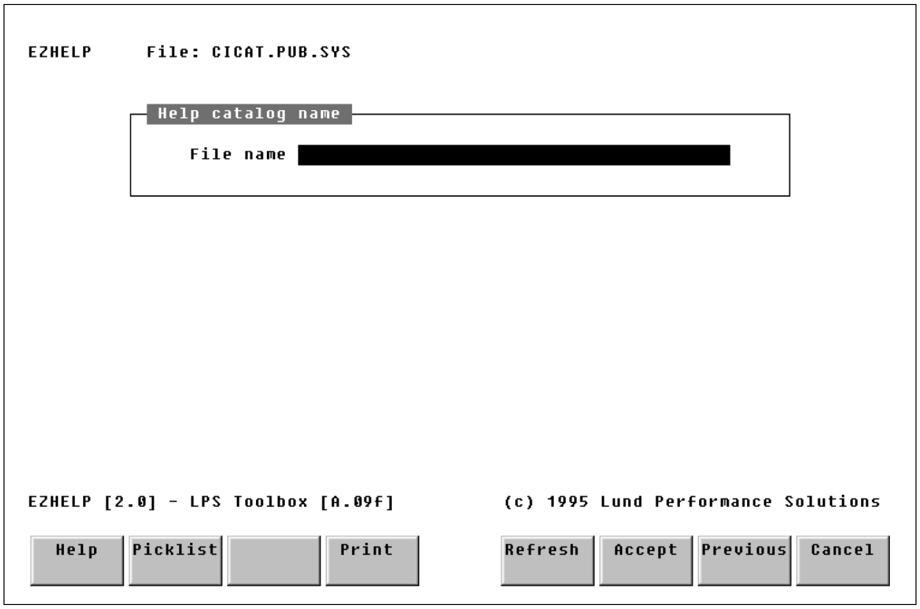
EZHELP will open the file you specify, dynamically arranging the file contents into information window displays and pop up lists from which you may select items and topics as needed.



```
EZHELP      File: CICAT.PUB.SYS
 Display..                                                  Exit Program

  System help

  Open..
  About E   Help file
  Exit me   Exit menu


EZHELP [2.0] - LPS Toolbox [A.09f]          (c) 1995 Lund Performance Solutions
```

Figure 7.9 *The Open Pull-down Menu*

Once you select **Open**, a pull-down menu listing the available options appears. Use the arrow keys to choose an option and then press **Return**. The Help file option is used to specify the name of the catalog file you wish to view. When you choose this option, a window is displayed that prompts you for a filename.



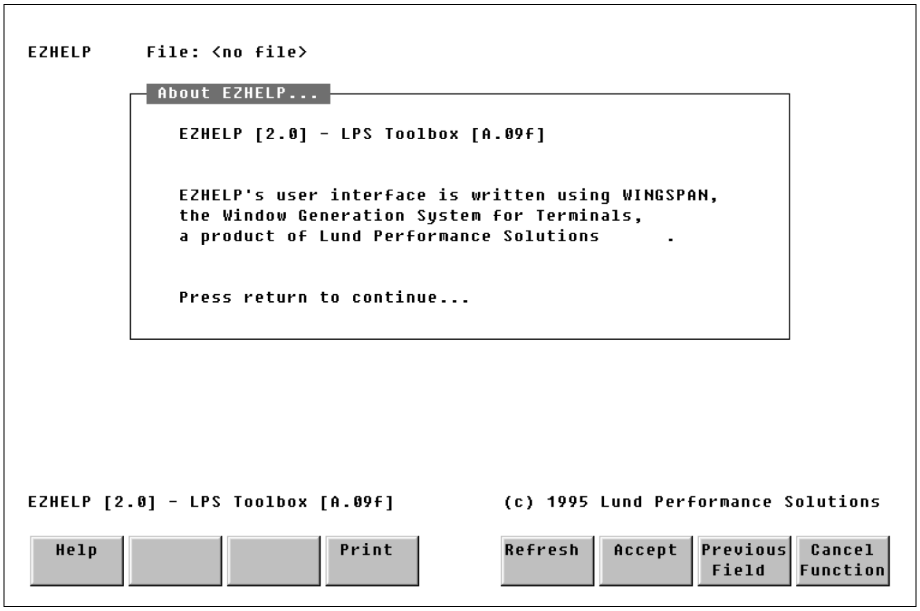
```
EZHELP     File: CICAT.PUB.SYS

        Help catalog name
             File name

EZHELP [2.0] - LPS Toolbox [A.09f]        (c) 1995 Lund Performance Solutions
```

Figure 7.10 *The Filename Specification Field*

Enter the filename of the help catalog you wish to view under the EZHELP interface in the **File name** field and press **Return**; or press F2 [**Picklist**] for the System Help Files Picklist. To select a help catalog file, use the arrow key to highlight the filename, then press **Return** to open the file.

Other EZHELP Options

Other options that are selectable in the EZHELP menus include the **About EZHELP** option in the Display pull-down menu. The **About EZHELP** option simply lists the version information for the current release of EZHELP. This screen display is shown next.



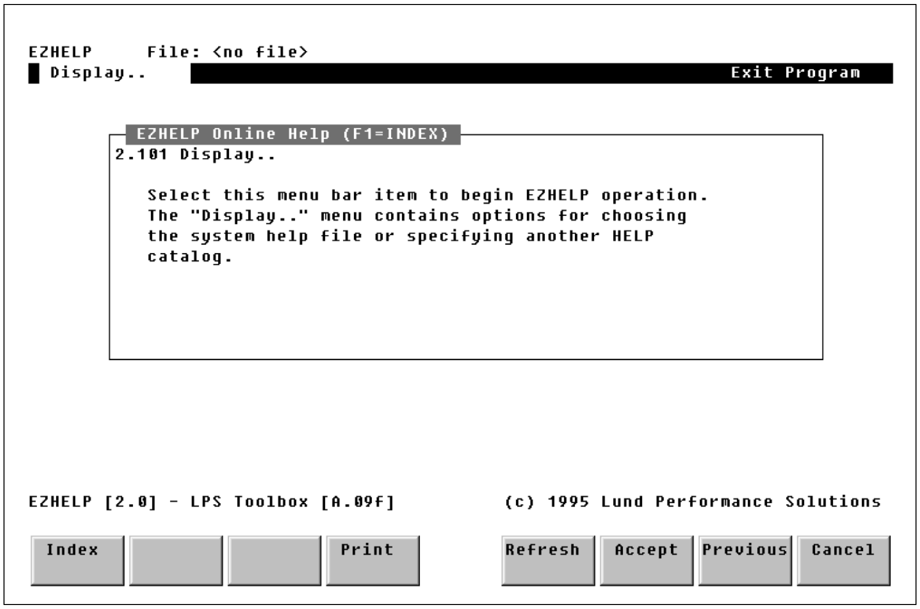
```
EZHELP     File: <no file>

        About EZHELP...

           EZHELP [2.0] - LPS Toolbox [A.09f]


           EZHELP's user interface is written using WINGSPAN,
           the Window Generation System for Terminals,
           a product of Lund Performance Solutions      .


           Press return to continue...


EZHELP [2.0] - LPS Toolbox [A.09f]          (c) 1995 Lund Performance Solutions
```

Figure 7.11 *About EZHELP*

Should you need assistance in navigating through EZHELP, the context-sensitive help facility is always available to provide information about the task at hand.

To access Help, simply press the F1 [**Help**] function key. The help screen shown next is produced whenever you press F1 while the Display menu option in the EZHELP main menu is highlighted.

]



```
EZHELP      File: <no file>
 Display..                                                  Exit Program

        EZHELP Online Help (F1=INDEX)
        2.101 Display..

           Select this menu bar item to begin EZHELP operation.
           The "Display.." menu contains options for choosing
           the system help file or specifying another HELP
           catalog.

EZHELP [2.0] - LPS Toolbox [A.09f]         (c) 1995 Lund Performance Solutions
```

Figure 7.12 *Using EZHELP's Context-Sensitive Help*

8

THE FASTLIB TOOL

FASTLIB is a library of fast replacements for the standard intrinsics: ASCII, BINARY, CTRANSLATE, DASCII, and DBINARY. Both CM and NM versions of these routines are available. Given that many applications call these intrinsics hundreds of thousands of times, using FASTLIB provides significant savings in CPU time. FASTLIB intrinsics are provided in libraries for both the Classic machine and the Spectrum machine.

Operation

The five FASTLIB intrinsics are "plug-compatible" with the standard intrinsics. ASCII and DASCII provide an extra output base, 16, which works in the same manner as base 8, except that the output is in hexadecimal.

NOTE See the *MPE/iX Intrinsics Reference Manual* for an explanation of base 10, -10, and 8.

FASTLIB comes as an NMOBJ file (for linking into your NM programs), an NMXL (for load-time linking with your NM programs), and as a USL (for prepping with your CM programs).

In addition to offering plug compatible replacements for these intrinsics, we also provide another choice. If your application does not use the return condition codes, you may choose to use FASTRLIB library of intrinsics. Functionally equivalent to the standard FASTLIB intrinsics, these intrinsics differ only in that they omit the step of setting the return condition code which yields even greater performance.

The FASTLIB intrinsics differ from the standard intrinsics in only two ways:

- They are much faster;
- If a standard intrinsic wants to abort your process, it will do so with a nice "intrinsic abort" message. The FASTLIB intrinsics will abort you in the same circumstances as the standard intrinsics, but without the same abort message (either a "FASTLIB abort" message will be displayed or an "invalid virtual address" message will appear).

The FASTLIB libraries that you have received are:

FASTLIB.XL	Native Mode executable libraries, sets condition codes

FASTRLIB.XL	Native Mode executable libraries, no condition codes set
FASTLIB.O	Native Mode object file, sets condition codes
FASTRLIB.O	Native Mode object file, no condition codes set
FASTLIB.USL	USL (classic) object file, sets condition codes

Capabilities

No special capabilities are required to run FASTLIB.

Usage

Usage information is provided in two sections: one section for Native Mode usage, and a second section for Compatibility Mode.

Native Mode Usage

There are two ways to use FASTLIB from your Native Mode application.

Easiest

Simply run existing programs with an extra option at the end of the RUN command. For example:

```
RUN MYPROG; XL = "FASTLIB.XL.LPSTOOLS"
```

If you specify a "XL=" command when you link a program, then you can omit the "XL=" at RUN time. For example:

```
:PASXL MYPROG, MYPROG.O, $NULL

:LINK FROM=MYPROG.O; TO=MYPROG.PUB; XL = FASTLIB.XL.LPSTOOLS
```

If you have the ability to re-link your programs, see the next note for a faster way of using FASTLIB.

Fastest

LINK the FASTLIB routines along with your program. In the LINK command, add the phrase ",FASTLIB.O.LPSTOOLS" to the FROM= option.

Example:

```
:PASXL FOO, FOO.O, $NULL

:LINK FROM=FOO.O, FASTLIB.O.LPSTOOLS; TO = FOO.PUB
```

Linking FASTLIB into your program will save about 20 instructions per call to each FASTLIB procedure.

Without Condition Codes

Many users do not need the condition codes set by BINARY, DBINARY, and CTRANSLATE. These users might begrudge the time spent by these FASTLIB procedures in setting the condition code.

NOTE CTRANSLATE always sets the condition code to CCE, no other value is possible!

FASTRLIB is a version of FASTLIB that does not set condition codes for any of the procedures. It is used exactly like FASTLIB (i.e.: it can either be linked into a program or accessed via an "XL=" option on the RUN command).

Compatibility Mode Usage

For compatibility mode users the only option that Lund Performance Solutions provides for the use of the FASTLIB intrinsics is to PREP them into your program. For example:

SEGMENTER

- USL MYUSL
- AUX FASTLIB.USL.LPSTOOLS
- COPY SEGMENT,FASTLIB

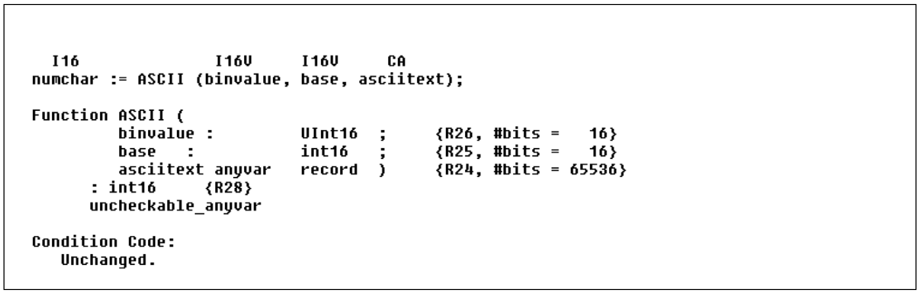
FASTRLIB comes as an NMOBJ file (for linking into your NM programs), and as an NMXL (for load-time linking with your NM programs). A USL version is not provided because setting the condition code in CM code requires only 4 instructions.

What's Next

The following section illustrates the calling sequence for each intrinsic in FASTLIB and FASTRLIB (see The CSEQ Tool to find how this was generated) from the viewpoint of Pascal/XL and the hardware, plus a brief description of each intrinsic. For a more detailed discussion, please refer to the *MPE/iX Intrinsic Reference Manual*.

ASCII

Purpose: Convert a 16-bit number into the equivalent ASCII string.

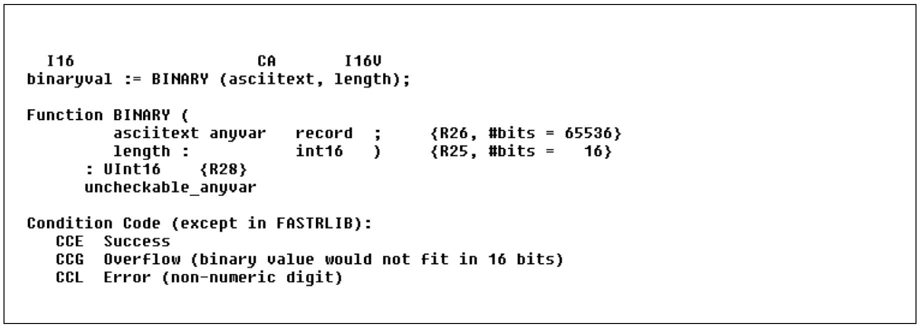


```
  I16              I16V     I16V     CA
numchar := ASCII (binvalue, base, asciitext);

Function ASCII (
         binvalue :         UInt16  ;     {R26, #bits =   16}
         base   :           int16   ;     {R25, #bits =   16}
         asciitext anyvar   record  )     {R24, #bits = 65536}
      : int16     {R28}
      uncheckable_anyvar

Condition Code:
   Unchanged.
```

Figure 8.1 *Convert to ASCII*

BINARY

Purpose: Convert an ASCII string into a 16-bit number.

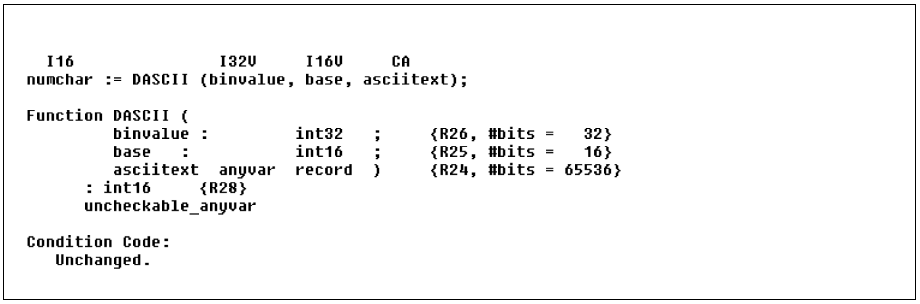


```
  I16                   CA       I16V
binaryval := BINARY (asciitext, length);

Function BINARY (
         asciitext anyvar   record  ;     {R26, #bits = 65536}
         length :           int16   )     {R25, #bits =   16}
      : UInt16    {R28}
      uncheckable_anyvar

Condition Code (except in FASTRLIB):
   CCE  Success
   CCG  Overflow (binary value would not fit in 16 bits)
   CCL  Error (non-numeric digit)
```

Figure 8.2 *Convert from ASCII*

DASCII

Purpose: Convert a 32-bit number into the equivalent ASCII string.

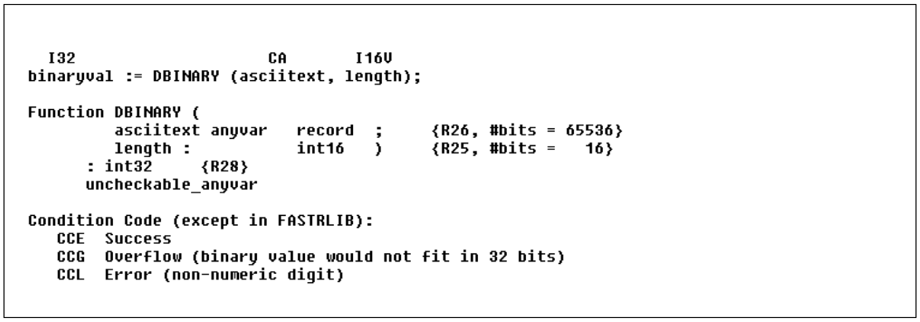


```
  I16               I32V     I16V     CA
numchar := DASCII (binvalue, base, asciitext);

Function DASCII (
         binvalue :         int32    ;    {R26, #bits =    32}
         base   :           int16    ;    {R25, #bits =    16}
         asciitext  anyvar  record   )    {R24, #bits = 65536}
      : int16     {R28}
      uncheckable_anyvar

Condition Code:
   Unchanged.
```

Figure 8.3 *Convert to Equivalent ASCII String*

DBINARY

Purpose: Convert an ASCII string into a 32-bit number.

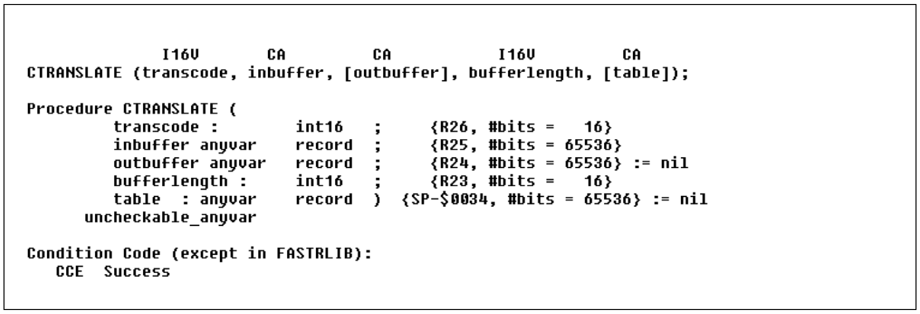


```
  I32                     CA       I16V
binaryval := DBINARY (asciitext, length);

Function DBINARY (
         asciitext anyvar   record   ;    {R26, #bits = 65536}
         length :           int16    )    {R25, #bits =    16}
      : int32     {R28}
      uncheckable_anyvar

Condition Code (except in FASTRLIB):
   CCE  Success
   CCG  Overflow (binary value would not fit in 32 bits)
   CCL  Error (non-numeric digit)
```

Figure 8.4 *Converted ASCII String*

CTRANSLATE

Purpose: Converts a string of EBCDIC or ASCII characters from one to the other, or between EBCDIK and KANA8. Or, translate via a user-supplied translation table.



```
              I16V       CA         CA            I16V         CA
CTRANSLATE (transcode, inbuffer, [outbuffer], bufferlength, [table]);

Procedure CTRANSLATE (
         transcode :        int16   ;     {R26, #bits =    16}
         inbuffer anyvar    record  ;     {R25, #bits = 65536}
         outbuffer anyvar   record  ;     {R24, #bits = 65536} := nil
         bufferlength :     int16   ;     {R23, #bits =    16}
         table  : anyvar    record  )  {SP-$0034, #bits = 65536} := nil
      uncheckable_anyvar

Condition Code (except in FASTRLIB):
   CCE  Success
```

Figure 8.5 *Conversion*

TOOLBOX STANDARDS

The ToolBox collections from Lund Performance Solutions have a uniform user interface. As a result, in addition to the commands specific to each Toolbox tool, most tools allow the commands described in "TOOLBOX STANDARDS" on page 213.

Timing

How much faster are the FASTLIB intrinsics? When they are originally written, the FASTLIB intrinsics were up to 20 times faster than the system intrinsics. Although the system intrinsics have been optimized since FASTLIB first became available, FASTLIB intrinsics should still be considered as a high-performance alternative.

Two test programs are provided so that you can measure the performance gains provided by the FASTLIB intrinsics.

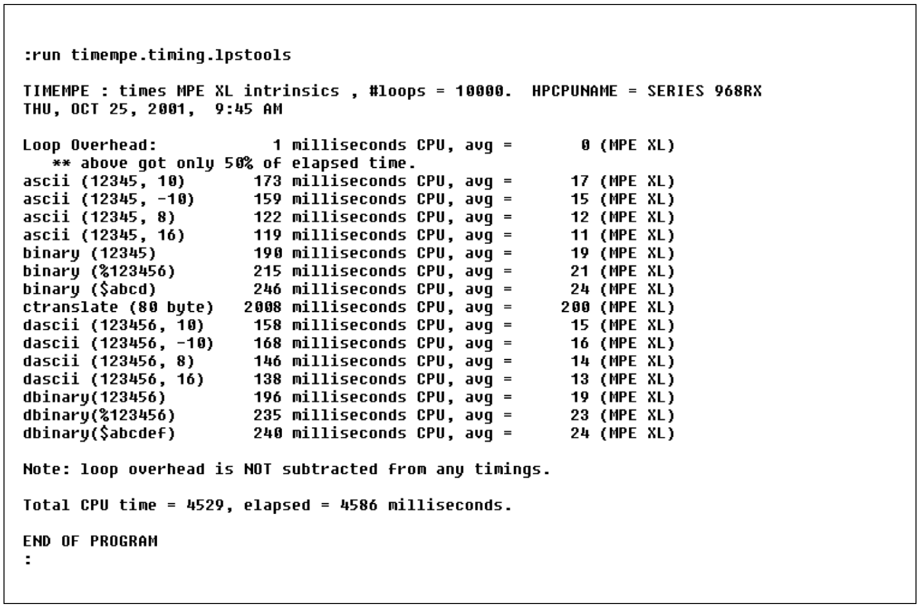
TIMEMPE.TIMING	test program uses HP intrinsics
TIMEFAST.TIMING	same program, uses FASTLIB intrinsics

To run these programs just use one of the RUN statements below:

```
RUN TIMEMPE.TIMING.LPSTOOLS

RUN TIMEFAST.TIMING.LPSTOOLS
```

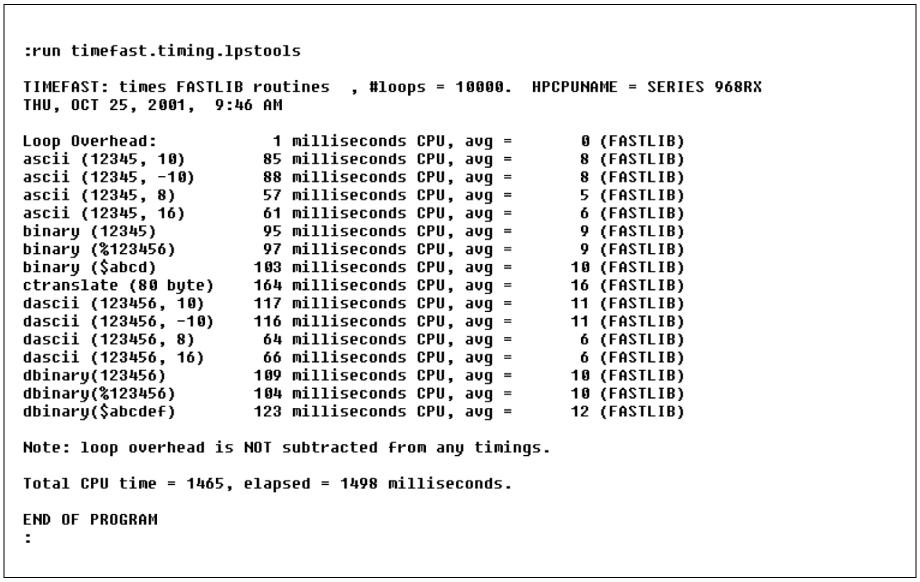
FASTLIB Examples

Following are some examples of the FASTLIB tool:



```
:run timempe.timing.lpstools

TIMEMPE : times MPE XL intrinsics , #loops = 10000.  HPCPUNAME = SERIES 968RX
THU, OCT 25, 2001,  9:45 AM

Loop Overhead:            1 milliseconds CPU, avg =       0 (MPE XL)
   ** above got only 50% of elapsed time.
ascii (12345, 10)       173 milliseconds CPU, avg =      17 (MPE XL)
ascii (12345, -10)      159 milliseconds CPU, avg =      15 (MPE XL)
ascii (12345, 8)        122 milliseconds CPU, avg =      12 (MPE XL)
ascii (12345, 16)       119 milliseconds CPU, avg =      11 (MPE XL)
binary (12345)          190 milliseconds CPU, avg =      19 (MPE XL)
binary (%123456)        215 milliseconds CPU, avg =      21 (MPE XL)
binary ($abcd)          246 milliseconds CPU, avg =      24 (MPE XL)
ctranslate (80 byte)   2008 milliseconds CPU, avg =     200 (MPE XL)
dascii (123456, 10)     158 milliseconds CPU, avg =      15 (MPE XL)
dascii (123456, -10)    168 milliseconds CPU, avg =      16 (MPE XL)
dascii (123456, 8)      146 milliseconds CPU, avg =      14 (MPE XL)
dascii (123456, 16)     138 milliseconds CPU, avg =      13 (MPE XL)
dbinary(123456)         196 milliseconds CPU, avg =      19 (MPE XL)
dbinary(%123456)        235 milliseconds CPU, avg =      23 (MPE XL)
dbinary($abcdef)        240 milliseconds CPU, avg =      24 (MPE XL)

Note: loop overhead is NOT subtracted from any timings.

Total CPU time = 4529, elapsed = 4586 milliseconds.

END OF PROGRAM
:
```

Figure 8.6 *Running the TIMEMPE Program*



```
:run timefast.timing.lpstools

TIMEFAST: times FASTLIB routines   , #loops = 10000.  HPCPUNAME = SERIES 968RX
THU, OCT 25, 2001,  9:46 AM

Loop Overhead:            1 milliseconds CPU, avg =       0 (FASTLIB)
ascii (12345, 10)        85 milliseconds CPU, avg =       8 (FASTLIB)
ascii (12345, -10)       88 milliseconds CPU, avg =       8 (FASTLIB)
ascii (12345, 8)         57 milliseconds CPU, avg =       5 (FASTLIB)
ascii (12345, 16)        61 milliseconds CPU, avg =       6 (FASTLIB)
binary (12345)           95 milliseconds CPU, avg =       9 (FASTLIB)
binary (%123456)         97 milliseconds CPU, avg =       9 (FASTLIB)
binary ($abcd)          103 milliseconds CPU, avg =      10 (FASTLIB)
ctranslate (80 byte)    164 milliseconds CPU, avg =      16 (FASTLIB)
dascii (123456, 10)     117 milliseconds CPU, avg =      11 (FASTLIB)
dascii (123456, -10)    116 milliseconds CPU, avg =      11 (FASTLIB)
dascii (123456, 8)       64 milliseconds CPU, avg =       6 (FASTLIB)
dascii (123456, 16)      66 milliseconds CPU, avg =       6 (FASTLIB)
dbinary(123456)         109 milliseconds CPU, avg =      10 (FASTLIB)
dbinary(%123456)        104 milliseconds CPU, avg =      10 (FASTLIB)
dbinary($abcdef)        123 milliseconds CPU, avg =      12 (FASTLIB)

Note: loop overhead is NOT subtracted from any timings.

Total CPU time = 1465, elapsed = 1498 milliseconds.

END OF PROGRAM
:
```

Figure 8.7 *Running the TIMEFAST Program*

FASTLIB Error Messages

Errors generated by FASTLIB are the same as those generated by their HP equivalents. See the *HP Intrinsic Reference Manual* for possible error conditions.

9

THE WILDCARD TOOL

The WILDCARD tool is a library of procedures that provide functionality not inherent in any programming language or environment. Functionally, the WILDCARD library provides solutions for two common programming tasks. First, it offers the ability to build a fileset from a complex fileset specification. This ability expands on LISTF-style operations so that you can add, subtract, or otherwise qualify groups of files for use in your programs. Second, WILDCARD provides a way to match patterns in string expressions (i.e., filename expressions).

The WILDCARD tool, then, is actually two groups of callable procedures: FILESET procedures and PATTERN procedures.

The FILESET procedures include:

getfileset

buildfileset

buildfilename

fileseterrmsg

fs_version

The PATTERN procedures include:

pattern_build

pattern_match

pattern_fga_match

check_fga_wildcard

NOTE POSIX (HFS) file structures are not currently supported.

FILESET Procedures

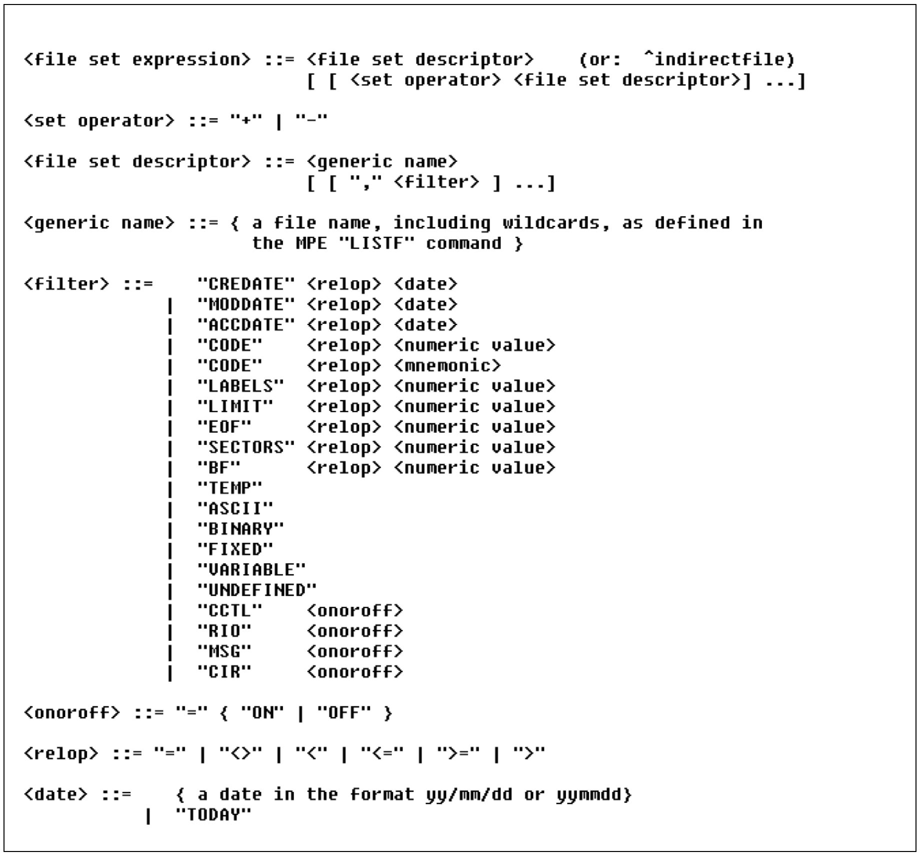
In order to provide maximum flexibility, the FILESET building tasks have been broken into five separate procedure calls. The generated fileset is stored in an ASCII flat file so that you can access it as best suits your needs. The section called "Output Format" provides details on the layout of the file. The complete syntax that can be used to specify a file is described in the section called "Fileset Syntax". If you are familiar with the MAGNET o BLAZE tools (included in the ***System Manager's Toolbox***), then you may be familiar with this syntax already.

The **getfileset** procedure allows you to build a fileset with a single procedure call. If this call is not flexible enough for your needs, you may want to use the procedures **buildfilename** and **buildfileset**. These procedures provide more latitude for building the fileset the way you need it.

The remaining procedures, fs_version and fileseterrmsg, are used to provide the version string of the FILESET procedures and the error text for a specified error code.

FILESET Syntax

This section outlines the syntax used in the various fileset procedures.



```
<file set expression> ::= <file set descriptor>    (or:  ^indirectfile)
                          [ [ <set operator> <file set descriptor>] ...]

<set operator> ::= "+" | "-"

<file set descriptor> ::= <generic name>
                          [ [ "," <filter> ] ...]

<generic name> ::= { a file name, including wildcards, as defined in
                     the MPE "LISTF" command }

<filter> ::=    "CREDATE" <relop> <date>
             |  "MODDATE" <relop> <date>
             |  "ACCDATE" <relop> <date>
             |  "CODE"    <relop> <numeric value>
             |  "CODE"    <relop> <mnemonic>
             |  "LABELS"  <relop> <numeric value>
             |  "LIMIT"   <relop> <numeric value>
             |  "EOF"     <relop> <numeric value>
             |  "SECTORS" <relop> <numeric value>
             |  "BF"      <relop> <numeric value>
             |  "TEMP"
             |  "ASCII"
             |  "BINARY"
             |  "FIXED"
             |  "VARIABLE"
             |  "UNDEFINED"
             |  "CCTL"    <onoroff>
             |  "RIO"     <onoroff>
             |  "MSG"     <onoroff>
             |  "CIR"     <onoroff>

<onoroff> ::= "=" { "ON" | "OFF" }

<relop> ::= "=" | "<>" | "<" | "<=" | ">=" | ">"

<date> ::=    { a date in the format yy/mm/dd or yymmdd}
           |  "TODAY"
```

Figure 9.1 *WILDCARD Extended Fileset Syntax*

NOTE All literals are case-insensitive.

For further information, you may wish to refer to Appendix B, which features a list of the more common file codes, and Appendix C, which provides a convenient reference for LISTF WILDCARD syntax.

Output Format

This section presents the output format of the FILESET procedures.

File structure: 80 byte, fixed, ASCII

Table 9.1 *Output Format*

Bytes	Item
0 .. 7	Account name
8 .. 15	Group name
16 .. 23	File name
24 .. 28	File code
29 .. 37	Record size
38 .. 41	File type
42 .. 53	End-of-file
54 .. 64	File limit
65 .. 68	Blocking factor
69 .. 79	Sectors

Operation

All of the FILESET procedures are callable from either Native Mode or Compatibility Mode.

The Native Mode version follows the Procedure Calling Convention established by Hewlett-Packard and is therefore callable from any language following these conventions.

For Compatibility Mode, follow the rules established by Hewlett-Packard for parameter passing and segmentation (i.e., not callable from CCS/C CREL format programs).

Two levels of integration are provided so that you can choose the method that best suits your needs. The first level is simply to call the procedures as you would call an intrinsic. FILESET procedures can be accessed in much the same manner as intrinsics are accessed. The second method may be a better choice if a greater level of control is desired. In this case, you would merge the declaration files into your source, and then recompile and link the program.

GETFILESET

The purpose of this procedure is to build a fileset based on the fileset specification string that is passed to this procedure.

Syntax:

```
short int getfileset (expression)
```

Return Value

getfileset returns a 16-bit integer encoded as follows:

Table 9.2 *Getfileset's Return Values*

Code	Definition	Description
< 0	:Error	where the absolute value is the error number. This can be passed **fileseterrmsg** to retrieve the error text.
0	:No error	where the resulting fileset is in the temporary file FILES.
> 0	:Warning	where the value is the number of characters processed from the provided fileset specification string.

Parameters

expression Byte array (required). It contains the NULL (ASCII 0) terminated fileset specification string. For a complete discussion of fileset specifications see Appendix C.

Operation

To use this routine, all that is required is to declare **getfileset** as an external procedure. Depending on the language used, this may occur automatically. Then, compile your application and link with the either the WILDCARD object file, relocatable library or executable library. After calling getfileset, check the return value for errors. If no error occurred, the resultant fileset can be accessed through the temporary file called "FILES".

NOTE FILES cannot be file equated. See the sample code TESTGFS.C.LPSTOOLS for an example.

BUILDFILENAME

This procedure is used to complete the building of the filename based on the specified mode. Five different modes are available ranging from fully-qualifying the filename to generating a unique filename. No errors are possible. The filename will be constructed using the standard MPE filename format (i.e., filename.group.account).

Syntax

```
    buildfilename (filename, mode, terminator);
```

The Parameter Set is listed next, where each parameter is either an integer, character array, or integer array.

Table 9.3 *BUILDFILENAME's Parameter Set*

Parameter	Name	Type	Comment
1	filename	character array	Required
2	mode	short int	Required
3	terminator	short int	Required

Return Value

There is no Return Value.

Parameters

filename

Byte array (required). Modes 0, 1, 2, and 3 contain the space character (ASCII 32) terminated filename. For mode 4, this filename will contain the unique filename generated by the call.

For all modes, the array will be terminated with the character provided in the parameter terminator. The dot (**.**) separator should not be specified. No filename validation will occur.

modes

Short int (required). Recognized values range from 0 to 4. The definitions are as follows:

0 = Append the logon group and account to the specified filename

1 = Append the logon account to the specified filename

2 = Append the program group and account to the specified filename

3 = Append the program account to the specified filename

4 = Generate a unique filename in logon group

If an unknown mode is given, then the terminator is appended to **filename**.

terminator

Short int (required). It is used to specify the character that will be used to terminate the byte array filename.

Operation

To use this routine, declare **buildfilename** as an external procedure. Depending on the language, this may occur automatically. Compile your application and link it with either the WILDCARD object file, relocatable library or executable library. Before calling **buildfilename**, determine which mode you want to use. Then, for modes 0 through 3, initialize the parameter filename. For all modes, initialize the terminator parameter before calling **buildfilename**.

The result of all operation will be in the byte array filename. The format for the filename will be in MPE format. Any values filled in by the call will be in uppercase. Groups, accounts and filenames will be separated by dots (**.**). The filename will be terminated with the character specified by the terminator parameter.

No errors are possible; calling **buildfilename** with an invalid mode will simply result in the filename being terminated by the terminator you provided. Also, calling **buildfilename** without a filename (for modes 0 through 3) will not cause an error, however, the resulting filename may not be very useful. For example, see the sample code TESTFS.C.LPSTOOLS or TESTFS.SPL.LPSTOOLS.

BUILDFILESET

This procedure will generate the fileset specified by the expression. The fileset will be stored in the file given by the parameter filename and the domain will be determined by the boolean value of perm. The **stat** parameter is a two element array. The 0th element contains the status, the 1st element contains an error code if the 0th element is non-zero. The procedure return value equals the number of characters processed from the **expression** parameter.

Syntax

```
    short int buildfileset (expression, filename, perm, stat);
```

The Parameter Set is listed next, where each parameter is either an integer, character array, or integer array.

Table 9.4 *BUILDFILESET's Parameter Set*

Parameter	Name	Type	Comment
1	expression	character array	Required
2	filename	character array	Required
3	perm	logical	Required
4	stat	short int	Required

Return Value

buildfileset returns a 16-bit integer that represents the number of characters processed from the expression string. Nominally, this equals the length of **expression**.

Parameters

expression — Byte array (required). This parameter contains the NULL (ASCII 0) terminated fileset specification string. See the Fileset Specification Syntax in Appendix C for a complete discussion of fileset specifications.

filename — Byte array (required). It contains the NULL (ASCII 0) terminated string used to build a file to hold the result of the **buildfileset** call. It cannot be file equated.

perm — Logical (required). It contains a value of true (even) or false (odd) used to indicate if the output file should be a permanent or temporary file.

stat — Short int array (required). It contains the status of the call to **buildfileset**. Stat(0) returns the status of the call. A nonzero value indicates an error. The nonzero code can be optionally passed to **fileseterrmsg** to retrieve the error text.

Operation

To use this routine, declare **buildfileset** as an external procedure. Depending on the language, this may occur automatically. Compile your application and link it with either the WILDCARD object file, relocatable library or executable library. After calling **buildfileset**, check the status variable **stat** to determine if the call was successful. Also, check the return value to determine if the entire expression was processed. If the variable stat equals zero, then the resultant can be accessed through the file specified by the parameter **filename** (see the sample code TESTFS.C.LPSTOOLS or TESTFS.SPL.LPSTOOLS).

FILESETERRMSG

The purpose of **fileseterrmsg** is to provide and format the text describing the error returned from a **buildfileset** or **getfileset** call.

Syntax

```
    short int fileseterrmsg (status, buffer);
```

The Parameter Set is listed next, where each parameter is either an integer array or character array.

Table 9.5 *FILESETERRMSG's Parameter Set*

Parameter	Name	Type	Comment
1	status	short int array	Required
2	buffer	character array	Required

Return Value

The integer value returned by **fileseterrmsg** is the byte length of the text that has been placed in buffer.

Parameters

status — Short int array (required). It contains the status of the call to **buildfileset**.

status(0) is the error number and it is used to look up the text of the error message.

status(1) (if non-zero) is appended to the end of the error text. The format used is: **info: <status(1)>**. Its use is purely informational. Most of the time when **status(1)** is non-zero, it will represent the error number returned by the intrinsic FCHECK plus some kind of file system error.

buffer — Byte array (required). The length must be at least 80 bytes.

Operation

To use this entry point, declare **fileseterrmsg** as an external. Depending on the language, this may occur automatically. Compile your application and link it with either the WILDCARD object file, relocatable library or executable library (see the sample code TESTFS.C.LPSTOOLS or TESTFS.SPL.LPSTOOLS).

FS_VERSION

This procedure will obtain the FILESET version string.

Syntax

```
fs_version (buffer);
```

Return Value

There are no Return Values.

Parameters

buffer — Byte array (required). The length must be at least 80 bytes.

Operation

To use this routine, declare **fs_version** as an external procedure. Depending on the language, this may occur automatically. Compile your application and link it with either the WILDCARD object file, relocatable library or executable library. After calling **fs_version**, the byte array will contain the ASCII version string. This can be used to test FILESET versions to ensure compatibility of applications that use FILESET (see the sample code TESTFS.C.LPSTOOLS or TESTFS.SPL.LPSTOOLS).

Fileset Error Numbers and Meanings

Table 9.6 *Fileset Error Numbers and Meanings*

stat(0)	meaning	stat(1) meaning
7	error during fclose	error number from FCHECK
8	error during fcontrol	error number from FCHECK
9	error during fopen, new	error number from FCHECK
10	error during fopen, old	error number from FCHECK
11	error during fread	error number from FCHECK
12	error during file rename	error number from FCHECK
13	error saving file	error number from FCHECK
14	error during fwrite	error number from FCHECK
21	error closing listf temporary file	error number from FCHECK
22	error opening listf temporary file	error number from FCHECK
23	error reading listf temporary file	error number from FCHECK
2	error from command intrinsic	error number from COMMAND
29	error during listf command	error number from COMMAND
1	expected alphabetic or numeric	not used
3	expected date	not used
4	bad filename part	not used
5	bad groupname part	not used
6	bad accountname part	not used

stat(0)	meaning	stat(1) meaning
15	bad 16 bit integer	not used
16	same as #1 & # 15	not used
17	bad 32 bit integer	not used
18	error converting to 32 bit integer	not used
19	same as #18, except value	not used
20	unknown keyword	not used
24	expected keyword "on" or "off"	not used
25	unexpected value in expression	not used
26	unknown relational operator	not used
27	unbalanced right parenthesis	not used
28	expected keyword "today"	not used

PATTERN Procedures

The WILDCARD Pattern Matching collection contains four procedures used for building and checking for pattern matches. Three of the procedures (**pattern_build**, **pattern_match**, **pattern_fga_match**) provide a low-level approach for integration into your application. The fourth procedure (**check_fga_wildcard**) provides a higher-level approach.

The procedures that start with the string **pattern_** are easily callable from either Pascal or C. The other procedure can be called from any Native Mode language.

Conceptually, any of the **pattern_** procedures could be called from any Native Mode language. Given that the data structure passed into a **pattern_** procedure is fairly complex, you should be aware that calling these types of procedures from either COBOL or SPLash! can be tricky. Conceptually, the **pattern_match** procedure can be used for matching strings of any length. However, it was really designed for matching strings that contain fully-qualified filenames.

Operation

The first thing that must be done to use those procedures is to initialize the PATTERN_TYPE data structures. This is done by calling the **pattern_build** procedure with the appropriate parameters.

Once the PATTERN_TYPE data-structures have been successfully initialized, the **pattern_match** or **pattern_fga_match** procedures can be called repeatedly to check for as many matches as you need. This approach is nice since the **pattern_build** procedure is only called once to set up the pattern (the **check_fga_wildcard** procedure uses both **pattern_build** and **pattern_fga_match**). This approach also makes it possible to initialize several WILDCARD patterns up front and then use them as needed.

CHECK_FGA_WILDCARD

This procedure is very simple to use. Simply pass in the Wildcard string and the filename string and this procedure will return either true or false. True means the filename was represented by the Wildcard, and False means it wasn't. Additionally, if the return value is negative, it will contain an error number.

CHECK_WILDCARD

This procedure will simplify the use of the WILDCARD Pattern matching procedures. This procedure is particularly useful if only one (or a few) filename(s) are being tested. Also, this procedure reduces some of the programming necessary to use the pattern matching procedures.

Syntax

```
    int check_wildcard (wildcard, filename);
```

The Parameter Set is listed next, where each parameter is either an integer array or character array.

Table 9.7 *CHECK_WILDCARD's Parameter Set*

Parameter	Name	Type	Comment
1	wildcard	character array	Required
2	filename	character array	Required

Return Value

Check_wildcard returns a 32-bit integer encoded as follow:

Table 9.8 *CHECK_WILDCARD's Return Values*

Code	Definition	Error Description
< 0	:Filename is matched by wildcard	
= 0	:Nomatch	

Code	Definition	Error Description
> 0	:Error	-1 = Missing 1st "." delimiter -2 = Error initializing filename pattern -3 = Missing 2nd "." delimiter -4 = Error initializing groupname pattern -5 = Missing accountname -6 = Error initializing accountname pattern -7 = Missing filename -8 = Missing groupname

Parameters

wildcard Byte array (required). A fully-qualified string ASCII space terminated. This procedure expects that the components of the filename are separated by a dot (**.**). Also, the buffer containing the string should not contain any characters past the terminating space.

Example 1: **@.@.@**

Example 2: **@.pub.sys**

Example 3: **@foo@.???.s#96**

filename Byte array (required). A NULL (ASCII zero) terminated MPE fully-qualified filename.

Operation

Using this procedure can significantly reduce the amount of programming required to check fully-qualified MPE filenames. This is a stand-alone procedure and is not used in conjunction with any of the other WILDCARD Pattern procedures. See TESTCW.C.LPSTOOLS or TESTPAT.PASCAL.LPSTOOLS.

PATTERN_BUILD

This routine encodes a "pattern" into a special format to be used by the procedures **pattern_match** and **pattern_fga_match**. The "pattern" is returned in its encoded form in the variable of type PATTERN_TYPE. Both C and Pascal header files and example programs have been provided to assist in understanding how to use this procedure.

Syntax

```
int pattern_build (wp_pattern_string,
                   wp_pattern_length,
                    wp_pattern,
                    wp_error,
                    wp_wildcard_chars,
                    wp_options,
                    wp_chars_used);
```

The Parameter Set is listed next, where each parameter is either an integer array, character array, PATTERN_OPTIONS_TYPE, or PATTERN_TYPE:

Table 9.9 *PATTERN_BUILD's Parameter Set*

Parameter	Name	Type	Comment
1	wp_pattern_string	character array	Required
2	wp_pattern_length	integer 32-bit signed	Required
3	wp_pattern	PATTERN_TYPE	Required
4	wp_error	integer 32-bit signed	Required
5	wp_wildcard_chars	character array	Optional
6	wp_options	PATTERN_OPIONS_TYPE	Optional
7	wp_chars_used	integer 32-bit signed	Optional

Return Value

Pattern_build returns a 32-bit integer encoded as follow:

Table 9.10 *PATTERN_BUILD's Return Values*

Code	Definition	Error Description
<> 0	:Error	See the **wp_error** parameter.
= 0	:No error	

Parameters

wp_pattern_string

Byte array by reference (required). It contains the wildcard pattern that is to be initialized.

Example: **@.pub.sys**

wp_pattern_length

32-bit integer (required). It contains the byte length of the wildcard pattern stored in **wp_pattern_string**.

wp_pattern

PATTERN_TYPE data-structure by reference (required). This parameter is initialized within **pattern_build**, then subsequently passed to **pattern_match** or **pattern_fga_match**. The programmer is only responsible for declaring and passing this parameter.

wp_error

32-integer by reference (required). This parameter will contain an error number if the procedure return value is nonzero encoded as follows:

= 1: Too many firm (constant) characters in **wp_pattern_string** (see the following section of a discussion of firm characters).

= 2: Negative length

= 3: Too many parts (firms+wildcard characters) in **wp_pattern_string**

= 4: Escape, internal error (check **wp_pattern_string**)

wp_wildcard_chars

Byte array (optional). It contains the characters that will be used to represent wildcards.

byte 0: single character, default - ’?’

byte 1: multiple character wildcard, default = ’@’

byte 2: single digit wildcard, default = ’#’

byte 3: not used, must be an ASCII blank, default = ’ ’

See the following section for a discussion on setting this parameter.

wp_options

PATTERN_OPTIONS_TYPE data-structure by reference (optional). The parameter is used to select or deselect the following options:

- upshift before matching
- trim leading blanks
- trim trailing blanks

See the following section for a discussion on setting this parameter.

wp_chars_used

32-bit integer by reference (optional). It returns the number of characters used from **wp_pattern_string**. This normally equals the length of the pattern unless an error occurs.

Operation

FIRM CHARACTERS

A **firm character** is a character that is not a valid wildcard character. WILDCARD patterns are usually constructed of both wildcard and **firm characters**.

Example: **A@.PUB.W???**

The maximum number of **firm characters** that a pattern can contain is eight (8). Therefore the longest legal pattern is: **@1@2@3@4@5@6@7@8@**, or 17 characters long. If the pattern is longer than this, the **wp_error** parameter will be set up to three (**pb_err_many_parts**). If more than eight (8) firm characters are found, then the parameter **wp_error** will be set to one (**pb_err_many_firm**).

SETTING WP_OPTIONS

The default WILDCARD Pattern options are:

- Upshift pattern and strings before matching
- Trim (remove) leading spaces from strings before matching
- Trim (remove) trailing spaces from strings before matching

Each of these options are selected by enabling the appropriate entry in the PATTERN_OPTIONS_TYPE data structure.

Examples for the PATTERN_OPTIONS_TYPE data structure:

in C:

With the declaration

```
PATTERN_OPTIONS_TYPE          wp_options;
    wp_options.upshift               = 1      /* to select (default) */
    wp_options.upshift               = 0      /* to deselect */
    wp_options.trim_leading          = 1      /* to select (default) */
    wp_options.trim_leading          = 0      /* to deselect */
    wp_options.trim_trailing         = 1      /* to select (default) */
    wp_options.trim_trailing         = 0      /* to deselect */
```

In PASCAL:

With the declaration

```
wp_options: PATTERN_OPTIONS_TYPE;
   options        := options + [upshift]           { to select (default) }
   options        := options - [upshift]           { to deselect }
   options        := options + [trim_leading]      { to select (default) }
   options        := options - [trim_leading]      { to deselect }
   options        := options + [trim_trailing]     { to select (default) }
```

```
    options             := options - [trim_trailing]               { to deselect }
```

SETTING WP_WILDCARD _CHARS

The WILDCARD Pattern matching procedures can be programmed to accept any wildcard characters. By default, the WILDCARD Pattern matching procedures use the question mark (?) for any single character wildcard. The "at" sign (@) for any sequence of wildcards, and the "pound" sign (#) for any digit wildcard. MPE and DOS examples follow.

Examples for **wp_wildcard_chars** are:

In C:

With the declaration

```
    char pchars[4];
        strcpy(pchars,"?@#")        /* MPE style wildcards */
        strcpy(pchars,"?*#")        /* DOS style wildcards */
```

In Pascal:

With the declaration

```
    pchars:array[1..4] of char;
        pchars:="?@#";              { MPE style wildcards }
        pchars:="?*#";              { DOS style wildcards }
```

For example:

In C:

With the following declarations

```
    /* WILDCARD Pattern variables */

int
        wp_result,                          /* function returned */
        wp_error,                           /* error # if wp_result <> 0 */
        wp_buffer_length,                   /* function returned */
        wp_mismatches,                      /* returned by pattern_fga_match */
        wp_chars_used,                      /* # of chars used by wp_pattern_build */


char
```

```
        wp_buffer[256];                     /* buffer for passing strings to wp */
        wp_pchars[4];                       /* ptr to user definable wildcard set */

PATTERN_TYPE
        wp_pattern;                         /* internal representation */

PATTERN_OPTIONS_TYPE
        wp_options;                         /* used to select wp options */


    /* initialization code */
strcpy(wp_chars, "?@#");                    /* use default MPE wildcards */
wp_options.upshift=1;                       /* upshift before comparing */
wp_options.trim_leading=1;                  /* trim leading spaces */
wp_options.trim_trailing=1;                 /* trim trailing spaces */
strcpy(wp_buffer, "a##@");                  /* specify a pattern */
wp_buffer_length=strlen(wp_buffer);
wp_error=wp_chars_used=0;                   /* clear status variables - optional */
wp_result=pattern_build(wp_buffer, wp_buffer_length, &wp-file_pattern, &wp_error, wp_pchars
                    wp_options, &wp_chars_used);
If(wp_result != 0)
    /* report error */
strcpy(wp_buffer, "A69OUT");
wp_buffer_length=6;
wp_result=pattern_match(wp_buffer, wp_buffer_length, &wp_pattern);
If(wp_result == 0)
    /* report error */
else
    /* report no match */;
```

In Pascal:

With the following declarations

```
    { WILDCARD Pattern variables }

$include 'paspat.dec1.lpstools'$


var
        wp_buffer                  :packed array [1..80] of char;
        wp_error                   :integer;
        wp_buffer_length           :integer;
        wp_option                  :pattern_options_type;
        wp_pattern                 :pattern_type;
        wp_result                  :integer;
        wp_chars_used              :integer;
        wp_pchars                  :packed array [1..8] of char;

wp_chars := "?@#";                            { use default MPE wildcards }
wp_options := wp_options + [upshift]          { upshift before comparing }
wp_options := wp_options + [trim_leading]     { trim leading spaces }
wp_options := wp_options + [trim_trailing]    { trim trailing spaces }
wp_buffer := "a##@";                          { specify a pattern }
wp_buffer_length := 4;                        { the pattern's length }
wp_chars_used := 0;                           { clear status variables - optional }
wp_error := 0;
wp_result := pattern build (addr(wp_buffer), wp_buffer_length, wp_pattern, wp_error,
                        wp_pchars, wp_options, wp_chars_used);


if wp_result <> 0 then
    { report error }


wp_buffer := "A69OUT";
wp_buffer_length := 6;
wp-result := pattern_match(addr(wp_buffer), wp_buffer_length, wp_pattern);
```

```
if wp_result = 0 then
      { report error }
else
      { report no match };
```

PATTERN_FGA_MATCH

This procedure was specifically deigned to test a fully-qualified filename against a pattern. Since there are three components to an MPE fully-qualified filename, three patterns must be initialized (with **pattern_match**) before calling this procedure.

Syntax

```
int pattern_fga_match (fga_string,
                       file_pattern,
                       group_pattern,
                       account_pattern,
                       mismatches);
```

The Parameter Set is listed next, where each parameter is either an integer, character array, or PATTERN_TYPE:

Table 9.11 *PATTERN_FGA_MATCH's Parameter Set*

Parameter	Name	Type	Comment
1	fga_string	character array	Required
2	file_pattern	PATTERN_TYPE	Required
3	group_pattern	PATTERN_TYPE	Required
4	account_pattern	PATTERN_TYPE	Required
5	mismatches	integer 32-bit signed	Required

Return Value

Pattern_fga_match returns a 32-bit integer encoded as follow:

Table 9.12 *PATTERN_FGA_MATCH's Return Values*

Code	Definition	Error Description
= 0	:MATCH	
= 1	:NO MATCH	
= 2	: Internal error	(check input data for correctness)

See the C and Pascal header files for defines for the return values.

When a NO MATCH is returned, the variable mismatches can be tested to determine the components of the filename that failed to match.

Parameters

fga_string

Byte array (required). A fully-qualified string ASCII space terminated. This procedure expects that the components of the filename are separated by a dot (**.**). Also, the buffer containing this string should not contain any characters beyond the terminating space.

file_pattern

PATTERN_TYPE by reference (required). Initialized by a call to **pattern_build** with the desired filename wildcard pattern.

group_pattern

PATTERN_TYPE by reference (required). Initialized by a call to **pattern_build** with the desired groupname wildcard pattern.

account_pattern

PATTERN_TYPE by reference (required). Initialized by a call to **pattern_build** with the desired accountname wildcard pattern.

mismatches

32-bit integer by reference (required). This variable is used to determine which components of the filename failed. If the return value is zero, then the value of this variable should not be used. If the return value is 1 (NO MATCH), then this variable is encoded as follows:

= 1 : Account name NO MATCH

= 2 : Group name NO MATCH

= 3 : Account and group name NO MATCH

= 4 : File name NO MATCH

= 5 : Account and file name NO MATCH

= 6 : Group and file name NO MATCH

= 7 : Account and group and file name NO MATCH

Operation

As in the case with the **pattern_match** procedure, once the **pattern_build** procedure is used to build the **filename**, **groupname** and **accountname** patterns it can be called as many times as needed. A typical initialization sequence for this procedure might be:

```
wp_result = pattern_build(filename,...,filename_pattern...)

   ...

wp_result = pattern_build(groupname,...,groupname_pattern...)

   ...

wp_result = pattern_build(accountname,...,accountname_pattern...)

   ...

wp_result = pattern_fga_match(fga_string,...,filename_pattern)

   ...
```

See the file PATTEST.PASCAL.LPSTOOLS.

PATTERN_MATCH

This procedure is used to "test" for pattern matches. For input, it requires an initialized variable of type PATTERN_TYPE (see procedure **pattern_build**), a string to check, and the length of the string.

Syntax

```
int pattern_match (wp_buffer,

                   wp_buffer_length,

                   wp_pattern);
```

The Parameter Set is listed next, where each parameter is either an integer, character array, or PATTERN_TYPE:

Table 9.13 *PATTERN_MATCH's Parameter Set*

Parameter	Name	Type	Comment
1	wp_buffer	character array	Required
2	wp_buffer_length	integer 32-bit signed	Required
3	wp_pattern	PATTERN_TYPE	Required

Return Value

Pattern_match returns a 32-bit integer encoded as follow:

Table 9.14 *PATTERN_FGA_MATCH's Return Values*

Code	Definition	Error Description
= 0	:MATCH	
= 1	:NO MATCH	
= 2	: Internal error	(check input data for correctness)

See the C and Pascal header files for defines for the return values.

Parameters

wp_buffer

Byte array (required). It contains the string that is being tested for in the pattern (in wp_pattern)

NOTE Since the length is also given, the string doesn't have to be space or NULL terminated.

wp_buffer_length

32-bit integer (required). The length of the string in **wp_buffer**.

wp_pattern

PATTERN_TYPE data-structure by reference (required). This variable should have been initialized by a call to **pattern_build**.

Operation

Before calling this procedure, call the **pattern_build** procedure to initialize a variable of type PATTERN_TYPE. Then, initialize **wp_buffer** and **wp_buffer_length** and call **pattern_match**. Since wp_pattern is initialized external to this procedure, **pattern_match** can be called as many times as needed without reinitializing the **wp_pattern** variable (see the example for the **pattern_build** procedure).

10

The XDSMAP Tool

XDSMAP is a library of plug-compatible modules that intercept calls to extra data segment intrinsics and map these calls to mapped files. The result is that calls to DMOVIN and DMOVOUT can be up to 20 times faster than the original intrinsic.

Operation

XDSMAP does not use the Compatibility Mode extra data segments, but rather creates temporary files to store the data segments. All of the extra data segment intrinsics (GETDSEG, ALTDSEG, FREEDSEG, DMOVIN, and DMOVOUT) are intercepted by XDSMAP. Each intrinsic performs the same functional operation as the original. These intrinsics can be called from any native mode program that uses extra data segments.

NOTE Privileged access to extra data segments is not supported.

Each XDSMAP intrinsic functions in a manner consistent with the documented functionality in the Hewlett-Packard *Intrinsic Reference Manual*. For the sake of completeness, a brief description of each intrinsic is provided in this chapter. Also, a small test program has been provided so that you may test these intrinsics on your system. Results from tests run on an HP 3000 S/925 are provided for comparison.

Lastly, the XDSMAP intrinsics return all error codes and conditions as documented in the Hewlett-Packard *Intrinsic Reference Manual*.

Capabilities

No special capabilities are required.

Usage

XDSMAP is delivered as a Native Mode object file which can be either linked directly to your program (preferred method where performance issues are of primary concern) or placed in an executable library (NMXL) for run-time binding.

Relocatable Library

```
:link from=myprog.o,xdsmap.o;to=myprog;cap=ia,ba,ds;rl=xdsmap.rl
```

Executable Library

```
:run myprog;xl="XDSMAP.XL"
```

or

```
:link from=myprog.o,xdsmap.o;to=myprog;cap=ia,ba,ds;xl=xdsmap.xl
```

Intrinsic Summary

Listed below is a summary list of XDSMAP intrinsics.

Table 10.1 *XDSMAP Intrinsics*

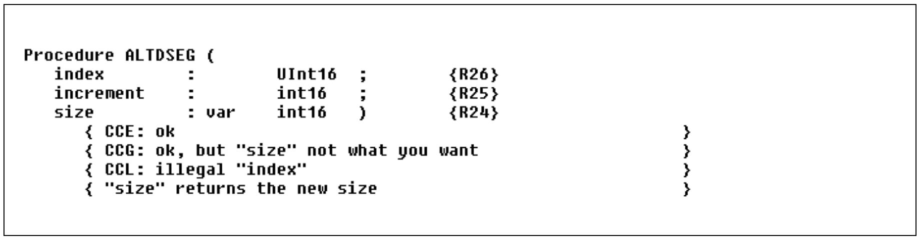
Intrinsic	Description
ALTDSEG	Adjusts size of extra data segment
DMOVIN	Copies data into caller's data area
DMOVOUT	Copies data into extra data segment
FREEDSEG	Deallocates memory
GETDSEG	Allocates extra data segment for process use

Intrinsics Definitions

Following is a detailed definition for each of the XDSMAP intrinsics.

ALTDSEG

This intrinsic is used to adjust the size (up or down) of an extra data segment. The size cannot be increased above the original value allocated by GETDSEG. The calling sequence is as follows:

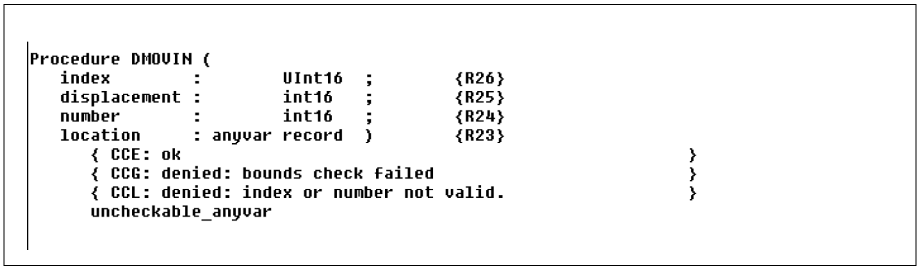


```
Procedure ALTDSEG (
   index        :        UInt16  ;        {R26}
   increment    :        int16   ;        {R25}
   size         : var    int16   )        {R24}
      { CCE: ok                                            }
      { CCG: ok, but "size" not what you want              }
      { CCL: illegal "index"                               }
      { "size" returns the new size                        }
```

Figure 10.1 *ALTDSEG Intrinsic*

DMOVIN

This intrinsic is used to copy data from the extra segment into the caller's data area. The calling sequence is as follows:

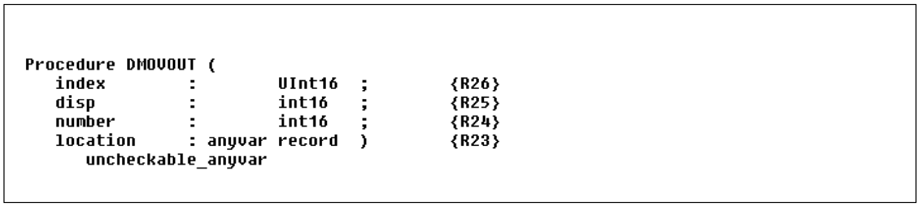


```
Procedure DMOVIN (
   index        :        UInt16  ;        {R26}
   displacement :        int16   ;        {R25}
   number       :        int16   ;        {R24}
   location     : anyvar record  )        {R23}
      { CCE: ok                                            }
      { CCG: denied: bounds check failed                   }
      { CCL: denied: index or number not valid.            }
      uncheckable_anyvar
```

Figure 10.2 *DMOVIN Intrinsic*

DMOVOUT

This intrinsic is used to copy data from the caller's data area into an extra data segment. The calling sequence is as follows:

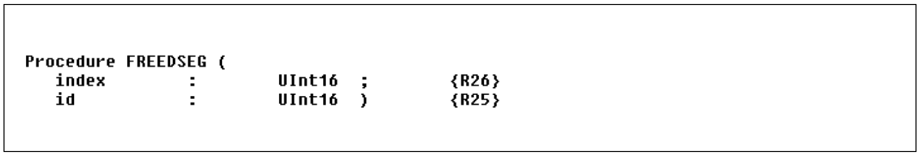


```
Procedure DMOVOUT (
   index        :        UInt16  ;        {R26}
   disp         :        int16   ;        {R25}
   number       :        int16   ;        {R24}
   location     : anyvar record  )        {R23}
      uncheckable_anyvar
```

Figure 10.3 *DMOVOUT Intrinsic*

FREEDSEG

This intrinsic is used to deallocate the memory allocated by the GETDSEG intrinsic. The calling sequence is as follows:

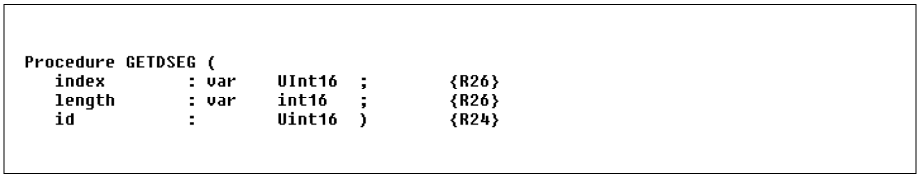


```
Procedure FREEDSEG (
   index        :        UInt16  ;        {R26}
   id           :        UInt16  )        {R25}
```

Figure 10.4 *FREEDSEG Intrinsic*

GETDSEG

This intrinsic is used for allocating or acquiring an extra data segment for use by a process. The calling sequence is as follows:

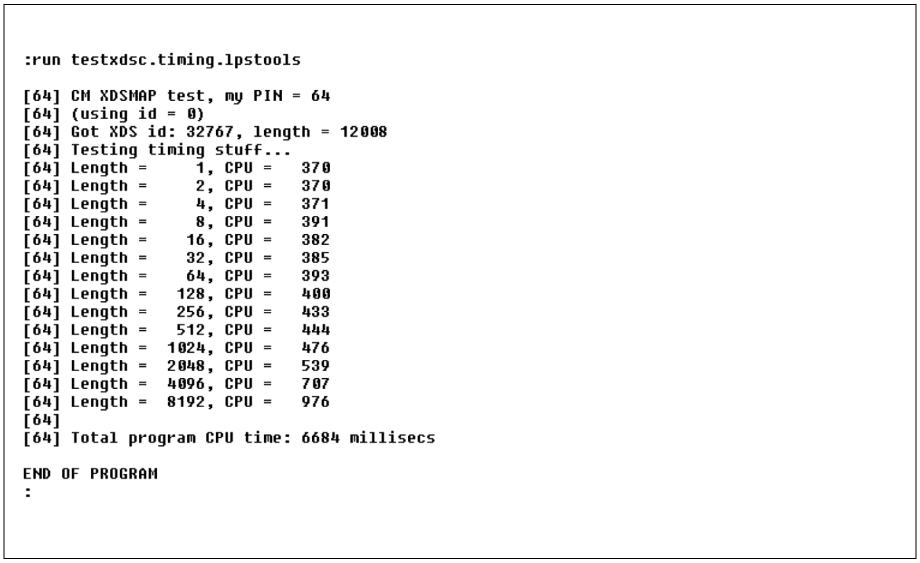


```
Procedure GETDSEG (
   index        : var    UInt16  ;        {R26}
   length       : var    int16   ;        {R26}
   id           :        Uint16  )        {R24}
```

Figure 10.5 *GETDSEG Intrinsic*

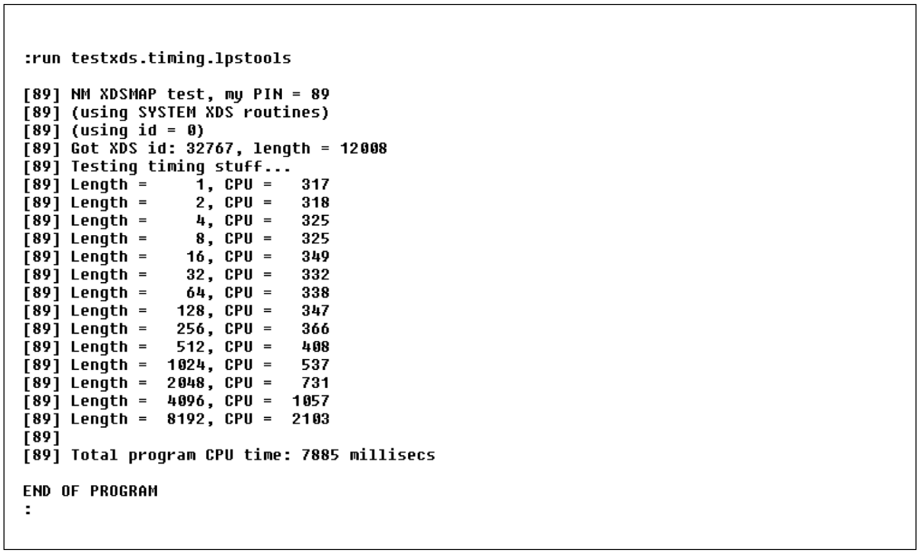
XDSMAP Examples

Sample test program results. See the file TESTXDS.SPL.LPSTOOLS.



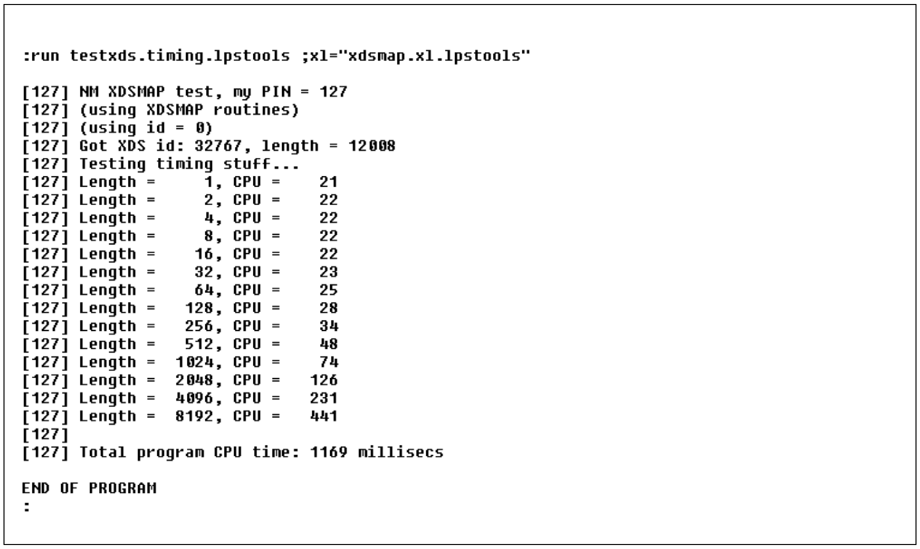
```
:run testxdsc.timing.lpstools

[64] CM XDSMAP test, my PIN = 64
[64] (using id = 0)
[64] Got XDS id: 32767, length = 12008
[64] Testing timing stuff...
[64] Length =     1, CPU =   370
[64] Length =     2, CPU =   370
[64] Length =     4, CPU =   371
[64] Length =     8, CPU =   391
[64] Length =    16, CPU =   382
[64] Length =    32, CPU =   385
[64] Length =    64, CPU =   393
[64] Length =   128, CPU =   400
[64] Length =   256, CPU =   433
[64] Length =   512, CPU =   444
[64] Length =  1024, CPU =   476
[64] Length =  2048, CPU =   539
[64] Length =  4096, CPU =   707
[64] Length =  8192, CPU =   976
[64]
[64] Total program CPU time: 6684 millisecs

END OF PROGRAM
:
```

Figure 10.6 *Compatibility Mode Output*



```
:run testxds.timing.lpstools

[89] NM XDSMAP test, my PIN = 89
[89] (using SYSTEM XDS routines)
[89] (using id = 0)
[89] Got XDS id: 32767, length = 12008
[89] Testing timing stuff...
[89] Length =     1, CPU =   317
[89] Length =     2, CPU =   318
[89] Length =     4, CPU =   325
[89] Length =     8, CPU =   325
[89] Length =    16, CPU =   349
[89] Length =    32, CPU =   332
[89] Length =    64, CPU =   338
[89] Length =   128, CPU =   347
[89] Length =   256, CPU =   366
[89] Length =   512, CPU =   408
[89] Length =  1024, CPU =   537
[89] Length =  2048, CPU =   731
[89] Length =  4096, CPU =  1057
[89] Length =  8192, CPU =  2103
[89]
[89] Total program CPU time: 7885 millisecs

END OF PROGRAM
:
```

Figure 10.7 *Native Mode Output Without XDSMAP*



```
:run testxds.timing.lpstools ;xl="xdsmap.xl.lpstools"

[127] NM XDSMAP test, my PIN = 127
[127] (using XDSMAP routines)
[127] (using id = 0)
[127] Got XDS id: 32767, length = 12008
[127] Testing timing stuff...
[127] Length =      1, CPU =    21
[127] Length =      2, CPU =    22
[127] Length =      4, CPU =    22
[127] Length =      8, CPU =    22
[127] Length =     16, CPU =    22
[127] Length =     32, CPU =    23
[127] Length =     64, CPU =    25
[127] Length =    128, CPU =    28
[127] Length =    256, CPU =    34
[127] Length =    512, CPU =    48
[127] Length =   1024, CPU =    74
[127] Length =   2048, CPU =   126
[127] Length =   4096, CPU =   231
[127] Length =   8192, CPU =   441
[127]
[127] Total program CPU time: 1169 millisecs

END OF PROGRAM
:
```

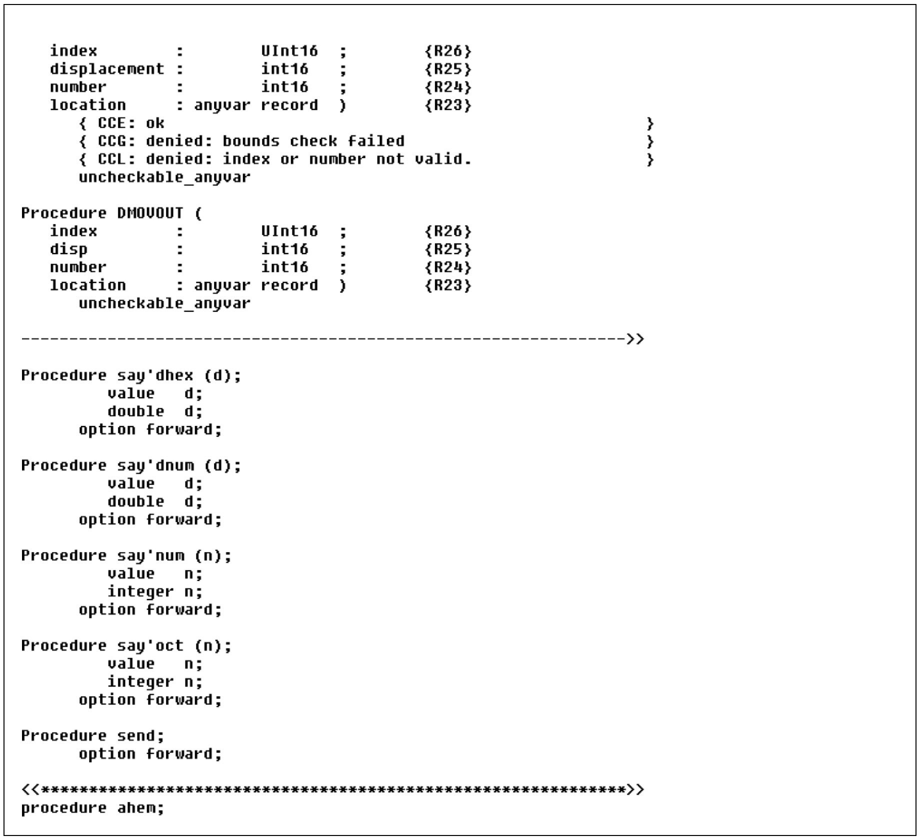
Figure 10.8 *Native Mode Output With XDSMAP*

The sample test program TESTXDS.SPL.LPSTOOLS:

```
<<testxds.spl  01/10/25       nm  cap=ds,ia,ba,ph,pm
ifcm -e testxds.cm
ifnm -o textxds.otemp
ifnm -e testxds.pub
>>

$control errors = 3

$set x9 = off                 ! OFF = SPL/V, ON = SPLash!
$if x9 = on or xsplash        ! SPL/V ignores the "OR XSPLASH"
$   set x9 = on               ! Yes, is SPLash!
$if

begin

equate
   max'pin        = 12000,    ! 7.0 Express 1 max (was 8192)
   max'xds'halfs  = max'pin + 8;    ! a few extra

$if x9 = on
define
   addr'type      = double #,
   virt           = virtual #;
$if x9 = off
define
   addr'type      = integer #,
   virt           = #;
$if               ! x9 = on/off

virt double array
   status         (0 : 0);

virt double pointer
   ptr'd;

real
   secs;

double
   ct;
```

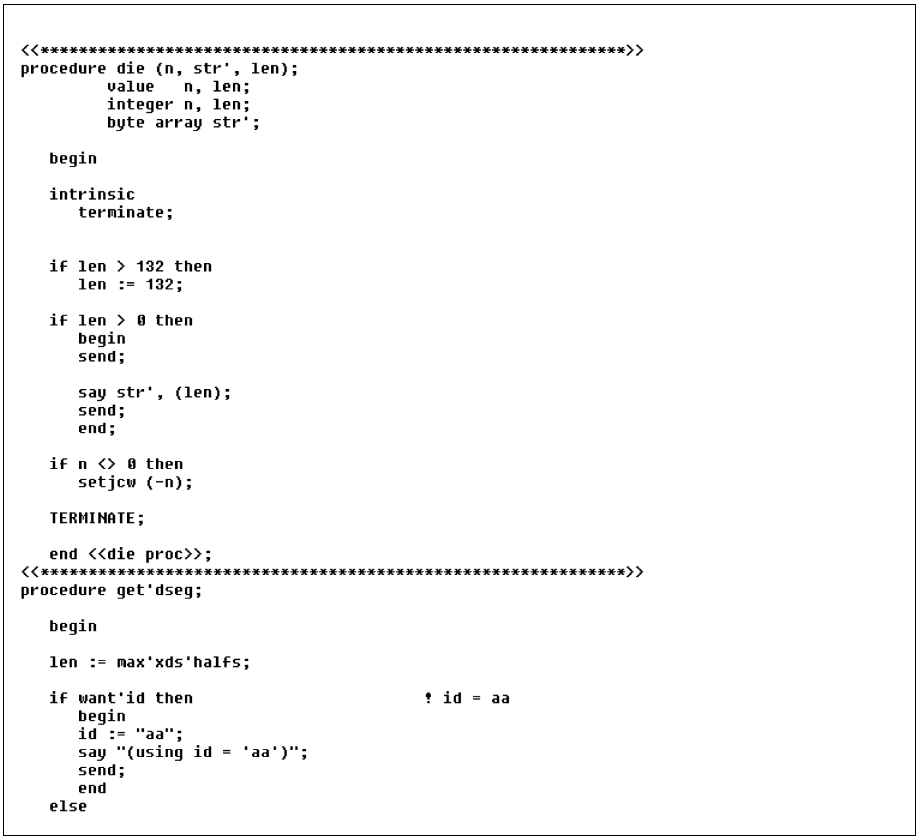
```
addr'type
   getdseg'addr,
   old'plabel;

integer array
   scratch        (0 : 255);

integer
   child'pin      := 0,
   err1,
   err2,
   icnt,
   len,
   my'pin         := 0,
   ocnt,
   outlen         := 0,
   parm           = q - 4,
   save'len,
   xds            := 0;

logical array
   outbuf         (0 : 255),
   xds'buf        (0 : max'xds'halfs);

logical
   child'parm     := 0,
   cy'hit         := false,
   flags          := 0,         ! set from PARM
   id             := 0,
   i'am'child     := false,
   i'am'dad       := false,
   save'flags     := 0;

byte array
   my'prog'       (0 : 27),
   outbuf'        (*) = outbuf,
   stringbuf'     (0 : 255);

define
   say            = outlen := outlen + move outbuf' (outlen) := #,
   sendstop       = begin
```

```
                    print (outbuf, - outlen, %320);
                    ahem;
                    end #,
   string         = stringbuf', move stringbuf' := #,

   want'debug     = flags.(01:01) #,
   want'i'am'child= flags.(02:01) #,
   want'quiet'child=flags.(03:01) #,
 ! ...
   want'id        = flags.(15:01) #,
   want'child     = flags.(14:01) #,
   want'pm        = flags.(13:01) #;

intrinsic
   activate,
   ascii,
   create,
   dascii,
   dmovin,
   dmovout,
   getdseg,
   getjcw,
   getprivmode,
   getprocinfo,
   getusermode,
   pause,
   print,
   procinfo,
   proctime,
   setjcw,
   resetcontrol,
   xcontrap;

$if x9 = on
Procedure give'up'cpu (status);
         double   status;
      option external, native, nocc;
$if               ! x9 = on

<<---------------------------------------------------------------

Procedure DMOVIN (
```



```
   index        :        UInt16  ;        {R26}
   displacement :        int16   ;        {R25}
   number       :        int16   ;        {R24}
   location     : anyvar record  )        {R23}
      { CCE: ok                                                    }
      { CCG: denied: bounds check failed                           }
      { CCL: denied: index or number not valid.                    }
      uncheckable_anyvar

Procedure DMOVOUT (
   index        :        UInt16  ;        {R26}
   disp         :        int16   ;        {R25}
   number       :        int16   ;        {R24}
   location     : anyvar record  )        {R23}
      uncheckable_anyvar

---------------------------------------------------------------->>

Procedure say'dhex (d);
         value   d;
         double  d;
      option forward;

Procedure say'dnum (d);
         value   d;
         double  d;
      option forward;

Procedure say'num (n);
         value   n;
         integer n;
      option forward;

Procedure say'oct (n);
         value   n;
         integer n;
      option forward;

Procedure send;
      option forward;

<<************************************************************>>
procedure ahem;
```

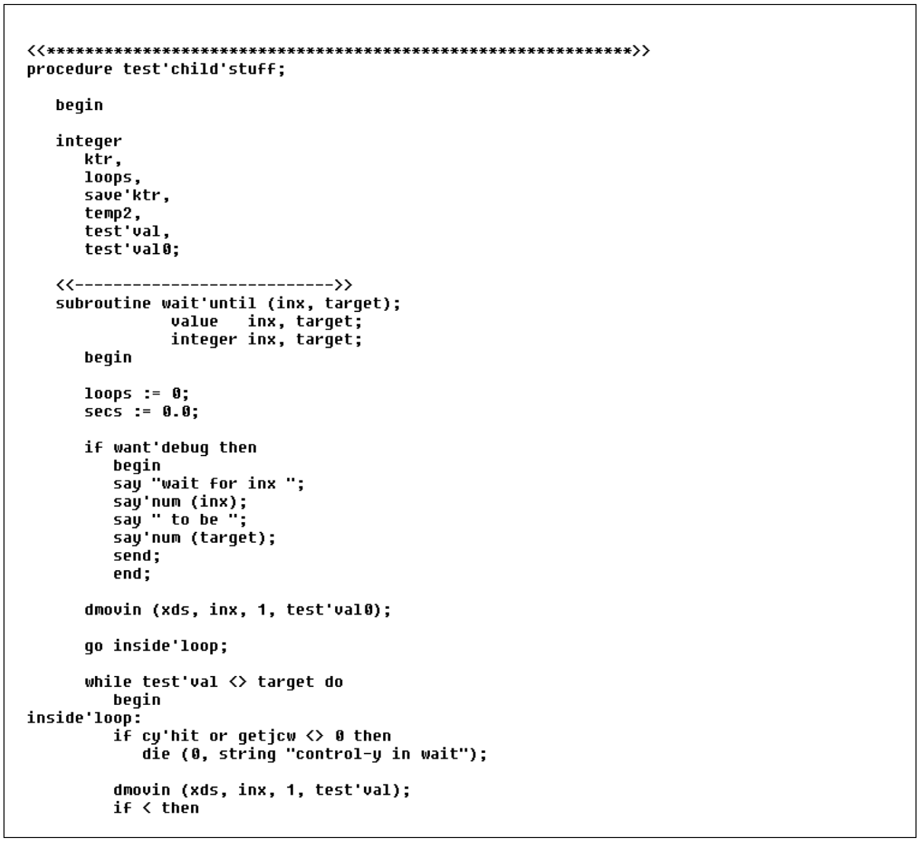
```
   begin

   outlen := 0;

   outbuf := "  ";
   move outbuf (1) := outbuf, (65);

   if my'pin <> 0 then
      begin
      say "[";
      if i'am'dad then
         say "dad  "
      else if i'am'child then
         say "child"
      else
         say'num (my'pin);
      say "] ";
      end;

   end <<ahem proc>>;
<<*************************************************************>>
procedure cy'handler;
$if x9 = on
      option native, nocc;
$if               ! x9 = on

   begin

   integer
      sdec        = q + 1;


   cy'hit := true;

   resetcontrol;

$if x9 = off
   tos := tos + %31400;
   assemble (xeq 0);
$if               ! x9 = off

   end <<cy'handler proc>>;
```



```
<<*************************************************************>>
procedure die (n, str', len);
         value   n, len;
         integer n, len;
         byte array str';

   begin

   intrinsic
      terminate;


   if len > 132 then
      len := 132;

   if len > 0 then
      begin
      send;

      say str', (len);
      send;
      end;

   if n <> 0 then
      setjcw (-n);

   TERMINATE;

   end <<die proc>>;
<<*************************************************************>>
procedure get'dseg;

   begin

   len := max'xds'halfs;

   if want'id then                        ! id = aa
      begin
      id := "aa";
      say "(using id = 'aa')";
      send;
      end
   else
```

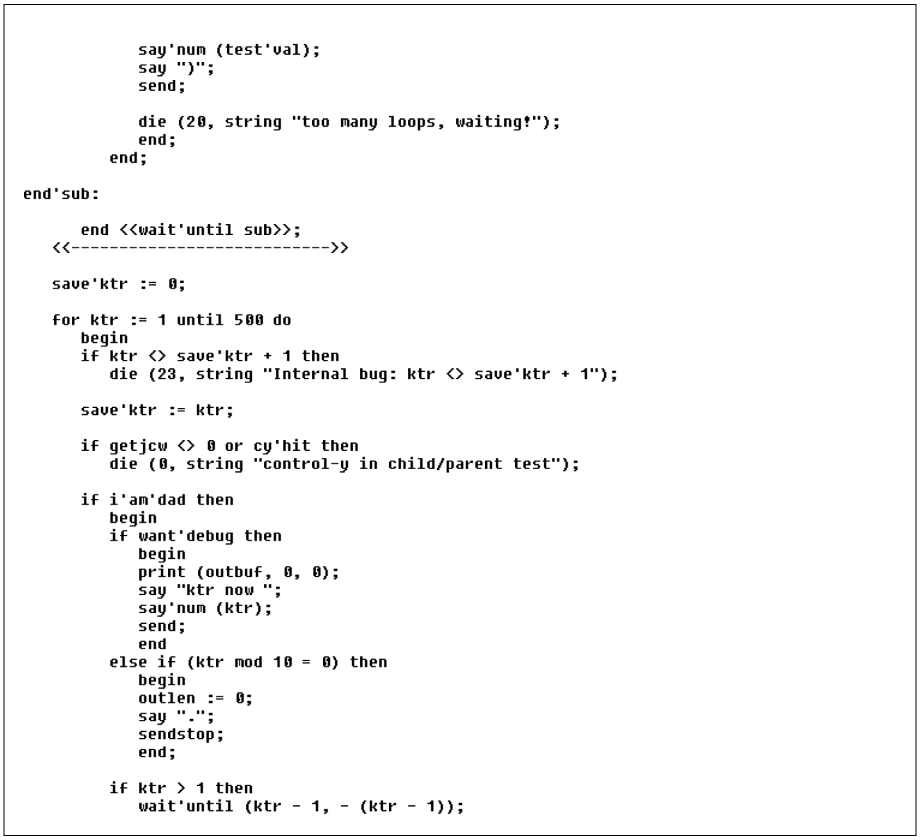
```
      begin
      say "(using id = 0)";
      send;
      id := 0;
      end;

   save'len := len;

   getdseg (xds, len, id);
   if < then
      begin
      say "GETDSEG failed, error: ";
      say'oct (xds);
      send;

      die (2, string "GETDSEG (20000, 'aa') failed");
      end;

   say "Got XDS id: ";
   say'num (xds);
   say ", length = ";
   say'num (len);
   if len <> save'len then
      begin
      say " (asked for ";
      say'num (save'len);
      say ")";
      end;
   send;

   end <<get'dseg proc>>;
<<*************************************************************>>
$if x9 = on
procedure priv'give'up'cpu;
      option native, privileged, nocc;

   begin

   give'up'cpu (status);

   end <<priv'give'up'cpu proc>>;
$if               ! x9 = on
<<*************************************************************>>
```

```
procedure say'dfmti (d, w);
         value   d, w;
         double  d;
         integer w;

   begin

   integer
      len;

   byte array
      scratch'    (0 : 15);


   if w > 32 then
      w := 32;

   len := dascii (d, 10, scratch');

         ! the following relies on outbuf' being blanked by ahem
   if w > len then
      outlen := outlen + (w - len);

   move outbuf' (outlen) := scratch', (len);
   outlen := outlen + len;

   end <<say'dfmti proc>>;
<<*************************************************************>>
procedure say'dhex (d);
         value   d;
         double  d;

   begin

   outbuf' (outlen) := "$";
   outlen := outlen + 1;
   dascii (d, 16, outbuf' (outlen));
   outlen := outlen + 8;

   end <<say'dhex proc>>;
<<*************************************************************>>
procedure say'dnum (d);
         value   d;
         double  d;

   begin

   outlen := outlen + dascii (d, 10, outbuf' (outlen));

   end <<say'dnum proc>>;
<<*************************************************************>>
procedure say'num (n);
         value   n;
         integer n;

   begin

   outlen := outlen + ascii (n, 10, outbuf' (outlen));

   end <<say'num proc>>;
<<*************************************************************>>
procedure say'oct (n);
         value   n;
         integer n;

   begin

   outbuf' (outlen) := "%";
   outlen := outlen + 1;

   ascii (n, 8, outbuf' (outlen));
   outlen := outlen + 6;

   end <<say'oct proc>>;
<<*************************************************************>>
procedure send;

   begin

   if i'am'child and want'quiet'child then
   else
      print (outbuf, - outlen, 0);

   ahem;

   end <<send proc>>;
```



```
<<*************************************************************>>
procedure test'child'stuff;

   begin

   integer
      ktr,
      loops,
      save'ktr,
      temp2,
      test'val,
      test'val0;

   <<---------------------------->>
   subroutine wait'until (inx, target);
               value   inx, target;
               integer inx, target;
      begin

      loops := 0;
      secs := 0.0;

      if want'debug then
         begin
         say "wait for inx ";
         say'num (inx);
         say " to be ";
         say'num (target);
         send;
         end;

      dmovin (xds, inx, 1, test'val0);

      go inside'loop;

      while test'val <> target do
         begin
inside'loop:
         if cy'hit or getjcw <> 0 then
            die (0, string "control-y in wait");

         dmovin (xds, inx, 1, test'val);
         if < then
```

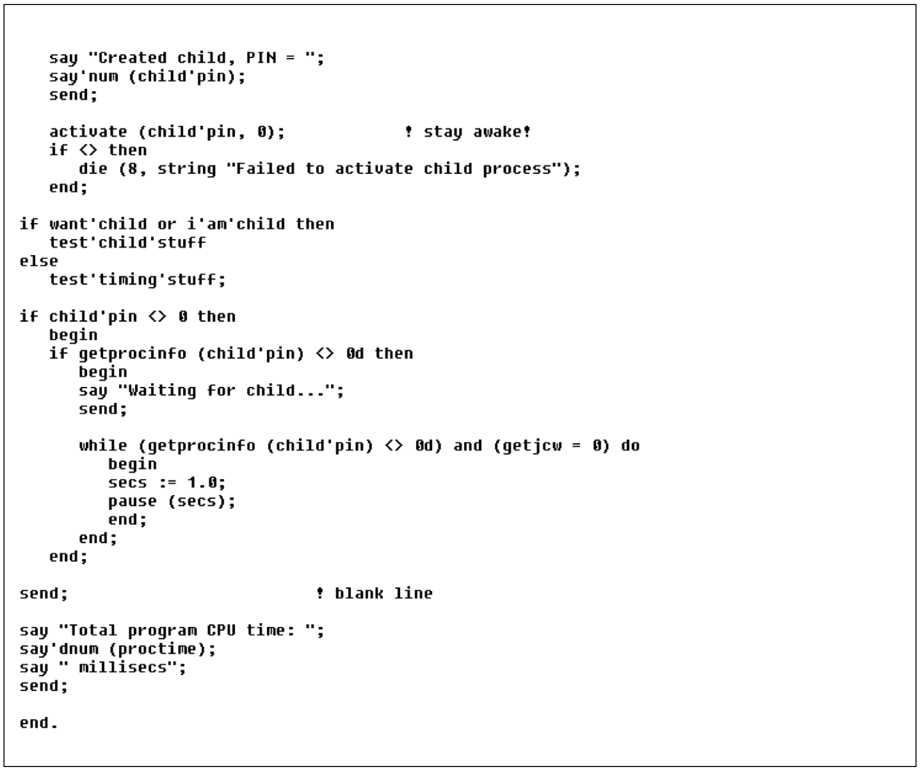
```
            die (11, string "dmovin (in wait) failed: CCL")
         else if > then
            die (11, string "dmovin (in wait) failed: CCG");

         if test'val = target then
            go end'sub;

$if x9 = on
         priv'give'up'cpu;
$if               ! x9 = on
         if (loops := loops + 1) >= 199 then
            begin
            secs := secs + 0.1;
            pause (secs);
            end;

         if loops > 203 then
            begin
            say "OOPS: too many waiting loops.  Last val = ";
            say'num (test'val);
            say ", looking for ";
            say'num (target);
            say " at inx ";
            say'num (inx);
            send;

            say "(first val found: ";
            say'num (test'val0);
            say ")";
            send;

            dmovin (xds, inx, 1, test'val);
            say "(value now ";
            say'num (test'val);
            say ")";
            send;

            secs := 2.0;

            pause (secs);

            dmovin (xds, inx, 1, test'val);
            say "(value now ";
```



```
            say'num (test'val);
            say ")";
            send;

            die (20, string "too many loops, waiting!");
            end;
         end;

end'sub:

      end <<wait'until sub>>;
   <<---------------------------->>

   save'ktr := 0;

   for ktr := 1 until 500 do
      begin
      if ktr <> save'ktr + 1 then
         die (23, string "Internal bug: ktr <> save'ktr + 1");

      save'ktr := ktr;

      if getjcw <> 0 or cy'hit then
         die (0, string "control-y in child/parent test");

      if i'am'dad then
         begin
         if want'debug then
            begin
            print (outbuf, 0, 0);
            say "ktr now ";
            say'num (ktr);
            send;
            end
         else if (ktr mod 10 = 0) then
            begin
            outlen := 0;
            say ".";
            sendstop;
            end;

         if ktr > 1 then
            wait'until (ktr - 1, - (ktr - 1));
         if want'debug then
            begin
            say "set ";
            say'num (ktr);
            say " to ";
            say'num (ktr);
            send;
            end;

         dmovout (xds, ktr, 1, ktr);
         end

      else
         begin
         wait'until (ktr, ktr);
         if want'debug then
            begin
            say "set ";
            say'num (ktr);
            say " to ";
            say'num (-ktr);
            send;
            end;

         temp2 := - ktr;
         dmovout (xds, ktr, 1, temp2);
         end;

      if ktr <> save'ktr then
         die (22, string "internal bug: ktr changed value!");
      end;                             ! for ktr...

   if i'am'dad then
      begin
      secs := 0.5;
      pause (secs);
      end;

   say "test ok!";
   send;

   end <<test'child'stuff proc>>;
<<************************************************************>>
```

```
procedure test'timing'stuff;

   begin

         ! init XDS with my PIN at index my'pin

   say "Testing timing stuff...";
   send;

   dmovout (xds, my'pin, 1, my'pin);
   if <> then
      die (3, string "dmovout #1 failed");

   dmovin (xds, my'pin, 1, scratch);
   if <> then
      die (4, string "dmovin #1 failed");

   if my'pin <> scratch then
      die (5, string "dmovin #1 data mismatch");

         ! time accessing xds...

   ocnt := 1;

   while ocnt <= max'xds'halfs do
      begin
      if getjcw <> 0 then
         cy'hit := true;

      if cy'hit then
         die (0, string "control-Y");

      ct := proctime;

      for icnt := 1 until 1000 do
         begin
         dmovout (xds, 0, ocnt, xds'buf);
         dmovin  (xds, 0, ocnt, xds'buf);
         if cy'hit then
            die (0, string "Control-Y");
         end;

      ct := proctime - ct;
      say "Length = ";
      say'dfmti (double (ocnt), 5);
      say ", CPU = ";
      say'dfmti (ct, 5);
      send;

      if (ct > 2000d) and (ocnt = 1) then
         die (6,
              string "Sorry...the XDS stuff seems to be *real* slow!");

      ocnt := ocnt * 2;
      end;

   end <<test'timing'stuff proc>>;
<<*************************************************************>>

setjcw (0);

flags := parm;

procinfo (err1, err2, 0 <<my pin>>, 1, my'pin);    ! get my PIN

ahem;

$if x9 = on
say "NM XDSMAP test, my PIN = ";
$if x9 = off
say "CM XDSMAP test, my PIN = ";
$if               ! x9 = on/off

say'num (my'pin);
send;

$if x9 = on
      ! get address of GETDSEG procedure (actually, of
      ! the XRT entry for it)

getdseg'addr := @getdseg;

      ! But, an XRT address is of the form:  $4xxxxxx1
      ! so we want to subtract that trailing "1"...
```

```
getdseg'addr := getdseg'addr - 1d;

@ptr'd := getdseg'addr;

if ptr'd = 10d then
   say "(using SYSTEM XDS routines)"
else
   say "(using XDSMAP routines)";
send;

if want'debug then                     ! debug
   begin
   say "GETDSEG @ ";
   say'dhex (ptr'd);
   say ".";
   say'dhex (ptr'd (1));
   send;
   end;
$if               ! x9 = on

if my'pin >= max'xds'halfs then
   begin
   say "Sorry, this test program cannot run...";
   send;
   say "It requires that the PIN of the process is <= ";
   say'num (max'xds'halfs);
   send;

   die (1, string "unexpectedly high PIN value");
   end;

xds'buf := 0;
move xds'buf (1) := xds'buf, (max'xds'halfs - 1);

i'am'child := want'i'am'child;

if want'pm then                        ! PM
   begin
   say "Using PM!";
   send;

   getprivmode;
```



```
   say "Created child, PIN = ";
   say'num (child'pin);
   send;

   activate (child'pin, 0);            ! stay awake!
   if <> then
      die (8, string "Failed to activate child process");
   end;

if want'child or i'am'child then
   test'child'stuff
else
   test'timing'stuff;

if child'pin <> 0 then
   begin
   if getprocinfo (child'pin) <> 0d then
      begin
      say "Waiting for child...";
      send;

      while (getprocinfo (child'pin) <> 0d) and (getjcw = 0) do
         begin
         secs := 1.0;
         pause (secs);
         end;
      end;
   end;

send;                         ! blank line

say "Total program CPU time: ";
say'dnum (proctime);
say " millisecs";
send;

end.
```

Figure 10.9 *Test Program for XDSMAP*

XDSMAP Error Messages

See the HP Intrinsic Reference Manual for possible error conditions.

A

UNSUPPORTED OPERATING SYSTEMS

If *Developer's Toolbox* is run on a version of the operating system that it doesn't know, it will terminate with either one of these messages:

```
This is an unknown version of MPE/iX
```

```
This version of MPE/iX is unfamiliar
```

The reason for these messages is that some of the tools may be sensitive to MPE/iX operating system changes. When these changes are detected, one of the warning messages will be displayed. If you get one of these messages, you may want to contact LPS to determine if the version of MPW/iX that you are running is compatible with tools operations.

There are two ways to override the operating system check, both of which involve setting a JCW.

At the MPE/iX prompt, type:

```
:setjcw LPSMPEOK 1
```

This allows the tool to acknowledge the unknown operating system's presence without terminating.

Or, you may type:

```
:setjcw LPSMPEOK 3
```

This allows the tool to quietly continue.

B

MPE File Codes

This appendix has been included in order to provide you with a convenient way to look up file code information that is displayed when you use Toolbox utilities like BLAZE, REP, or any other tool that presents filecode information.

File codes are recorded in the file label and are available to process accessing the file through the FFILEINFO or FGETINFO intrinsic. Although any user can specify a positive integer ranging from 0 to 32767 or the mnemonic name for this parameter, certain reserved integers and mnemonic have particular system-defined meanings. This table defines the MPE reserved integer and mnemonic values.

NOTE Default is file code 0.

Table B.1 *MPE reserved integer and mnemonic values*

Integer	Mnemonic	Meaning
1024	USL	User Subprogram Library
1025	BASD	Basic Data
1026	BASP	Basic Program
1027	BASFP	Basic Fast Program
1028	RL	Compatibility Mode Relocatable Library
1029	PROG	Compatibility Mode Program File
1030	NMPROG	Native Mode Program File
1031	SL	Segmented Library
1032	NMXL	Native Mode Executable Library
1033	NMRL	Native Mode Relocatable Library
1035	VFORM	VPLUS Forms File

Integer	Mnemonic	Meaning
1036	VFAST	VPLUS Fast Forms File
1037	VREF	VPLUS Reformat File
1040	XLSAV	Cross Loader ASCII File (SAVE)
1041	XLBIN	Cross Loader Relocated Binary File
1042	XLDSP	Cross Loader ASCII File (DISPLAY)
1050	EDITQ	Edit Quick File
1051	EDTCQ	Edit KEEPQ File (COBOL)
1052	EDTCT	Edit TEXT File (COBOL)
1054	TDPDT	TDP Diary File
1055	TDPQM	TDP Proof Marked File QMARKED
1056	TDPP	TDP Proof Marked non-COBOL File
1057	TDPCP	TDP Proof Marked COBOL File
1058	TDPQ	TDP Work File
1059	TDPXQ	TDP Work File COBOL
1060	RJEPN	RJE Punch File
1070	QPROC	QUERY Procedure File
1080	KSAMK	KSAM Key File
1083	GRAPH	GRAPH Specification File
1084	SD	Self-describing File
1090	LOG	User Logging Log File
1100	WDOC	HPWORD Document
1101	WDICT	HPWORD Hyphenation Dictionary
1102	WCONF	HPWORD Configuration File
1103	W2601	HPWORD Attended Printer Environment
1110	PCELL	IFS/3000 Character Cell File
1112	PENV	IFS/3000 Environment File

Integer	Mnemonic	Meaning
1113	PCCMP	IFS/3000 Compiled Character Cell File
1114	RASTR	Graphics Image in RASTER Format
1130	OPTLF	OPT/3000 Log File
1131	TEPES	TEPE/3000 Script File
1132	TEPEL	TEPE/3000 Log File
1133	SAMPL	APS/3000 Log File
1139	MPEDL	MPEDC/DRP Log File
1140	TSR	HPToolset Root File
1141	TSD	HPToolset Data File
1145	DRAW	Drawing File for HPDRAW
1146	FIG	Figure File for HPDRAW
1147	FONT	Reserved
1148	COLR	Reserved
1149	D48	Reserved
1152	SLATE	Compressed SLATE File
1153	SLATW	Expanded SLATE Work File
1156	DSTOR	RAPID/3000 DICTDBU Utility Store File
1157	TCODE	Code File for Transact/3000 Compiler
1158	RCODE	Code File for Report/3000 Compiler
1159	ICODE	Code File for Inform/3000 Compiler
1166	MDIST	HPDESK Distribution List
1167	MTEXT	HPDESK Text
1168	MARPA	ARPA Messages File
1169	MARPD	ARPA Distribution List
1170	MCMND	HPDESK Abbreviated Commands File
1171	MFRTM	HPDESK Diary Free Time List

Integer	Mnemonic	Meaning
1172	None	Reserved
1173	MEFT	HPDESK External File Transfer Messages File
1174	MCRPT	HPDESK Encrypted Item
1175	MSERL	HPDESK Serialized (Composite) Item
1176	VCSF	Version Control System File
1177	TTYPE	Terminal Type File
1178	TVFC	Terminal Vertical Format Control File
1192	NCONF	Network Configuration File
1193	NTRAC	Network Trace File
1194	NTLOG	Network Log File
1195	MIDAS	Reserved
1211	NDIR	Reserved
1212	INODE	Reserved
1213	INVRT	Reserved
1214	EXCEP	Reserved
1215	TAXON	Reserved
1216	QUERF	Reserved
1226	VC	VC File
1227	DIF	DIF File
1228	LANGD	Language Definition File
1229	CHARD	Character Set Definition File
1230	MGCAT	Formatted Application Message Catalog
1236	BMAP	Base Map Specification File
1242	BDATA	HP Business BASIC/V Data File
1243	BFORM	HP Business BASIC/V Field Order File for VPLUS

Integer	Mnemonic	Meaning
1244	BSAVE	HP Business BASIC/V SAVE Program File
1245	BCNFG	Configuration File for Default Options for HP Business
BASIC Programs		
1246	BKEY	Function Key Definition File for Terminal
1258	PFSTA	Pathflow STATIC File
1259	PFDYN	Pathflow Dynamic File
1270	RFDCA	Revisable From DCA Data Stream
1271	FFDCA	Final Form DCA Data Stream
1272	DIU	Document Interchange Unit File
1273	PDOC	HPWORD/150 Document
1275	DFI	DISOSS Filing Information File
1276	SRI	Search Restart Information File
1401	CWPTX	Chinese Word Processor Text File
1421	MAP	HPMAP/3000 Map Specification File
1422	GAL	Reserved
1425	TTX	Reserved
1428	RDIL	HP Business Report Writer (BRW) Dictionary File CM
1429	RSPEC	BRW Specification File
1430	RSPCF	BRW Specification File
1431	REXCL	BRW Execution File
1432	RJOB	BRW Report 509 File
1433	ROUTI	BRW Intermediate Report File
1434	ROUTD	BRW Dictionary Output
1435	PRINT	BRW Print File

Integer	Mnemonic	Meaning
1436	RCONF	BRW Configuration File
1437	RDICN	BRW NM Dictionary File
1438	REXNUM	BRW NM Execution File
1441	PIF	Reserved
1461	NMOBJ	Native Mode Object File
1462	PASLIB	Pascal XL Source Library
1476	TIFF	Tag Image File Format
1477	RDF	Revisable Document Format
1478	SOF	Serial Object File
1479	GPF	Chart File for Charting Gallery Chart
1480	GPD	Data File for Charting Gallery Chart
1483	VCGPM	Virtuoso Core Generator Processed Macro File
1484	FRMAT	Formatter
1485	DUMP	Dump Files Created and Used by IDAT and DPAN
1486	NNMD0	New Wave Mail Distribution List
1491	X4HDR	X.400 Header for HP Desk Manager
1500	WP1	Reserved
1501	WP2	Reserved
1502	LO123	Lotus 123 Spread Sheet
1514	FPCF	Form Tester Command Spec File
1515	INSP	Spooler XL Input Spoolfile
1516	OUTSP	Spooler XL Output Spoolfile
1517	CHKSP	Spooler XL Checkpoint Spoolfile
1521	DSKIT	HPDesk Intrinsics Transaction File
1526	MSACK	Man Server Acknowledgment

Integer	Mnemonic	Meaning
1527	MSNON	Man Server Non-Delivery Notification
1528	MSTRC	Man Server Trace File
3333		Reserved

C

LISTF FILESET

In some commands, you may substitute wildcard characters for certain parameters, or parts of parameters, in the list. The wildcard characters count toward the eight character limit for user, group, account, and file names. These wildcard characters are defined in the table below.

Wildcard Characters Definitions

Table C.1 *Wildcard Characters Definitions*

Character	Function
@	Specifies zero or more alphanumeric characters. When used by itself, @ denotes all possible members of the set.
#	Specifies one numeric character.
?	Specifies one alphanumeric character.

Wildcard Characters Examples

The above characters can be used as follows:

Table C.2 *Wildcard Characters Examples*

Example	Description
n@	Represents all items starting with the character "n".
@n	Represents all items ending with the character "n".
n@x	Represents all items starting with the character "n" and ending with the character "x".
n###	Represents all items starting with the character "n" followed by three digits, where each digit is represented by a single number (#) sign. (The "n" may be followed by up to seven number (#) signs.)

Example	Description
?n@	Represents all items whose second character is "n".
n?	Represents all two-character items starting with the character "n".
?n	Represents all two-character items ending with the character "n".

D

STANDARD WINDOWING TERMS AND FEATURES

This section explains the terminology used in describing windows-based tools. Standard features, like function keys, are discussed in the next appendix.

Table D.1 *Standard Windowing Terms*

Term	Description
User Input	Monospace typeface (i.e., enter EXFORMF into the filename field).
Window	A rectangular area that occupies a portion of the screen and is used to display information that can be viewed easily.
Menus	
Menu Bar	A single line menu (usually shaded) with the items displayed horizontally across the top of the display.
Pull-down Menu	A menu of various options that extends down from an item in a menu bar. Menu items whose names include an ellipsis (i.e., "Forms..") have pull-down menus attached to them. You may select items within a pull-down menu the same way you would in any menu.
Arrow Keys	Horizontal arrow keys are used to move along the menu bar and choose a menu to open. You may open a menu by highlighting it in the menu bar, then pressing **Enter/ Return**. A down arrow key can also be used to open a menu. To select an item within a menu, move up and down in the menu using the vertical arrow keys. You may also type the first letter of the menu item you want twice, in quick succession, and the cursor will jump to that item starts with the same letter, the cursor will jump to the first menu item it finds with that letter.

Term	Description
Menu Walking	If you have a pull-down menu already open, using the horizontal arrow keys will automatically open neighboring pull-down menus as you move to them.
Scrolling	
Vertical Scrolling	The fastest way to move through the file. Vertical scrolling is done by using the **Prev** and **Next** keys or the **PgUp** and **PgDn** keys on a PC.
Arrow Keys	The arrow keys also scroll, but only one line or column at a time.
Scroll Lock	If you have Scroll Lock enabled for your terminal or PC, your arrow keys will not function properly. It is best to leave the Scroll Lock off.

E

STANDARD FUNCTION KEYS

This section describes a few of the standard function keys typically found in a windows-based Toolbox. Non-standard function keys that are used for Toolbox-specific operations are not covered here. Only common keys, like **Help** and **Print**, are discussed here.

NOTE Function keys are context-sensitive. This means that depending on which screen is active, some or all of these functions will be available for you to use.

HELP

Context-sensitive **Help** is always assigned to the F1 key. When F1 is pressed, a pop-up help window appears on top of the current display. This window will have a title that describes the general subject of the help material. Within the window, the cursor keys, and keypad keys (**PgUp**, **Home**, etc.) can be used to navigate through the text.

For the most part, the help text displayed in the window is based on the action you are trying to accomplish. Once the text is displayed, you can browse through the entire Help subsystem.

The help text for a ***Toolbox*** utility is stored in a text file in the HELP group. If you want, this text can be modified to better suit your needs.

PRINT

Pressing **F2** outputs a "snap-shot" of the current screen display to either a printer of a disk file. The formal file designator for the output file is LP. Output can be directed to the system line printer by issuing the following file equation:

```
:FILE LP;DEV=LP
```

If no file equation is defined for LP, the output is directed to a disk file with the name LP. To direct output to a file with a different name, use a file equation of the form:

```
:FILE LPSLP=myfile
```

REFRESH

This function is not always available. When it is available, it is typically accessed through the F5 function key.

The purpose of this operation is to refresh the entire screen display. This is occasionally necessary due to "noisy" connections to the host computer or operator messages that may disrupt the screen.

Most windowed ***LPS-Tools*** usually operate in QUIET mode, so TELL messages will not corrupt the display. WARN messages, however, cannot be avoided.

ACCEPT

This function is not always available. When it is available, it is typically accessed through the F6 function key. The purpose of this function is to accept user input from a data entry form.

NOTE In **Character mode**, this key has the same effect as the **Return** key. In **Block mode**, this key is used instead of the **Return** key.

PREVIOUS and NEXT

PREVIOUS is used in data entry windows to return to the previous field or in menus to return to the previous menu option.

NEXT is used to move to the next data entry field or menu option.

CANCEL or EXIT

These functions are typically available through the F8 function key.

CANCEL is used to terminate the current activity and return you to the previous level of activity. EXIT simply terminates the program.

ZOOM

This function key provides two functions: ZOOM IN and ZOOM OUT. The function key label displays the active function.

ZOOM IN enlarges the current window to take up the entire screen, while ZOOM OUT returns the enlarged window to its original size.

F

THE MODIFY EDITOR

MODIFY is a single-line visual mode editor used for all REDO commands and, to a greater extent, in a few of the tools.

Operations

MODIFY displays your changes on the screen as you type. The cursor rests on the same line as the text you are editing. If you type any printable key, that key will either replace the character the cursor was on, or insert the key before that character, depending on the mode. Initially, you are in transparent mode. Here, a blank will simply cause the cursor to move one space to the right. Typing any other printable character terminates transparent-mode and puts you in overwrite-mode, so the character will replace the one the cursor is on.

The 3 basic modes are:

Table F.1 *Basic Modes*

Mode Types	To Enter Mode	To Exit Mode
transparent	^T	any printable char, ^B, ^O, or ^X
overwrite	^O	^T, ^B, or ^X
insert	^B or ^	^T, ^O, or ^X

You cannot create a line longer than the maximum specified by the calling program, nor can you accidentally "lose" characters off the right edge when using insert-mode. A beep will sound when you try to execute an illegal action.

The editor has an extensive set of commands, all of which are invoked via control-characters. MODIFY is case-sensitive. A few commands are meaningful only when this editor is used from within QEDIT from Robelle Consulting, Ltd. For more information on QEDIT, consult the documentation that comes with the product.

Char	Mnemonic	Description
^A	append	Goto end-of-line. Moves the cursor to just after the last character on the line. If the line is already at the maximum length, the cursor will be placed on the last character.
^B	before	Turn on insert-mode. Turns off overwrite-mode. If you enter a character while in insert-mode, it will be put before the character the cursor is on, and the rest of the line will move to the right one.
^	before	Control up-arrow (synonym of ^B). Use ^^ instead of ^B if you are on a system console.
^C	case	Change case of current character. If the current character is a lowercase letter, it will be changed to an uppercase letter and vice-versa.
^D	delete	Delete character. Typing ^D will cause the character under the cursor to be deleted and the rest of the line moved one space to the left.
^L^D	delete end	If the cursor is just past the last character (i.e., you just did a ^L or ^A), then the ^D will delete the last character of the line.
^E	erase	Erase to end of line. This will erase all of the text from the cursor to the end of line.
^F<c>	find	Find next occurrence of "c". The cursor will be moved to the first occurrence of the character "c" to the right of the cursor. If "c" is not found, you will hear a beep.
^F<n><c>	find	Find "nth" occurrence of "c" (1<=n<=8)
^G	goof	Undo all current modifications. Restores the line of text to its original form. **NOTE** ^V, ^K, ^T^D, and ^T^V cannot be undone.
^H	backspace	Move back one char (non-destructive)
^I	tab	Skip 10 characters to right.
^J	justify	Deletes blanks from cursor to the first non-blank (does not delete that character).

Char	Mnemonic	Description
^K	add	Requests QEDIT to add a line after current line. The current line will then be re-displayed for editing and you will get to edit the new line.
^L	lengthen	Goto end-of-line (synonym of ^A). Use ^L instead of ^A if you are on a Type Ahead Engine (TAE).
^M	return	Marks end of editing a line. Returns to the caller (i.e., QEDIT) the modified line. **NOTE** ^M is the same as the Return key.
^O	overwrite	Initiate overwrite-mode and also turn off insert-mode (^B). In overwrite-mode, if you enter a character it will replace the one on the screen (i.e., overwrite it).
^P<#> <dir>		Move up/down some number of lines of text (only applicable) from QEDIT). For example, "^P3-" moves back 3 lines.
^Q	query	Displays Help information.
^S<c>	scan	Find previous occurrence of "c". The cursor will be moved to the first occurrence of "c" to the left of the current cursor position. If "c" is not found, you will hear a beep.
^S<n><c>	scan	Find nth occurrence of "c" (1<=n<=8).
^T	transparent	Terminates insert-mode and overwrite-mode. After ^T, if you type blanks, the cursor will simply move right one space without affecting the text. Transparent-mode is always turned off automatically whenever a non-blank printable character is entered, then overwrite-mode is turned on.
^T^D	delete	If done at column 1, will request caller to delete the line.
^T^V	splice	If done at column 1, will request caller to join the next line to the end of the current line. The newly spliced line will be displayed for editing.

Char	Mnemonic	Description
^U	jUmpback	Move 10 characters to left. This is the opposite of ^I. As an aid to remembering them, ^I is the same as hitting the tab key, and ^U is just to the left of ^I on the keyboard.
^V	split	Split current line (at cursor) into two lines and modify both of them.
^X	eXamine	Examine (redisplay) current line.
^Y	Abort	Terminates modify mode without changing the current line.
^W	Wordproc	Shifts into "word processor" mode. In word processor mode, the next control character is used to select a function.

Word Processing Mode Functions

Table F.2 *Word Processing Mode Functions*

Char	Description
^W^C	Compress multiple blanks into single blanks.
^W^D	Delete Word. Deletes from the cursor to the next blank and then any following blanks up to (but not including) the next non-blank.
^W^H	Toggles a flag that remembers if you have an HP110 (flag is initially off). The flag is needed because the HP110 only implements a subset of the "standard" HP26xx escape sequences.
^W^L	Draws a ruled "line", like the "LT" command in QEDIT.
^W^N	Toggles "numbered" mode. A line-number prefix will be displayed in front of a line of text only both of the following are true: (1) line numbers have been requested (either via an M command from QEDIT or via ^W^N), and (2) the line-number was passed to QZMODIFY by QEDIT (i.e., you did an M command, not an MQ command).

Char	Description
^W<c>^D	Delete all characters from cursor up to, but not including character "c". **NOTE** "c" must be a "printable" ASCII character (character code > 31). If the cursor is currently on a "c", it is deleted immediately before looking for the first "c". If "c" is not found, nothing is deleted.
^W^P	Put next character into text. This is useful when you want to put a control-character into the text. All non-printable characters will be displayed as periods (.), so they will take up one space on the line.
^W^S^D	Down-case all letters from cursor to end of line.
^W^S^U	Up-case all letters from cursor to end of line.
^W^S^T	Toggle-case all letters from cursor to end of line.
^W^T	Toggles the TypeAhead Engine (if you have one) through three states: **disabled**, **enabled**, **ignored**.
^W^V	Prints the version id of this editor.
^W?	Display the ASCII character code for the character that the cursor is on, in decimal and octal.

Symbol Chart

The following is an explanation of the symbols used above:

Table F.3 *Symbol Chart*

Symbol	Explanation
<c>	Any single character. This character will be searched for. If <c> is ^W, the search will be for a "word" (words are delimited by blanks) instead of for a single character.
<#>	Zero or more digits. For example, **^P12+** would mean move forward 12 lines. **^P3-** would mean move back 3 lines.
<n>	One of: "^A, ^B, ..., ^H" and is interpreted as the number "1, 2, ..., 8" respectively.
<dir>	A "-" to move "back", or a "+" to move "forward".

NOTE When modifying a line longer than 79 characters, some commands (i.e., ^D, ^B, ^E) will not update any line of the screen display other than the one you are on. Whenever you want to see an accurate display of your text line, press "^X" to refresh the display.

You cannot use the special keys on an HP terminal (i.e., the cursor keys, insert char, delete char, clear). If you use them by accident, a ^X will refresh the display of the line you are editing.

TypeAhead

The remaining information applies only to those users who have TypeAhead Engines (from Telamon). The TypeAhead Engine (TAE) can be in one of three states from the editor’s viewpoint: **disabled**, **enabled**, or **ignored**. Each is defined below:

ignored — Editor will not do anything to either encourage or discourage the use of the TAE. This is the initial state (in most cases, however, see below).

enabled — Editor will place the TAE in single-character mode at entry and restore it to line mode at exit. This means that the HP3000 won’t lose typed ahead input anymore and that the special keys (i.e., cursor keys) will work nicely.

disabled — Editor will disable TypeAhead at entry (by sending ^A^V to the TAE) and enable it at exit. In this mode, the TAE is effectively taken out of the "circuit".

With QEDIT, you configure TAE-treatment as part of the SET MODIFY VEMODIFY command:

```
SET MOD VEMODIFY              {Ignore the TAE}

SET MOD VEMODIFY TAEOFF       {TAE exists, disable it.}

SET MOD VEMODIFY TAE          {TAE exists, enable it.}
```

Additional commands are available **only** when the TAE is present and enabled:

Table F.4 *Additional CommandsL*

Command	Explanation
^W^T	Toggles the TypeAhead Engine through three states: **disabled**, **enabled**, or **ignored**.
Left arrow	The HP26xx left arrow key will move the cursor one space to the left.

Command	Explanation
Right arrow	The HP26xx right arrow key will move the cursor one space to the right.
Up arrow	Move up to the prior line of text, leaving cursor in the same column. The CRT screen is scrolled DOWN, so the line you were just editing is moved down one.
Down arrow	Move down to the next line of text, leaving cursor in the same column. The CRT screen is scrolled UP, so the line you were just editing is moved up one.
Delete char	Deletes the character under the cursor (like ^D).
Insert char	Turns on insert mode (like ^B).
Home up	Move cursor to column 1 of current line.
Home down	Move cursor to last column of current line.
Insert line	Ask QEDIT to add a new line AFTER the current line.
Delete line	Ask QEDIT to delete the current line.
^leftarrow	Moves cursor LEFT to the blank just after the nearest token to the left of the cursor. Valid ONLY if a TypeAhead Engine is present and enabled. Only available on HP264x terminals.
^rightarrow	Moves cursor RIGHT until it hits the next token. Will not move past current end of text. Valid ONLY if a TypeAhead Engine is present and enabled. Only available on HP264x terminals.

G

SETTING OPTIONS

The following list covers the standard settings that you would commonly use with System Manager's Toolbox utility after you have started it and are at that tool's prompt. These options impact how the tools behave. Any user-defined customization is achieved through these special options.

The RESET and SET commands are used for enabling or disabling options. In general, SET is equivalent to "enable" and RESET is equivalent to "disable".

When to Use Setting Options

For tools that serve very pointed, specific tasks like finding a file or changing program capabilities. setting options never really becomes an issue because users are "in and out" of these programs so quickly. But for tools that have a more multi-dimensional purpose, a typical user session could last quite a while. So, knowing how these options can affect a given utility's operation is extremely useful.

For example, the EATEMPTY option, when enabled, ignores empty input lines and continues to display the results from the command last entered. If you need to look at several screens full of information then enabling this option is very useful.

TOOLBOX STANDARDS

The ToolBox collections from Lund Performance Solutions have a uniform user interface. As a result, in addition to the commands specific to each Toolbox tool, most tools allow the following commands: //, CAPTURE, CRON, CROFF, DO, LISTREDO, RESET, SET, USE.

//

// will terminate most tools immediately.

CAPTURE

This command has the following syntax:

```
CAPTURE <captureoptions>
```

The CAPTURE command will generate a hardcopy (or a disc copy) of all (or a portion) of the screen display. The ability to enter CAPTURE as a command to most tools can be enabled by entering SET CAPTUREOK and can be disabled by entering RESET CAPTUREOK.

The interactive tools maintain a shared session-local redo stack of 40 lines. The DO, LISTREDO, and REDO commands access this stack. The options REDOOK and REDOALL affect the operation of the commands.

DO

This command has the following syntax:

```
DO [ cmd# | relative_cmd# | start_text ]
```

The DO command causes the tool to re-use the selected saved input line without re-editing.

If no options follow DO, then the most recent line is reused.

If a cmd# (i.e.: DO 5) is used, then that command is retrieved and reused.

If a relative_cmd# (i.e.: DO -3) is used, then that line is retrieved and reused. A value of -1 means: most recent, -2 means second most recent, etc.

If start_text is specified, then the most recent command that started with the same text (regardless of case) is reused.

LISTREDO

This command has the following syntax:

```
LISTREDO [ALL | *]
```

Lists the REDO stack for a tool. The REDO stack is up to 40 lines long. If the REDOALL option is false, then only the saved input lines from the current tool will be listed, otherwise the last 40 lines (regardless of what tool saved them) will be listed.

If the ALL option is specified, then all saved input lines will be listed, regardless of REDOALL and tool identity.

If the "*" option is specified, then only the current tool's saved lines will be listed, regardless of REDOALL.

REDO

This command has the following syntax:

```
REDO [ cmd# | relative_cmd# | start_text ]
```

The REDO command is very similar to the DO command. After selecting a saved input line, it then displays it for editing. When editing is done, the line is used as input. The REDO can be abandoned by pressing CTRL+Y while editing.

HELP

This command has one of the following syntax:

```
HELP  or

HELP   ?  or

HELP   command
```

The HELP command (synonym: ?) provides help information about the program in general, or about a specific command. Commands may be abbreviated, in which case HELP will display information about every command that starts with the same characters.

HELP ? will display the entire help file for a tool.

Examples:

HELP STANDARDS	displays information about Toolbox standard interface.
? SE	displays information about the SET command, and any other command beginning with "SE".

SET/RESET

These commands have the following syntax:

```
SET     <options>

RESET   <options>
```

In addition to the various SET/RESET options provided by each tool, every tool supports the following options:

<options>:

BATCH CAPTUREOK COPYLP CRON CRONOK CRONPROMPT EATEMPTY EATPROMPT ECHO MPEOK PAGING QEDITOK REDO REDOALL TERMQUIET UPSHIFT USEOK 80 132

Some users like the set/reset paradigm for turning options on/off, while other users like the set option/NOoption paradigm. The SET and RESET commands provides both styles:

SET <option>	will set the option to true.
RESET <option>	will set the option to false.
SET NO<option>	will set the option to false.
RESET NO<option>	will set the option to true.

Some of the options that end in "OK" control whether or not certain commands will be automatically recognized by the Toolbox input routine.

These options are: CAPTUREOK, CRONOK, MPEOK, REDOOK, and USEOK.

[RE]SET BATCH

The BATCH option allows the user to tell a tool that it is in a job (SET BATCH) or in a session (RESET BATCH). Every tool initially determines the value of this option by calling the WHO intrinsic. The ability to override it with a SET/RESET command is intended as a development tool for Lund Performance Solutions.

[RE]SET CAPTUREOK

If CAPTUREOK is true, then the a "pscreen" (a hard copy of the screen's current contents) can be obtained by entering the command CAPTURE at most prompts. (See also: HELP CAPTURE)

CAPTUREOK can be turned on by entering: SET CAPTUREOK

CAPTUREOK can be turned off by entering: RESET CAPTUREOK

[RE]SET COPYLP

When COPYLP is true, then a copy of all terminal output (except for input prompts) is sent to LPSLP.

COPYLP can be turned on by entering: SET COPYLP

COPYLP can be turned off by entering: RESET COPYLP

[RE]SET CROFF

[RE]SET CRON

SET CROFF turns off the "CRON" option

When CRON is true, hitting a return with no other input on the line will cause a tool to re-use the last input line.

CRON can be turned on by entering: CRON, SET CRON, or RESET CROFF

CRON can be turned off by entering: CROFF, SET CROFF, or RESET CRON

NOTENOTE See CRONOK option for more information about the CRON and CROFF commands.

[RE]SET CRONOK

When CRONOK is true, the CRON and CROFF commands may be entered at any prompt.

When CRONOK is false, the CRON and CROFF commands are not allowed. (In this case, the [RE]SET CRON command can be used to turn CRON on and off.)

[RE]SET CRONPROMPT

When CRON is true, the tool will display the default input as part of the prompt if CRONPROMPT is true.

CRONPROMPT can be turned on by entering: SET CRONPROMPT

CRONPROMPT can be turned off by entering: RESET CRONPROMPT

[RE]SET EATEMPTY

When EATEMPTY is true (and CRON is false), the tool will not "see" empty input lines. Most tools set EATEMPTY to true by default.

EATEMPTY can be turned on by entering: SET EATEMPTY

EATEMPTY can be turned off by entering: RESET EATEMPTY

[RE]SET EATPROMPT

When EATPROMPT is true, then a tool will look at the beginning of every input line to see if you did something like:

> move cursor up; hit ENTER

If EATPROMPT is true, and the start of the input line matches the text you were last prompted with, then that text is stripped from your input. After the stripping is done, the remainder of the input line is treated as though it was freshly typed in. Most tools set EATPROMPT to true by default.

EATPROMPT can be turned on by entering: SET EATPROMPT

EATPROMPT can be turned off by entering: RESET EATPROMPT

[RE]SET ECHO

If ECHO is true, then all input read by the tool input routine is automatically echoed to LPSOUT. ECHO is set/reset automatically at the start of each tool, and is normally not changed by users. It is documented here for completeness.

[RE]SET MPEOK

If MPEOK is true, then any input line starting with a colon (:) is passed to the HPCICOMMAND intrinsic. Most tools set MPEOK to true by default.

MPEOK can be turned on by entering: SET MPEOK

MPEOK can be turned off by entering: RESET MPEOK

[RE]SET PAGING

If PAGING is true, and if the tool is running in a session, then most output will be "paged" (i.e.: it will pause every 22 lines or so). The HELP subsystem ALWAYS temporarily sets paging to true for sessions.

PAGING can be turned on by entering: SET PAGING

PAGING can be turned off by entering: RESET PAGING

[RE]SET PSCREENOK

PSCREENOK is a synonym for CAPTUREOK.

[RE]SET QEDITOK

If QEDITOK is true and REDOOK is true, then the 2-character sequence <escape>v will be treated as a synonym for LISTREDO. This character sequence is loaded into softkey 7 by QEDIT and labelled "Listredo".

[RE]SET REDOALL

The REDO stack maintained by the tool programs is a shared stack of 40 lines.

If REDOALL is true and REDOOK is true, then LISTREDO, DO, and REDO will see the entire stack.

If REDOALL is false and REDOOK is true, then LISTREDO, DO, and REDO will see only those redo stack entries that came from the current tool.

REDOALL is reset by default.

[RE]SET REDOOK

If REDOOK is true, then most tools support the commands DO, LISTREDO, and REDO.

REDOOK can be turned on by entering: SET REDOOK

REDOOK can be turned off by entering: RESET REDOOK

[RE]SET UPSHIFT

If UPSHIFT is true, then input will be automatically shifted to uppercase.

UPSHIFT can be turned on by entering: SET UPSHIFT

UPSHIFT can be turned off by entering: RESET UPSHIFT

[RE]SET USEOK

If USEOK is true, then most tools will allow the USE command. (See also: HELP USE)

USEOK can be turned on by entering: SET USEOK

USEOK can be turned off by entering: RESET USEOK

USE

This command has the following syntax:

```
    USE[Q] <filename>
```

The USE command causes the tool to read subsequent input from the specified disc file. USE will echo its input, USEQ will not.

Because USE files may not be nested, a USE command within a USE file will close the first USE file and switch to the second file.

H

CHRONOS MODES

All mode numbers are in hexadecimal format. Refer to the key provided for an expanded description of format codes used in this list.

Key

MDY	= Month/Day/Year
DMY	= Day/Month/Year
YMD	= Year/Month/Day
WD	= Weekday

Inc. = Increment (see Table H.1 on page 221)

Table H.1 *CHRONOS Modes*

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0008	SYSTEM LOCAL	CHRONOS-STAMP	MDY	MDY	FALSE	
0808	SYSTEM LOCAL	CHRONOS-STAMP	MDY	MDY	TRUE	DATE
1808	SYSTEM LOCAL	CHRONOS-STAMP	MDY	MDY	TRUE	TIME
0108	SYSTEM LOCAL	CHRONOS-STAMP	MDY	DMY	FALSE	
0908	SYSTEM LOCAL	CHRONOS-STAMP	MDY	DMY	TRUE	DATE
1908	SYSTEM LOCAL	CHRONOS-STAMP	MDY	DMY	TRUE	TIME
0208	SYSTEM LOCAL	CHRONOS-STAMP	MDY	YMD	FALSE	
0A08	SYSTEM LOCAL	CHRONOS-STAMP	MDY	YMD	TRUE	DATE
1A08	SYSTEM LOCAL	CHRONOS-STAMP	MDY	YMD	TRUE	TIME
0048	SYSTEM LOCAL	CHRONOS-STAMP	DMY	MDY	FALSE	
0848	SYSTEM LOCAL	CHRONOS-STAMP	DMY	MDY	TRUE	DATE

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
1848	SYSTEM LOCAL	CHRONOS-STAMP	DMY	MDY	TRUE	TIME
0148	SYSTEM LOCAL	CHRONOS-STAMP	DMY	DMY	FALSE	
0948	SYSTEM LOCAL	CHRONOS-STAMP	DMY	DMY	TRUE	DATE
1948	SYSTEM LOCAL	CHRONOS-STAMP	DMY	DMY	TRUE	TIME
0248	SYSTEM LOCAL	CHRONOS-STAMP	DMY	YMD	FALSE	
0A48	SYSTEM LOCAL	CHRONOS-STAMP	DMY	YMD	TRUE	DATE
1A48	SYSTEM LOCAL	CHRONOS-STAMP	DMY	YMD	TRUE	TIME
0088	SYSTEM LOCAL	CHRONOS-STAMP	YMD	MDY	FALSE	
0888	SYSTEM LOCAL	CHRONOS-STAMP	YMD	MDY	TRUE	DATE
1888	SYSTEM LOCAL	CHRONOS-STAMP	YMD	MDY	TRUE	TIME
0188	SYSTEM LOCAL	CHRONOS-STAMP	YMD	DMY	FALSE	
0988	SYSTEM LOCAL	CHRONOS-STAMP	YMD	DMY	TRUE	DATE
1988	SYSTEM LOCAL	CHRONOS-STAMP	YMD	DMY	TRUE	TIME
0288	SYSTEM LOCAL	CHRONOS-STAMP	YMD	YMD	FALSE	
0A88	SYSTEM LOCAL	CHRONOS-STAMP	YMD	YMD	TRUE	DATE
1A88	SYSTEM LOCAL	CHRONOS-STAMP	YMD	YMD	TRUE	TIME
0010	SYSTEM LOCAL	FORMATTED	MDY	MDY	FALSE	
0810	SYSTEM LOCAL	FORMATTED	MDY	MDY	TRUE	DATE
1810	SYSTEM LOCAL	FORMATTED	MDY	MDY	TRUE	TIME
0110	SYSTEM LOCAL	FORMATTED	MDY	DMY	FALSE	
0910	SYSTEM LOCAL	FORMATTED	MDY	DMY	TRUE	DATE
1910	SYSTEM LOCAL	FORMATTED	MDY	DMY	TRUE	TIME
0210	SYSTEM LOCAL	FORMATTED	MDY	YMD	FALSE	
0A10	SYSTEM LOCAL	FORMATTED	MDY	YMD	TRUE	DATE

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
1A10	SYSTEM LOCAL	FORMATTED	MDY	YMD	TRUE	TIME
0050	SYSTEM LOCAL	FORMATTED	DMY	MDY	FALSE	
0850	SYSTEM LOCAL	FORMATTED	DMY	MDY	TRUE	DATE
1850	SYSTEM LOCAL	FORMATTED	DMY	MDY	TRUE	TIME
0150	SYSTEM LOCAL	FORMATTED	DMY	DMY	FALSE	
0950	SYSTEM LOCAL	FORMATTED	DMY	DMY	TRUE	DATE
1950	SYSTEM LOCAL	FORMATTED	DMY	DMY	TRUE	TIME
0250	SYSTEM LOCAL	FORMATTED	DMY	YMD	FALSE	
0A50	SYSTEM LOCAL	FORMATTED	DMY	YMD	TRUE	DATE
1A50	SYSTEM LOCAL	FORMATTED	DMY	YMD	TRUE	TIME
0090	SYSTEM LOCAL	FORMATTED	YMD	MDY	FALSE	
0890	SYSTEM LOCAL	FORMATTED	YMD	MDY	TRUE	DATE
1890	SYSTEM LOCAL	FORMATTED	YMD	MDY	TRUE	TIME
0190	SYSTEM LOCAL	FORMATTED	YMD	DMY	FALSE	
0990	SYSTEM LOCAL	FORMATTED	YMD	DMY	TRUE	DATE
1990	SYSTEM LOCAL	FORMATTED	YMD	DMY	TRUE	TIME
0290	SYSTEM LOCAL	FORMATTED	YMD	YMD	FALSE	
0A90	SYSTEM LOCAL	FORMATTED	YMD	YMD	TRUE	DATE
1A90	SYSTEM LOCAL	FORMATTED	YMD	YMD	TRUE	TIME
0018	SYSTEM LOCAL	UNFORMATTED	MDY	MDY	FALSE	
0818	SYSTEM LOCAL	UNFORMATTED	MDY	MDY	TRUE	DATE
1818	SYSTEM LOCAL	UNFORMATTED	MDY	MDY	TRUE	TIME
0118	SYSTEM LOCAL	UNFORMATTED	MDY	DMY	FALSE	
0918	SYSTEM LOCAL	UNFORMATTED	MDY	DMY	TRUE	DATE

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
1918	SYSTEM LOCAL	UNFORMATTED	MDY	DMY	TRUE	TIME
0218	SYSTEM LOCAL	UNFORMATTED	MDY	YMD	FALSE	
0A18	SYSTEM LOCAL	UNFORMATTED	MDY	YMD	TRUE	DATE
1A18	SYSTEM LOCAL	UNFORMATTED	MDY	YMD	TRUE	TIME
0058	SYSTEM LOCAL	UNFORMATTED	DMY	MDY	FALSE	
0858	SYSTEM LOCAL	UNFORMATTED	DMY	MDY	TRUE	DATE
1858	SYSTEM LOCAL	UNFORMATTED	DMY	MDY	TRUE	TIME
0158	SYSTEM LOCAL	UNFORMATTED	DMY	DMY	FALSE	
0958	SYSTEM LOCAL	UNFORMATTED	DMY	DMY	TRUE	DATE
1958	SYSTEM LOCAL	UNFORMATTED	DMY	DMY	TRUE	TIME
0258	SYSTEM LOCAL	UNFORMATTED	DMY	YMD	FALSE	
0A58	SYSTEM LOCAL	UNFORMATTED	DMY	YMD	TRUE	DATE
1A58	SYSTEM LOCAL	UNFORMATTED	DMY	YMD	TRUE	TIME
0098	SYSTEM LOCAL	UNFORMATTED	YMD	MDY	FALSE	
0898	SYSTEM LOCAL	UNFORMATTED	YMD	MDY	TRUE	DATE
1898	SYSTEM LOCAL	UNFORMATTED	YMD	MDY	TRUE	TIME
0198	SYSTEM LOCAL	UNFORMATTED	YMD	DMY	FALSE	
0998	SYSTEM LOCAL	UNFORMATTED	YMD	DMY	TRUE	DATE
1998	SYSTEM LOCAL	UNFORMATTED	YMD	DMY	TRUE	TIME
0298	SYSTEM LOCAL	UNFORMATTED	YMD	YMD	FALSE	
0A98	SYSTEM LOCAL	UNFORMATTED	YMD	YMD	TRUE	DATE
1A98	SYSTEM LOCAL	UNFORMATTED	YMD	YMD	TRUE	TIME
0020	SYSTEM LOCAL	JULIAN	MDY	MDY	FALSE	
0820	SYSTEM LOCAL	JULIAN	MDY	MDY	TRUE	DATE

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
1820	SYSTEM LOCAL	JULIAN	MDY	MDY	TRUE	TIME
0120	SYSTEM LOCAL	JULIAN	MDY	DMY	FALSE	
0920	SYSTEM LOCAL	JULIAN	MDY	DMY	TRUE	DATE
1920	SYSTEM LOCAL	JULIAN	MDY	DMY	TRUE	TIME
0220	SYSTEM LOCAL	JULIAN	MDY	YMD	FALSE	
0A20	SYSTEM LOCAL	JULIAN	MDY	YMD	TRUE	DATE
1A20	SYSTEM LOCAL	JULIAN	MDY	YMD	TRUE	TIME
0060	SYSTEM LOCAL	JULIAN	DMY	MDY	FALSE	
0860	SYSTEM LOCAL	JULIAN	DMY	MDY	TRUE	DATE
1860	SYSTEM LOCAL	JULIAN	DMY	MDY	TRUE	TIME
0160	SYSTEM LOCAL	JULIAN	DMY	DMY	FALSE	
0960	SYSTEM LOCAL	JULIAN	DMY	DMY	TRUE	DATE
1960	SYSTEM LOCAL	JULIAN	DMY	DMY	TRUE	TIME
0260	SYSTEM LOCAL	JULIAN	DMY	YMD	FALSE	
0A60	SYSTEM LOCAL	JULIAN	DMY	YMD	TRUE	DATE
1A60	SYSTEM LOCAL	JULIAN	DMY	YMD	TRUE	TIME
00A0	SYSTEM LOCAL	JULIAN	YMD	MDY	FALSE	
08A0	SYSTEM LOCAL	JULIAN	YMD	MDY	TRUE	DATE
18A0	SYSTEM LOCAL	JULIAN	YMD	MDY	TRUE	TIME
01A0	SYSTEM LOCAL	JULIAN	YMD	DMY	FALSE	
09A0	SYSTEM LOCAL	JULIAN	YMD	DMY	TRUE	DATE
19A0	SYSTEM LOCAL	JULIAN	YMD	DMY	TRUE	TIME
02A0	SYSTEM LOCAL	JULIAN	YMD	YMD	FALSE	
0AA0	SYSTEM LOCAL	JULIAN	YMD	YMD	TRUE	DATE
1AA0	SYSTEM LOCAL	JULIAN	YMD	YMD	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0028	SYSTEM LOCAL	CHRONOS-STRING	MDY	STRING-WD-MDY	FALSE	
0828	SYSTEM LOCAL	CHRONOS-STRING	MDY	STRING-WD-MDY	TRUE	DATE
1828	SYSTEM LOCAL	CHRONOS-STRING	MDY	STRING-WD-MDY	TRUE	TIME
0128	SYSTEM LOCAL	CHRONOS-STRING	MDY	STRING-WD-DMY	FALSE	
0928	SYSTEM LOCAL	CHRONOS-STRING	MDY	STRING-WD-DMY	TRUE	DATE
1928	SYSTEM LOCAL	CHRONOS-STRING	MDY	STRING-WD-DMY	TRUE	TIME
0228	SYSTEM LOCAL	CHRONOS-STRING	MDY	STRING-MDY	FALSE	
0A28	SYSTEM LOCAL	CHRONOS-STRING	MDY	STRING-MDY	TRUE	DATE
1A28	SYSTEM LOCAL	CHRONOS-STRING	MDY	STRING-MDY	TRUE	TIME
0328	SYSTEM LOCAL	CHRONOS-STRING	MDY	STRING-DMY	FALSE	
0B28	SYSTEM LOCAL	CHRONOS-STRING	MDY	STRING-DMY	TRUE	DATE
1B28	SYSTEM LOCAL	CHRONOS-STRING	MDY	STRING-DMY	TRUE	TIME
0068	SYSTEM LOCAL	CHRONOS-STRING	DMY	STRING-WD-MDY	FALSE	
0868	SYSTEM LOCAL	CHRONOS-STRING	DMY	STRING-WD-MDY	TRUE	DATE
1868	SYSTEM LOCAL	CHRONOS-STRING	DMY	STRING-WD-MDY	TRUE	TIME
0168	SYSTEM LOCAL	CHRONOS-STRING	DMY	STRING-WD-DMY	FALSE	
0968	SYSTEM LOCAL	CHRONOS-STRING	DMY	STRING-WD-DMY	TRUE	DATE
1968	SYSTEM LOCAL	CHRONOS-STRING	DMY	STRING-WD-DMY	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0268	SYSTEM LOCAL	CHRONOS-STRING	DMY	STRING-MDY	FALSE	
0A68	SYSTEM LOCAL	CHRONOS-STRING	DMY	STRING-MDY	TRUE	DATE
1A68	SYSTEM LOCAL	CHRONOS-STRING	DMY	STRING-MDY	TRUE	TIME
0368	SYSTEM LOCAL	CHRONOS-STRING	DMY	STRING-DMY	FALSE	
0B68	SYSTEM LOCAL	CHRONOS-STRING	DMY	STRING-DMY	TRUE	DATE
1B68	SYSTEM LOCAL	CHRONOS-STRING	DMY	STRING-DMY	TRUE	TIME
00A8	SYSTEM LOCAL	CHRONOS-STRING	YMD	STRING-WD-MDY	FALSE	
08A8	SYSTEM LOCAL	CHRONOS-STRING	YMD	STRING-WD-MDY	TRUE	DATE
18A8	SYSTEM LOCAL	CHRONOS-STRING	YMD	STRING-WD-MDY	TRUE	TIME
01A8	SYSTEM LOCAL	CHRONOS-STRING	YMD	STRING-WD-DMY	FALSE	
09A8	SYSTEM LOCAL	CHRONOS-STRING	YMD	STRING-WD-DMY	TRUE	DATE
19A8	SYSTEM LOCAL	CHRONOS-STRING	YMD	STRING-WD-DMY	TRUE	TIME
02A8	SYSTEM LOCAL	CHRONOS-STRING	YMD	STRING-MDY	FALSE	
0AA8	SYSTEM LOCAL	CHRONOS-STRING	YMD	STRING-MDY	TRUE	DATE
1AA8	SYSTEM LOCAL	CHRONOS-STRING	YMD	STRING-MDY	TRUE	TIME
03A8	SYSTEM LOCAL	CHRONOS-STRING	YMD	STRING-DMY	FALSE	
0BA8	SYSTEM LOCAL	CHRONOS-STRING	YMD	STRING-DMY	TRUE	DATE
1BA8	SYSTEM LOCAL	CHRONOS-STRING	YMD	STRING-DMY	TRUE	TIME
0009	CHRONOS-STAMP	CHRONOS-STAMP	MDY	MDY	FALSE	
0809	CHRONOS-STAMP	CHRONOS-STAMP	MDY	MDY	TRUE	DATE
1809	CHRONOS-STAMP	CHRONOS-STAMP	MDY	MDY	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0109	CHRONOS-STAMP	CHRONOS-STAMP	MDY	DMY	FALSE	
0909	CHRONOS-STAMP	CHRONOS-STAMP	MDY	DMY	TRUE	DATE
1909	CHRONOS-STAMP	CHRONOS-STAMP	MDY	DMY	TRUE	TIME
0209	CHRONOS-STAMP	CHRONOS-STAMP	MDY	YMD	FALSE	
0A09	CHRONOS-STAMP	CHRONOS-STAMP	MDY	YMD	TRUE	DATE
1A09	CHRONOS-STAMP	CHRONOS-STAMP	MDY	YMD	TRUE	TIME
0049	CHRONOS-STAMP	CHRONOS-STAMP	DMY	MDY	FALSE	
0849	CHRONOS-STAMP	CHRONOS-STAMP	DMY	MDY	TRUE	DATE
1849	CHRONOS-STAMP	CHRONOS-STAMP	DMY	MDY	TRUE	TIME
0149	CHRONOS-STAMP	CHRONOS-STAMP	DMY	DMY	FALSE	
0949	CHRONOS-STAMP	CHRONOS-STAMP	DMY	DMY	TRUE	DATE
1949	CHRONOS-STAMP	CHRONOS-STAMP	DMY	DMY	TRUE	TIME
0249	CHRONOS-STAMP	CHRONOS-STAMP	DMY	YMD	FALSE	
0A49	CHRONOS-STAMP	CHRONOS-STAMP	DMY	YMD	TRUE	DATE
1A49	CHRONOS-STAMP	CHRONOS-STAMP	DMY	YMD	TRUE	TIME
0089	CHRONOS-STAMP	CHRONOS-STAMP	YMD	MDY	FALSE	
0889	CHRONOS-STAMP	CHRONOS-STAMP	YMD	MDY	TRUE	DATE
1889	CHRONOS-STAMP	CHRONOS-STAMP	YMD	MDY	TRUE	TIME
0189	CHRONOS-STAMP	CHRONOS-STAMP	YMD	DMY	FALSE	
0989	CHRONOS-STAMP	CHRONOS-STAMP	YMD	DMY	TRUE	DATE
0989	CHRONOS-STAMP	CHRONOS-STAMP	YMD	DMY	TRUE	TIME
0289	CHRONOS-STAMP	CHRONOS-STAMP	YMD	YMD	FALSE	
0A89	CHRONOS-STAMP	CHRONOS-STAMP	YMD	YMD	TRUE	DATE
1A89	CHRONOS-STAMP	CHRONOS-STAMP	YMD	YMD	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0011	CHRONOS-STAMP	FORMATTED	MDY	MDY	FALSE	
0811	CHRONOS-STAMP	FORMATTED	MDY	MDY	TRUE	DATE
1811	CHRONOS-STAMP	FORMATTED	MDY	MDY	TRUE	TIME
0111	CHRONOS-STAMP	FORMATTED	MDY	DMY	FALSE	
0911	CHRONOS-STAMP	FORMATTED	MDY	DMY	TRUE	DATE
1911	CHRONOS-STAMP	FORMATTED	MDY	DMY	TRUE	TIME
0211	CHRONOS-STAMP	FORMATTED	MDY	YMD	FALSE	
0A11	CHRONOS-STAMP	FORMATTED	MDY	YMD	TRUE	DATE
1A11	CHRONOS-STAMP	FORMATTED	MDY	YMD	TRUE	TIME
0051	CHRONOS-STAMP	FORMATTED	DMY	MDY	FALSE	
0851	CHRONOS-STAMP	FORMATTED	DMY	MDY	TRUE	DATE
1851	CHRONOS-STAMP	FORMATTED	DMY	MDY	TRUE	TIME
0151	CHRONOS-STAMP	FORMATTED	DMY	DMY	FALSE	
0951	CHRONOS-STAMP	FORMATTED	DMY	DMY	TRUE	DATE
1951	CHRONOS-STAMP	FORMATTED	DMY	DMY	TRUE	TIME
0251	CHRONOS-STAMP	FORMATTED	DMY	YMD	FALSE	
0A51	CHRONOS-STAMP	FORMATTED	DMY	YMD	TRUE	DATE
1A51	CHRONOS-STAMP	FORMATTED	DMY	YMD	TRUE	TIME
0091	CHRONOS-STAMP	FORMATTED	YMD	MDY	FALSE	
0891	CHRONOS-STAMP	FORMATTED	YMD	MDY	TRUE	DATE
1891	CHRONOS-STAMP	FORMATTED	YMD	MDY	TRUE	TIME
0191	CHRONOS-STAMP	FORMATTED	YMD	DMY	FALSE	
0991	CHRONOS-STAMP	FORMATTED	YMD	DMY	TRUE	DATE
1991	CHRONOS-STAMP	FORMATTED	YMD	DMY	TRUE	TIME
0291	CHRONOS-STAMP	FORMATTED	YMD	YMD	FALSE	

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0A91	CHRONOS-STAMP	FORMATTED	YMD	YMD	TRUE	DATE
1A91	CHRONOS-STAMP	FORMATTED	YMD	YMD	TRUE	TIME
0019	CHRONOS-STAMP	UNFORMATTED	MDY	MDY	FALSE	
0819	CHRONOS-STAMP	UNFORMATTED	MDY	MDY	TRUE	DATE
1819	CHRONOS-STAMP	UNFORMATTED	MDY	MDY	TRUE	TIME
0119	CHRONOS-STAMP	UNFORMATTED	MDY	DMY	FALSE	
0919	CHRONOS-STAMP	UNFORMATTED	MDY	DMY	TRUE	DATE
1919	CHRONOS-STAMP	UNFORMATTED	MDY	DMY	TRUE	TIME
0219	CHRONOS-STAMP	UNFORMATTED	MDY	YMD	FALSE	
0A19	CHRONOS-STAMP	UNFORMATTED	MDY	YMD	TRUE	DATE
1A19	CHRONOS-STAMP	UNFORMATTED	MDY	YMD	TRUE	TIME
0059	CHRONOS-STAMP	UNFORMATTED	DMY	MDY	FALSE	
0859	CHRONOS-STAMP	UNFORMATTED	DMY	MDY	TRUE	DATE
1859	CHRONOS-STAMP	UNFORMATTED	DMY	MDY	TRUE	TIME
0159	CHRONOS-STAMP	UNFORMATTED	DMY	DMY	FALSE	
0959	CHRONOS-STAMP	UNFORMATTED	DMY	DMY	TRUE	DATE
1959	CHRONOS-STAMP	UNFORMATTED	DMY	DMY	TRUE	TIME
0259	CHRONOS-STAMP	UNFORMATTED	DMY	YMD	FALSE	
0A59	CHRONOS-STAMP	UNFORMATTED	DMY	YMD	TRUE	DATE
1A59	CHRONOS-STAMP	UNFORMATTED	DMY	YMD	TRUE	TIME
0099	CHRONOS-STAMP	UNFORMATTED	YMD	MDY	FALSE	
0899	CHRONOS-STAMP	UNFORMATTED	YMD	MDY	TRUE	DATE
1899	CHRONOS-STAMP	UNFORMATTED	YMD	MDY	TRUE	TIME
0199	CHRONOS-STAMP	UNFORMATTED	YMD	DMY	FALSE	

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0999	CHRONOS-STAMP	UNFORMATTED	YMD	DMY	TRUE	DATE
1999	CHRONOS-STAMP	UNFORMATTED	YMD	DMY	TRUE	TIME
0299	CHRONOS-STAMP	UNFORMATTED	YMD	YMD	FALSE	
0A99	CHRONOS-STAMP	UNFORMATTED	YMD	YMD	TRUE	DATE
1A99	CHRONOS-STAMP	UNFORMATTED	YMD	YMD	TRUE	TIME
0021	CHRONOS-STAMP	JULIAN	MDY	MDY	FALSE	
0821	CHRONOS-STAMP	JULIAN	MDY	MDY	TRUE	DATE
1821	CHRONOS-STAMP	JULIAN	MDY	MDY	TRUE	TIME
0121	CHRONOS-STAMP	JULIAN	MDY	DMY	FALSE	
0921	CHRONOS-STAMP	JULIAN	MDY	DMY	TRUE	DATE
1921	CHRONOS-STAMP	JULIAN	MDY	DMY	TRUE	TIME
0221	CHRONOS-STAMP	JULIAN	MDY	YMD	FALSE	
0A21	CHRONOS-STAMP	JULIAN	MDY	YMD	TRUE	DATE
1A21	CHRONOS-STAMP	JULIAN	MDY	YMD	TRUE	TIME
0061	CHRONOS-STAMP	JULIAN	DMY	MDY	FALSE	
0861	CHRONOS-STAMP	JULIAN	DMY	MDY	TRUE	DATE
1861	CHRONOS-STAMP	JULIAN	DMY	MDY	TRUE	TIME
0161	CHRONOS-STAMP	JULIAN	DMY	DMY	FALSE	
0961	CHRONOS-STAMP	JULIAN	DMY	DMY	TRUE	DATE
1961	CHRONOS-STAMP	JULIAN	DMY	DMY	TRUE	TIME
0261	CHRONOS-STAMP	JULIAN	DMY	YMD	FALSE	
0A61	CHRONOS-STAMP	JULIAN	DMY	YMD	TRUE	DATE
1A61	CHRONOS-STAMP	JULIAN	DMY	YMD	TRUE	TIME
00A1	CHRONOS-STAMP	JULIAN	YMD	MDY	FALSE	

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
08A1	CHRONOS-STAMP	JULIAN	YMD	MDY	TRUE	DATE
18A1	CHRONOS-STAMP	JULIAN	YMD	MDY	TRUE	TIME
01A1	CHRONOS-STAMP	JULIAN	YMD	DMY	FALSE	
09A1	CHRONOS-STAMP	JULIAN	YMD	DMY	TRUE	DATE
19A1	CHRONOS-STAMP	JULIAN	YMD	DMY	TRUE	TIME
02A1	CHRONOS-STAMP	JULIAN	YMD	YMD	FALSE	
0AA1	CHRONOS-STAMP	JULIAN	YMD	YMD	TRUE	DATE
1AA1	CHRONOS-STAMP	JULIAN	YMD	YMD	TRUE	TIME
0029	CHRONOS-STAMP	CHRONOS-STRING	MDY	STRING-WD-MDY	FALSE	
0829	CHRONOS-STAMP	CHRONOS-STRING	MDY	STRING-WD-MDY	TRUE	DATE
1829	CHRONOS-STAMP	CHRONOS-STRING	MDY	STRING-WD-MDY	TRUE	TIME
0129	CHRONOS-STAMP	CHRONOS-STRING	MDY	STRING-WD-DMY	FALSE	
0929	CHRONOS-STAMP	CHRONOS-STRING	MDY	STRING-WD-DMY	TRUE	DATE
1929	CHRONOS-STAMP	CHRONOS-STRING	MDY	STRING-WD-DMY	TRUE	TIME
0229	CHRONOS-STAMP	CHRONOS-STRING	MDY	STRING-MDY	FALSE	
0A29	CHRONOS-STAMP	CHRONOS-STRING	MDY	STRING-MDY	TRUE	DATE
1A29	CHRONOS-STAMP	CHRONOS-STRING	MDY	STRING-MDY	TRUE	TIME
0329	CHRONOS-STAMP	CHRONOS-STRING	MDY	STRING-DMY	FALSE	
0B29	CHRONOS-STAMP	CHRONOS-STRING	MDY	STRING-DMY	TRUE	DATE
1B29	CHRONOS-STAMP	CHRONOS-STRING	MDY	STRING-DMY	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0069	CHRONOS-STAMP	CHRONOS-STRING	DMY	STRING-WD-MDY	FALSE	
0869	CHRONOS-STAMP	CHRONOS-STRING	DMY	STRING-WD-MDY	TRUE	DATE
1869	CHRONOS-STAMP	CHRONOS-STRING	DMY	STRING-WD-MDY	TRUE	TIME
0169	CHRONOS-STAMP	CHRONOS-STRING	DMY	STRING-WD-DMY	FALSE	
0969	CHRONOS-STAMP	CHRONOS-STRING	DMY	STRING-WD-DMY	TRUE	DATE
1969	CHRONOS-STAMP	CHRONOS-STRING	DMY	STRING-WD-DMY	TRUE	TIME
0269	CHRONOS-STAMP	CHRONOS-STRING	DMY	STRING-MDY	FALSE	
0A69	CHRONOS-STAMP	CHRONOS-STRING	DMY	STRING-MDY	TRUE	DATE
1A69	CHRONOS-STAMP	CHRONOS-STRING	DMY	STRING-MDY	TRUE	TIME
0369	CHRONOS-STAMP	CHRONOS-STRING	DMY	STRING-DMY	FALSE	
0B69	CHRONOS-STAMP	CHRONOS-STRING	DMY	STRING-DMY	TRUE	DATE
1B69	CHRONOS-STAMP	CHRONOS-STRING	DMY	STRING-DMY	TRUE	TIME
00A9	CHRONOS-STAMP	CHRONOS-STRING	YMD	STRING-WD-MDY	FALSE	
08A9	CHRONOS-STAMP	CHRONOS-STRING	YMD	STRING-WD-MDY	TRUE	DATE
18A9	CHRONOS-STAMP	CHRONOS-STRING	YMD	STRING-WD-MDY	TRUE	TIME
01A9	CHRONOS-STAMP	CHRONOS-STRING	YMD	STRING-WD-DMY	FALSE	
09A9	CHRONOS-STAMP	CHRONOS-STRING	YMD	STRING-WD-DMY	TRUE	DATE
19A9	CHRONOS-STAMP	CHRONOS-STRING	YMD	STRING-WD-DMY	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
02A9	CHRONOS-STAMP	CHRONOS-STRING	YMD	STRING-MDY	FALSE	
0AA9	CHRONOS-STAMP	CHRONOS-STRING	YMD	STRING-MDY	TRUE	DATE
1AA9	CHRONOS-STAMP	CHRONOS-STRING	YMD	STRING-MDY	TRUE	TIME
03A9	CHRONOS-STAMP	CHRONOS-STRING	YMD	STRING-DMY	FALSE	
0BA9	CHRONOS-STAMP	CHRONOS-STRING	YMD	STRING-DMY	TRUE	DATE
1BA9	CHRONOS-STAMP	CHRONOS-STRING	YMD	STRING-DMY	TRUE	TIME
000A	FORMATTED	CHRONOS-STAMP	MDY	MDY	FALSE	
080A	FORMATTED	CHRONOS-STAMP	MDY	MDY	TRUE	DATE
180A	FORMATTED	CHRONOS-STAMP	MDY	MDY	TRUE	TIME
010A	FORMATTED	CHRONOS-STAMP	MDY	DMY	FALSE	
090A	FORMATTED	CHRONOS-STAMP	MDY	DMY	TRUE	DATE
190A	FORMATTED	CHRONOS-STAMP	MDY	DMY	TRUE	TIME
020A	FORMATTED	CHRONOS-STAMP	MDY	YMD	FALSE	
0A0A	FORMATTED	CHRONOS-STAMP	MDY	YMD	TRUE	DATE
1A0A	FORMATTED	CHRONOS-STAMP	MDY	YMD	TRUE	TIME
004A	FORMATTED	CHRONOS-STAMP	DMY	MDY	FALSE	
084A	FORMATTED	CHRONOS-STAMP	DMY	MDY	TRUE	DATE
184A	FORMATTED	CHRONOS-STAMP	DMY	MDY	TRUE	TIME
014A	FORMATTED	CHRONOS-STAMP	DMY	DMY	FALSE	
094A	FORMATTED	CHRONOS-STAMP	DMY	DMY	TRUE	DATE
194A	FORMATTED	CHRONOS-STAMP	DMY	DMY	TRUE	TIME
024A	FORMATTED	CHRONOS-STAMP	DMY	YMD	FALSE	
0A4A	FORMATTED	CHRONOS-STAMP	DMY	YMD	TRUE	DATE
1A4A	FORMATTED	CHRONOS-STAMP	DMY	YMD	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
008A	FORMATTED	CHRONOS-STAMP	YMD	MDY	FALSE	
088A	FORMATTED	CHRONOS-STAMP	YMD	MDY	TRUE	DATE
188A	FORMATTED	CHRONOS-STAMP	YMD	MDY	TRUE	TIME
018A	FORMATTED	CHRONOS-STAMP	YMD	DMY	FALSE	
098A	FORMATTED	CHRONOS-STAMP	YMD	DMY	TRUE	DATE
198A	FORMATTED	CHRONOS-STAMP	YMD	DMY	TRUE	TIME
028A	FORMATTED	CHRONOS-STAMP	YMD	YMD	FALSE	
0A8A	FORMATTED	CHRONOS-STAMP	YMD	YMD	TRUE	DATE
1A8A	FORMATTED	CHRONOS-STAMP	YMD	YMD	TRUE	TIME
0012	FORMATTED	FORMATTED	MDY	MDY	FALSE	
0812	FORMATTED	FORMATTED	MDY	MDY	TRUE	DATE
1812	FORMATTED	FORMATTED	MDY	MDY	TRUE	TIME
0112	FORMATTED	FORMATTED	MDY	DMY	FALSE	
0912	FORMATTED	FORMATTED	MDY	DMY	TRUE	DATE
1912	FORMATTED	FORMATTED	MDY	DMY	TRUE	TIME
0212	FORMATTED	FORMATTED	MDY	YMD	FALSE	
0A12	FORMATTED	FORMATTED	MDY	YMD	TRUE	DATE
1A12	FORMATTED	FORMATTED	MDY	YMD	TRUE	TIME
0052	FORMATTED	FORMATTED	DMY	MDY	FALSE	
0852	FORMATTED	FORMATTED	DMY	MDY	TRUE	DATE
1852	FORMATTED	FORMATTED	DMY	MDY	TRUE	TIME
0152	FORMATTED	FORMATTED	DMY	DMY	FALSE	
0952	FORMATTED	FORMATTED	DMY	DMY	TRUE	DATE
1952	FORMATTED	FORMATTED	DMY	DMY	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0252	FORMATTED	FORMATTED	DMY	YMD	FALSE	
0A52	FORMATTED	FORMATTED	DMY	YMD	TRUE	DATE
1A52	FORMATTED	FORMATTED	DMY	YMD	TRUE	TIME
0092	FORMATTED	FORMATTED	YMD	MDY	FALSE	
0892	FORMATTED	FORMATTED	YMD	MDY	TRUE	DATE
1892	FORMATTED	FORMATTED	YMD	MDY	TRUE	TIME
0192	FORMATTED	FORMATTED	YMD	DMY	FALSE	
0992	FORMATTED	FORMATTED	YMD	DMY	TRUE	DATE
1992	FORMATTED	FORMATTED	YMD	DMY	TRUE	TIME
0292	FORMATTED	FORMATTED	YMD	YMD	FALSE	
0A92	FORMATTED	FORMATTED	YMD	YMD	TRUE	DATE
1A92	FORMATTED	FORMATTED	YMD	YMD	TRUE	TIME
001A	FORMATTED	UNFORMATTED	MDY	MDY	FALSE	
081A	FORMATTED	UNFORMATTED	MDY	MDY	TRUE	DATE
181A	FORMATTED	UNFORMATTED	MDY	MDY	TRUE	TIME
011A	FORMATTED	UNFORMATTED	MDY	DMY	FALSE	
091A	FORMATTED	UNFORMATTED	MDY	DMY	TRUE	DATE
191A	FORMATTED	UNFORMATTED	MDY	DMY	TRUE	TIME
021A	FORMATTED	UNFORMATTED	MDY	YMD	FALSE	
0A1A	FORMATTED	UNFORMATTED	MDY	YMD	TRUE	DATE
1A1A	FORMATTED	UNFORMATTED	MDY	YMD	TRUE	TIME
005A	FORMATTED	UNFORMATTED	DMY	MDY	FALSE	
085A	FORMATTED	UNFORMATTED	DMY	MDY	TRUE	DATE
185A	FORMATTED	UNFORMATTED	DMY	MDY	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
015A	FORMATTED	UNFORMATTED	DMY	DMY	FALSE	
095A	FORMATTED	UNFORMATTED	DMY	DMY	TRUE	DATE
195A	FORMATTED	UNFORMATTED	DMY	DMY	TRUE	TIME
025A	FORMATTED	UNFORMATTED	DMY	YMD	FALSE	
0A5A	FORMATTED	UNFORMATTED	DMY	YMD	TRUE	DATE
1A5A	FORMATTED	UNFORMATTED	DMY	YMD	TRUE	TIME
009A	FORMATTED	UNFORMATTED	YMD	MDY	FALSE	
089A	FORMATTED	UNFORMATTED	YMD	MDY	TRUE	DATE
189A	FORMATTED	UNFORMATTED	YMD	MDY	TRUE	TIME
019A	FORMATTED	UNFORMATTED	YMD	DMY	FALSE	
099A	FORMATTED	UNFORMATTED	YMD	DMY	TRUE	DATE
199A	FORMATTED	UNFORMATTED	YMD	DMY	TRUE	TIME
029A	FORMATTED	UNFORMATTED	YMD	YMD	FALSE	
0A9A	FORMATTED	UNFORMATTED	YMD	YMD	TRUE	DATE
1A9A	FORMATTED	UNFORMATTED	YMD	YMD	TRUE	TIME
0022	FORMATTED	JULIAN	MDY	MDY	FALSE	
0822	FORMATTED	JULIAN	MDY	MDY	TRUE	DATE
1822	FORMATTED	JULIAN	MDY	MDY	TRUE	TIME
0122	FORMATTED	JULIAN	MDY	DMY	FALSE	
0922	FORMATTED	JULIAN	MDY	DMY	TRUE	DATE
1922	FORMATTED	JULIAN	MDY	DMY	TRUE	TIME
0222	FORMATTED	JULIAN	MDY	YMD	FALSE	
0A22	FORMATTED	JULIAN	MDY	YMD	TRUE	DATE
1A22	FORMATTED	JULIAN	MDY	YMD	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0062	FORMATTED	JULIAN	DMY	MDY	FALSE	
0862	FORMATTED	JULIAN	DMY	MDY	TRUE	DATE
1862	FORMATTED	JULIAN	DMY	MDY	TRUE	TIME
0162	FORMATTED	JULIAN	DMY	DMY	FALSE	
0962	FORMATTED	JULIAN	DMY	DMY	TRUE	DATE
1962	FORMATTED	JULIAN	DMY	DMY	TRUE	TIME
0262	FORMATTED	JULIAN	DMY	YMD	FALSE	
0A62	FORMATTED	JULIAN	DMY	YMD	TRUE	DATE
1A62	FORMATTED	JULIAN	DMY	YMD	TRUE	TIME
00A2	FORMATTED	JULIAN	YMD	MDY	FALSE	
08A2	FORMATTED	JULIAN	YMD	MDY	TRUE	DATE
18A2	FORMATTED	JULIAN	YMD	MDY	TRUE	TIME
01A2	FORMATTED	JULIAN	YMD	DMY	FALSE	
09A2	FORMATTED	JULIAN	YMD	DMY	TRUE	DATE
19A2	FORMATTED	JULIAN	YMD	DMY	TRUE	TIME
02A2	FORMATTED	JULIAN	YMD	YMD	FALSE	
0AA2	FORMATTED	JULIAN	YMD	YMD	TRUE	DATE
1AA2	FORMATTED	JULIAN	YMD	YMD	TRUE	TIME
002A	FORMATTED	CHRONOS-STRING	MDY	STRING-WD-MDY	FALSE	
082A	FORMATTED	CHRONOS-STRING	MDY	STRING-WD-MDY	TRUE	DATE
182A	FORMATTED	CHRONOS-STRING	MDY	STRING-WD-MDY	TRUE	TIME
012A	FORMATTED	CHRONOS-STRING	MDY	STRING-WD-DMY	FALSE	

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
092A	FORMATTED	CHRONOS-STRING	MDY	STRING-WD-DMY	TRUE	DATE
192A	FORMATTED	CHRONOS-STRING	MDY	STRING-WD-DMY	TRUE	TIME
022A	FORMATTED	CHRONOS-STRING	MDY	STRING-MDY	FALSE	
0A2A	FORMATTED	CHRONOS-STRING	MDY	STRING-MDY	TRUE	DATE
1A2A	FORMATTED	CHRONOS-STRING	MDY	STRING-MDY	TRUE	TIME
032A	FORMATTED	CHRONOS-STRING	MDY	STRING-DMY	FALSE	
0B2A	FORMATTED	CHRONOS-STRING	MDY	STRING-DMY	TRUE	DATE
1B2A	FORMATTED	CHRONOS-STRING	MDY	STRING-DMY	TRUE	TIME
006A	FORMATTED	CHRONOS-STRING	DMY	STRING-WD-MDY	FALSE	
086A	FORMATTED	CHRONOS-STRING	DMY	STRING-WD-MDY	TRUE	DATE
186A	FORMATTED	CHRONOS-STRING	DMY	STRING-WD-MDY	TRUE	TIME
016A	FORMATTED	CHRONOS-STRING	DMY	STRING-WD-DMY	FALSE	
096A	FORMATTED	CHRONOS-STRING	DMY	STRING-WD-DMY	TRUE	DATE
196A	FORMATTED	CHRONOS-STRING	DMY	STRING-WD-DMY	TRUE	TIME
026A	FORMATTED	CHRONOS-STRING	DMY	STRING-MDY	FALSE	
0A6A	FORMATTED	CHRONOS-STRING	DMY	STRING-MDY	TRUE	DATE
1A6A	FORMATTED	CHRONOS-STRING	DMY	STRING-MDY	TRUE	TIME
036A	FORMATTED	CHRONOS-STRING	DMY	STRING-DMY	FALSE	
0B6A	FORMATTED	CHRONOS-STRING	DMY	STRING-DMY	TRUE	DATE
1B6A	FORMATTED	CHRONOS-STRING	DMY	STRING-DMY	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
00AA	FORMATTED	CHRONOS-STRING	YMD	STRING-WD-MDY	FALSE	
08AA	FORMATTED	CHRONOS-STRING	YMD	STRING-WD-MDY	TRUE	DATE
18AA	FORMATTED	CHRONOS-STRING	YMD	STRING-WD-MDY	TRUE	TIME
01AA	FORMATTED	CHRONOS-STRING	YMD	STRING-WD-DMY	FALSE	
09AA	FORMATTED	CHRONOS-STRING	YMD	STRING-WD-DMY	TRUE	DATE
19AA	FORMATTED	CHRONOS-STRING	YMD	STRING-WD-DMY	TRUE	TIME
02AA	FORMATTED	CHRONOS-STRING	YMD	STRING-MDY	FALSE	
0AAA	FORMATTED	CHRONOS-STRING	YMD	STRING-MDY	TRUE	DATE
1AAA	FORMATTED	CHRONOS-STRING	YMD	STRING-MDY	TRUE	TIME
03AA	FORMATTED	CHRONOS-STRING	YMD	STRING-DMY	FALSE	
0BAA	FORMATTED	CHRONOS-STRING	YMD	STRING-DMY	TRUE	DATE
1BAA	FORMATTED	CHRONOS-STRING	YMD	STRING-DMY	TRUE	TIME
000B	UNFORMATTED	CHRONOS-STAMP	MDY	MDY	FALSE	
080B	UNFORMATTED	CHRONOS-STAMP	MDY	MDY	TRUE	DATE
180B	UNFORMATTED	CHRONOS-STAMP	MDY	MDY	TRUE	TIME
010B	UNFORMATTED	CHRONOS-STAMP	MDY	DMY	FALSE	
090B	UNFORMATTED	CHRONOS-STAMP	MDY	DMY	TRUE	DATE
190B	UNFORMATTED	CHRONOS-STAMP	MDY	DMY	TRUE	TIME
020B	UNFORMATTED	CHRONOS-STAMP	MDY	YMD	FALSE	
0A0B	UNFORMATTED	CHRONOS-STAMP	MDY	YMD	TRUE	DATE
1A0B	UNFORMATTED	CHRONOS-STAMP	MDY	YMD	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
004B	UNFORMATTED	CHRONOS-STAMP	DMY	MDY	FALSE	
084B	UNFORMATTED	CHRONOS-STAMP	DMY	MDY	TRUE	DATE
184B	UNFORMATTED	CHRONOS-STAMP	DMY	MDY	TRUE	TIME
014B	UNFORMATTED	CHRONOS-STAMP	DMY	DMY	FALSE	
094B	UNFORMATTED	CHRONOS-STAMP	DMY	DMY	TRUE	DATE
194B	UNFORMATTED	CHRONOS-STAMP	DMY	DMY	TRUE	TIME
024B	UNFORMATTED	CHRONOS-STAMP	DMY	YMD	FALSE	
0A4B	UNFORMATTED	CHRONOS-STAMP	DMY	YMD	TRUE	DATE
1A4B	UNFORMATTED	CHRONOS-STAMP	DMY	YMD	TRUE	TIME
008B	UNFORMATTED	CHRONOS-STAMP	YMD	MDY	FALSE	
088B	UNFORMATTED	CHRONOS-STAMP	YMD	MDY	TRUE	DATE
188B	UNFORMATTED	CHRONOS-STAMP	YMD	MDY	TRUE	TIME
018B	UNFORMATTED	CHRONOS-STAMP	YMD	DMY	FALSE	
098B	UNFORMATTED	CHRONOS-STAMP	YMD	DMY	TRUE	DATE
198B	UNFORMATTED	CHRONOS-STAMP	YMD	DMY	TRUE	TIME
028B	UNFORMATTED	CHRONOS-STAMP	YMD	YMD	FALSE	
0A8B	UNFORMATTED	CHRONOS-STAMP	YMD	YMD	TRUE	DATE
1A8B	UNFORMATTED	CHRONOS-STAMP	YMD	YMD	TRUE	TIME
0013	UNFORMATTED	FORMATTED	MDY	MDY	FALSE	
0813	UNFORMATTED	FORMATTED	MDY	MDY	TRUE	DATE
1813	UNFORMATTED	FORMATTED	MDY	MDY	TRUE	TIME
0113	UNFORMATTED	FORMATTED	MDY	DMY	FALSE	
0913	UNFORMATTED	FORMATTED	MDY	DMY	TRUE	DATE
1913	UNFORMATTED	FORMATTED	MDY	DMY	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0213	UNFORMATTED	FORMATTED	MDY	YMD	FALSE	
0A13	UNFORMATTED	FORMATTED	MDY	YMD	TRUE	DATE
1A13	UNFORMATTED	FORMATTED	MDY	YMD	TRUE	TIME
0053	UNFORMATTED	FORMATTED	DMY	MDY	FALSE	
0853	UNFORMATTED	FORMATTED	DMY	MDY	TRUE	DATE
1853	UNFORMATTED	FORMATTED	DMY	MDY	TRUE	TIME
0153	UNFORMATTED	FORMATTED	DMY	DMY	FALSE	
0953	UNFORMATTED	FORMATTED	DMY	DMY	TRUE	DATE
1953	UNFORMATTED	FORMATTED	DMY	DMY	TRUE	TIME
0253	UNFORMATTED	FORMATTED	DMY	YMD	FALSE	
0A53	UNFORMATTED	FORMATTED	DMY	YMD	TRUE	DATE
1A53	UNFORMATTED	FORMATTED	DMY	YMD	TRUE	TIME
0093	UNFORMATTED	FORMATTED	YMD	MDY	FALSE	
0893	UNFORMATTED	FORMATTED	YMD	MDY	TRUE	DATE
1893	UNFORMATTED	FORMATTED	YMD	MDY	TRUE	TIME
0193	UNFORMATTED	FORMATTED	YMD	DMY	FALSE	
0993	UNFORMATTED	FORMATTED	YMD	DMY	TRUE	DATE
1993	UNFORMATTED	FORMATTED	YMD	DMY	TRUE	TIME
0293	UNFORMATTED	FORMATTED	YMD	YMD	FALSE	
0A93	UNFORMATTED	FORMATTED	YMD	YMD	TRUE	DATE
1A93	UNFORMATTED	FORMATTED	YMD	YMD	TRUE	TIME
001B	UNFORMATTED	UNFORMATTED	MDY	MDY	FALSE	
081B	UNFORMATTED	UNFORMATTED	MDY	MDY	TRUE	DATE
181B	UNFORMATTED	UNFORMATTED	MDY	MDY	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
011B	UNFORMATTED	UNFORMATTED	MDY	DMY	FALSE	
091B	UNFORMATTED	UNFORMATTED	MDY	DMY	TRUE	DATE
191B	UNFORMATTED	UNFORMATTED	MDY	DMY	TRUE	TIME
021B	UNFORMATTED	UNFORMATTED	MDY	YMD	FALSE	
0A1B	UNFORMATTED	UNFORMATTED	MDY	YMD	TRUE	DATE
1A1B	UNFORMATTED	UNFORMATTED	MDY	YMD	TRUE	TIME
005B	UNFORMATTED	UNFORMATTED	DMY	MDY	FALSE	
085B	UNFORMATTED	UNFORMATTED	DMY	MDY	TRUE	DATE
185B	UNFORMATTED	UNFORMATTED	DMY	MDY	TRUE	TIME
015B	UNFORMATTED	UNFORMATTED	DMY	DMY	FALSE	
095B	UNFORMATTED	UNFORMATTED	DMY	DMY	TRUE	DATE
195B	UNFORMATTED	UNFORMATTED	DMY	DMY	TRUE	TIME
025B	UNFORMATTED	UNFORMATTED	DMY	YMD	FALSE	
0A5B	UNFORMATTED	UNFORMATTED	DMY	YMD	TRUE	DATE
1A5B	UNFORMATTED	UNFORMATTED	DMY	YMD	TRUE	TIME
009B	UNFORMATTED	UNFORMATTED	YMD	MDY	FALSE	
089B	UNFORMATTED	UNFORMATTED	YMD	MDY	TRUE	DATE
189B	UNFORMATTED	UNFORMATTED	YMD	MDY	TRUE	TIME
019B	UNFORMATTED	UNFORMATTED	YMD	DMY	FALSE	
099B	UNFORMATTED	UNFORMATTED	YMD	DMY	TRUE	DATE
199B	UNFORMATTED	UNFORMATTED	YMD	DMY	TRUE	TIME
029B	UNFORMATTED	UNFORMATTED	YMD	YMD	FALSE	
0A9B	UNFORMATTED	UNFORMATTED	YMD	YMD	TRUE	DATE
1A9B	UNFORMATTED	UNFORMATTED	YMD	YMD	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0023	UNFORMATTED	JULIAN	MDY	MDY	FALSE	
0823	UNFORMATTED	JULIAN	MDY	MDY	TRUE	DATE
1823	UNFORMATTED	JULIAN	MDY	MDY	TRUE	TIME
0123	UNFORMATTED	JULIAN	MDY	DMY	FALSE	
0923	UNFORMATTED	JULIAN	MDY	DMY	TRUE	DATE
1923	UNFORMATTED	JULIAN	MDY	DMY	TRUE	TIME
0223	UNFORMATTED	JULIAN	MDY	YMD	FALSE	
0A23	UNFORMATTED	JULIAN	MDY	YMD	TRUE	DATE
1A23	UNFORMATTED	JULIAN	MDY	YMD	TRUE	TIME
0063	UNFORMATTED	JULIAN	DMY	MDY	FALSE	
0863	UNFORMATTED	JULIAN	DMY	MDY	TRUE	DATE
1863	UNFORMATTED	JULIAN	DMY	MDY	TRUE	TIME
0163	UNFORMATTED	JULIAN	DMY	DMY	FALSE	
0963	UNFORMATTED	JULIAN	DMY	DMY	TRUE	DATE
1963	UNFORMATTED	JULIAN	DMY	DMY	TRUE	TIME
0263	UNFORMATTED	JULIAN	DMY	YMD	FALSE	
0A63	UNFORMATTED	JULIAN	DMY	YMD	TRUE	DATE
1A63	UNFORMATTED	JULIAN	DMY	YMD	TRUE	TIME
00A3	UNFORMATTED	JULIAN	YMD	MDY	FALSE	
08A3	UNFORMATTED	JULIAN	YMD	MDY	TRUE	DATE
18A3	UNFORMATTED	JULIAN	YMD	MDY	TRUE	TIME
01A3	UNFORMATTED	JULIAN	YMD	DMY	FALSE	
09A3	UNFORMATTED	JULIAN	YMD	DMY	TRUE	DATE
19A3	UNFORMATTED	JULIAN	YMD	DMY	TRUE	TIME
02A3	UNFORMATTED	JULIAN	YMD	YMD	FALSE	

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0AA3	UNFORMATTED	JULIAN	YMD	YMD	TRUE	DATE
1AA3	UNFORMATTED	JULIAN	YMD	YMD	TRUE	TIME
002B	UNFORMATTED	CHRONOS-STRING	MDY	STRING-WD-MDY	FALSE	
082B	UNFORMATTED	CHRONOS-STRING	MDY	STRING-WD-MDY	TRUE	DATE
182B	UNFORMATTED	CHRONOS-STRING	MDY	STRING-WD-MDY	TRUE	TIME
012B	UNFORMATTED	CHRONOS-STRING	MDY	STRING-WD-DMY	FALSE	
092B	UNFORMATTED	CHRONOS-STRING	MDY	STRING-WD-DMY	TRUE	DATE
192B	UNFORMATTED	CHRONOS-STRING	MDY	STRING-WD-DMY	TRUE	TIME
022B	UNFORMATTED	CHRONOS-STRING	MDY	STRING-MDY	FALSE	
0A2B	UNFORMATTED	CHRONOS-STRING	MDY	STRING-MDY	TRUE	DATE
1A2B	UNFORMATTED	CHRONOS-STRING	MDY	STRING-MDY	TRUE	TIME
032B	UNFORMATTED	CHRONOS-STRING	MDY	STRING-DMY	FALSE	
0B2B	UNFORMATTED	CHRONOS-STRING	MDY	STRING-DMY	TRUE	DATE
1B2B	UNFORMATTED	CHRONOS-STRING	MDY	STRING-DMY	TRUE	TIME
006B	UNFORMATTED	CHRONOS-STRING	DMY	STRING-WD-MDY	FALSE	
086B	UNFORMATTED	CHRONOS-STRING	DMY	STRING-WD-MDY	TRUE	DATE
186B	UNFORMATTED	CHRONOS-STRING	DMY	STRING-WD-MDY	TRUE	TIME
016B	UNFORMATTED	CHRONOS-STRING	DMY	STRING-WD-DMY	FALSE	

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
096B	UNFORMATTED	CHRONOS-STRING	DMY	STRING-WD-DMY	TRUE	DATE
196B	UNFORMATTED	CHRONOS-STRING	DMY	STRING-WD-DMY	TRUE	TIME
026B	UNFORMATTED	CHRONOS-STRING	DMY	STRING-MDY	FALSE	
0A6B	UNFORMATTED	CHRONOS-STRING	DMY	STRING-MDY	TRUE	DATE
1A6B	UNFORMATTED	CHRONOS-STRING	DMY	STRING-MDY	TRUE	TIME
036B	UNFORMATTED	CHRONOS-STRING	DMY	STRING-DMY	FALSE	
0B6B	UNFORMATTED	CHRONOS-STRING	DMY	STRING-DMY	TRUE	DATE
1B6B	UNFORMATTED	CHRONOS-STRING	DMY	STRING-DMY	TRUE	TIME
00AB	UNFORMATTED	CHRONOS-STRING	YMD	STRING-WD-MDY	FALSE	
08AB	UNFORMATTED	CHRONOS-STRING	YMD	STRING-WD-MDY	TRUE	DATE
18AB	UNFORMATTED	CHRONOS-STRING	YMD	STRING-WD-MDY	TRUE	TIME
01AB	UNFORMATTED	CHRONOS-STRING	YMD	STRING-WD-DMY	FALSE	
09AB	UNFORMATTED	CHRONOS-STRING	YMD	STRING-WD-DMY	TRUE	DATE
19AB	UNFORMATTED	CHRONOS-STRING	YMD	STRING-WD-DMY	TRUE	TIME
02AB	UNFORMATTED	CHRONOS-STRING	YMD	STRING-MDY	FALSE	
0AAB	UNFORMATTED	CHRONOS-STRING	YMD	STRING-MDY	TRUE	DATE
1AAB	UNFORMATTED	CHRONOS-STRING	YMD	STRING-MDY	TRUE	TIME
03AB	UNFORMATTED	CHRONOS-STRING	YMD	STRING-DMY	FALSE	
0BAB	UNFORMATTED	CHRONOS-STRING	YMD	STRING-DMY	TRUE	DATE
1BAB	UNFORMATTED	CHRONOS-STRING	YMD	STRING-DMY	TRUE	TIME

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
000C	JULIAN	CHRONOS-STAMP	MDY	MDY	FALSE	
080C	JULIAN	CHRONOS-STAMP	MDY	MDY	TRUE	DATE
180C	JULIAN	CHRONOS-STAMP	MDY	MDY	TRUE	TIME
010C	JULIAN	CHRONOS-STAMP	MDY	DMY	FALSE	
090C	JULIAN	CHRONOS-STAMP	MDY	DMY	TRUE	DATE
190C	JULIAN	CHRONOS-STAMP	MDY	DMY	TRUE	TIME
020C	JULIAN	CHRONOS-STAMP	MDY	YMD	FALSE	
0A0C	JULIAN	CHRONOS-STAMP	MDY	YMD	TRUE	DATE
1A0C	JULIAN	CHRONOS-STAMP	MDY	YMD	TRUE	TIME
004C	JULIAN	CHRONOS-STAMP	DMY	MDY	FALSE	
084C	JULIAN	CHRONOS-STAMP	DMY	MDY	TRUE	DATE
184C	JULIAN	CHRONOS-STAMP	DMY	MDY	TRUE	TIME
014C	JULIAN	CHRONOS-STAMP	DMY	DMY	FALSE	
094C	JULIAN	CHRONOS-STAMP	DMY	DMY	TRUE	DATE
194C	JULIAN	CHRONOS-STAMP	DMY	DMY	TRUE	TIME
024C	JULIAN	CHRONOS-STAMP	DMY	YMD	FALSE	
0A4C	JULIAN	CHRONOS-STAMP	DMY	YMD	TRUE	DATE
1A4C	JULIAN	CHRONOS-STAMP	DMY	YMD	TRUE	TIME
008C	JULIAN	CHRONOS-STAMP	YMD	MDY	FALSE	
088C	JULIAN	CHRONOS-STAMP	YMD	MDY	TRUE	DATE
188C	JULIAN	CHRONOS-STAMP	YMD	MDY	TRUE	TIME
018C	JULIAN	CHRONOS-STAMP	YMD	DMY	FALSE	
098C	JULIAN	CHRONOS-STAMP	YMD	DMY	TRUE	DATE
198C	JULIAN	CHRONOS-STAMP	YMD	DMY	TRUE	TIME
028C	JULIAN	CHRONOS-STAMP	YMD	YMD	FALSE	

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0A8C	JULIAN	CHRONOS-STAMP	YMD	YMD	TRUE	DATE
1A8C	JULIAN	CHRONOS-STAMP	YMD	YMD	TRUE	TIME
0014	JULIAN	FORMATTED	MDY	MDY	FALSE	
0814	JULIAN	FORMATTED	MDY	MDY	TRUE	DATE
1814	JULIAN	FORMATTED	MDY	MDY	TRUE	TIME
0114	JULIAN	FORMATTED	MDY	DMY	FALSE	
0914	JULIAN	FORMATTED	MDY	DMY	TRUE	DATE
1914	JULIAN	FORMATTED	MDY	DMY	TRUE	TIME
0214	JULIAN	FORMATTED	MDY	YMD	FALSE	
0A14	JULIAN	FORMATTED	MDY	YMD	TRUE	DATE
1A14	JULIAN	FORMATTED	MDY	YMD	TRUE	TIME
0054	JULIAN	FORMATTED	DMY	MDY	FALSE	
0854	JULIAN	FORMATTED	DMY	MDY	TRUE	DATE
1854	JULIAN	FORMATTED	DMY	MDY	TRUE	TIME
0154	JULIAN	FORMATTED	DMY	DMY	FALSE	
0954	JULIAN	FORMATTED	DMY	DMY	TRUE	DATE
1954	JULIAN	FORMATTED	DMY	DMY	TRUE	TIME
0254	JULIAN	FORMATTED	DMY	YMD	FALSE	
0A54	JULIAN	FORMATTED	DMY	YMD	TRUE	DATE
1A54	JULIAN	FORMATTED	DMY	YMD	TRUE	TIME
0094	JULIAN	FORMATTED	YMD	MDY	FALSE	
0894	JULIAN	FORMATTED	YMD	MDY	TRUE	DATE
1894	JULIAN	FORMATTED	YMD	MDY	TRUE	TIME
0194	JULIAN	FORMATTED	YMD	DMY	FALSE	

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0994	JULIAN	FORMATTED	YMD	DMY	TRUE	DATE
1994	JULIAN	FORMATTED	YMD	DMY	TRUE	TIME
0294	JULIAN	FORMATTED	YMD	YMD	FALSE	
0A94	JULIAN	FORMATTED	YMD	YMD	TRUE	DATE
1A94	JULIAN	FORMATTED	YMD	YMD	TRUE	TIME
001C	JULIAN	UNFORMATTED	MDY	MDY	FALSE	
081C	JULIAN	UNFORMATTED	MDY	MDY	TRUE	DATE
181C	JULIAN	UNFORMATTED	MDY	MDY	TRUE	TIME
011C	JULIAN	UNFORMATTED	MDY	DMY	FALSE	
091C	JULIAN	UNFORMATTED	MDY	DMY	TRUE	DATE
191C	JULIAN	UNFORMATTED	MDY	DMY	TRUE	TIME
021C	JULIAN	UNFORMATTED	MDY	YMD	FALSE	
0A1C	JULIAN	UNFORMATTED	MDY	YMD	TRUE	DATE
1A1C	JULIAN	UNFORMATTED	MDY	YMD	TRUE	TIME
005C	JULIAN	UNFORMATTED	DMY	MDY	FALSE	
085C	JULIAN	UNFORMATTED	DMY	MDY	TRUE	DATE
185C	JULIAN	UNFORMATTED	DMY	MDY	TRUE	TIME
015C	JULIAN	UNFORMATTED	DMY	DMY	FALSE	
095C	JULIAN	UNFORMATTED	DMY	DMY	TRUE	DATE
195C	JULIAN	UNFORMATTED	DMY	DMY	TRUE	TIME
025C	JULIAN	UNFORMATTED	DMY	YMD	FALSE	
0A5C	JULIAN	UNFORMATTED	DMY	YMD	TRUE	DATE
1A5C	JULIAN	UNFORMATTED	DMY	YMD	TRUE	TIME
009C	JULIAN	UNFORMATTED	YMD	MDY	FALSE	

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
089C	JULIAN	UNFORMATTED	YMD	MDY	TRUE	DATE
189C	JULIAN	UNFORMATTED	YMD	MDY	TRUE	TIME
019C	JULIAN	UNFORMATTED	YMD	DMY	FALSE	
099C	JULIAN	UNFORMATTED	YMD	DMY	TRUE	DATE
199C	JULIAN	UNFORMATTED	YMD	DMY	TRUE	TIME
029C	JULIAN	UNFORMATTED	YMD	YMD	FALSE	
0A9C	JULIAN	UNFORMATTED	YMD	YMD	TRUE	DATE
1A9C	JULIAN	UNFORMATTED	YMD	YMD	TRUE	TIME
0024	JULIAN	JULIAN	MDY	MDY	FALSE	
0824	JULIAN	JULIAN	MDY	MDY	TRUE	DATE
1824	JULIAN	JULIAN	MDY	MDY	TRUE	TIME
0124	JULIAN	JULIAN	MDY	DMY	FALSE	
0924	JULIAN	JULIAN	MDY	DMY	TRUE	DATE
1924	JULIAN	JULIAN	MDY	DMY	TRUE	TIME
0224	JULIAN	JULIAN	MDY	YMD	FALSE	
0A24	JULIAN	JULIAN	MDY	YMD	TRUE	DATE
1A24	JULIAN	JULIAN	MDY	YMD	TRUE	TIME
0064	JULIAN	JULIAN	DMY	MDY	FALSE	
0864	JULIAN	JULIAN	DMY	MDY	TRUE	DATE
1864	JULIAN	JULIAN	DMY	MDY	TRUE	TIME
0164	JULIAN	JULIAN	DMY	DMY	FALSE	
0964	JULIAN	JULIAN	DMY	DMY	TRUE	DATE
1964	JULIAN	JULIAN	DMY	DMY	TRUE	TIME
0264	JULIAN	JULIAN	DMY	YMD	FALSE	

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0A64	JULIAN	JULIAN	DMY	YMD	TRUE	DATE
1A64	JULIAN	JULIAN	DMY	YMD	TRUE	TIME
00A4	JULIAN	JULIAN	YMD	MDY	FALSE	
08A4	JULIAN	JULIAN	YMD	MDY	TRUE	DATE
18A4	JULIAN	JULIAN	YMD	MDY	TRUE	TIME
01A4	JULIAN	JULIAN	YMD	DMY	FALSE	
09A4	JULIAN	JULIAN	YMD	DMY	TRUE	DATE
19A4	JULIAN	JULIAN	YMD	DMY	TRUE	TIME
02A4	JULIAN	JULIAN	YMD	YMD	FALSE	
0AA4	JULIAN	JULIAN	YMD	YMD	TRUE	DATE
1AA4	JULIAN	JULIAN	YMD	YMD	TRUE	TIME
002C	JULIAN	CHRONOS-STRING	MDY	STRING-WD-MDY	FALSE	
082C	JULIAN	CHRONOS-STRING	MDY	STRING-WD-MDY	TRUE	DATE
182C	JULIAN	CHRONOS-STRING	MDY	STRING-WD-MDY	TRUE	TIME
012C	JULIAN	CHRONOS-STRING	MDY	STRING-WD-DMY	FALSE	
092C	JULIAN	CHRONOS-STRING	MDY	STRING-WD-DMY	TRUE	DATE
192C	JULIAN	CHRONOS-STRING	MDY	STRING-WD-DMY	TRUE	TIME
022C	JULIAN	CHRONOS-STRING	MDY	STRING-MDY	FALSE	
0A2C	JULIAN	CHRONOS-STRING	MDY	STRING-MDY	TRUE	DATE
1A2C	JULIAN	CHRONOS-STRING	MDY	STRING-MDY	TRUE	TIME
032C	JULIAN	CHRONOS-STRING	MDY	STRING-DMY	FALSE	

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
0B2C	JULIAN	CHRONOS-STRING	MDY	STRING-DMY	TRUE	DATE
1B2C	JULIAN	CHRONOS-STRING	MDY	STRING-DMY	TRUE	TIME
006C	JULIAN	CHRONOS-STRING	DMY	STRING-WD-MDY	FALSE	
086C	JULIAN	CHRONOS-STRING	DMY	STRING-WD-MDY	TRUE	DATE
186C	JULIAN	CHRONOS-STRING	DMY	STRING-WD-MDY	TRUE	TIME
016C	JULIAN	CHRONOS-STRING	DMY	STRING-WD-DMY	FALSE	
096C	JULIAN	CHRONOS-STRING	DMY	STRING-WD-DMY	TRUE	DATE
196C	JULIAN	CHRONOS-STRING	DMY	STRING-WD-DMY	TRUE	TIME
026C	JULIAN	CHRONOS-STRING	DMY	STRING-MDY	FALSE	
0A6C	JULIAN	CHRONOS-STRING	DMY	STRING-MDY	TRUE	DATE
1A6C	JULIAN	CHRONOS-STRING	DMY	STRING-MDY	TRUE	TIME
036C	JULIAN	CHRONOS-STRING	DMY	STRING-DMY	FALSE	
0B6C	JULIAN	CHRONOS-STRING	DMY	STRING-DMY	TRUE	DATE
1B6C	JULIAN	CHRONOS-STRING	DMY	STRING-DMY	TRUE	TIME
00AC	JULIAN	CHRONOS-STRING	YMD	STRING-WD-MDY	FALSE	
08AC	JULIAN	CHRONOS-STRING	YMD	STRING-WD-MDY	TRUE	DATE
18AC	JULIAN	CHRONOS-STRING	YMD	STRING-WD-MDY	TRUE	TIME
01AC	JULIAN	CHRONOS-STRING	YMD	STRING-WD-DMY	FALSE	
09AC	JULIAN	CHRONOS-STRING	YMD	STRING-WD-DMY	TRUE	DATE

Mode	Source	Destination	Source Format	Destination Format	Inc. Flag	Inc. Type
19AC	JULIAN	CHRONOS-STRING	YMD	STRING-WD-DMY	TRUE	TIME
02AC	JULIAN	CHRONOS-STRING	YMD	STRING-MDY	FALSE	
0AAC	JULIAN	CHRONOS-STRING	YMD	STRING-MDY	TRUE	DATE
1AAC	JULIAN	CHRONOS-STRING	YMD	STRING-MDY	TRUE	TIME
03AC	JULIAN	CHRONOS-STRING	YMD	STRING-DMY	FALSE	
0BAC	JULIAN	CHRONOS-STRING	YMD	STRING-DMY	TRUE	DATE
1BAC	JULIAN	CHRONOS-STRING	YMD	STRING-DMY	TRUE	TIME

INDEX