symbolics™

Volume 1B
System Index

Volume 1B. System Index

#996012

Copyright © 1984, Symbolics, Inc. of Cambridge, Massachusetts. All rights reserved.
Printed in USA. This document may not be reproduced in whole or in part without the prior written consent of Symbolics, Inc.

Design: Schafer/LaCasse
Cover and title page typography: Litho Composition Co.
Text typography: Century Schoolbook and Helvetica produced on a Symbolics 3600 Lisp Machine from Bitstream, Inc., outlines; text master printed on Symbolics LGP-1 Laser Graphics Printer.

The first Lisp Machine system was a product of the efforts of many people at the M.I.T. Artificial Intelligence Laboratory, and of the unique environment there. Portions of earliest versions of many of the documents in this documentation set were written at the AI Lab.

Contents

System Index

RN
Release Notes/
Patch Notes

NEWS
Newsletters/
Bug Reports

symbolics™

RN Release Notes/ Patch Notes

Cambridge, Massachusetts

Release 5.0 Release Notes
 # 995050

March 1984

This document corresponds to Release 5.0.

This document was prepared by the Documentation Group of Symbolics, Inc.

No representation or affirmation of fact contained in this document should be construed as a warranty by Symbolics, and its contents are subject to change without notice. Symbolics, Inc. assumes no responsibility for any errors that might appear in this document.

Symbolics software described in this document is furnished only under license, and may be used only in accordance with the terms of such license. Title to, and ownership of, such software shall at all times remain in Symbolics, Inc. Nothing contained herein implies the granting of a license to make, use, or sell any Symbolics equipment or software.

Symbolics is a trademark of Symbolics, Inc., Cambridge, Massachusetts.
UNIX is a trademark of Bell Laboratories, Inc.
VAX, TOPS-20, and VMS are trademarks of Digital Equipment Corporation.
TENEX is a registered trademark of Bolt Beranek and Newman Inc.

Copyright © 1984, Symbolics, Inc. of Cambridge, Massachusetts.
All rights reserved. Printed in USA.
This document may not be reproduced in whole or in part without the prior written consent of Symbolics, Inc.

Printing year and number: 87 86 85 84 9 8 7 6 5 4 3 2 1

Table of Contents

Page

1. Release 5.0: Introduction and Highlights 1

1.1 New Microcode in Release 5.0: 270 on 3600, 998 on LM-2 3

2. Changes to the Lisp Language and Compiler in Release 5.0 5

2.1 Incompatible Changes to Lisp in Release 5.0 5
2.1.1 Changes to **login** 5
2.1.2 Changes to Packages 6
2.1.3 Symbols in **global** and **keyword** packages with the same names 7
2.1.4 Symbols moved to or from **global** package 9
2.1.5 Keyword Symbols Are Self-evaluating 13
2.1.6 Functions moved from the **si** package to **global**: **deallocate-whole-resource**, **map-resource** 13
2.1.7 New special forms **catch** and **throw** replace ***catch** and ***throw** 14
2.1.8 Nonkeyword form of **make-array** is obsolete 16
2.1.9 **string-length** uses same coercion rules as **string** 16
2.1.10 Change in type of array returned by **string-append** 16
2.1.11 Changes to Readtable, Reader, and Printer for Common Lisp 16
2.1.12 Changes to **make-syn-stream** 20
2.1.13 **format** directives ~**@T** and ~**@*** replace ~**X** and ~**G** 21
2.1.14 Changes to **format:ochar** 21
2.1.15 Incompatible Changes to the Input Editor (Rubout Handler) 21
2.1.16 Changes to **open** 24
2.1.17 Changes to **renamef** and **copyf** 26
2.1.18 Changes to Host Determination in Pathnames 30
2.1.19 Meaning of argument changed for **fs:parse-pathname** 32
2.1.20 Arguments changed for **fs:user-homedir** and **fs:init-file-pathname** 34
2.1.21 Init File Pathnames Standardized 34
2.1.22 **:init** canonical pathname type removed 35
2.1.23 Changes to Logical Pathnames 35
2.1.24 **fs:make-logical-pathname-host** replaces **fs:add-logical-pathname-host** 37
2.1.25 Previously undocumented function: **fs:set-logical-pathname-host** 38
2.1.26 **load-file-list** obsolete 39
2.1.27 Change in arguments to **print-herald** 39
2.1.28 Change in arguments to **unadvise** 39

2.1.29 Window System Changes Associated with Mouse Input 40
2.1.30 **:clear-screen, :clear-eol,** and **:clear-eof** messages to windows renamed 44
2.2 New Features in Lisp in Release 5.0 44
2.2.1 New function: **eql** 44
2.2.2 New special form: **defconstant** 45
2.2.3 New special forms: **block** and **tagbody** 46
2.2.4 New special forms: **multiple-value-call** and **multiple-value-prog1** 47
2.2.5 3600 Supports Ieee Single- and Double-precision Floating Point 48
2.2.6 New function: **mod** 49
2.2.7 New functions: **byte, byte-size, byte-position** 50
2.2.8 New Metering Tools for the 3600 50
2.2.9 New Meters for the LM-2 52
2.2.10 New special form: **define-symbol-macro** 53
2.2.11 New function: **undefflavor** 53
2.2.12 New option for **defflavor: :required-init-keywords** 53
2.2.13 New option for **defflavor: :mixture** 54
2.2.14 New **format** directives: ~→ and ~← 55
2.2.15 New special form: **format:defformat** 56
2.2.16 New Features Associated with the Input Editor (Rubout Handler) 57
2.2.17 New macro: **sys:with-open-file-search** 69
2.2.18 New condition flavor: **fs:multiple-file-not-found** 69
2.2.19 New condition flavor: **fs:rename-across-hosts** 70
2.2.20 New variable: **fs:*remember-passwords*** 70
2.2.21 New function: **si:patch-loaded-p** 70
2.2.22 New functions: **si:make-process-queue, si:process-enqueue, si:process-dequeue, si:process-queue-locker, si:reset-process-queue** 71
2.2.23 New function: **applyhook** 72
2.2.24 New variable: **gc-on** 73
2.2.25 New initialization list: **:after-full-gc** 73
2.2.26 New variable: **dbg:*debug-io-override*** 74
2.2.27 New message to conditions: **:special-command-p** 74
2.2.28 New macro: **tv:with-mouse-grabbed-on-sheet** 74
2.2.29 New variable: **tv:cold-load-stream-old-selected-window** 74
2.2.30 New flavor: **tv:margin-space-mixin** 74
2.2.31 New font: **fonts:cptfonti** 75
2.2.32 New Choose-variable-values Keywords 76
2.3 Improvements to Lisp in Release 5.0 77
2.3.1 Previously undocumented special form: **destructuring-bind** 77
2.3.2 Invisible blocks in **progs** and **dos** 78
2.3.3 Previously undocumented function: **clear-resource** 78
2.3.4 Multidimensional Arrays on the 3600 Remember Actual Dimensions 78

2.3.5 New options for **make-plane**: **:initial-dimensions**, **:initial-origins** 80
2.3.6 New optional arguments to **string-upcase** and **string-downcase** 80
2.3.7 Previously undocumented function: **string-compare** 81
2.3.8 3600 select-methods handle **:operation-handled-p** and **:send-if-handles** 81
2.3.9 Compiler Performs Style Checking on All Forms 81
2.3.10 **sys:dump-forms-to-file** always puts package attribute into binary file 82
2.3.11 Previously undocumented macro: **swapf** 82
2.3.12 Compiler now warns about implicit **progns** in **loops** 82
2.3.13 Some Methods Can Use Combination Type as Method Type 83
2.3.14 Previously undocumented reader macro: #| and |# 83
2.3.15 New function to be called by reader macros: **si:read-recursive** 83
2.3.16 New optional arguments to **read-from-string** 84
2.3.17 Changes to **prompt-and-read** 84
2.3.18 Previously Undocumented Feature: Coroutine Streams 89
2.3.19 **format** ~\ directives can have package prefixes 92
2.3.20 Wildcard Directory Mapping Available 92
2.3.21 Previously undocumented function: **describe-system** 94
2.3.22 Improvements to **make-system**: **error-restart**, selective transformations 94
2.3.23 Second argument to **si:install-microcode** now optional 94
2.3.24 Change in argument to **process-wait-with-timeout** 95
2.3.25 New option for **si:sb-on**: **:mouse** (3600 only) 95
2.3.26 New format for **trace** output 96
2.3.27 Recursion in Bound and Default Handlers Eliminated 97
2.3.28 **:proceed** methods can now return **nil** 97
2.3.29 New clause for **condition-call**: **:no-error** 98
2.3.30 New message to arithmetic errors: **:operands** 98
2.3.31 Change in Debugger special command for **fs:directory-not-found** 98
2.3.32 New optional argument to **gc-immediately** 98
2.3.33 New optional arguments to **print-notifications** 99
2.3.34 **:draw-filled-in-circle** uses same algorithm as **:draw-circle** 99
2.3.35 Previously undocumented variables: **sys:mouse-x-scale-array** and **sys:mouse-y-scale-array** (LM-2 only) 100
2.3.36 New optional argument to **tv:mouse-wait** 101
2.3.37 New flavors: **tv:truncatable-lines-mixin**, **tv:truncating-lines-mixin** 102
2.3.38 New variable: **tv:*mouse-modifying-keystates*** 102
2.3.39 Shifted Mouse Clicks Can Now Be Used for Editor Commands 103

2.3.40 Previously undocumented functions: **tv:add-to-system-menu-programs-column, tv:add-to-system-menu-create-menu** 103
2.3.41 Argument to **:menu** type menu items can be a menu or a form 105
2.3.42 Clicking Middle Edits Current String in Choose-variable-values Windows 105
2.3.43 **tv:scroll-maintain-list** init function can take arguments 105

3. Changes to Networks in Release 5.0 **107**

3.1 Incompatible Changes to Networks in Release 5.0 107
3.1.1 Network Namespace System 107
3.1.2 **chaos:stream, chaos:close,** and **chaos:finish** renamed 107
3.1.3 **neti:reset, neti:enable,** and **neti:disable** replace **chaos:reset, chaos:enable,** and **chaos:disable** 107
3.1.4 Changes to **chaos:open-stream** 108
3.1.5 **chaos:send-unc-pkt** automatically returns the packet to the free pool 108
3.2 New Features in Networks in Release 5.0 108
3.2.1 New function: **chaos:conn-finished-p** 109
3.2.2 Changes to VMS Chaosnet 109
3.2.3 Changes to Serial I/O: Parity Recovery and Xon/Xoff Character Setting 109
3.2.4 Hdlc Serial I/O on the 3600 110
3.2.5 Interface to the Vadic Modem 111

4. Changes to Utilities in Release 5.0 **113**

4.1 Incompatible Changes to Utilities in Release 5.0 113
4.1.1 Default Font Format Now Bfd 113
4.1.2 Changes to Font Editor File Commands 113
4.1.3 Changes to FUNCTION C, FUNCTION M, and FUNCTION Q 113
4.2 New Features in Utilities in Release 5.0 114
4.2.1 New feature: Flavor Examiner (SELECT X) 114
4.2.2 New terminal program (SELECT T) 114
4.2.3 Show Hardcopy Status (M-X) replaces **chaos:print-lgp-queue** 115
4.3 Improvements to Utilities in Release 5.0 116
4.3.1 Font Editor and Inspector use ESCAPE to evaluate forms 116
4.3.2 Debugger C-M creates a process 116
4.3.3 M-SUSPEND selects frame with **break** read function for Debugger 116
4.3.4 END and C-END swapped in Converse 116
4.3.5 Changes to Converse Notifications 117

5. Changes to the File System in Release 5.0 **119**

5.1 Incompatible Changes to the File System in Release 5.0 119
5.1.1 New Default LMFS Translation Table for Sys Hosts 119
5.1.2 LMFS Dumper Supports Accordion Wildcards 119
5.2 New Features in the File System in Release 5.0 120
5.2.1 LMFS Now Supports Directory Links 120
5.2.2 LMFS Accordion Wildcards 120
5.2.3 Dumper Restarting and Append-to-tape Default 121

6. Changes to Zmacs in Release 5.0 123

6.1 Incompatible Changes to Zmacs in Release 5.0 123
6.1.1 Both default pathnames for Source Compare (m-X) now use **:newest** version 123
6.1.2 Changes to Add Patch Changed Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X) 123
6.1.3 Set Package (m-X) offers to create a package 123
6.1.4 Change in numeric arguments to Copy File (m-X) 124
6.2 New Features in Zmacs in Release 5.0 124
6.2.1 New Zmacs command: Resume Patch (m-X) 124
6.2.2 New Zmacs command: Start Private Patch (m-X) 124
6.2.3 New Zmacs command: Source Compare Newest Definition (m-X) 124
6.2.4 New Buffer-history Mechanism in Zmacs 125
6.2.5 New Zwei command: Comment Out Region (c-X c-;) 125
6.2.6 New Zwei command: Find Files in Tag Table (m-X) 125
6.2.7 New Zwei commands: Lowercase Code in [Region/Buffer] (m-X), Uppercase Code in [Region/Buffer] (m-X) 125
6.2.8 New canonical file type: **:mss** 126
6.3 Improvements to Zmacs in Release 5.0 126
6.3.1 Default File Name Changed for Commands in Dired Buffer 126
6.3.2 Major-mode-setting Commands Now Query About Updating File Attribute List 126
6.3.3 Change in Zmacs command Modified Two Windows (c-X 4) 127
6.3.4 Internal changes to macros **zwei:defmajor** and **zwei:defminor** 127

7. Changes to Zmail in Release 5.0 129

7.1 Incompatible Changes to Zmail in Release 5.0 129
7.1.1 Zmail Init File Pathnames Standardized 129
7.1.2 Babyl files with **summary-window-format** other than **t** or **nil** need to be edited 129
7.1.3 Ramifications of Host Colon Change for Babyl Files 129
7.2 New Features in Zmail in Release 5.0 130
7.2.1 Sorting by Conversations Available 130
7.2.2 New [map Over] Menu Item: [reply] 130
7.2.3 New [map Over] Menu Item: [select Conversation] 130

7.3 Improvements to Zmail in Release 5.0 130
7.3.1 Previously undocumented commands: Delete Conversation By References (m-X), Append Conversation By References (m-X) 130
7.3.2 Rfc822 Domain Addressing Supported 130

8. Changes to the FEP in Release 5.0 133

8.1 FEP Version 14: New Features 133
8.1.1 FEP Supports Hdlc Serial I/O 133
8.2 FEP Version 15: Incompatible Changes 133
8.2.1 h-c-*upper-left* stops execution of Lisp 133
8.2.2 **si:halt** replaces **sys:%halt** 133
8.2.3 >Configuration.fep Files Are Now Called >Boot.boot 134
8.2.4 New Defaults for FEP Commands 134
8.2.5 Disk Format Command Asks Different Question 134
8.3 FEP Version 15: New Features 135
8.3.1 Loading Sync Programs 135
8.4 FEP Version 15: Improvements 135
8.4.1 Show Configuration Command Displays More Information 135
8.4.2 Memory Board Not Needed in Lbus Slot 0 136
8.5 FEP Version 16: New Features 136
8.5.1 New FEP Commands: Add Disk-type and Clear Disk-types 136
8.6 FEP Version 16: Improvements 137
8.6.1 Unplugging Lemo Cables Should Not Halt the FEP 137
8.6.2 Continue Command Sends an All-keys-up Character to Lisp 137
8.6.3 More Information Available on Causes of Crashes 137
8.7 FEP Version 17: Improvements 140
8.7.1 Show Status Command Displays More Useful Information 140
8.8 FEP Version 18: Improvements 140
8.8.1 h-c-*upper-left* waits for Lisp to stop itself 140

9. Release 5.0: Notes and Clarifications 143

9.1 Clarifications and Corrections for Release 5.0 143
9.1.1 What happens when you cold boot 143
9.1.2 **sort** predicate should return **nil** for equal elements 144
9.1.3 **store** not supported on the 3600 145
9.1.4 Using **copy-array-portion** on the same array 145
9.1.5 **bitblt** width from the destination array 145
9.1.6 Inspecting hash arrays of **eq** hash tables not permitted 145
9.1.7 Known problem: **char-upcase** and **char-downcase** undefined for modified characters 145
9.1.8 How to use the **sys:function-parent** declaration 145
9.1.9 Use **record-source-file-name** instead of (**remprop** *symbol* **':source-file-name**) 147
9.1.10 Use **cdr** with locatives returned by **locf** 148
9.1.11 **rplaca** can be used with stack lists 148

9.1.12 FUNCTION 2 W displays current process name in status line 148
9.1.13 Known problem with **si:gc-reclaim-immediately** 148
9.1.14 **tv:set-default-font** not supported 149
9.1.15 Avoid Errors in the Mouse Process 149
9.1.16 **nil** not a valid menu item 149

10. Release 5.0: Operations and Site Management 151

10.1 Notes on Operations in Release 5.0 151
10.1.1 Backup Tape Reliability 151
10.1.2 Site Configuration for Dialnet 151

Index 153

1. Release 5.0: Introduction and Highlights

These notes accompany the release of Release 5.0. They describe changes made since Release 4.5. The changes are organized in the following sections. Within each section, the material is organized into incompatible changes, new features, and improvements.

Changes to the Lisp Language and Compiler in Release 5.0
This section describes changes relevant to the Lisp language and compiler. The biggest changes are these:

- Packages have completely changed for compatibility with Common Lisp. The **keyword** package is now separate from **user**, and it does not inherit from **global**. Files compiled in earlier systems will not work in Release 5.0 and should be recompiled.
- The procedure for logging in has changed as a result of a new network namespace system.
- The rubout handler has been renamed to the input editor and has been extensively changed.
- The readtable, reader, printer, and **open** have changed for compatibility with Common Lisp.
- The 3600 now supports IEEE-standard single- and double-precision floating point numbers.
- Several window system flavors and methods related to mouse input have been changed.
- Logical pathnames and translations have changed.
- Init file pathnames have been standardized.
- The first colon in a pathname now always delimits the host.

Changes to Networks in Release 5.0
This section describes changes in network implementation, interface, and protocols. The biggest change is the introduction of a network namespace system.

Changes to Utilities in Release 5.0
This section describes changes in what any other computer would call the operating system and utilities. This includes the

Debugger, the garbage collector, network support, and various system keyboard features. The most important changes are:

- Changes in the location of some keyboard keys on the 3600.
- A new system for yanking input in Zwei and the input editor.
- A new terminal program that incorporates Telnet and Supdup.
- A new utility, the Flavor Examiner, for finding information about flavors and methods.
- A new carry tape system.

Changes to the File System in Release 5.0
This section describes changes in the Lisp Machine file system. The major change is the introduction of accordion wildcards.

Changes to Zmacs in Release 5.0
This section describes changes in the Zmacs editor. The biggest change is the new yank system. See the document *Using the Input Editor*.

Changes to Zmail in Release 5.0
This section describes changes in the Zmail mail reading and sending program.

Changes to the FEP in Release 5.0
This section describes changes in the FEP. Release 5.0 requires FEP version number 17 or higher.

Release 5.0: Notes and Clarifications
This section contains explanations and clarifications of items that people found confusing in previous releases and documentation.

Release 5.0: Operations and Site Management
This section describes changes to the system and site configuration features of the system. These changes are important to the people who are responsible for the software at each site.

You can find all the incompatible changes by reading the first part of each section. A complete list of changes appears in the Table of Contents.

As in previous releases, many minor bugs have been fixed and performance in some areas has been improved. Only the more important or visible changes are mentioned here.

1.1 New Microcode in Release 5.0: 270 on 3600, 998 on LM-2

Release 5.0 world loads must be run with microcode version 270 on the 3600 and version 998 on the LM-2. The old world loads do not work with the new microcode, and the new world loads do not work with the old microcode.

2. Changes to the Lisp Language and Compiler in Release 5.0

2.1 Incompatible Changes to Lisp in Release 5.0

2.1.1 Changes to login

The **login** function has changed for Release 5 for compatibility with the new network naming scheme. The arguments are different. If you type a *user-name* argument that isn't the name of a known user in the network namespace, you are asked whether to supply a specific host to log into this time. Before **login** finishes, you are also prompted to add a user object to the network database, using **tv:edit-namespace-object**.

login *user-name* &key *host* (*load-init-file* **t**) *Function*

Note that although you enter the same user id for *user-name* as in previous releases, the user object that contains it now also contains the name of the host where your mail and init files reside. Therefore, you seldom need to supply a *host* argument to **login**. See the section "Network Database".

user-name is the name of a user. *host* is a particular host computer. If the value of *load-init-file* is **t**, as it is by default, the user's init file is loaded. If the value of *load-init-file* is **nil** the init file is not loaded.

You can log in as a registered user by not specifying a host, or you can log in to a specific host as a user on that host, not registered in the Lisp Machine namespace database.

If *host* requires passwords for logging in, you are asked for a password. When logging in to a TOPS-20 host, typing an asterisk before your password enables any special capabilities you may be authorized to use.

If anyone is logged into the machine already, **login** logs that user out before logging in *user-name*. See the function **logout**. **login** also runs the **login-initialization-list**. See the section "System Initialization Lists".

When **login** loads an init file, it looks for a file whose name depends on the host. See the section "Init File Naming Conventions". Init files should be written using **login-forms** so that **logout** can undo them. Usually, however, you cold boot the machine before logging in, to remove any traces of the previous user.

login returns **t**.

A typical use of **login** now looks like this:

```
(login 'djones)
```

If you supply an unknown user id and don't specify **:host**, you are given an opportunity to specify a particular host for the current login session, and to add the user object thus created to the network database (accomplished via **tv:edit-namespace-object**) for subsequent logins. You can instead select the Retry option, which is useful when the namespace server did not respond to your initial **login** request.

2.1.2 Changes to Packages

Packages have completely changed for Release 5.0. Formerly, packages were arranged in a hierarchy of *superpackages* and *subpackages*. This hierarchy no longer exists. Instead, symbols within a package are divided into *internal* and *external* symbols. One package can inherit the external symbols of another by *using* the second package. A package can also *import* or *export* symbols.

An important result of this reorganization is that the **keyword** package is no longer the same as **user**, and it does not inherit from **global** or any other packages. Thus, **foo** in the **user** package is no longer the same symbol as **:foo**, and **foo** in the **global** package is no longer the same symbol as **:foo**. The **fonts** package also no longer inherits from **global**.

You must change any symbols in your programs that are now in the wrong package. In particular, you must add package prefixes (colons) to symbols that are in the **global** or **user** package but should be in **keyword**, and you must remove package prefixes from symbols that are in **keyword** but should be in **global** or **user**.

You might have difficulty making these corrections because the editor currently signals an error when it parses a file in Lisp Mode that contains some symbols in the wrong package. To edit this kind of file, use Find File in Fundamental Mode (m-X).

All programs compiled in earlier systems should be recompiled in Release 5.0. The package information in code compiled in earlier systems is no longer valid. Unless your programs use symbol names that depend on the old package hierarchy, you should not have to rewrite programs to work in Release 5.0. Some functions and special forms have changed, but most changes are upward compatible.

Font files from previous releases load in Release 5.0, giving several warnings. You should then use the Font Editor to write files of type BFD. See the section "Default Font Format Now Bfd".

The *Lisp Machine Manual* chapter "Packages" has been rewritten. It describes both general changes to the package system and new functions, special forms, and condition flavors not documented here. See the document *Packages*.

2.1.2.1 New special form: defpackage

The special form **defpackage** replaces **package-declare**. **package-declare** still exists for compatibility with earlier systems.

See the special form **defpackage**.

2.1.2.2 New function: make-package

The function **make-package** replaces **pkg-create-package**. **pkg-create-package** still exists for compatibility with earlier systems.

See the function **make-package**.

2.1.2.3 intern, intern-local, intern-soft, and intern-local-soft return two values

The functions **intern**, **intern-local**, **intern-soft**, and **intern-local-soft** return two values instead of three. The second value is different but upward compatible.

See the function **intern**. See the function **intern-local**. See the function **intern-soft**. See the function **intern-local-soft**.

2.1.2.4 Optional argument to mapatoms-all and where-is eliminated

mapatoms-all and **where-is** no longer take an optional argument defaulting to the **global** package. They now always process all packages that are not invisible. The function **package-used-by-list** can help if you need to process only the subset of all packages that use some particular package.

See the function **mapatoms-all**. See the function **where-is**. See the function **package-used-by-list**.

2.1.3 Symbols in global and keyword packages with the same names

Before Release 5.0 the **keyword** package inherited from **global**. Symbols in the **global** and **keyword** packages with the same names were the same symbols. In Release 5.0 the keyword package does not inherit from **global**, and symbols in these packages with the same names are different symbols. Following is a partial list of symbols in the **keyword** package with the same names as symbols in **global**. Along with each symbol is the reason for its existence. Some other symbols that have the same names exist in both packages, but for these you would nearly always use the symbol in **global**.

Symbol	*Use*
:and	Method combination type
:append	Method combination type
:array	**defstruct** type, data type
:array-leader	**defstruct** type
:assoc	Choose-variable-values item type
:atom	Data type

:base	File attribute
:beep	Message to windows
:bitblt	Message to windows
:byte-size	**open** option
:cadr	#+ feature
:character	Data type
:close	Message to streams
:closure	Data type
:compile	**make-system** option
:cond	**trace** option
:defun-method	Function spec type
:delete	Message to pathnames (et al.)
:describe	Message to objects
:documentation	Property for **defvar** documentation
:equal	Message to (some) pathnames
:eval-when	**defstruct** option
:export	**defpackage** option
:fill-pointer	**make-array** option
:fix	Data type
:fixnum	Data type
:float	Data type
:flonum	Data type
:font	Internal; data type name is **font**
:funcall	Menu item type; constraint frames
:function	**trace** option
:get	Message to objects with property lists
:get-handler-for	Message to objects
:getl	Message to objects with property lists
:ibase	Obsolete file attribute
:import	**defpackage** option
:lambda-macro	Function spec type
:list	Method combination type; **defstruct** type
:list*	**defstruct** type
:make-array	**defstruct** option
:named-structure-symbol	**make-array** option
:nconc	Method combination type
:null	Data type
:open	Message to pathnames
:or	Method combination type
:otherwise	For **:case** method combination; use **otherwise** with **selectq**
:package	File attribute
:plist	Message to objects with property lists
:print	Message to streams
:progn	Method combination type
:prompt-and-read	Message to streams

:putprop	Message to objects with property lists
:random	Not a valid data type, but can be returned by one-argument **typep**
:read	Message to streams
:readline	**fquery** type
:remprop	Message to objects with property lists
:rubout-handler	Message to streams
:shadow	**defpackage** option
:shadowing-import	**defpackage** option
:step	**trace** option
:string-trim	**prompt-and-read** type
:symbol	Data type
:string	Data type
:tyi	Message to streams
:tyipeek	Message to streams
:tyo	Message to streams
:unbound-function	Internal to **who-calls** (not the special argument to it)

2.1.4 Symbols moved to or from global package

The following symbols were added to **global**:

block
byte
byte-position
byte-size
deallocate-whole-resource
defconstant
define-symbol-macro
defpackage
do-all-symbols
do-external-symbols
do-local-symbols
do-symbols
eql
export
find-all-symbols
import
keywordp
make-package
map-resource
mod
multiple-value-prog1
package-external-symbols
package-shadowing-symbols
package-use-list

package-used-by-list
pkg-add-relative-name
pkg-delete-relative-name
pkg-keyword-package
read-delimited-string
readline-or-nil
shadow
shadowing-import
tagbody
undefflavor
unexport
unuse-package
use-package
with-input-editing

The following symbols were added to **global** on the 3600 only:

%%arg-desc-quoted
%%arg-desc-rest-arg
%%q-fixnum
%%q-flonum
%%q-high-type
dfloat

The following symbols were removed from **global**:

◊g
◊p
bignum
def-open-coded
include
locative
open-code
page-table-area
pkg-contained-in
pkg-debug-copy
pkg-is-loaded-p
pkg-load
pkg-refname-alist
pkg-super-package
plane-ar-n
plane-as-n
process-class
screen-xgp-hardcopy
sg-area
sg-return-unsafe
small-flonum

source-file-name
supdup
telnet
unbind
with-resource

The following LM-2-specific symbols were removed from **global** on the 3600 only:

%%arg-desc-evaled-rest
%%arg-desc-fef-bind-hair
%%arg-desc-fef-quote-hair
%%arg-desc-quoted-rest
%%q-flag-bit
%%q-high-half
%%q-low-half
%24-bit-difference
%24-bit-plus
%24-bit-times
%activate-open-call-block
%allocate-and-initialize
%allocate-and-initialize-array
%arg-desc-evaled-rest
%arg-desc-fef-bind-hair
%arg-desc-fef-quote-hair
%arg-desc-quoted-rest
%assure-pdl-room
%divide-double
%float-double
%mar-high
%mar-low
%microcode-version-number
%multiply-fractions
%open-call-block
%p-deposit-field
%p-deposit-field-offset
%p-flag-bit
%p-mask-field
%p-mask-field-offset
%p-store-flag-bit
%remainder-double
%structure-boxed-size
%unibus-read
%unibus-write
%xbus-read
%xbus-write
***dif**
***plus**

***quo**
***times**
***unwind-stack**
ap-3
ar-3
art-32b
art-error
art-float
art-fps-float
art-half-fix
art-reg-pdl
art-special-pdl
art-stack-group-head
as-3
catch-all
cdr-error
clear-mar
dtp-array-header
dtp-array-pointer
dtp-entity
dtp-fef-pointer
dtp-free
dtp-header
dtp-instance-header
dtp-instance-variable-pointer
dtp-select-method
dtp-small-flonum
dtp-stack-group
dtp-symbol-header
dtp-trap
dtp-u-entry
enable-trapping
entity
entityp
fasd-update-file
fasl-append
fasload
font-next-plane
font-rasters-per-word
font-words-per-char
get-list-pointer-into-array
get-list-pointer-into-struct
get-locative-pointer-into-array
macro-compiled-program
mar-break
mar-mode

number-gc-on
print-error-mode
q-data-types
qc-file
qc-file-load
read-meter
return-next-value
set-current-band
set-current-microload
set-error-mode
set-mar
set-memory-size
small-float
small-floatp
swap-sv-of-sg-that-calls-me
swap-sv-on-call-out
trapping-enabled-p
write-meter
xstore

The following 3600-specific symbols were removed from **global** on the LM-2 only:

compiled-function-area
constants-area
control-tables
page-table-area
pname-area
property-list-area
stack-area
symbol-area
wired-control-tables

2.1.5 Keyword Symbols Are Self-evaluating

Keyword symbols now evaluate to themselves. You no longer have to quote them. The compiler takes account of this self-evaluation to produce efficient compiled code.

2.1.6 Functions moved from the si package to global: deallocate-whole-resource, map-resource

The functions **si:deallocate-whole-resource** and **si:map-resource** are now in the **global** package. These functions were previously undocumented.

deallocate-whole-resource *resource-name* *Function*

Deallocate all allocated objects of the resource specified by *resource-name*, returning them to the free-object list of the resource. You should use this function with caution. It marks all allocated objects as free, even if they are

still in use. If you call **deallocate-whole-resource** when objects are still in use, future calls to **allocate-resource** might allocate those same objects for another purpose.

map-resource *resource-name function* &rest *args* *Function*

Calls *function* once for every object in the resource specified by *resource-name*. *function* is called with the following arguments:

- The object
- **t** if the object is in use, or **nil** if it is free
- *resource-name*
- Any additional arguments specified by *args*

2.1.7 New special forms catch and throw replace *catch and *throw

The new special forms **catch** and **throw** are recommended for making nonlocal exits. ***catch** and ***throw** are supported for compatibility with earlier releases.

catch and **throw** differ from ***catch** and ***throw** mainly in the returned values: **catch** returns multiple values from its last body form when it exits normally, and **throw** causes **catch** and ***catch** to return multiple values that result from its second subform. You can use **catch** with ***throw** and ***catch** with **throw** (although this is not recommended). If control exits normally, the returned values depend on whether **catch** or ***catch** is used. If control exits abnormally, the returned values depend on whether **throw** or ***throw** is used.

The old Maclisp **catch** and **throw** macros are no longer supported on the LM-2. They were never supported on the 3600.

catch *tag body...* *Special Form*

catch is used with **throw** for nonlocal exits. **catch** first evaluates *tag* to obtain an object that is the "tag" of the catch. Then the *body* forms are evaluated in sequence, and **catch** returns the (possibly multiple) values of the last form in the body.

However, a **throw** or ***throw** form might be evaluated during the evaluation of one of the forms in *body*. In that case, if the throw "tag" is **eq** to the catch "tag" and if this **catch** is the innermost **catch** with that tag, the evaluation of the body is immediately aborted, and **catch** returns values specified by the **throw** or ***throw** form.

If the **catch** exits abnormally because of a **throw** form, it returns the (possibly multiple) values that result from evaluating **throw**'s second subform. If the **catch** exits abnormally because of a ***throw** form, it returns two values: the first is the result of evaluating ***throw**'s second subform, and the second is the result of evaluating ***throw**'s first subform (the tag thrown to).

On the LM-2 only, ***throw** and ***unwind-stack** cause the **catch** to return two additional values. If ***throw** is used, the third and fourth values are **nil**. If ***unwind-stack** is used, the third and fourth values are the third and fourth arguments to ***unwind-stack** (the active-frame-count and the action).

(catch 'foo *form*) catches a **(throw 'foo** *form*) but not a **(throw 'bar** *form*). It is an error if **throw** is done when no suitable **catch** exists.

The scope of the *tags* is dynamic. That is, the **throw** does not have to be lexically within the **catch** form; it is possible to throw out of a function that is called from inside a **catch** form.

On the LM-2 only, the values **t** and **nil** for *tag* are special: A **catch** whose tag is one of these values catches throws to any tag. These are for internal use only: **unwind-protect** uses **t**, and **catch-all** uses **nil**. The only difference between **t** and **nil** is in the error checking; **t** implies that after a "cleanup handler" is executed, control will be thrown again to the same tag. Thus, it is an error if a specific catch for this tag does not exist higher up the stack. With **nil**, the error check is not done.

Example:

```
(catch 'negative
  (mapcar (function (lambda (x)
                      (cond ((minusp x)
                             (throw 'negative x))
                            (t (f x)) )))
          y))
```

which returns a list of **f** of each element of **y** if they are all positive, otherwise the first negative member of **y**.

throw *tag form* *Special Form*

throw is used with **catch** to make nonlocal exits. It first evaluates *tag* to obtain an object that is the "tag" of the throw. It next evaluates *form* and saves the (possibly multiple) values. It then finds the innermost **catch** or ***catch** whose "tag" is **eq** to the "tag" that results from evaluating *tag*. It causes the **catch** or ***catch** to abort the evaluation of its body forms and to return all values that result from evaluating *form*. In the process, dynamic variable bindings are undone back to the point of the **catch**, and any **unwind-protect** cleanup forms are executed. An error is signalled if no suitable **catch** is found.

The scope of the *tags* is dynamic. That is, the **throw** does not have to be lexically within the **catch** form; it is possible to throw out of a function that is called from inside a **catch** form.

On the 3600, the value of *tag* cannot be the symbol **sys:unwind-protect-tag**; that is reserved for internal use. On the LM-2,

the values **t**, **nil**, and **0** for *tag* are reserved for internal use. At present you cannot use **t** and **nil** for *tag* on the 3600; this will be changed in a future release.

2.1.8 Nonkeyword form of make-array is obsolete

The nonkeyword form of **make-array** documented on page 113 of the *Lisp Machine Manual* is obsolete. All programs should use the form that takes keyword arguments in addition to the array dimensions.

See the function **make-array**.

2.1.9 string-length uses same coercion rules as string

string-length now uses the same rules as **string** in interpreting its argument as a string. In particular, it is an error for the argument to be a floating-point number, and **string-length** can now be used for an instance that handles the **:string-for-printing** message.

string-length *string* *Function*

string-length returns the number of characters in *string*. This function uses the same coercion rules as **string** in interpreting *string* as a string. **string-length** returns the **array-active-length** if *string* is a string, or the **array-active-length** of the pname if *string* is a symbol.

2.1.10 Change in type of array returned by string-append

Previously, when the first argument to **string-append** was an array, the result was an array of the same type. Now the result is an array of the same type as the argument with the greatest number of bits per element.

string-append &rest *strings* *Function*

Any number of strings are copied and concatenated into a single string. With a single argument, **string-append** simply copies it. The result is an array of the same type as the argument with the greatest number of bits per element. For example, if the arguments are arrays of type **art-string** and **art-fat-string**, an array of type **art-fat-string** is returned. **string-append** can be used to copy and concatenate any type of one-dimensional array. Example:

```
(string-append #/! "foo" #/!) => "!foo!"
```

2.1.11 Changes to Readtable, Reader, and Printer for Common Lisp

Changes have been made to the readtable, reader, and printer in preparation for the introduction of Common Lisp, which is not available in Release 5.0.

2.1.11.1 Reader Accepts Common Lisp Floating Point Exponents

The reader now accepts all exponent characters in Common Lisp.

Following is a summary of floating-point exponent characters and the way numbers containing them are read on the 3600 and LM-2.

Character	*3600*	*LM-2*
B or b	single-precision	flonum
D or d	double-precision	flonum
E or e	depends on value of **cl:*read-default-float-format***	depends on value of **cl:*read-default-float-format***
F or f	single-precision	flonum
L or l	double-precision	flonum
S or s	single-precision	small-flonum

2.1.11.2 New variable: cl:*read-default-float-format*

cl:*read-default-float-format* *Variable*

Controls how floating-point numbers with no exponent or an exponent preceded by "E" or "e" are read. Following is a summary of the way possible values cause these numbers to be read on the 3600 and LM-2:

Value	*3600*	*LM-2*
cl:single-float	single-precision	flonum
cl:double-float	double-precision	flonum
cl:short-float	single-precision	small-flonum
cl:long-float	double-precision	flonum

The default value is **cl:single-float**.

2.1.11.3 New descriptions: si:bitscale, si:digitscale, si:non-terminating-macro

Three syntax descriptions for characters have been added. **si:bitscale** and **si:digitscale** describe "left shift" operations on integers.
si:non-terminating-macro describes a character that can be a macro character or a symbol constituent. You can use **set-syntax-from-description** to set the syntax of characters to these descriptions.

set-syntax-from-description *char description* &optional *readtable* *Function*

Sets the syntax of *char* in *readtable* to be that described by the symbol *description*. The following descriptions are defined in the standard readtable:

si:alphabetic An ordinary character such as "A".

si:break A token separator such as "(". (Obviously left parenthesis has other properties besides being a break.)

si:whitespace A token separator that can be ignored, such as " ".

si:single A self-delimiting single-character symbol. The initial readtable does not contain any of these.

si:slash The character quoter. In the initial readtable this is "/".

si:verticalbar The symbol print-name quoter. In the initial readtable this is "|".

si:doublequote The string quoter. In the initial readtable this is "".

si:macro A macro character. Do not use this; use **set-syntax-macro-char**.

si:circlecross The octal escape for special characters. In the initial readtable this is "⊗".

si:bitscale A character that causes the fixnum to its left to be doubled the number of times indicated by the fixnum to its right. In the initial readtable this is "_". See the section "What the Reader Accepts".

si:digitscale A character that causes the fixnum to its left to be multiplied by **ibase** the number of times indicated by the fixnum to its right. In the initial readtable this is "^". See the section "What the Reader Accepts".

si:non-terminating-macro
A macro character that is not a token separator. This is a macro character if seen alone but is just a symbol constituent inside a symbol. You can use it as a character of a symbol other than the first without slashing it. (# would be one of these if it were not built into the reader.)

These symbols will probably be moved to the standard keyword package at some point. *readtable* defaults to the current readtable.

2.1.11.4 New reader macro: #B

#B*rational* reads *rational* (an integer or a ratio) in binary (radix 2). Examples:

```
#B1101 <=> 13.
#B1100\100 <=> 3
```

2.1.11.5 New reader macro: #:

#:*name* reads *name* as an uninterned symbol. It always creates a new symbol. Like all package prefixes, **#:** can be followed by any expression. Example: **#:(a b c)**.

2.1.11.6 Printing Uninterned Symbols

By default, uninterned symbols print with a **#:** prefix. You can make them print without a prefix by evaluating **(setf (si:pttbl-uninterned-prefix readtable) "")**.

2.1.11.7 #/ and #\ now identical

The reader macros #/ and #\ are now identical. You can use either to read a printing character (such as **#\A**) or the name of a nonprinting character (such as **#/c-X**). Unless you are using Common Lisp, #/ and #\ continue to return integers to represent characters.

The editor, however, still distinguishes between these two syntaxes. #/ prevents the editor from treating the character immediately following as a special character; #\ does not. Thus, to include an open parenthesis character in your program, use #/(, not #\(.

See the section "Sharp-sign Abbreviations".

2.1.11.8 Reading and Printing Character Objects

For Common Lisp only, the reader macros #/ and #\ can create character objects, and a double-quoted string can create a string of characters, depending on a property of the readtable. Regardless of this, character objects print with a #\ prefix.

2.1.11.9 Ratios read in current ibase and print in current base

Two integers separated by \ are read as a ratio of the integers. Ratios are now read in the base that is the value of **ibase** and printed in the base that is the value of **base**, not in decimal.

2.1.11.10 New Rules for Reading Ambiguous Tokens

Some tokens could be read as either symbols or integers in a base larger than 10. The variables **si:*read-extended-ibase-unsigned-number*** and **si:*read-extended-ibase-signed-number*** determine how these tokens are read.

si:*read-extended-ibase-unsigned-number* *Variable*

Controls how a token that could be a number or a symbol, and does not start with a + or - sign, is interpreted when **ibase** is greater than ten.

nil	It is never a number.
t	It is always a number.
:sharpsign	It is a symbol at top level, but a number after **#X** or **#***n***R**.
:single	It is a symbol except immediately after **#X** or **#***n***R**.

The default value is **:single**.

si:*read-extended-ibase-signed-number* *Variable*

Controls how a token that could be a number or a symbol, and starts with a + or - sign, is interpreted when **ibase** is greater than ten.

nil	It is never a number.
t	It is always a number.
:sharpsign	It is a symbol at top level, but a number after **#X** or **#***n***R**.
:single	It is a symbol except immediately after **#X** or **#***n***R**.

The default value is **:sharpsign**.

Some tokens could be read as either integers or floating-point numbers. Such tokens are always read as integers. Thus, **1e0** is an integer if the value of **ibase** is at least 15. But **1.0e0** is always read as a floating-point number because of the decimal point.

2.1.12 Changes to make-syn-stream

The argument to **make-syn-stream** can now be a symbol or a locative. **make-syn-stream** returns an uninterned symbol.

make-syn-stream *symbol* *Function*

make-syn-stream creates and returns a "synonym stream" (syn for short). *symbol* can be either a symbol or a locative.

If *symbol* is a symbol, the synonym stream is actually an uninterned symbol named **#:***symbol*-**syn-stream**. This generated symbol has a property that declares it to be a legitimate stream. This symbol is the value of *symbol*'s **si:syn-stream** property, and its function definition is forwarded to the value cell of *symbol* using a **dtp-external-value-cell-pointer**. Any operations sent to this stream are redirected to the stream that is the value of *symbol*.

If *symbol* is a locative, the synonym stream is an uninterned symbol named **#:syn-stream**. This generated symbol has a property that declares it to be a legitimate stream. The function definition of this symbol is forwarded to the cell designated by *symbol*. Any operations sent to this stream are redirected to the stream that is the contents of the cell to which *symbol* points.

2.1.13 format directives ~@T and ~@* replace ~X and ~G

A new format directive, **~@T**, replaces **~X**, and the new directive **~@*** replaces **~G**. For the time being **~X** and **~G** still work as in previous releases, but in a future release they will become number-formatting directives for compatibility with Common Lisp.

~@T outputs a space. ~*n*@T outputs *n* spaces.

~*n*@* "goes to" the *n*th argument (0 is the first). **~@*** or **~0@*** goes back to the first argument in *args*. Directives after a ~*n*@* will take sequential arguments after the one gone to. When within a **~{** construct, the "goto" is relative to the list of arguments being processed by the iteration. This is an "absolute goto"; for a "relative goto", see **~***.

2.1.14 Changes to format:ochar

The **:sail** style keyword for **format:ochar** has been replaced by **:brief**, which has slightly different consequences.

format:ochar *character* &optional *style top-explain minwidth* *Function*
&rest *options*

format:ochar outputs *character* in one of three styles, selected by the *style* argument. *minwidth* and *options* control padding as usual.

If *style* is **:read**, **nil**, or not specified, then the character is printed using #/ or #\ so that it could be read back in.

If *style* is **:editor**, then the output is in the style of the string "Meta-Rubout".

If *style* is **:brief**, a somewhat more abbreviated style is used in which "c-", "m-", and the like, are used to represent "Control" and "Meta", and shorter names for characters are also used when possible. See the section "The Character Set".

top-explain is useful with the **:editor** and **:brief** styles. It says that any character which has to be typed using the Symbol key should be followed by an explanation of how to type it. For example: "**α (Symbol-shift-A)**".

2.1.15 Incompatible Changes to the Input Editor (Rubout Handler)

The input editor, formerly called the rubout handler, has been extensively redesigned. This section describes only the incompatible changes for Release 5.0. For new features: See the section "New Features Associated with the Input Editor (Rubout Handler)".

2.1.15.1 Changes to Input Editor User Interface

The history mechanism of the input editor has changed. Each stream that uses the input editor now maintains an infinite-length input history. A global kill history is maintained, shared with other streams that use the input editor and with Zwei-based editors. Commands for yanking input from these histories have changed. The histories are now emptied before cold booting or disk saving, not when logging out. See the document *Using the Input Editor.*

Formerly, when you yanked previous input to the input editor, that input was displayed without the final character. You could reevaluate a previous form by yanking it and then typing the final character.

Now the reader uses *activation characters.* The input editor displays yanked input fully. You can edit the input if you wish and "activate" it by pressing END from anywhere within the input you are editing. If you want to put an activation character into the input, "quote" it using c-Q.

readline and the Converse pop-up window also use activation. You can press RETURN anywhere within a line to activate it for **readline**. You can press END anywhere within a Converse pop-up reply to send the reply.

2.1.15.2 Changes to input editor options :do-not-echo, :pass-through, :prompt, :reprompt

:do-not-echo now implies that the character arguments are activation characters.

:do-not-echo *char1 char2 ...* *Option*

The characters *char1*, *char2*, and so on, are interpreted as activation characters. The comparison is done with =, not **char-equal**, so that the control and meta bits are not masked off. The characters are not inserted into the input buffer and are not interpreted as input editor commands. When one of these characters is typed, the final **:tyi** value returned is the character, not a blip.

This option exists only for compatibility with earlier releases. New programs should use the **:activation** option.

:pass-through is now allowed only for characters with no modifier bits set.

:pass-through *char1 char2 ...* *Option*

The characters *char1*, *char2*, and so on are not to be treated as special by the input editor. This option is used to pass format effectors (such as HELP or CLEAR-INPUT) through to the reading function instead of interpreting them as input editor commands. **:pass-through** is allowed only for characters with no modifier bits set, that is, for character codes 0 through 377 (octal). For characters that have modifier bits set and must be visible to the reading function, use **:do-not-echo** or **:activation.**

The second argument to the **:prompt** and **:reprompt** functions is now a keyword instead of a character.

:prompt *function-or-string* *Option*

If *function-or-string* is a function and it is time for the user to be prompted, *function-or-string* is called with two arguments. The first is a stream it may print on; the second is a keyword that indicates the origin of the function call:

Keyword	*Function called from*
:prompt	**:rubout-handler** method of **tv:stream-mixin**
:restore	**:restore-rubout-handler-buffer** method of **tv:stream-mixin**
:insert, :overwrite, :temporary	**:finish-typeout** method of **tv:stream-mixin**
:refresh	Body of the input editor

If *function-or-string* is a string and it is time for the user to be prompted, *function-or-string* is displayed as a prompt.

The difference between **:prompt** and **:reprompt** is that the latter does not call the prompt function or display the string when the input editor is first entered, but only when the input is redisplayed (for example, after a screen clear). If both options are specified, **:reprompt** overrides **:prompt** except when the input editor is first entered.

:reprompt *function-or-string* *Option*

Like **:prompt** but calls *function-or-string* (if it is a function) or displays *function-or-string* (if it is a string) only when the input is redisplayed (for example, after a screen clear), not when the input editor is first entered. If both **:prompt** and **:reprompt** are specified, **:reprompt** overrides **:prompt** except when the input editor is first entered.

2.1.15.3 New error flavors: sys:parse-error and sys:parse-ferror

The input editor no longer catches all flavors of error. It now catches only errors built on the flavors **sys:parse-error** and **sys:parse-ferror**.

sys:parse-error *Flavor*

This flavor is built on **error** and is the type of error caught by the input editor. This flavor accepts the init keyword **:correct-input**. If the value is **t**, which is the default, the input editor prints "Type RUBOUT to correct your input" and does not erase the message until a non-self-inserting character is typed. If the value is **nil**, no message is printed, and any typeout from the read function is erased immediately after the next character is typed. Syntax errors signalled by read functions should be built on top of this flavor.

sys:parse-ferror *Flavor*

This flavor is built on **sys:parse-error** and **ferror**. It accepts the init keywords **:format-string** and **:format-args** as well as **:correct-input**. This flavor exists for read functions that do not have a special flavor of error defined for them.

2.1.15.4 New function: sys:parse-ferror

sys:parse-ferror *format-string* &rest *format-args* *Function*

Signals an error of flavor **sys:parse-ferror**. *format-string* and *format-args* are passed as the **:format-string** and **:format-args** init options to the error object.

See the flavor **sys:parse-ferror**.

2.1.16 Changes to open

2.1.16.1 Changes to open option :direction

:direction :probe replaces **:direction nil**. New permissible values are **:probe-directory** and **:probe-link**. Various misspelled values such as **:in** that used to work for some file hosts are no longer supported for consistency.

:direction *Option*

The **:direction** option allows the following values:

:input	The file is being opened for input. This is the default.
:output	The file is being opened for output.
:probe	A "probe" opening; no data are to be transferred, and the file is being opened only to gain access to or change its properties. Returns the truename of the object at the end of a link or chain of links. (**probef** is usually preferable to an explicit probe opening.)
:probe-link	The same as **:probe** except that links are not chased. Returns the truename of the object named, even if it is a link.
:probe-directory	The pathname is being opened to find out about the existence of its *directory* component. Otherwise, the semantics are the same as **:probe**. If the directory is not found, a file lookup error is signalled.
:probe-link	The same as **:probe** except that links are not chased. Returns the truename of the object named, even if it is a link.

2.1.16.2 New open option: :estimated-length

:estimated-length *Option*

The value of the **:estimated-length** option may be **nil** (the default), which means there is no estimated length, or a number of bytes indicating the estimated length of a file to be written. Some file systems use this to optimize disk allocation.

2.1.16.3 New open options: :if-exists and :if-does-not-exist

:if-exists determines what happens if the specified file already exists; **:if-does-not-exist** determines what happens if it does not exist. The ability to append files has been added.

:if-exists *Option*

Specifies the action to be taken if the **:direction** is **:output** and a file of the specified name already exists. If the direction is **:input** or **:probe** (or any of the **:probe**-like directions), this argument is ignored.

The following values are allowed:

:error Signals an error. This is the default when the version component of the filename is not **:newest**.

:new-version Creates a new file with the same file name but a larger version number. This is the default when the version component of the filename is **:newest**. File systems without version numbers may choose to implement this by effectively treating it as **supersede**.

:rename Renames the existing file to some other name, and then creates a new file with the specified name. On most file systems, this renaming happens at the time of a successful close.

:rename-and-delete Renames the existing file to some other name and then deletes it (but doesn't expunge it, on those systems that distinguish deletion from expunging). Then creates a new file with the specified name. On most file systems, this renaming happens at the time of a successful close.

:overwrite The existing file is used, and output operations on the stream destructively modify the file. The file pointer is initially positioned at the beginning of the file; however, the file is not truncated back to length zero when it is opened.

:truncate The existing file is used, and output operations on the stream destructively modify the file. The file pointer is initially positioned at the beginning of the file; at that

time, the file is truncated to length zero, and disk storage occupied by it is freed.

:append The existing file is used, and output operations on the stream destructively modify the file. The file pointer is initially positioned at the current end of the file.

:supersede Supersedes the existing file. If possible, the file system does not destroy the old file until the new stream is closed, against the possibility that the stream will be closed in "abort" mode. This differs from **:new-version** in that **:supersede** creates a new file with the same name as the old one, rather than a file name with a higher version number.

nil Does not create a file or even a stream. Instead, simply returns **nil** to indicate failure.

:if-does-not-exist *Option*

Specifies the action to be taken if the file does not already exist. The following values are allowed:

:error Signals an error. This is the default if the **:direction** is **:input**, **:probe**, or any of the **:probe**-like modes, or if the **:if-exists** argument is **:overwrite**, **:truncate**, or **:append**.

:create Creates an empty file with the specified name, and then proceeds as if it had already existed. This is the default if the **:direction** is **:output** and the **:if-exists** argument is anything but **:overwrite**, **:truncate**, or **:append**.

nil Does not create a file or even a stream. Instead, simply returns **nil** to indicate failure.

2.1.17 Changes to renamef and copyf

The following changes have been made to **renamef** and **copyf**:

- **renamef** now returns three values: the target pathname, the truename of the source pathname, and the truename of the target pathname. The **:rename** messages to streams and pathnames return the second and third of these values.

- For **renamef** and **copyf**, the target pathname defaults against the *link-opaque truename* of the source pathname.

- The **copyf** option **:copy-author** now defaults to **t**.

- A new **copyf** option, **:create-directories**, determines whether or not **copyf** tries to create a directory for the copy target.

renamef *file new-name* &optional *(error-p* **t***)* *Function*

renamef is a function for renaming one file to another. The Rename File (M-X) command in the editor uses this function.

file can be a pathname, a string, or a stream that is open to a file. The specified file is renamed to *new-name* (a pathname or string). If *error-p* is **t**, when an error occurs it is signalled as a Lisp error. If *error-p* is **nil** and an error occurs, the error object is returned; otherwise the three values described below are returned.

file must refer to a unique file; it cannot contain any wild components. *new-name* can contain wild components, which are eliminated after merging the defaults by means of **:translate-wild-pathname**. **renamef** first attempts to open *file*. When that has happened successfully, it parses *new-name* and merges it (using **fs:merge-pathnames**) against the *link-opaque truename* of *file* and version of **:newest**. This has the following result for version numbers.

Source	*Target*	*Result*
>foo>a.b.newest	>bar>	Retains the version number
>foo>a.b.newest	>bar>x	Makes a new version of >bar>x.b

The defaults for *new-name* come from the link-opaque truename of *file*. For systems without links, this is indistinguishable from the truename. Otherwise, the link-opaque truename depends on whether *file* contains an **:oldest** or **:newest** version. If it does not and if it is fully defaulted, with no wild components, the pathname is its own link-opaque truename. If a pathname *x* contains an **:oldest** or **:newest** version, the link-opaque truename is the pathname of the file or link that corresponds to *x*, with the version number filled in. For example, renaming the LMFS file `>a>p1.lisp` to `>b>` results in `>b>p1.lisp`, with the version of `>a>p1.lisp.newest` inherited. This is so whether `>a>p1.lisp.newest` is a real file, a link, or a rename-through link.

renamef returns three values:

1. The pathname produced by merging and defaulting *new-name*. This is the attempted result of the renaming.

2. The pathname of the object that was actually renamed. This might not be the same as *file*. For example, *file* might have an **:oldest** or **:newest** version, or LMFS rename-through links might be involved. This pathname never has an **:oldest** or **:newest** version.

3. The actual pathname that resulted from the renaming. This might not be the same as *new-name*. For example, *new-name* might have an **:oldest** or **:newest** version, or LMFS create-through links might be involved.

The **:rename** message to streams and pathnames returns the second and third of these values.

Examples:

This example is as simple as possible. Using LMFS, on host johnny, with no links involved:

```
(renamef "johnny:>a>foo.lisp" "bar") =>
#<LMFS-PATHNAME "johnny:>a>bar.lisp">
#<LMFS-PATHNAME "johnny:>a>foo.lisp.17">
#<LMFS-PATHNAME "johnny:>a>bar.lisp.1">
```

This example is as complex as possible. Using LMFS, on host eddie, with links

```
>abel>moe.lisp.4 => >baker>larry.lisp (rename-through) (latest)
>baker>larry.lisp.4 =>
  >charlie>sam.lisp.19 (not rename- or create-through) (latest)
>david>jerry.lisp.5 => >earl>ted.lisp (create-through) (latest)

(renamef "eddie:>abel>moe.lisp.4" "eddie:>david>jerry") =>
#<LMFS-PATHNAME "eddie:>david>jerry.lisp">
#<LMFS-PATHNAME "eddie:>baker>larry.lisp.4">
#<LMFS-PATHNAME "eddie:>earl>ted.lisp.1">
```

copyf *from-path to-path* &key (*characters* **':default**) (*byte-size* **nil**) *Function*
(*copy-creation-date* **t**) (*copy-author* **t**)
(*report-stream* **nil**) (*create-directories* **':query**)

copyf is a function for copying one file to another. Copy File (M-X) in the editor uses this function.

from-path and *to-path* are the source and destination pathnames, which can be file specs. *from-path* must refer to a unique file; it cannot contain any wild components. *to-path* can contain wild components, which are eliminated after merging the defaults by means of **:translate-wild-pathname**. **copyf** first attempts to open *from-path*. When that has happened successfully, it parses *to-path* and merges it (using **fs:merge-pathnames**) against the *link-opaque truename* of *from-path* and version of **:newest**. This has the following result for version numbers.

Source	*Target*	*Result*
>foo>a.b.newest	>bar>	Retains the version number
>foo>a.b.newest	>bar>x	Makes a new version of >bar>x.b

The defaults for *to-path* come from the *link-opaque truename* of *from-path*. For systems without links, this is indistinguishable from the truename. Otherwise, the link-opaque truename depends on whether *from-path* contains an **:oldest** or **:newest** version. If it does not and if it is fully defaulted, with no wild components, the pathname is its own link-opaque truename. If a pathname *x* contains an **:oldest** or **:newest** version, the link-opaque

truename is the pathname of the file or link that corresponds to *x*, with the version number filled in. For example, copying the LMFS file `>a>p1.lisp` to `>b>` results in `>b>p1.lisp`, with the version of `>a>p1.lisp.newest` inherited. This is so whether `>a>p1.lisp.newest` is a real file, a link, or a rename-through link.

By default, **copyf** copies the creation date and author of the file.

Following is a description of the other options:

:characters Possible values:

:default	**copyf** decides whether this is a binary or character transfer according to the canonical type of *from-path*. You do not need to supply this argument for standard file types. For types that are not known canonical types, it opens *from-path* in **:default** mode. In that case, the server for the file system containing *from-path* makes the character-or-binary decision.
t	Specifies that the transfer must be in character mode.
nil	Specifies that the transfer must be binary mode (in this case, you must supply *byte-size* if using a byte size other than 16).

:byte-size Specifies the byte size with which both files will be opened for binary transfers. You must supply **:byte-size** when **:characters** is **nil** and the byte size is other than 16. Otherwise, **copyf** determines the byte size from the file type for *from-path*. When *from-path* is a binary file with a known canonical type, it determines the byte size from the **:binary-file-byte-size** property of the type. When the file does not have a known type, it requests the byte size for *from-path* from the file server. When the server for the file system containing *from-path* cannot supply the byte size, it assumes that the byte size is 16.

:report-stream When **:report-stream** is **nil** (the default), the copying takes place with no messages. Otherwise, the value must be a stream for reporting the start and successful completion of the copying. The completion message contains the truename of *to-path*.

:create-directories
Determines whether directories should be created, if

needed, for the target of the copy. Permissible values are as follows:

t	Try to create the target directory of the copy and all superiors. Report directory creation to **standard-output**.
nil	Do not try to create directories. If the directory does not exist, handle this condition like any other error.
:query	If the directory does not exist, ask whether or not to create it. This is the default.

2.1.18 Changes to Host Determination in Pathnames

An important incompatible change has been made in the way the pathname system determines the host for a pathname being parsed. The first colon in a string to be parsed as a pathname now always delimits the host. You can "quote" embedded colons that are not intended to delimit the host by inserting a colon at the beginning of the string.

An upwardly compatible change has been made as well. From either a 3600 or an LM-2, you can now use the syntax "*host*|FEP*n*" to refer to a FEP file system on a remote 3600. *host* is the name of the host, and *n* is the disk unit number.

Following are the rules for host determination in a pathname.

Two important operations of the pathname system are *parsing* and *merging*. Parsing is the conversion of a string, which might have been typed by the user when asked to supply the name of a file, into a pathname object. This involves finding out for which host the pathname is intended, using the file name syntax conventions of that host to parse the string into the standard pathname components, and constructing such a pathname. Merging is the operation which takes a pathname with missing components and supplies values for those components from a set of defaults.

Since each kind of file system has its own character string representation of names of its files, there has to be a different parser for each of these representations, capable of examining such a character string and determining the value of each component. The parsers, therefore, all work differently. How does the parsing operation know which parser to use? It determines for which host the pathname is intended, and uses the appropriate parser. A filename character string may specify a host explicitly, by having the name of the host, followed by a colon, at the beginning of the string, or it may assume a default, if there is no host name followed by a colon at the beginning of the string.

Here is how the pathname system determines for which host a pathname being parsed is intended. The first colon in a pathname being parsed *always* delimits the

host name. You can also enter pathname strings that are for a specific host and do not contain any host name. In that case, a *default host* is used. Normally, the identity of the default host is displayed to the user entering a pathname. See the section "Defaults and Merging".

However, it is possible to have pathnames that have colons in them that do not designate hosts, such as filenames constructed from clock times, and the like. Some systems use the colon character to delimit devices. This creates a problem in parsing such pathnames. See the function **fs:parse-pathname**. The standard Lisp Machine user interface does not use such pathnames, but they may be used by particular programs, especially programs that deal with files whose format is defined by a foreign operating system.

The rule for parsing file names containing colons is, again, that any string used before a colon is *unconditionally* interpreted as a file computer. If the string cannot be interpreted as a host, an error is signalled.

If you must type one of these peculiar pathnames that have embedded colons *not* meaning hosts, you omit the host and place a colon at the beginning of the string. This "null host" tells the parser that it should *not* look further for a colon, but instead assume the host from the defaults. Examples:

- SS:<FOO>BAR refers to a host named "SS". :SS:<FOO>BAR refers to no explicit host; if parsed relative to a TOPS-20 default, "SS" probably refers to a device.
- 09:25:14.data refers to a host named "09". :09:25:14.data refers to no explicit host.
- AI: COMMON; GEE WHIZ refers to a host named "AI".
- AI: ARC: USERS1; FOO BAR refers to a host named "AI". "ARC" is the name of a device in the ITS operating system.
- EE:PS:<COMMON>GEE.WHIZ.5 specifies host EE (TOPS-20).
- PS:<COMMON>GEE.WHIZ.5 specifies a host named PS, which is almost certainly not what is intended! The user probably intended the "PS" device on some TOPS-20 host.
- :PS:<COMMON>GEE.WHIZ.5, assuming that the default host is some TOPS-20, specifies a device named "PS" on that host.

There are a handful of "pseudo-" host names, which are recognized as host names even though they are not actually the names of hosts. They are "local", "FEP", "FEP*n*", and "*host*|FEP*n*".

"local" This pseudo-host name always refers to the local file system

(LMFS) of the machine that you are using. It does not matter whether or not a local file system actually exists on that machine; an attempt will be made to reference it. "Local" is always equivalent to the name of the local host.

"FEP" (3600 only) This pseudo-host name always refers to a FEP (Front-End Processor) file system on the machine you are using, specifically, the one on the disk unit from which the system was booted.

"FEP*n*" (3600 only) This pseudo name always refers to a FEP file system on the machine you are using. The single digit *n* specifies the disk unit number; there is a separate FEP file system on each drive. This can access the boot unit, or any other disk unit, when multiple units are present.

"*host*|FEP*n*" *host* must be a valid host name. This pseudo-host name refers to a FEP file system on a remote 3600. This may be used from LM-2's, as well as 3600's, to reference FEP file systems of remote 3600's. The syntax "*host*|FEP" is *not* acceptable: you may not access the "boot unit" of a remote 3600 in this fashion. You must know the disk unit number. The disk unit number of a host having only one disk unit is **0**.

If the string to be parsed does not specify a host explicitly, the parser assumes that some particular host is the one in question, and it uses the parser for that host's file system. The optional arguments passed to the parsing function (**fs:parse-pathname**) tell it which host to assume.

2.1.19 Meaning of argument changed for fs:parse-pathname

The meaning of the second, *with-respect-to* argument to **fs:parse-pathname** has changed. *with-respect-to* specifies a host against which to parse the first argument, *thing*. Formerly, if *thing* contained a host name and *with-respect-to* was not **nil**, an error was signalled when the two hosts were not the same. When they were the same, the host name was removed from *thing* before it was parsed. Now this is true only if *thing* is a Maclisp-style list.

Now, if *thing* is a string and *with-respect-to* is not **nil**, *thing* is taken as a true string for the host specified by *with-respect-to*. Host names are not removed from *thing* before it is parsed, and *thing* is parsed against the host specified by *with-respect-to*. Thus, when *with-respect-to* is not **nil**, *thing* should not contain host names.

This change was made necessary by the change in host naming conventions. See the section "Changes to Host Determination in Pathnames".

fs:parse-pathname *thing* &optional *with-respect-to (defaults* **fs:*default-pathname-defaults***)* *Function*

This turns *thing*, which can be a pathname, a string, a symbol, or a Maclisp-style name list, into a pathname. Most functions that are advertised to take a pathname argument call **fs:parse-pathname** on it so that they will accept anything that can be turned into a pathname (most, however, do it indirectly, by calling **fs:merge-pathnames**).

This function does *not* do defaulting, even though it has an argument named *defaults*; it only does parsing. The *with-respect-to* and *defaults* arguments are there because in order to parse a string into a pathname, it is necessary to know what host it is for so that it can be parsed with the file name syntax peculiar to that host.

If *with-respect-to* is supplied, it should be a host or a string to be parsed as the name of a host. If *thing* is a string or symbol, it is then parsed as a true string for that host; host names specified as part of *thing* are not removed. Thus, when *with-respect-to* is not **nil**, *thing* should not contain a host name.

If *with-respect-to* is not supplied or is **nil**, any host name inside *thing* is parsed and used as the host. If *with-respect-to* is **nil** and no host is specified as part of *thing*, the host is taken from *defaults*.

Examples, using a LMFS host named Q:

```
(fs:parse-pathname "a:>b.c" "q") => #<LMFS-PATHNAME "Q:a:>b.c">  ;(wrong)
(fs:parse-pathname "q:>b.c" "q") => #<LMFS-PATHNAME "Q:q:>b.c">  ;(wrong)
(fs:parse-pathname "q:>b.c")     => #<LMFS-PATHNAME "Q:>b.c">
(fs:parse-pathname ">b.c" "q")   => #<LMFS-PATHNAME "Q:>b.c">
```

Note that this causes correct parsing of a TOPS-20 pathname when *thing* contains a device but no host and when *with-respect-to* is not **nil**. (Warning: If *thing* contains a device but no host and if *with-respect-to* is **nil** or not supplied, the device is interpreted as a host.) In the following example, X is a TOPS-20 host and A is a device:

```
(fs:parse-pathname "a:<b>c.d" "x") => #<TOPS20-PATHNAME "X:A:<B>C.D">
(fs:parse-pathname "a:<b>c.d")     => Error: "a" is not a known file
                                             server host.
```

In the same TOPS-20 example, if *with-respect-to* is **nil** and the host is to taken from *defaults*, the pathname string must be preceded by a colon to be parsed correctly:

```
(fs:parse-pathname ":a:<b>c.d" nil "x:") => #<TOPS20-PATHNAME "X:A:<B>C.D">
(fs:parse-pathname "a:<b>c.d" nil "x:")  => Error: "a" is not a known file
                                                   server host.
```

If *thing* is a list, *with-respect-to* is specified, and *thing* contains a host name, an error is signalled if the hosts from *with-respect-to* and *thing* are not the same.

2.1.20 Arguments changed for fs:user-homedir and fs:init-file-pathname

The second, optional argument to **fs:user-homedir** has been removed. **fs:init-file-pathname** now has two optional arguments: the canonical type of the init file, and the host.

fs:user-homedir &optional *(host* **fs:user-login-machine***)* *Function*

Returns the pathname of the logged-in user's home directory on *host*, which defaults to the host the user logged in to. For a registered user (one who logged in without using the **:host** argument to **login**), the host is the user's **home-host** attribute. Home directory is a somewhat system-dependent concept, but from the point of view of the Lisp Machine it is the directory where the user keeps personal files such as init files and mail. This function returns a pathname without any name, type, or version component (those components are all **nil**).

fs:init-file-pathname *program-name* &optional *(canonical-type* **nil***)* *Function*
(host **fs:user-login-machine***)*

Returns the pathname of the logged-in user's init file for the program *program-name*, on the *host*, which defaults to the host the user logged in to. Programs that load init files containing user customizations call this function to find where to look for the file, so that they need not know the separate init file name conventions of each host operating system. The *program-name* "LISPM" is used by the **login** function. *canonical-type* is the canonical type of the init file. It should be **nil** when the returned pathname is being passed to **load** so that **load** can look for a file of the appropriate type.

2.1.21 Init File Pathnames Standardized

The names of init files have been standardized, and init files are now of canonical type **:lisp** for source files and **:bin** and **:qbin** for compiled files. You must change the names of your init files for Release 5.0.

Init files are of canonical type **:lisp** for source files and **:bin** or **:qbin** for compiled files. For hosts that support long file names, the init file name consists of *program-name* with "-INIT" appended. Thus, the standard file name for a lispm init file is LISPM-INIT; for a Zmail init file, it is ZMAIL-INIT. Hosts that do not support long file names have conventions peculiar to each system.

Following are the names of lispm init source files on some hosts:

Host system	*File name*
LMFS/TOPS-20	LISPM-INIT.LISP
UNIX	lispm-init.l
VMS	LISPMINI.LSP

Multics	lispm-init.lisp
ITS	If user has own directory: LISPM >. If user does not have own directory: *USER* LISPM.

2.1.22 :init canonical pathname type removed

The **:init** canonical type has been removed. Init files are now of canonical type **:lisp** for source files and **:bin** and **:qbin** for compiled files.

2.1.23 Changes to Logical Pathnames

2.1.23.1 Logical Pathname Name, Type, and Version Now Separated by Periods

The name, type, and version of logical pathnames are now separated by periods. Spaces are accepted on input for compatibility.

2.1.23.2 New Default Representations for Newest and Oldest Logical Pathname Versions

The default representation in a logical pathname for the **:newest** version is the string "NEWEST". The default representation for the **:oldest** version is the string "OLDEST". On input, ">" is accepted for **:newest** and "<" for **:oldest** for compatibility.

2.1.23.3 Logical Pathnames Now Hierarchical

Logical pathnames can now have hierarchical directories. Each directory level is separated by a semicolon.

2.1.23.4 Changes to Logical Pathname Translations

The procedure for translating logical to physical pathnames has changed to conform to the rules for wildcard pathname matching. Logical directory names in translation lists should now be terminated by semicolons (though **fs:set-logical-pathname-host** accepts logical directory names without semicolons).

This section explains the format of the "translations" list of logical pathnames and the rules for translating a logical pathname to a physical pathname.

Each element of the list (one translation) specifies two wildcard pathnames, the first on the logical host and the second on the physical. In the Lisp form (in the file sys:site;*host*.translations) that specifies this form, they are given as strings to be parsed against these respective hosts. As they are parsed, they are merged with a pathname of wild name, wild type, and wild version.

Following is an example of a translations list:

```
'(("L-BIN;" ">lmach>fasl>")
  ("L-COMPILER;" ">sys>l-compiler>")
  ("L-SYS;" ">lmach>")
  ("L-*;" ">lmach>*>")
  ("LMFS-PATCH;" ">sys>lmfs>patch>")
  ("*;" ">sys>*>")))
```

Note that logical directory names should be followed by semicolons (though **fs:set-logical-pathname-host** accepts names without semicolons).

The method of translating a logical pathname consists of matching it against each first element of each translation, in succession. The order in the list is thus very important. At the first match, the translated pathname is produced by sending the **:translate-wild-pathname** message to the logical pathname with the first element of the translation as the source pattern and the second element of the translation as the target pattern. See the section "Wildcard Pathname Mapping". See the section "Wildcard Directory Mapping".

Note that it is possible to have a translation that matches "everything else", as in the example below. In the presence of such a translation, it is impossible to have an undefined translation.

Back-translation is performed by searching the second elements of the translations list, and translating in the other direction.

A special version of wild pathname translation, called "reversible wild pathname translation", is used. The difference between regular wild pathname translation and reversible translation is in the treatment of a target wildcard pattern consisting solely of *. In regular translation, a target pattern of **:wild** causes the source component to be copied verbatim. This is a useful user-interface feature, but it causes dropping of information and resultant noninvertibility of the transformation. In reversible mapping, this feature is not present. Logical pathname translation and back-translation is done in this mode.

Example:

Type	*Source pattern*	*Source instance*	*Target pattern*	*Result*
Regular	foo*	foolish	*	foolish
Reversible	foo*	foolish	*	lish
Either	*	bar	foo-*	foo-bar

Note that the inverse translation of foo-bar to bar cannot be accomplished under regular translation.

2.1.23.5 Flavor fs:undefined-logical-pathname-translation replaces fs:undefined-logical-pathname-directory

fs:undefined-logical-pathname-translation is a new condition flavor signalled when a logical pathname is referenced but has no translation to a physical pathname. It replaces **fs:undefined-logical-pathname-directory**.

fs:undefined-logical-pathname-translation *Flavor*

A logical pathname was referenced but is not defined. The **:logical-pathname** message returns the logical pathname. This flavor has a **:define-directory** proceed type, which prompts for a physical pathname whose directory component is the translation of the logical directory on the given host.

2.1.24 fs:make-logical-pathname-host replaces fs:add-logical-pathname-host

The function **fs:add-logical-pathname-host** is obsolete, though currently supported for compatibility. You should now use **fs:make-logical-pathname-host** to define a logical pathname host. This function loads the file sys:site;*host-name*.translations, which should contain a single form: a call to **fs:set-logical-pathname-host**.

fs:make-logical-pathname-host *host-name* *Function*

Defines *host-name*, which should be a string or symbol, to be the name of a logical pathname host. *host-name* should not conflict with the name of any existing host, logical or physical.

This function loads the file sys:site;*host-name*.translations. This file should contain a single form: a call to **fs:set-logical-pathname-host**. The file is always loaded into the **file-system** package. See the function **fs:set-logical-pathname-host**.

fs:make-logical-pathname-host not only loads this file but also arranges for the same file to be reloaded in the future. **load-patches** checks the translations file for each logical host that is defined in the current world; if any file has been changed it is reloaded. **load-patches** does this if and only if no specific systems are specified in its arguments.

fs:make-logical-pathname-host alters the **logical-pathnames-translation-files** system so that it contains the translations files for all logical hosts defined in the current world. **load-patches** loads updated translations files by calling **make-system** on this system.

When a world load is taken to a new site, the translation file for each logical host that is defined in the current world is reloaded from the new site's sys:site; directory. This changes all logical pathnames to map into the new set of physical pathnames defined by the new site.

An **fs:make-logical-pathname-host** form often appears in the file

sys:site;*system-name*.system. **make-system** looks for this file when given the name of an unknown system. In addition to a call to **fs:make-logical-pathname-host**, this file should contain a call to **si:set-system-source-file**, which specifies the logical pathname of the file containing the **defsystem** form.

Example:

Following are the contents of the file sys:site;cube.system:

```
;;; -*-  Mode: LISP; Package: USER -*-

(fs:make-logical-pathname-host "cube")
(si:set-system-source-file "cube" "cube: cube; cubpkg")
```

2.1.25 Previously undocumented function: fs:set-logical-pathname-host

fs:set-logical-pathname-host creates a logical host and defines translations from logical directories on that host to physical directories on a physical host. A call to this function is the only form in the file sys:site;*logical-host*.translations, which is loaded by **fs:make-logical-pathname-host**.

fs:set-logical-pathname-host *logical-host* &key *physical-host translations no-translate* *Function*

fs:set-logical-pathname-host creates a logical host named *logical-host* if it does not already exist. It then establishes translations of logical directories on *logical-host* to physical directories on *physical-host*. *translations* is a "translations" list of two-element lists of strings representing associated logical directories (source patterns) and physical directories (target patterns). For the format of the lists and the translation rules: See the section "Logical Pathname Translation".

Logical directory names should be terminated by semicolons, but **fs:set-logical-pathname-host** accepts names without semicolons. Host names can appear in the strings in the translations list, but each logical host in a string must refer to the same host as *logical-host*, and each physical host in a string must refer to the same host as *physical-host*. If the physical pathname is on a TOPS-20 or VMS device, you must include the host name (either explicitly or implicitly, with an initial colon) so that the device is not taken to be the host.

If *no-translate* is **nil** or unsupplied, the translation of every interned logical pathname is checked. Properties are copied from the old physical pathname to the the new one, and logical pathnames that now have no corresponding physical pathnames are uninterned. If *no-translate* is not **nil** this mapping is suppressed, and some physical pathnames might not get the properties of the logical pathname. The consequences of this are unknown.

A call to **fs:set-logical-pathname-host** is usually the only form in the file

sys:site;*logical-host*.translations. This file is loaded by **fs:make-logical-pathname-host** (always in the **file-system** package), which also arranges for it to be reloaded in the future. **load-patches** checks this file for all logical hosts in the current world and reloads the file if it has changed.

Example:

Following is the contents of the file sys:site;cube.translations:

```
;;; -*- Mode: LISP; Package: FILE-SYSTEM -*-

(set-logical-pathname-host "cube"
  ':physical-host "pointer"
  ':translations '(("cube;" ">cube>")))
```

2.1.26 load-file-list obsolete

The function **load-file-list** is obsolete. Use **make-system** instead.

2.1.27 Change in arguments to print-herald

print-herald no longer accepts an optional argument of the stream for display; display now always goes to **standard-output**. Instead, it accepts two keyword arguments: **:as-if-band** is used by **disk-save**, and **:verbose** controls the display of system version numbers.

print-herald &key *as-if-band verbose* *Function*

Prints out the herald message to **standard-output**. The herald message is what the machine prints when it is cold booted. It shows you the name of the FEP file or partition for the current world load, any comment added to the herald, a measure of the physical memory and swapping space available, the versions of the systems that are running, the site name, and the machine's own host name.

:as-if-band is used by **disk-save** to supply the name of the FEP file or partition of the saved world. **:verbose** controls the system version information displayed: if **t**, the version numbers of all systems are displayed; if **nil**, the version numbers of only those systems that differ from the release are displayed.

2.1.28 Change in arguments to unadvise

The three subforms of **unadvise** are now independent. If *function* is **nil** but *class* or *position* is not, **unadvise** removes only the specified classes or positions of advice for all functions.

unadvise &optional *function class position* *Special Form*

Removes pieces of advice. None of its subforms are evaluated. *function* and *class* have the same meaning as they do in the function **advise**. *position* specifies which piece of advice to remove. It can be the numeric index (0 means the first one) or it can be the name of the piece of advice.

unadvise can remove more than one piece of advice if some of its arguments are missing or **nil**. The arguments *function*, *class*, and *position* all act independently. A missing value or **nil** means all possibilities for that aspect of advice. For example, the following form removes all **:before**, **:after**, and **:around** advice named **negative-arg-check** on the **factorial** function.

```
(unadvise factorial nil negative-arg-check)
```

In this example **unadvise** removes all **:around** advice on all functions in all positions with all names.

```
(unadvise nil :around)
```

In this example **unadvise** removes all classes of advice named **my-personal-advice** on all functions.

```
(unadvise nil nil my-personal-advice)
```

(unadvise) removes all advice on all functions, since *function*, *class*, and *position* take on all possible values.

2.1.29 Window System Changes Associated with Mouse Input

2.1.29.1 Flavors tv:any-tyi-mixin and tv:list-tyi-mixin obsolete

The flavors **tv:any-tyi-mixin** and **tv:list-tyi-mixin** are obsolete. The **:tyi** and **:tyi-no-hang** methods of **tv:stream-mixin** have been renamed to **:any-tyi** and **:any-tyi-no-hang**. The **:tyi**, **:tyi-no-hang**, **:list-tyi**, **:mouse-or-kbd-tyi**, and **:mouse-or-kbd-tyi-no-hang** methods of **tv:any-tyi-mixin** and **tv:list-tyi-mixin** have been moved to **tv:stream-mixin**. The **tv:any-tyi-mixin** and **tv:list-tyi-mixin** flavors still exist for compatibility, but they have no effect.

2.1.29.2 Changes to :tyi, :tyi-no-hang, :list-tyi, :mouse-or-kbd-tyi, and :mouse-or-kbd-tyi-no-hang methods of tv:stream-mixin

The **:tyi** method of **tv:stream-mixin** has been renamed to **:any-tyi**, and **:tyi-no-hang** has been renamed to **:any-tyi-no-hang**. The behavior of **:any-tyi** is somewhat more complex because of interactions with the input editor. If you want to receive input that might be integers (character codes) or blips (such as mouse clicks or activation blips), you should send these messages instead of **:tyi** and **:tyi-no-hang**. The **:tyi** and **:tyi-no-hang** methods now always discard blips; previously they did so only if **tv:any-tyi-mixin** was a component of the window flavor. You can send these messages if you want to receive only keyboard input. If you want to receive only blips, send the **:list-tyi** message.

The **:mouse-or-kbd-tyi** and **:mouse-or-kbd-tyi-no-hang** methods are of use to Zwei but probably not to any other programs.

:any-tyi &optional *eof-action* of **tv:stream-mixin** *Method*

Read and return the next character of input from the window, waiting if there is none. Where the character comes from depends on the value of the variable **rubout-handler**. Following is a summary of actions for each possible value of **rubout-handler**:

nil	If the input buffer contains unscanned input, take the next character from there. Otherwise, take the next character from the window's I/O buffer.
:read	If the input buffer contains unscanned input, take the next character from there. Otherwise, if an activation blip or character is present, return that. Otherwise, enter the input editor.
:tyi	Take the next character from the window's I/O buffer.

If *eof-action* is **nil**, an error is signalled when an end-of-file is encountered. Otherwise, the method returns **nil** when an end-of-file is encountered.

:any-tyi-no-hang &optional *eof-action* of **tv:stream-mixin** *Method*

Check the window's I/O buffer and return the next character if it is immediately available. If no characters are immediately available, return **nil**. It is an error to call this method from inside the input editor (that is, if the value of **rubout-handler** is not **nil**). *eof-action* is ignored. This is used by programs that continuously do something until a key is typed, then look at the key and decide what to do next.

:tyi &optional *eof-action* of **tv:stream-mixin** *Method*

If called from outside the input editor, this is the same as **:any-tyi**, except that only integers and **nil** can be returned. Blips are discarded, unless the first element of the blip is **:mouse-button** and the second element is **#\mouse-r-1**; in this case, the method pops up a system menu. If called from inside the input editor with **:full-rubout** specified and if an activation blip is read when the input buffer is empty, the method causes control to be returned from the input editor.

:tyi-no-hang &optional *eof-action* of **tv:stream-mixin** *Method*

This is like **:any-tyi-no-hang**, except that only integers and **nil** can be returned. Blips are discarded, unless the first element of the blip is **:mouse-button** and the second element is **#\mouse-r-1**; in this case, the method pops up a system menu.

:list-tyi of **tv:stream-mixin** *Method*

This is like **:any-tyi** except that it only returns blips and never returns integers. If it encounters any integers in the input stream, it discards them entirely (they are removed from the I/O buffer and the program never sees them).

:mouse-or-kbd-tyi of **tv:stream-mixin** *Method*

This is like **:any-tyi**, except that it returns two values, and it discards all blips but those whose first element is the symbol **:mouse**. In this case it returns the third element of the blip and the blip itself. Otherwise, if it sees an integer or **nil**, it returns that as both returned values. Blips whose first element is **:mouse** are produced by the user's clicking on the mouse while inside the editor. This method is used only by Zwei.

:mouse-or-kbd-tyi-no-hang of **tv:stream-mixin** *Method*

This is like **:any-tyi-no-hang**, except that it returns two values, and it discards all blips but those whose first element is the symbol **:mouse**. In this case it returns the third element of the blip and the blip itself. Otherwise, if it sees an integer or **nil**, it returns that as both returned values. Blips whose first element is **:mouse** are produced by the user's clicking on the mouse while inside the editor. This method is used only by Zwei.

2.1.29.3 Flavors tv:list-mouse-buttons-mixin and tv:kbd-mouse-buttons-mixin obsolete

The flavors **tv:list-mouse-buttons-mixin** and **tv:kbd-mouse-buttons-mixin** are obsolete. The facilities they provided for interpreting mouse clicks have been incorporated into the **:mouse-click** method of **tv:essential-mouse**. The flavors still exist for compatibility, but they have no effect.

2.1.29.4 Changes to :mouse-click method of tv:essential-mouse

The **:mouse-click** method of **tv:essential-mouse** now always sends blips to any window with an I/O buffer. The blips are of the form previously provided by **tv:list-mouse-buttons-mixin**. When the click is **#\mouse-r-1**, this method no longer pops up a system menu.

This change allows programs to receive blips from any windows, including Lisp Listeners, without having to define special flavors of window.

:mouse-click *buttons x y* of **tv:essential-mouse** *Method*

This method is called by the **:mouse-buttons** method of **tv:essential-mouse**, which is called by the default mouse handler when mouse buttons are pushed. *buttons* is an encoded integer representing the buttons pushed; use reader macros like **#\mouse-r-1** to handle these integers in your program. *x* and *y* represent the position of the mouse at the time of the click, in the window's outside coordinates.

If the click is **#\mouse-r-2**, the **:mouse-buttons** method pops up a system menu. Otherwise, if the window has an I/O buffer, **:mouse-click** sends it a blip of the form (**:mouse-button** *buttons window x y*). In addition, if the click is **#\mouse-l-1**, the window is selected.

:mouse-click methods are combined using **:or** combination, so the

:mouse-click method of **tv:essential-mouse** runs only if no earlier method handles the message (and all earlier methods return **nil**).

The following example illustrates the use of the **:any-tyi** message to receive both mouse and keyboard input to windows. It is a simple drawing program whose command loop accepts single keystroke or mouse click commands. This program does not require any special flavor of window in order to run. It runs using any window that can become the value of **terminal-io**.

```
(defun draw-help ()
  (send terminal-io ':clear-window)
  (format t "Click the left mouse button to draw a square.~@
             Click the middle mouse button to draw a circle.~@
             Click the right mouse button to draw a triangle.~@
             Type REFRESH to clear the screen.~@
             Type END to exit.~@
             Type HELP for documentation.~%"))

(defun draw ()
  (draw-help)
  (loop for command = (send terminal-io ':any-tyi)
        do (cond ((fixp command)
                  (selectq command
                    (#\refresh (send terminal-io ':clear-window))
                    (#\end (return))
                    (#\help (draw-help))
                    (t (beep))))
                 ((eq (car command) ':mouse-button)
                  (destructuring-bind (click nil x y) (cdr command)
                    (selectq click
                      (#\mouse-l-1 (send terminal-io ':draw-rectangle 20 20 x y))
                      (#\mouse-m-1 (send terminal-io ':draw-circle x y 10))
                      (#\mouse-r-1 (send terminal-io ':draw-triangle
                                         x y (- x 10) (+ y 20) (+ x 10) (+ y 20)))
                      (t (beep)))))
                 (t (beep)))))
```

2.1.29.5 Flavor tv:preemptable-read-any-tyi-mixin obsolete

The flavor **tv:preemptable-read-any-tyi-mixin** is obsolete. It has been replaced by the **:preemptable** input editor option. The flavor still exists for compatibility, but it has no effect. The **:preemptable-read** message is supported by **tv:stream-mixin** for compatibility.

:preemptable *token* *Option*

A blip in the input stream causes control to be returned from the input editor immediately. Two values are returned: the blip and *token*, which is usually a keyword symbol. Any unscanned input typed before the blip remains in the input buffer, available to the next read operation from the stream.

2.1.30 :clear-screen, :clear-eol, and :clear-eof messages to windows renamed

The following messages to windows have been renamed:

- **:clear-screen** is now **:clear-window**
- **:clear-eol** is now **:clear-rest-of-line**
- **:clear-eof** is now **:clear-rest-of-window**

In Release 5.0, windows continue to accept the old messages for compatibility. The cold-load stream does not accept the new messages.

In a future release, **:clear-eof** will become a no-op so that the meaning of this message will be compatible with that for noninteractive streams. The cold-load stream will accept the new messages.

The **:clear-eof** message was renamed because it had two different meanings. For windows, it meant to clear the window from the cursor position to the bottom. For noninteractive streams, it meant to read the EOF indicator, so that data past the EOF could be read. The other two messages were renamed to be consistent with modern naming conventions.

:clear-window of **tv:sheet** *Method*

Erase the whole window and move the cursor position to the upper left corner of the window.

:clear-rest-of-line of **tv:sheet** *Method*

Erase from the current cursor position to the end of the current line; that is, erase a rectangle horizontally from the cursor position to the inside right edge of the window, and vertically from the cursor position to one line-height below the cursor position.

:clear-rest-of-window of **tv:sheet** *Method*

Erase from the current cursor position to the bottom of the window. In more detail, first do a **:clear-rest-of-line**, and then clear all of the window past the current line.

2.2 New Features in Lisp in Release 5.0

2.2.1 New function: eql

eql *x y* *Function*

eql returns **t** if is arguments are **eq**, or if they are numbers of the same type with the same value, or (in Common Lisp) if they are character objects that represent the same character. The predicate = compares the values of

two numbers even if the numbers are of different types. Use **equal** or **string-equal** to compare the characters of two strings.

Examples:

```
(eql 'a 'a) => t
(eql 3 3)   => t
(eql 3 3.0) => nil
(eql 3.0 3.0) => t
(eql #/a #/a) => t
(eql (cons 'a 'b) (cons 'a 'b)) => nil
(eql "foo" "FOO") => nil
```

The following expressions might return either **t** or **nil**:

```
(eql '(a . b) '(a . b))
(eql "foo" "foo")
```

In Zetalisp:

```
(eql 1.0s0 1.0d0) => nil
(eql 0.0 -0.0) => nil
```

2.2.2 New special form: defconstant

The special form **defconstant** is used to declare a named constant.

defconstant *variable initial-value [documentation]* *Special Form*

defconstant declares the use of a named constant in a program. *initial-value* is evaluated and *variable* set to the result. The value of *variable* is then fixed. It is an error if *variable* has any special bindings at the time the **defconstant** form is executed. Once a special variable has been declared constant by **defconstant**, any further assignment to or binding of that variable is an error.

The compiler is free to build assumptions about the value of the variable into programs being compiled. If the compiler does replace references to the name of the constant by the value of the constant in code to be compiled, the compiler takes care that such "copies" appear to be **eql** to the object that is the actual value of the constant. For example, the compiler may freely make copies of numbers, but it exercises care when the value is a list.

In Zetalisp, **defconstant** and **defconst** are essentially the same if the value is other than a number, a character, or an interned symbol. However, if the variable being declared already has a value, **defconst** freely changes the value, whereas **defconstant** queries before changing the value (unless the **defconstant** form is in a patch file). **defconstant** assumes that changing the value is dangerous because the old value might have been incorporated into compiled code, which would be out of date if the value changed.

In general, you should use **defconstant** to declare constants whose value is a

number, character, or interned symbol and is guaranteed not to change. An example is π. The compiler can optimize expressions that contain references to these constants. If the value is another type of Lisp object or if it might change, you should use **defconst** instead.

documentation, if provided, should be a string. It is accessible to the **documentation** function.

2.2.3 New special forms: block and tagbody

block is a primitive special form used with **return-from** for premature exit from a piece of code. **tagbody** is a primitive special form used with **go** for unstructured transfer of control. **prog**, **do**, and their variants are effectively constructed out of **let**, **block**, and **tagbody** forms.

block *name form...* *Special Form*

block evaluates each *form* in sequence and normally returns the (possibly multiple) values of the last *form*. However, **(return-from** *name value***)** or one of its variants (a **return** or **return-list** form) might be evaluated during the evaluation of some *form*. In that case, the (possibly multiple) values that result from evaluating *value* are immediately returned from the innermost block that has the same name and that lexically contains the **return-from** form. Any remaining forms in that block are not evaluated.

name is not evaluated. It must be a symbol.

The scope of *name* is lexical. That is, the **return-from** form must be inside the block itself (or inside a block that that block lexically contains), not inside a function called from the block.

do, **prog**, and their variants establish implicit blocks around their bodies; you can use **return-from** to exit from them. These blocks are named **nil** unless you specify a name explicitly.

For example, the following two forms are equivalent:

```
(cond ((predicate x)
       (do-one-thing))
      (t
       (format t "The value of X is ~S~%" x)
       (do-the-other-thing)
       (do-something-else-too)))

(block deal-with-x
  (when (predicate x)
    (return-from deal-with-x (do-one-thing)))
  (format t "The value of X is ~S~%" x)
  (do-the-other-thing)
  (do-something-else-too))
```

tagbody *tag-or-statement...* *Special Form*

The body of a **tagbody** form is a series of *tags* or *statements*. A *tag* is a symbol; a *statement* is a list. **tagbody** processes each element of the body in sequence. It ignores *tags* and evaluates *statements*, discarding the results. If it reaches the end of the body, it returns **nil**.

If a **(go** *tag***)** form is evaluated during evaluation of a *statement*, **tagbody** searches its body and the bodies of any **tagbody** forms that lexically contain it. Control is transferred to the innermost *tag* that is **eq** to the *tag* in the **go** form. Processing continues with the next *tag* or *statement* that follows the *tag* to which control is transferred.

The scope of the *tags* is lexical. That is, the **go** form must be inside the **tagbody** construct itself (or inside a **tagbody** form that that **tagbody** lexically contains), not inside a function called from the **tagbody**.

do, **prog**, and their variants use implicit **tagbody** constructs. You can provide *tags* within their bodies and use **go** forms to transfer control to the *tags*.

For example, the following two forms are equivalent:

```
(dotimes (i n) (print i))

(let ((i 0))
  (when (plusp n)
    (tagbody
      loop
      (print i)
      (setq i (1+ i))
      (when (< i n) (go loop)))))
```

2.2.4 New special forms: multiple-value-call and multiple-value-prog1

multiple-value-call and **multiple-value-prog1** are new special forms for returning multiple values. **multiple-value-call** evaluates forms and uses the (possibly multiple) values as arguments to a function. **multiple-value-prog1** is like **prog1** except that it can return multiple values from its first form.

multiple-value-call *function body ...* *Special Form*

multiple-value-call first evaluates *function* to obtain a function. It then evaluates all the forms in *body*, gathering together all the values of the forms (not just one value from each). It gives these values as arguments to the function and returns whatever the function returns.

For example, suppose the function **frob** returns the first two elements of a list of numbers:

```
(multiple-value-call #'+ (frob '(1 2 3)) (frob '(4 5 6)))
  <=> (+ 1 2 4 5) => 12.
```

(Defun frob (some-list)
  (values (car some-list) (cadr some-list)))

multiple-value-prog1 *first-form body...* *Special Form*

multiple-value-prog1 is like **prog1**, except that if its first form returns multiple values, **multiple-value-prog1** returns those values.

2.2.5 3600 Supports Ieee Single- and Double-precision Floating Point

The 3600 supports IEEE-standard single-precision and double-precision floating-point numbers. Single-precision floating-point numbers have a precision of 24 bits, or about 7 decimal digits. Their range is from 1.1754944e-38 to 3.4028235e38. Double-precision floating-point numbers have a precision of 53 bits, or about 16 decimal digits. Their range is from 2.2250738585072014d-308 to 1.7976931348623157d308.

Number objects exist that are outside the upper and lower limits of the ranges for single and double precision. Larger than the largest number is +1e= (or +1d= for doubles). Smaller than the smallest number is -1e= (or -1d= for doubles). Smaller than the smallest normalized positive number but larger than zero are the "denormalized" numbers. Some floating-point objects are Not-a-Number (NaN); they are the result of (// 0.0 0.0) (with trapping disabled) and like operations.

IEEE numbers are symmetric about zero, so the negative of every representable number is also a representable number (on the 3600 only). Zeros are signed in IEEE format, but +0.0 and -0.0 act the same arithmetically. For example:

```
(= +0.0 -0.0) => t
(plusp 0.0)   => nil
(plusp -0.0)  => nil
(zerop -0.0)  => t
(eq 0.0 -0.0) => nil
```

See the IEEE standard: Microprocessor Standards Committee, IEEE Computer Society, "A Proposed Standard for Binary Floating-Point Arithmetic: Draft 8.0 of IEEE Task P754," *Computer*, March 1981, pp. 51-62.

Some related functions have been added or extended. The mathematical functions, such as **sin** and **log**, have been modified to accept both single- and double-precision arguments.

2.2.5.1 float returns a single-precision number

float always coerces its argument to a single-precision floating-point number, even if the argument is a double-precision number.

float *x* *Function*

Converts any kind of number to a flonum on the LM-2 and to a single-precision floating-point number on the 3600. Note that, on the 3600, **float** reduces a double-precision argument to single precision.

2.2.5.2 New function: dfloat

dfloat converts its argument to a double-precision floating-point number.

dfloat *x* *Function*

(3600 only) Converts any kind of number to a double-precision floating-point number.

2.2.5.3 New data types: :single-float and :double-float

On the 3600, an object of type **:single-float** is a single-precision floating-point number. An object of type **:double-float** is a double-precision floating-point number. The **:float** data type is the union of these two types.

See the function **typep**.

2.2.5.4 floatp returns t for any floating-point number

floatp returns **t** if its argument is either a single-precision or a double-precision floating-point number.

floatp *arg* *Function*

floatp returns **t** if its argument is a floating-point number, that is, a flonum or a small flonum on the LM-2 or a single- or double-precision floating-point number on the 3600. Otherwise it returns **nil**.

2.2.5.5 New functions: sys:single-float-p, sys:double-float-p

sys:single-float-p and **sys:double-float-p** are predicates to distinguish between single- and double-precision floating-point numbers.

sys:single-float-p *arg* *Function*

(3600 only) Returns **t** if *arg* is a single-precision floating-point number, otherwise **nil**.

sys:double-float-p *arg* *Function*

(3600 only) Returns **t** if *arg* is a double-precision floating-point number, otherwise **nil**.

2.2.6 New function: mod

mod *x y* *Function*

The same as **remainder**, except that the returned value has the sign of the *second* argument instead of the first. When there is no remainder, the returned value is **0**.

Examples:

```
(mod -3 2)  => 1
(mod 3 -2)  => -1
(mod -3 -2) => -1
(mod 4 -2)  => 0
```

2.2.7 New functions: byte, byte-size, byte-position

byte creates a byte specifier; **byte-size** extracts the size field of a byte specifier; **byte-position** extracts the position field of a byte specifier.

byte *size position* *Function*

Creates a byte specifier for a byte *size* bits wide, *position* bits from the right-hand (least-significant) end of the word.

Example:

```
(ldb (byte 3 4) #o12345) => 6
```

byte-size *byte-specifier* *Function*

Extracts the size field of *byte-specifier*. You can use **setf** on this form:

```
(setq a (byte 3 4))
(setf (byte-size a) 2)
(byte-size a) => 2
```

byte-position *byte-specifier* *Function*

Extracts the position field of *byte-specifier*. You can use **setf** on this form:

```
(setq a (byte 3 4))
(setf (byte-position a) 2)
(byte-position a) => 2
```

2.2.8 New Metering Tools for the 3600

A new set of program counter (PC) metering tools is available on the 3600.

Program counter (PC) metering is a tool to allow the user to determine where time is being spent in a given program.

PC metering essentially produces a histogram. At regular intervals, the Front End Processor (FEP) causes the main processor to task switch to special microcode. This microcode looks up the macro PC that contains the virtual address of the macroinstruction that the processor is currently executing. If this virtual address falls outside the *monitored range*, the microcode increments a count of the number of PCs that missed the monitored range. If the address is within the monitored range, the microcode subtracts the bottom of the monitored range from the PC, leaving a word offset. It then divides the word offset by the number of words per *bucket* and uses that as an index into the *monitor array*. Next, it increments that indexed element of the monitor array. This can only measure statistically where the macro PC is pointing; for the results to be valid, a relatively large number of samples per bucket must be available. FEP version 13 samples at about 170 samples per second, so the PC monitoring with that version is probably valid only for sessions that take longer than five to ten seconds.

You specify some range of the program to be monitored. The range is specified by lower and upper bounding addresses, and compiled functions that lie between those

addresses are monitored. The range is divided into some number of buckets. The relative amount of time that the program spends executing in each bucket is measured.

The parameters you specify are the range of addresses to be monitored, the number of buckets, and an array with one word for each bucket.

Some of the metering functions deal with *compiled functions*. In this context a compiled function is either a compiled code object or an **art-16b** array, into which escape functions (small, internal operations used by the microcode) compile.

meter:make-pc-array *size* *Function*
Makes a PC array with *size* number of buckets. This storage is wired, so you probably do not want this to be more than about 64. pages, or **(* 64. sys:page-size)** words.

meter:monitor-all-functions *Function*
Changes the microcode parameters so that the monitor array refers to every possible function in the Lisp world at the time of the execution of **meter:monitor-all-functions**. This usually causes many functions to map into a single bucket, and is therefore useful in obtaining a first estimate of which functions are using a significant portion of the execution time.

meter:expand-range *start-bucket* &optional (*end-bucket start-bucket*) *Function*
Changes the microcode parameters so that the entire monitor array refers only to the functions previously contained within the range specified by *start-bucket* and *end-bucket*. *start-bucket* and *end-bucket* are inclusive bounds.

meter:report &optional *function-list* *Function*
Prints a summary of the data collected into the monitor array. You should not have to supply the *function-list* argument.

meter:start-monitor &optional (*clear* **t**) *Function*
Enables collection of PC data. If *clear* is not **nil**, the contents of the monitor array are cleared. If *clear* is **nil**, the array is not modified, so that the new samples are simply added to the old.

meter:stop-monitor *Function*
Disables further collection of PC data.

meter:print-functions-in-bucket *bucket* *Function*
Prints all the compiled functions that map into the specified *bucket*.

meter:list-functions-in-bucket *bucket* *Function*
Returns a list of all the compiled functions that map into the specified *bucket*.

meter:range-of-bucket *bucket* *Function*
Returns the virtual address range that maps into the specified *bucket*.

meter:with-monitoring *clear body...* *Macro*
Enables monitoring around the execution of *body*. If *clear* is not **nil**, clears the monitor array first. See the function **meter:start-monitor**.

meter:map-over-functions-in-bucket *bucket function* &rest *args* *Function*
Calls *function* for every compiled function in the specified *bucket*. The first argument to *function* should be the compiled function, and any remaining arguments are *args*.

meter:function-range *function* *Function*
Returns two values, the buckets that contain the first and last instructions of *function*.

meter:function-name-with-escapes *object* *Function*
If *object* is a compiled function, returns the function spec of the compiled function. Otherwise, returns **nil**.

2.2.9 New Meters for the LM-2

Five new microcode meters have been added for the LM-2.

sys:%tv-clock-counter *Meter*
Counts down every 60th of a second. When it reaches zero it resets from the **sys:%tv-clock-rate** meter and causes a sequence break if enabled.

sys:%count-disk-page-read-operations-in-transporter *Meter*
The number of page faults that went to the disk in the transporter (part of the garbage collector).

sys:%count-disk-page-read-operations-in-scavenger *Meter*
The number of page faults that went to the disk in the scavenger (part of the garbage collector).

sys:%transporter-run-time *Meter*
The number of microseconds spent in the transporter (part of the garbage collector).

sys:%scavenger-run-time *Meter*
The number of microseconds spent in the scavenger (part of the garbage collector).

2.2.10 New special form: define-symbol-macro

A symbol macro translates a symbol into a substitute form. When the Lisp evaluator is given a symbol, it checks whether the symbol has been defined as a symbol macro. If so, it evaluates the symbol's replacement form instead of the symbol itself. Use **define-symbol-macro** to define a symbol macro.

define-symbol-macro *name form* *Special Form*

This special form defines a symbol macro. *name* is a symbol to be defined as a symbol macro. *form* is a Lisp form to be substituted for the symbol when the symbol is evaluated. A symbol macro is more like a subst than a macro: *form* is the form to be substituted for the symbol, not a form whose evaluation results in the substitute form.

A symbol defined as a symbol macro cannot be used in the context of a variable. You cannot use **setq** on it, and you cannot bind it. You can use **setf** on it: **setf** substitutes the replacement form, which should access something, and expands into the appropriate update function. Example:

```
(define-symbol-macro foo (+ 3 bar))
(setq bar 2)
foo => 5
```

Here is a more complex example. Suppose you want to define some new instance variables and methods for a flavor. You want to test the methods using existing instances of the flavor. For testing purposes, you might use hash tables to simulate the instance variables, using one hash table per instance variable with the instance as the key. You could then implement an instance variable **x** as a symbol macro:

```
(defvar x-hash-table (make-hash-table))
(define-symbol-macro x (send x-hash-table ':get-hash self))
```

To simulate setting a new value for **x**, you could use **(setf x** *value*), which would expand into **(send x-hash-table ':put-hash self** *value*).

2.2.11 New function: undefflavor

undefflavor reverses the effect of a **defflavor**.

undefflavor *flavor-name* *Function*

Removes the flavor named by *flavor-name*.

2.2.12 New option for defflavor: :required-init-keywords

:required-init-keywords declares that some init keywords must be specified when making an instance of a flavor.

:required-init-keywords *Option*

The arguments are keywords. It is an error to try to make an instance of this flavor or any incorporating it without specifying these keywords as arguments to **make-instance** (or **instantiate-flavor**) or a **:default-init-plist** option in a component flavor. This error can often be detected at compile time.

2.2.13 New option for defflavor: :mixture

:mixture lets you define a family of flavors and use init options to select the member of the family to instantiate.

:mixture *Option*

Defines a family of related flavors. When **make-instance** (or **instantiate-flavor**) is called, it uses keywords in the init-plist to decide which flavor of the family to instantiate. Thus, init options can be used to select the flavor as well as instance-variable values.

The *ancestral* flavor is the one that includes the **:mixture** option in its **defflavor**. The flavors in the family are automatically constructed by mixing various mixins with the ancestral flavor. The names for the family members are chosen automatically. The name of such an automatically constructed flavor is a concatenation of the names of its components, separated by hyphens; however, obvious redundancies are removed heuristically.

defflavor of the ancestral flavor also defines the automatically constructed flavors. **compile-flavor-methods** of the ancestral flavor also compiles combined methods of the automatically constructed flavors.

The **:mixture** option has the following form:

(:mixture *spec spec ...***)**

Each *spec* is processed independently, and all the resulting mixins are mixed together. A *spec* may be any of the following:

(keyword mixin)
Add mixin if the value of *keyword* is **t**; add nothing if **nil**.

(keyword (value mixin) (value mixin) ...)
Look up the value of *keyword* in this alist and add the specified mixin.

(keyword mixin subspec subspec ...)

(keyword (value mixin subspec subspec ...) ...)
Subspecs take on the same forms as specs. Subspecs are processed only when the specified keyword has the specified value. Use them when there are interdependencies among keywords.

A *mixin* is one of the following:

symbol	The name of a flavor to be mixed in
nil	No flavor needs to be mixed in if the keyword takes on this value
string	This value is illegal: Signal an error with the string as the message

make-instance and **instantiate-flavor** check that the keywords are given with legal values.

Example:

```
(defflavor cereal-stream (...) (stream)
  ...
  (:init-keywords :characters :direction
                  :ascii :hang-up-when-close)
  (:mixture (:characters
             (t nil (:direction
                      (:in buffered-line-input-stream)
                      (:out buffered-output-character-stream))
                    (:ascii ascii-translating-character-stream))
             (nil nil (:direction (:in buffered-input-stream)
                                  (:out buffered-output-stream))
                      (:ascii "Ascii translation is not
                               meaningful for binary streams")))
            (:hang-up-when-close hang-up-when-close-mixin)))
```

Note the need for an **:init-keywords** declaration for any keywords that are used only in the **:mixture** declaration.

In this declaration, any kind of stream may have a **:hang-up-when-close** option. The **:characters** option does not itself add any mixins (hence the **nil**), but the processing of the **:direction** option depends on whether it is used with a character stream or a binary stream. The **:ascii** option is allowed only for character streams, and we specify an error message if it is used with a binary stream. If **:ascii** had not been mentioned in the **:characters nil** case, the keyword would have been ignored by **make-instance** on the assumption that an **:init** method was going to do something with it.

2.2.14 New format directives: ~→ and ~←

~→*text*~← is useful for indenting *text*.

~→*text*~← indents *text* at the cursor position that is current at the time of the ~→. A ~→ must be terminated with a ~←. ~→ and ~← can be nested like ~[~] and ~<~>. This directive is especially useful in making error messages indent properly.

Example:

```
(format t "~&Error:  ~→~A~←" "File not found
for FOO.LISP.1")
```

prints

```
Error:  File not found
        for FOO.LISP.1
```

~← terminates a ~→. It is undefined elsewhere.

2.2.15 New special form: format:defformat

format:defformat defines a new **format** directive. The function associated with the directive should send its output to the value of **format:*format-output***. Directives that were written in the old style and that send their output to **standard-output** still work, but you should begin to convert them to the new form.

format:defformat *directive (arg-type) arglist body ...* *Special Form*

Defines a new **format** directive.

directive is a symbol that names the directive. If *directive* is longer than one character, it must be enclosed in backslashes in calls to **format**:

```
(format t "~\foo\" ...)
```

directive is usually in the **format** package; if it is in another package, the user must specify the package in calls to **format**:

```
(format t "~\foo:bar\" ...)
```

format:defformat defines a function to be called when **format** is called using *directive*. *body* is the body of the function definition. *arg-type* is a keyword that determines the arguments to be passed to the function as *arglist*:

:no-arg The directive uses no arguments. The function is passed one argument, a list of parameters to the directive. The value returned by the function is ignored.

:one-arg The directive uses one argument. The function is passed two arguments: the argument associated with the directive and a list of parameters to the directive. The value returned by the function is ignored.

:multi-arg The directive uses a variable number of arguments. The function is passed two arguments. The first is a list of the first argument associated with the directive and all the remaining arguments to **format**. The second is a list of parameters to the directive. The function should **cdr** down the list of arguments, using as many as it wants, and return the tail of the list so that the remaining arguments can be given to other directives.

The function can examine the values of **format:colon-flag** and **format:atsign-flag**. If **format:colon-flag** is not **nil**, the directive was given a **:** modifier. If **format:atsign-flag** is not **nil**, the directive was given a **@** modifier.

The function should send its output to the stream that is the value of **format:*format-output***.

Here is an example of a **format** directive that takes one argument and prints a number in base 7:

```
(format:defformat format:base-7 (:one-arg) (argument parameters)
  parameters                                        ;ignored
  (let ((base 7))
    (princ argument format:*format-output*)))
```

Now:

```
(format nil "> ~\base-7\ <" 8)  =>  "> 11 <"
```

2.2.16 New Features Associated with the Input Editor (Rubout Handler)

This section describes new features in the input editor and reading functions. For incompatible changes: See the section "Incompatible Changes to the Input Editor (Rubout Handler)".

2.2.16.1 New input editor options: :no-input-save, :activation, :command, :preemptable

The **:no-input-save** option causes the input editor not to save the scanned contents of the buffer on the input history.

:no-input-save *Option*

The input editor does not save the scanned contents of the input buffer on the input history when returning from the reading function. This is intended for use by functions such as **fquery** that use the input editor to ask simple questions whose responses are not worth saving. **yes-or-no-p** uses **:no-input-save** by default.

:activation allows a reading function (like **read** or **readline**) to recognize activation characters.

:activation *function* &rest *arguments* *Option*

For each character typed, the input editor invokes *function* with the character as the first argument and *arguments* as the remaining arguments. If the function returns **nil**, the input editor processes the character as it normally would. Otherwise, the cursor is moved to the end of the input buffer, a rescan of the input is forced (if one is pending), and the blip (**:activation** *character numeric-arg*) is returned by the final sending of the **:any-tyi** message to the stream. Activation characters are not inserted into

the input buffer, nor are they echoed by the input editor. It is the responsibility of the reading function to do any echoing. For instance, **readline**, not the input editor, types a Newline at the end of the input buffer when RETURN, END, or LINE is pressed.

:command allows control to be returned from the input editor on reading special characters.

:command *function* &rest *arguments* *Option*

This option is used to implement nonediting single-keystroke commands. For each character typed, the input editor invokes *function* with the character as the first argument and *arguments* as the remaining arguments. If the function returns **nil**, the input editor processes the character as it normally would. Otherwise, control is returned from the input editor immediately. Two values are returned: a blip of the form (**:command** *character numeric-arg*) and the keyword **:command**. Any unscanned input typed before the command character remains in the input buffer, available to the next read operation from the stream.

:preemptable allows control to be returned from the input editor on reading blips.

:preemptable *token* *Option*

A blip in the input stream causes control to be returned from the input editor immediately. Two values are returned: the blip and *token*, which is usually a keyword symbol. Any unscanned input typed before the blip remains in the input buffer, available to the next read operation from the stream.

This example illustrates the use of the **:command**, **:preemptable**, and **:prompt** input editor options. It is a simple command loop that reads different kinds of commands -- typed Lisp expressions, single-keystroke commands, and mouse clicks. The Lisp expressions are read using the **si:read-or-end** function. You can provide four kinds of input:

Input	*Action*
END	Exit the command loop
Lisp form	Print form on next line
Mouse click	Display type of click and mouse coordinates
Single-key command	Display keystroke

The predicate for detecting a single-keystroke command simply checks for the Super bit. In a more complex program, it might look up the character in a command table.

```
(defun command-char-p (c) (ldb-test %%kbd-super c))

(defun command-loop ()
  (loop
    do (multiple-value-bind (value flag)
           (si:read-or-end standard-input nil '((:command command-char-p)
                                                (:preemptable :blip)
                                                (:prompt "Command loop input: ")))
         (selectq flag
           (:end
            (format t "Done")
            (return t))
           (:blip
            (selectq (car value)
              (:mouse-button
               (destructuring-bind (click nil x y) (cdr value)
                 (format t "~C click at ~D, ~D" click x y)))
              (otherwise (format t "Random blip -- ~S" value))))
           (:command
            (format t "Execute ~:C command" (second value)))
           (otherwise
            (format t "~&Value is ~S" value))))))
```

2.2.16.2 New macro: with-input-editing

with-input-editing defines a context in which a reading function can be called using the input editor.

with-input-editing (*stream-var input-editor-options parameters keyword*) *body...* *Macro*

This macro provides a convenient way of invoking the input editor for use by a reading function. It establishes a context in which input editing should be provided. Use this macro instead of sending a **:rubout-handler** message directly.

All the "arguments" are optional. *stream-var* is a variable bound to the stream from which characters are read; if *stream-var* is not provided or is **nil**, **standard-input** is used. *input-editor-options*, if provided and not **nil**, should evaluate to a list of options suitable for the first argument to the **:rubout-handler** message.

keyword determines the activation characters for the input editor:

Value	*Activation characters*
nil	None
:end-activation	**#\end**
:line-activation	**#\end**, **#\return**, and **#\line**

:line **#\end**, **#\return**, and **#\line**. In addition, a Newline is echoed after the reading function returns.

The macro defines an internal function with *body* as its body and (*stream-var* . *parameters*) as its argument list. When the function containing the **with-input-editing** form is called from outside the input editor, the stream is sent a **:rubout-handler** message if it handles it. The arguments to the **:rubout-handler** message are the list of *input-editor-options*, including any activation option determined by *keyword*; the internal function; and any arguments to the internal function. If the function containing the **with-input-editing** form is called from inside the input editor or if the stream does not handle the **:rubout-handler** message, the internal function is called instead.

parameters must be **nil** or a list of arguments and local variables that are defined outside the scope of the **with-input-editing** form but referenced within it. If a lexically external variable **x** is referenced within *body* but does not appear in the *parameters* list, the compiler issues the warning, "Lexical scoping not implemented for the variable **x**."

The following example defines a reading function.

```
(defun read-number
       (&optional (stream standard-input) input-editor-options
        input-radix or-nil)
  (with-input-editing (stream input-editor-options (input-radix or-nil)
                              :line)
    (loop with number
          with ibase = (or input-radix ibase)
          for string = (readline-or-nil stream)
          do (cond ((not string)
                    (if or-nil (return nil)))
                   ((numberp (setq number (ignore-errors
                                            (read-from-string string))))
                    (return number))
                   (t (sys:parse-ferror
                        "A~:[~; decimal~] number~:[~;, or <Return> for none,~] ~
                        is required"
                        (= ibase 10.) or-nil))))))
```

2.2.16.3 New function: read-delimited-string

read-delimited-string allows reading from a stream until a delimiter character is encountered.

read-delimited-string &optional (*delimiters* **#\end**) (*stream* **standard-input**) (*eof* **nil**) (*input-editor-options* **nil**) &rest (*make-array-args* **'(100. :type art-string)**) *Function*

delimiter is either a character or a list of characters. Characters are read

from *stream* until one of the delimiter characters is encountered. The characters read up to the delimiter are returned as a string. This function may be invoked from inside or outside the input editor. If invoked from outside the input editor, the delimiter characters are set up as activation characters. The *eof* argument is treated the same way as the *eof* argument to the **:tyi** message to noninteractive streams. *input-editor-options* are passed on as the first argument to the **:rubout-handler** message, after having an **:activation** entry prepended. *make-array-args* are arguments to be passed to **make-array** when constructing the string to return.

read-delimited-string returns four values:

- The string
- An *eof* flag, if the *eof* parameter was **nil**
- The character that delimited the string
- Any numeric argument given the delimiter character

This function is used by **readline, qsend,** and the **:delimited-string** option for **prompt-and-read.**

Examples:

The following reads characters until END is typed and returns a string at least 200. characters long with a leader-length of 3:

```
(read-delimited-string #\end standard-input nil nil 200. :leader-length 3)
```

The following is the same as **(readline),** except that it does not echo a Newline after the string is activated:

```
(read-delimited-string '(#\return #\line #\end))
```

A simple word parser:

```
(read-delimited-string '(#\space #/, #/. #/?))
```

For a more complex example of a sentence parser that uses *read-delimited-string*: See the section "Examples of Use of the Input Editor".

2.2.16.4 New optional argument to read

A list of input editor options can now be passed to **read** as an optional argument.

read &optional (*stream* **standard-input**) *eof-option* *Function*
input-editor-options

read reads in the printed representation of a Lisp object from *stream*, builds a corresponding Lisp object, and returns the object. For details: See the section "Input Functions".

(This function can take its arguments in the other order, for Maclisp compatibility only.)

2.2.16.5 New function: si:read-or-end

read-or-end &optional (*stream* **standard-input**) *eof-option* *input-editor-options* *Function*

This function is like **read**, except that if it is reading from an interactive stream and the user presses END as the first character or the first character after only whitespace characters, it returns two values, **nil** and **:end**. If it encounters any nonwhitespace characters, END has the same meaning as for **read**. *eof-option* has the same meaning as for other reading functions. *input-editor-options* are passed to the input editor if the stream supports it.

The **:expression-or-end** and **:eval-form-or-end** options for **prompt-and-read** invoke **si:read-or-end**.

2.2.16.6 readline and readline-trim return additional values

readline and **readline-trim** now return four values: a string; an *eof* flag; the delimiter that terminated the string; and any numeric argument given the delimiter.

readline &optional (*stream* **standard-input**) *eof-option* *input-editor-options* *Function*

readline reads in a line of text. If called from inside the input editor or if reading from a stream that does not support the input editor, the line is terminated by a Newline character. If the stream supports the input editor and **readline** is called from outside the input editor, the line is terminated by RETURN, LINE, or END.

This function is usually used to get a line of input from the user. If *stream* supports the input editor, **readline** calls **read-delimited-string**, and *input-editor-options* is passed as the list of options to the input editor.

readline returns four values:

- The line as a character string, without the Newline character.
- An *eof* flag, if *eof-option* was **nil**. This is **t** if the line was terminated because end-of-file was encountered, or **nil** if it was terminated because of a RETURN, LINE, or END character.
- The character that delimited the string.
- Any numeric argument given the delimiter character.

See the function **read-delimited-string**.

readline-trim &optional (*stream* **standard-input**) *eof-option* *input-editor-options* *Function*

readline-trim trims leading and trailing whitespace from string input. "Whitespace" means spaces, tabs, or newlines. It takes the same arguments as the normal **readline** and returns the same four values.

Examples:

```
(readline-trim)   exciting option   RETURN =>
"exciting option"
NIL
141
NIL

(readline-trim)RETURN =>
""
NIL
141
NIL
```

The **:string-trim** option for **prompt-and-read** and **tv:choose-variable-values** uses **readline-trim**.

2.2.16.7 New function: readline-or-nil

readline-or-nil returns a string trimmed of white space, or else **nil** if the string is empty.

readline-or-nil &optional (*stream* **standard-input**) *eof-option* *Function*
input-editor-options

Like **readline-trim**, except that it returns a first value of **nil** instead of the empty string if the input string is empty.

The **:string-or-nil** option for **prompt-and-read** and the **:string-or-nil** choose-variable-values keyword use **readline-or-nil**.

See the function **readline-trim**.

2.2.16.8 New methods of tv:stream-mixin: :start-typeout, :finish-typeout, :rescanning-p, :force-rescan, :replace-input, :read-bp

Six new methods have been added to **tv:stream-mixin** for communication between the input editor and sophisticated reading functions that offer typeout and completion. The methods are **:start-typeout**, **:finish-typeout**, **:rescanning-p**, **:force-rescan**, **:replace-input**, and **:read-bp**.

:start-typeout *type* &optional *spacing* of **tv:stream-mixin** *Method*

Informs the input editor that typeout to the window will follow. The word "typeout" is used in the name of this message because this is very similar to typeout in the editor, even though typeout windows are not actually used. *type* can be one of the following keywords:

Keyword	*Action*
:insert	Typeout is inserted before the current input, as is done with notifications or input editor documentation.
:overwrite	Like **:insert**, but the next time **:insert** or **:overwrite** typeout is performed, this typeout is overwritten.

:append	Typeout appears after the current input, which remains visible before the typeout. This is the style used by **break**.
:temporary	Typeout appears after the current input and is erased after the user types a character.

spacing can be one of the following keywords:

Keyword	*Action*
:none	No spacing before typeout.
:fresh-line	Typeout begins at the beginning of a line.
:blank-line	A blank line precedes typeout.

If *spacing* is not specified, a default that depends on *type* is computed.

:finish-typeout &optional *spacing erase?* of **tv:stream-mixin** *Method*

Completes typeout to the window and causes the input buffer to be refreshed. In the case of **:temporary** typeout, the *erase?* parameter is used to indicate whether or not the typeout overwrote part of the current input by wrapping around the screen. It is the responsibility of the program doing the typeout to keep track of how much is output.

spacing can be one of the following keywords:

Keyword	*Action*
:none	No spacing before typeout.
:fresh-line	Typeout begins at the beginning of a line.
:blank-line	A blank line precedes typeout.

If *spacing* is not specified, a default that depends on the *type* argument to the **:start-typeout** method is computed.

:rescanning-p of **tv:stream-mixin** *Method*

This message can be sent by a read function that uses the input editor to determine whether the next character returned by **:tyi** will come from the input buffer or from the keyboard. If **t** is returned, the input is being rescanned and the next character will come from the input buffer. If **nil** is returned, the next character will come from the keyboard.

:force-rescan of **tv:stream-mixin** *Method*

This message can be sent by a read function that uses the input editor to force a rescan of the current input. Before this message is sent, usually some global state has changed and the contents of the input buffer are interpreted differently.

:replace-input *n-chars string* &optional *(begin* **0***) end* of **tv:stream-mixin** *Method*

This message can be sent by a read function that uses the input editor to provide completion of the current input. *n-chars* is the number of characters to be removed from the end of the input buffer and erased from the screen. The substring of *string* determined by *begin* and *end* is then displayed on the screen. The scan pointer is left after the string, and a rescan does not take place. If a rescan takes place at some later time, the characters in *string* will be seen as input.

:read-bp of **tv:stream-mixin** *Method*

Returns the value of the scan pointer. This is for the benefit of read functions that might want to return a pointer into the input buffer when signalling an error of type **sys:parse-error**.

2.2.16.9 New variable: tv:rh-typeout-default

tv:rh-typeout-default controls the style of typeout performed by the input editor.

tv:rh-typeout-default *Variable*

Controls the style of typeout performed by the input editor. Permissible values are the keywords acceptable as the *type* argument to the **:start-typeout** method of **tv:stream-mixin**. These are **:insert**, **:overwrite**, **:append**, and **:temporary**. The default value is **:insert**.

2.2.16.10 Using the Input Editor: Examples

This series of examples shows several different ways of using the input editor, gradually increasing in complexity. The examples are also available in the file sys: examples; interaction.lisp.

We refer to functions whose names begin with "read-" as "reading functions" or "readers", since they read individual characters and construct a Lisp object as a returned value. Examples of readers the Lisp system provides are **read**, **readline**, and **read-delimited-string**. **read** returns Lisp objects of many types. **readline** and **read-delimited-string** return strings.

read-two-lines-1 reads two lines of input from the console. You type each line in its own editing context. After you enter the first line by pressing RETURN, LINE, or END, you can no longer rub out or otherwise edit any of the characters in the first line. You can type and edit only the second line at that point.

```
(defun read-two-lines-1 () (list (readline) (readline)))
```

read-two-lines-2 lets you edit both lines in a single context by using the **with-input-editing** macro. Even after entering the first line you can edit it. For example, the M-< input editor command moves the cursor to the first character of the first line. **read-two-lines-2** also adds a stream parameter so that you can read from different streams without having to bind **standard-input**. You can also use this function for reading from noninteractive streams, such as file streams.

```
(defun read-two-lines-2 (&optional (stream standard-input))
  (with-input-editing (stream) (list (readline stream) (readline stream))))
```

read-two-lines-3 demonstrates the use of the **:prompt** input editor option and the **:end-activation** option for **with-input-editing**. When you invoke this function on an interactive stream you receive a prompt. This prompt is redisplayed if typeout to the stream occurs. This might happen if you press HELP or the window receives a notification.

The **:end-activation** option defines **#\end** as an activation character. This lets you activate previous input to **read-two-lines-3**, after yanking and editing it, by pressing END. The **:prompt** and **:end-activation** options have no effect on the behavior of the function for noninteractive streams.

```
(defun read-two-lines-3 (&optional (stream standard-input))
  (with-input-editing (stream '((:prompt "Type two lines: ")) () :end-activation)
    (list (readline stream) (readline stream))))
```

read-n-lines-1 is like **read-two-lines** except that you specify the number of lines to be read using the **n-lines** argument. This example illustrates passing a parameter into the body of the **with-input-editing** form.

```
(defun read-n-lines-1 (n-lines &optional (stream standard-input))
  (with-input-editing (stream '((:prompt "Type some lines: ")) (n-lines) :end-activation)
    (loop repeat n-lines collect (readline stream))))
```

read-n-lines-2 shows a different way of passing the **n-lines** parameter into the **with-input-editing** body. It uses a prompt function instead of a string to generate the prompt, and it passes the **n-lines** parameter to that function.

```
(defvar *n-lines*)

(defun read-n-lines-prompt (stream ignore)
  (format stream "Type ~R line~:P:~%" *n-lines*))

(defun read-n-lines-2 (*n-lines* &optional (stream standard-input))
  (with-input-editing (stream '((:prompt read-n-lines-prompt)) () :end-activation)
    (loop repeat *n-lines* collect (readline stream))))
```

Next is an example of a simple sentence parser. It builds a list of strings and symbols that represent the words and punctuation marks of the sentence. A sentence may be any number of lines long. It is delimited by a period or a question mark. Words are delimited by a space, newline, or punctuation mark. This is also an example of a reading function written entirely in terms of **:tyi** as the primitive input operation.

```
(defun read-sentence-1 (&optional (stream standard-input))
  (with-input-editing (stream '((:prompt "Type a sentence:  ")))
    (loop named sentence
          with sentence = nil
          for word = (make-array 20. ':type art-string ':fill-pointer 0)
          do (loop for char = (send stream ':tyi)
                   do (cond ((memq char '(#\space #\return #/. #/? #/,))
                             (if (not (equal word ""))
                                 (push word sentence))
                             (selectq char
                               ((#\space #\return)
                                (return))
                               (#\.
                                (push ':period sentence)
                                (return-from sentence (nreverse sentence)))
                               (#\?
                                (push ':question-mark sentence)
                                (return-from sentence (nreverse sentence)))))
                            (t (array-push-extend word char)))))))
```

Following is a different sentence parser that calls **read-delimited-string** to accumulate characters into a string. It uses the **:end-activation** option for **with-input-editing** so that previous input to **read-sentence-2** can be yanked, edited, and activated using the END key. When it detects incorrect uses of punctuation, it calls **sys:parse-ferror** to signal an error caught by the input editor.

```
(defun read-sentence-2 (&optional (stream standard-input))
  (with-input-editing (stream '((:prompt "Type a sentence:  ")) () :end-activation)
    (loop with sentence = nil
          do (multiple-value-bind (word nil delimiter)
                 (read-delimited-string '(#\space #\return #/. #/? #/, #/: #/;) stream)
               (if (not (equal word ""))
                   (push word sentence))
               (cond ((memq delimiter '(#\space #\return)))
                     ((null sentence)
                      (if (eq delimiter #\end)
                          (return nil)
                          (sys:parse-ferror
                            "The punctuation mark /"~C/" occurred at the ~
                             beginning of the sentence."
                            delimiter)))
                     ((symbolp (car sentence))
                      (sys:parse-ferror
                        "The punctuation mark /"~C/" was typed after a ~@^."
                        delimiter (car sentence)))
                     (t (selectq delimiter
                          (#/,
                           (push ':comma sentence))
                          (#/:
                           (push ':colon sentence))
                          (#/;
                           (push ':semicolon sentence))
                          (#/.
                           (push ':period sentence)
                           (return (nreverse sentence)))
                          (#/?
                           (push ':question-mark sentence)
                           (return (nreverse sentence)))))))))
```

Sometimes an error in parsing is detected not by the function that invokes the input editor, but by some function that it calls. In the next example, **read-time** invokes **time:parse-universal-time** to do its parsing. If we did not use the **condition-case** form in **read-time**, we would enter the Debugger when **time:parse-universal-time** encountered incorrect input. The **condition-case** form encapsulates the original error in one of flavor **sys:parse-ferror** so that the input editor catches it. Alternately, we could define **time:parse-error** to be a subflavor of **sys:parse-error**.

```
(defun read-time (&optional (stream standard-input) input-editor-options)
  (with-input-editing (stream input-editor-options () :line)
    (let ((string (readline-or-nil stream)))
      (when string
        (condition-case (error)
            (time:parse-universal-time string)
          (time:parse-error
            (sys:parse-ferror "~A" error)))))))
```

2.2.17 New macro: sys:with-open-file-search

sys:with-open-file-search is like **with-open-file**, but it searches for a file with one of the types in a list of file types. **load** uses this to search first for a binary file and then for a source file.

sys:with-open-file-search (*stream-variable* *Macro*
(*operation defaults auto-retry*)
(*type-list-function pathname* . *type-list-args*) .
open-options) *body...*

sys:with-open-file-search performs a **with-open-file**, searching for a file with one of the types in a list of file types. **load** uses this macro when not given a specific file type to search first for a binary file and then for a source file.

The body is evaluated with *stream-variable* bound to a stream that reads or writes the file. *open-options* are alternating keywords and values to be passed to **open**.

type-list-function should be a function whose first argument is *pathname* and whose remaining arguments are *type-list-args*. The function should return two values: a list of file types to be searched, in order of preference, and a base pathname to be merged with the types and *defaults* in searching for the file. *defaults* can be a pathname or a defaults alist; if omitted, the defaults come from **fs:*default-pathname-defaults***. The macro uses **fs:merge-pathname-defaults** for merging.

If no file is found with any of the types in the list of types, **fs:multiple-file-not-found** is signalled. *operation* is the name of the operation that failed; usually this is the name of the function that contains the **sys:with-open-file-search** form. If *auto-retry* is not **nil** and the condition is not handled, the user is prompted for a new pathname.

2.2.18 New condition flavor: fs:multiple-file-not-found

fs:multiple-file-not-found is signalled when none of a number of possible files is found. This condition is signalled by **sys:with-open-file-search** when it fails to find any file.

fs:multiple-file-not-found *Flavor*

None of a number of possible files was found. This flavor is built on **fs:file-lookup-error**. It is signalled when **load** is not given a specific file type but cannot find either a source or a binary file to load.

The flavor allows three init keywords of its own. These are also the names of messages that return the following:

:operation	The operation that failed
:pathname	The pathname given to the operation
:pathnames	A list of pathnames that were sought unsuccessfully

The condition has a **:new-pathname** proceed type to prompt for a new pathname.

2.2.19 New condition flavor: fs:rename-across-hosts

fs:rename-across-hosts *Flavor*

The hosts of the initial and target pathnames are not the same. This flavor is built on **fs:rename-failure**.

2.2.20 New variable: fs:*remember-passwords*

fs:*remember-passwords*, if not **nil**, causes the first password for each file access path to be remembered and suppresses prompting for passwords when the same person uses that access path again.

fs:*remember-passwords* *Variable*

If not **nil**, causes the first password for each file access path to be remembered. This suppresses prompting for passwords on subsequent attempts by the same user to use that access path. The default value is **nil**.

Note that if you set this variable in an init file, your first login password, typed before the init file is loaded, is not remembered.

Caution: Remembered passwords are accessible. Even after you log out the remembered password for each access path is accessible. If password security is important, you probably should not set this variable to a non-**nil** value.

2.2.21 New function: si:patch-loaded-p

si:patch-loaded-p is a predicate that tells whether the loaded version of a system is at or past a specified patch level.

si:patch-loaded-p *major-version minor-version* *Function*
&optional *(system* **"System"***)*

A predicate that tells whether the loaded version of *system* is past (or at) the specified patch level. Returns **t** if:

- the major version loaded is *major-version* and the minor version loaded is greater than or equal to *minor-version*

- the major version loaded is greater than *major-version*

Otherwise, the function returns **nil**.

2.2.22 New functions: si:make-process-queue, si:process-enqueue, si:process-dequeue, si:process-queue-locker, si:reset-process-queue

A process queue is a new facility for round-robin locking. Each process that requests a lock via a queue enters itself on the queue if the lock is not free. Processes are given a chance to seize the lock in the order in which they request it.

si:make-process-queue *name size* *Function*

Makes and returns a queue for processes requesting a lock. *name* is an external name for the queue and is used only in printing the queue. *size* is the size of the queue. This is the maximum number of processes that will be guaranteed to lock the queue in exact requesting order.

si:process-enqueue *queue* &optional *queue-value* (*whostate* **"Lock"**) *Function*

Locks *queue*. *queue-value* is an object to enter on the queue; if *queue-value* is **nil** or unsupplied, the object is the current process. If *queue* is empty, seizes the lock immediately by inserting *queue-value* on the queue and returning. If *queue* is not full but other processes are on the queue waiting for the lock to be free, inserts *queue-value* at the end of the queue, waits for the lock to be free, and then seizes the lock by returning. If *queue* is full, waits until *queue* is not full and tries again to seize the lock. *whostate* is displayed in the status line while waiting to seize the lock. Signals an error if *queue-value* has already seized the lock.

si:process-dequeue *queue* &optional *queue-value* (*error-p* **t**) *Function*

Unlocks *queue*. *queue-value* is an object on the queue. If *queue-value* is **nil** or unsupplied, it is the current process; if not **nil**, it should be the same as the *queue-value* given to the matching call to **si:process-enqueue**. If *queue-value* has the lock, unlocks the lock by removing *queue-value* from *queue* and giving the next process on the queue a chance to seize the lock. If *queue-value* does not have the lock and *error-p* is not **nil**, signals an error.

si:process-queue-locker *queue* *Function*

Returns the *queue-value* for the process that holds the lock on *queue*, or **nil** if the lock is free.

si:reset-process-queue *queue* *Function*

Unlocks *queue* and removes all processes on the queue.

2.2.23 New function: applyhook

applyhook provides a hook into **apply**, much as **evalhook** provides a hook into **eval**. It is useful for printing debugging information about **apply** operations.

applyhook *Variable*

When the value of this variable is not **nil** and **eval** calls **apply**, **applyhook** is bound to **nil** and the function that was its value is applied to two arguments: the function that **eval** gave to **apply** and the list of arguments to that function. The value it returns is returned from the evaluator.

applyhook *function args evalhook applyhook* *Function*

function is applied to *args* with **evalhook** lambda-bound to the function *evalhook* and with **applyhook** lambda-bound to the function *applyhook*. Like the **evalhook** function, this bypasses the first place where the relevant hook would normally be triggered. Either of the last two arguments can be nil.

The function **evalhook** now takes an optional third argument, an *applyhook* function to be called by **eval** after **eval** has evaluated the arguments to a function.

evalhook *form evalhook* &optional *applyhook* *Function*

evalhook is a function that helps exploit the **evalhook** feature. The *form* is evaluated with **evalhook** lambda-bound to the function *evalhook*. The checking of **evalhook** is bypassed in the evaluation of *form* itself, but not in any subsidiary evaluations, for instance of arguments in the *form*. This is like a "one-instruction proceed" in a machine-language debugger.

```
Example:
;; This function evaluates a form while printing debugging
;; information.
(defun hook (x)
   (terpri)
   (evalhook x 'hook-function))

;; Notice how this function calls evalhook to evaluate the
;;  form f, so as to hook the subforms.
(defun hook-function (f)
   (let ((v (evalhook f 'hook-function)))
     (format t "form: ~s~%value: ~s~%" f v)
     v))

;; This isn't a very good program, since if f returns multiple
;; values, it will not work.
```

The following output might be seen from **(hook '(cons (car '(a . b)) 'c))**:

```
form: (quote (a . b))
value: (a . b)
form: (car (quote (a . b)))
value: a
form: (quote c)
value: c
(a . c)
```

Normally after **eval** has evaluated the arguments to a function, it calls the function. If *applyhook* exists, however, **eval** calls the hook with two arguments: the function and its list of arguments. The values returned by the hook constitute the values for the form. The hook could use **apply** on its arguments to do what **eval** would have done normally. This hook is active for special forms as well as for real functions.

Whenever either an evalhook or applyhook is called, both hooks are bound off. The evalhook itself can be **nil** if only an applyhook is needed.

applyhook catches only **apply** operations done by **eval**. It does not catch **apply** called in other parts of the interpreter or **apply** or **funcall** operations done by other functions such as **mapcar**. In general, such uses of **apply** can be dealt with by intercepting the call to **mapcar**, using the applyhook, and substituting a different first argument.

The argument list is like an &rest argument: it might be stack-allocated but is not guaranteed to be. Hence you cannot perform side-effects on it and you cannot store it in any place that does not have the same dynamic extent as the call to *applyhook*.

2.2.24 New variable: gc-on

gc-on *Variable*

The value of this variable is **t** when the garbage collector is turned on and **nil** when it is turned off. **gc-on** is useful in finding out whether the garbage collector has turned itself off (as it does when not enough free space remains to be able to complete a copying garbage collection).

2.2.25 New initialization list: :after-full-gc

si:full-gc runs the **:after-full-gc** initialization list after it collects all the garbage. It runs the previously undocumented **:full-gc** initialization list before it collects the garbage.

See the section "System Initialization Lists".

2.2.26 New variable: dbg:*debug-io-override*

dbg:*debug-io-override* can be used to direct the Debugger to a stream other than that designated by **debug-io**. This is useful mainly in complex debugging using the cold-load stream.

dbg:*debug-io-override* *Variable*

If the value of this variable is **nil** (the default), the Debugger uses the stream that is the value of **debug-io**. But if the value of **dbg:*debug-io-override*** is not **nil**, the Debugger uses the stream that is the value of this variable instead. This variable should always be set (using **setq**), not bound, so all processes and stack groups can see it.

2.2.27 New message to conditions: :special-command-p

You can send the **:special-command-p** message to a condition object to determine whether a command is one of the Debugger special commands for that object.

:special-command-p *command-type* of **condition** *Method*

Returns **t** if *command-type* is a valid Debugger special command for this condition object; otherwise, returns **nil**.

2.2.28 New macro: tv:with-mouse-grabbed-on-sheet

tv:with-mouse-grabbed-on-sheet grabs the mouse and confines it to a window. This is usually preferable to using **tv:with-mouse-grabbed**.

tv:with-mouse-grabbed-on-sheet (*sheet*) *body...* *Macro*

Evaluates *body* with the mouse grabbed and confined to *sheet*. During execution the variables **tv:mouse-x** and **tv:mouse-y** are relative to the window's outside coordinates. The default value of *sheet* is **self**.

2.2.29 New variable: tv:cold-load-stream-old-selected-window

This variable tells you which window was selected at the time you entered the cold load stream.

tv:cold-load-stream-old-selected-window *Variable*

At a cold-load stream break, the value of this variable is the value of **tv:selected-window** at the time you entered the cold-load stream.

2.2.30 New flavor: tv:margin-space-mixin

tv:margin-space-mixin lets you leave some blank space in the margins of a window. You can use the **:space** init option and the **:space** and **:set-space** messages to determine the amount of blank space to be left.

tv:margin-space-mixin *Flavor*

This flavor provides a margin item that just leaves some blank space. It might be useful if you're using scroll bars, and you want to leave a little white space between the scroll bar and the inside of the window.

:space (for **tv:margin-space-mixin**) *Init Option*

Initializes the amount of blank space in the margins of the window. Possible values:

nil	No space
t	One pixel blank in each of the four margins
n	*n* pixels of space in each of the four margins (*n* is an integer)

(*left top right bottom*)
left pixels blank in the left margin, *top* pixels blank in the top margin, and so on (values are integers)

:space of **tv:margin-space-mixin** *Method*

Returns a list of four elements, (*left top right bottom*). These are integers representing the number of pixels of blank space in the four margins of the window.

:set-space *new-space* of **tv:margin-space-mixin** *Method*

Specifies the amount of blank space to be left in the margins of the window. Possible values of *new-space*:

nil	No space
t	One pixel blank in each of the four margins
n	*n* pixels of space in each of the four margins (*n* is an integer)

(*left top right bottom*)
left pixels blank in the left margin, *top* pixels blank in the top margin, and so on (values are integers)

2.2.31 New font: fonts:cptfonti

A new tv font, **fonts:cptfonti**, is available. This is a fixed-width italic font of the same width and shape as **fonts:cptfont**, the default screen font. It is most useful for italicizing running text along with **fonts:cptfont**.

2.2.32 New Choose-variable-values Keywords

The following choose-variable-values keywords are new in Release 5.0 or were previously undocumented: **:choose-multiple, :string-or-nil, :decimal-number, :decimal-number-or-nil, :date-or-never, :past-date, :time-interval-or-never, :pathname, :pathname-or-nil, :pathname-list, :host, :host-list, :pathname-host, :keyword-list,** and **:font-list.**

Keyword *Action*

:choose-multiple *values-list print-function*
This type takes arguments like **:assoc** but permits the user to choose more than one element in the values-list. The variable is set to a list of all the values chosen.

:string-or-nil This value is a string or **nil** if the user just presses RETURN, LINE, or END.

:decimal-number This value is a decimal number, read and printed in radix 10.

:decimal-number-or-nil
This value is a decimal number, read and printed in radix 10, or **nil** if the user just presses RETURN, LINE, or END.

:date-or-never This value is a universal date-time or **nil** if the user types "never". An ambiguous date is interpreted as being in the future.

:past-date The value is a universal date-time. An ambiguous date is interpreted as being in the past.

:time-interval-or-never
The value is an integer representing the number of seconds in a time interval, or **nil** if the user types "never". The interval is read and printed as either "never" or alternating numbers and units of time; the units can include seconds, minutes, hours, days, weeks, or years.

:pathname The value is a pathname, represented as a string. The pathname read is merged with the defaults in **fs:*default-pathname-defaults*** and has a default version of **:newest.**

:pathname-or-nil The value is a pathname, represented as a string, or **nil** if the user just presses RETURN, LINE, or END. The pathname read is merged with the defaults in **fs:*default-pathname-defaults*** and has a default version of **:newest.**

:pathname-list The value is a list of pathnames, read as a series of pathnames separated by commas and optional spaces, and merged with the defaults in **fs:*default-pathname-defaults***. The default version is **:newest.** The list is printed as a series of pathnames separated by commas and spaces.

:host The value is a network host, read and printed as the name of the host.

:host-list The value is a list of network hosts, read as a series of host names separated by commas or spaces, and printed as a series of host names separated by commas and spaces.

:pathname-host The value is a pathname host, read and printed as the name of the host. The name can be "local", "sys", or the name of another logical host as well as the name of a physical host.

:keyword-list The value is a list of symbols in the **keyword** package, read as a series of symbol names separated by commas or spaces, and printed as a series of symbol names separated by spaces. Symbol names are read and printed without package prefixes (that is, not preceded by colons).

:font-list The value is a list of fonts, read as a series of font names separated by commas or spaces, and printed as a series of font names separated by commas and spaces. Font names are read and printed without package prefixes (that is, not preceded by **fonts:**).

2.3 Improvements to Lisp in Release 5.0

2.3.1 Previously undocumented special form: destructuring-bind

destructuring-bind *variable-pattern data body ...* *Special Form*

destructuring-bind binds variables to values, using **defmacro**'s destructuring facilities, and evaluates the body forms in the context of those bindings.

First *data* is evaluated. If *variable-pattern* is a symbol, it is bound to the result of evaluating *data*. If *variable-pattern* is a t the result of evaluating *data* should be a tree of the same shape. The trees are disassembled, and each variable that is a component of *variable-pattern* is bound to the value that is the corresponding element of the tree that results from evaluating *data*. If not enough values are supplied, the remaining variables are bound to **nil**. If too many values are supplied, the excess values are ignored. Finally, the body forms are evaluated sequentially, the old values of the variables are restored, and the result of the last body form is returned.

As with the pattern in a **defmacro** form, *variable-pattern* actually resembles the **lambda**-list of a function; it can have **&**-keywords. See the section "Advanced Features of **defmacro**".

Example:

```
(destructuring-bind (a (b) &optional (c 'd))
                    '((x y) (z))
  (values a b c))
```

returns **(x y)**, **z**, and **d**.

2.3.2 Invisible blocks in progs and dos

You can now make a block invisible to **returns** in any kind of **prog** or **do** form by including immediately within it the form **(declare (invisible-block t))**. This feature is intended for macro expansions, not for user code.

See the special form **do-named**.

2.3.3 Previously undocumented function: clear-resource

clear-resource causes **allocate-resource** to ignore existing objects in the resource and make new objects when called.

clear-resource *resource-name* *Function*

Forget all of the objects being remembered by the resource specified by *resource-name*. Future calls to **allocate-resource** create new objects. This function is useful if something about the resource has been changed incompatibly, such that the old objects are no longer usable. If an object of the resource is in use when **clear-resource** is called, an error is signalled when that object is deallocated.

2.3.4 Multidimensional Arrays on the 3600 Remember Actual Dimensions

Arrays of more than one dimension on the 3600 now store their dimensions. This allows multidimensional indirect arrays to have *conformal indirection*. A new **make-array** option, **:displaced-conformally**, has been added. The window system now uses conformal indirect arrays for screen arrays.

Multidimensional arrays on the 3600 remember their actual dimensions, separately from the magic numbers by which to multiply the subscripts before adding them together to get the index into the array.

As a result of this, multidimensional indirect arrays on the 3600 can have *conformal indirection*. If A is indirected to B, and they do not have the same width, then normally the part of B that is shared with A does not have the same shape as A. If conformal indirection is used, then it does have the same shape and there are gaps between the rows of A. For example:

```
(setq b (make-array '(10. 20.)))
(setq a (make-array '(3 5) ':displaced-to b ':displaced-index-offset 12.))
```

Now:

```
(aref a 1 0) = (aref b 3 1) and (aref a 1 1) = (aref b 6 1).
```

In contrast:

```
(setq a (make-array '(3 5) ':displaced-to b
                           ':displaced-index-offset 12.
                           ':displaced-conformally t))
```

(aref a 1 0) = (aref b 3 1) still, but (aref a 1 1) = (aref b 3 2). Each row of A corresponds to part of a row of B, always starting at the same column (2).

A graphic illustration:

```
(setq a (make-array '(6 20.))
      b (make-array '(3 5) ':displaced-to a ':displaced-index-offset 22.)
      c (make-array '(3 5) ':displaced-to a ':displaced-index-offset 22.
                           ':displaced-conformally t))
```

```
  Normal case                    Conformal case
  0                   19         0                   19
  +--------------------+         +--------------------+
 0|aaaaaaaaaaaaaaaaaaaa|        0|aaaaaaaaaaaaaaaaaaaa|
  |aaBBBBBBBBBBBBBBBBaaa|         |aaCCCCCaaaaaaaaaaaaa|
  |aaaaaaaaaaaaaaaaaaaa|         |aaCCCCCaaaaaaaaaaaaa|
  |aaaaaaaaaaaaaaaaaaaa|         |aaCCCCCaaaaaaaaaaaaa|
  |aaaaaaaaaaaaaaaaaaaa|         |aaaaaaaaaaaaaaaaaaaa|
 5|aaaaaaaaaaaaaaaaaaaa|        5|aaaaaaaaaaaaaaaaaaaa|
  +--------------------+         +--------------------+
```

Arrays are stored in column-major order, so the units in which the index-offset is measured should be read first from left to right and then from top to bottom.

The meaning of **adjust-array-size** for conformal indirect arrays is undefined.

The window system now uses conformal indirect arrays for its screen arrays. This means that on the 3600 the bit-array in which a window saves its bits when it is not visible no longer has to be the full width of the screen; now it is just the width of the window, rounded up to the next multiple of 32 bits. On the LM-2, screen arrays and bit-save arrays are still the full width of the screen.

Some associated internal changes for both the 3600 and the LM-2:

- The **locations-per-line** instance variable changes when expose and deexpose happen
- The arguments to the **:create-screen-array** message to screens have changed
- The **:adjust-screen-array** message to sheets replaces the **:redirect-screen-array** message to screens
- New message to screens: **:inferior-screen-array-adjusted**

Screen arrays no longer use multilevel indirection; the screen array of a nonscreen sheet always indirects either to a bit-save array or to the screen array of its screen. The screen array of a screen is always a displaced array to the hardware screen buffer.

2.3.5 New options for make-plane: :initial-dimensions, :initial-origins

make-plane has two new keyword options: **:initial-dimensions** and **:initial-origins**.

make-plane *rank* &rest *options* *Function*

Creates and returns a plane. *rank* is the number of dimensions. *options* is a list of alternating keyword symbols and values. The allowed keywords are:

:type The array type symbol (for example, **art-1b**) specifying the type of the array out of which the plane is made.

:default-value
The default component value.

:extension
The amount by which to extend the plane. See the section "Planes".

:initial-dimensions
A list of dimensions for the initial creation of the plane. You might want to use this option to create a plane whose first dimension is a multiple of 32, so you can use **bitblt** on it. Default: the result returned by **(make-list** *rank* **':initial-value 1)**.

:initial-origins
A list of origins for the initial creation of the plane. Default: the result returned by **(make-list** *rank* **':initial-value 0)**.

Example:

```
(make-plane 2 ':type 'art-4b ':default-value 3)
```

creates a two-dimensional plane of type **art-4b**, with default value **3**.

2.3.6 New optional arguments to string-upcase and string-downcase

string-upcase and **string-downcase** now take three optional arguments: a starting index and limit for changing case in substrings, and an indicator of whether the string should be copied or modified directly.

string-upcase *string* &optional (*from* **0**) *to* (*copy-p* **t**) *Function*

If *copy-p* is not **nil**, returns a copy of *string*, with lowercase alphabetic characters replaced by the corresponding uppercase characters. If *copy-p* is **nil**, uppercases characters in *string* itself and then returns the modified *string*. *from* is the index in *string* at which to begin uppercasing characters. If *to* is supplied, it is used in place of **(array-active-length** *string***)** as the index one greater than the last character to be uppercased.

string-downcase *string* &optional (*from* **0**) *to* (*copy-p* **t**) *Function*

If *copy-p* is not **nil**, returns a copy of *string*, with uppercase alphabetic

characters replaced by the corresponding lowercase characters. If *copy-p* is **nil**, lowercases characters in *string* itself and then returns the modified *string*. *from* is the index in *string* at which to begin lowercasing characters. If *to* is supplied, it is used in place of **(array-active-length** *string***)** as the index one greater than the last character to be lowercased.

2.3.7 Previously undocumented function: string-compare

string-compare compares two strings using dictionary order and returns a number that depends on whether or not the strings are equal.

string-compare *string1 string2* &optional (*idx1* **0**) (*idx2* **0**) *lim1 lim2* *Function*

Compares the characters of *string1* starting at *idx1* and ending just below *lim1* with the characters of *string2* starting at *idx2* and ending just below *lim2*. The comparison is in alphabetical order. *lim1* and *lim2* default to the lengths of the strings. **string-compare** returns:

- a positive number if *string1* > *string2*
- zero if *string1* = *string2*
- a negative number if *string1* < *string2*

If the strings are not equal, the absolute value of the number returned is one more than the index (in *string1*) at which the difference occurred.

string-compare uses the same rules as **string** in coercing *string1* and *string2* into strings.

2.3.8 3600 select-methods handle :operation-handled-p and :send-if-handles

Select-methods on the 3600 now handle the **:operation-handled-p** and **:send-if-handles** messages. Methods for these messages are generated automatically when the **defselect** form is evaluated.

See the special form **defselect**. See the message **:operation-handled-p**. See the message **:send-if-handles**.

2.3.9 Compiler Performs Style Checking on All Forms

The compiler no longer fails to perform style checking on the results of macro expansions and optimizations.

The compiler performs style checking on all forms. Style checking is implemented by the **compiler:style-checker** property on a symbol; the value of the property is called on all forms whose **car** is that symbol, except those immediately enclosed in **inhibit-style-warnings**.

2.3.10 sys:dump-forms-to-file always puts package attribute into binary file

sys:dump-forms-to-file always puts a package attribute into the binary file it writes. If you do not specify the *attribute-list* argument, or if *attribute-list* does not contain a **:package** attribute, the function uses the **user** package. This is to ensure that package prefixes on symbols are always interpreted when they are loaded as they were intended when the file was dumped.

2.3.11 Previously undocumented macro: swapf

swapf exchanges the value of one generalized variable with that of another.

swapf *a b* *Macro*

Exchanges the value of one generalized variable with that of another. *a* and *b* are access-forms suitable for **setf**. The returned value is not defined. All the caveats that apply to **incf** apply to **swapf** as well: Forms within *a* and *b* may be evaluated more than once. Examples:

```
(swapf a b)
   ==> (setf a (prog1 b (setf b a)))
   ==> (setq a (prog1 b (setq b a)))

(swapf (car (foo)) (car (bar)))
   ==> (setf (car (foo)) (prog1 (car (bar)) (setf (car (bar)) (car (foo)))))
   ==> (rplaca (foo) (prog1 (car (bar)) (rplaca (bar) (car (foo)))))
```

Note that in the second example the functions **foo** and **bar** are called twice.

2.3.12 Compiler now warns about implicit progns in loops

An expression in a **loop** clause can be a single form or a series of forms that constitute an implicit **progn**. When an implicit **progn** appears, it is often an error caused by omitting a **do**. Because this error is so frequent, the intentional use of implicit **progn**s in most clauses is considered obsolete and dangerous. If you intend to use a **progn**, use an explicit one. To help you detect implicit **progn**s, the compiler now issues a warning whenever it encounters one in a context where it is likely to be a mistake.

Consider the following example:

```
(defun frob (list)
  (loop for thing in list
        collect (string thing)
        (format t "~&~A" thing)))
```

The returned value is a list of **nil**s, one for each element of *list*. The author most likely intended to return a list of the elements of *list*, coerced to strings, but omitted a **do** before the **(format ...)** form. When this definition is compiled, the compiler issues this warning:

```
For function FROB:
  The use of multiple forms with an implicit PROGN in this context is
  considered obsolete, but is still supported for the time being.
  If you did not intend to use multiple forms here, you probably
  omitted a DO.  If the use of multiple forms was intentional, put a
  PROGN in your code.
  The offending clause -- LIST (STRING THING) (FORMAT T "~&~A" THING)
```

2.3.13 Some Methods Can Use Combination Type as Method Type

Methods used with combination types that formerly allowed only untyped methods can now use the combination type as the method type.

Methods used with **:progn**, **:append**, **:nconc**, **:and**, **:or**, **:list**, **:inverse-list**, and **:pass-on** combination types can use the combination type as the method type. This is useful in documenting how the method is used.

In the following example, **(:method foo :or :find-frabjous-frob)** could have been defined as **(:method foo :find-frabjous-frob)**. The only difference is one of style: Using **:or** as the method type makes it clear that the methods are combined using **:or** combination.

```
(defflavor foo (frobl) (bar)
  (:method-combination (:or :base-flavor-last :find-frabjous-frob)))

(defmethod (foo :or :find-frabjous-frob) (type)
  (dolist (frob frobl)
    (when (send frob ':frabjous-p type)
      (return frob))))
```

2.3.14 Previously undocumented reader macro: #| and |#

#| begins a comment for the Lisp reader, and |# ends one.

#| begins a comment for the Lisp reader. The reader ignores everything until the next |#, which closes the comment. Note that if the |# is inside a comment that begins with a semicolon, it is *not* ignored; it closes the comment that began with the preceding #|. #| and |# can be on different lines, and #|...|# pairs can be nested.

2.3.15 New function to be called by reader macros: si:read-recursive

Reader macros that call a read function should now call **si:read-recursive** instead of **read**.

si:read-recursive *stream* *Function*

si:read-recursive should be called by reader macros that need to call a function to read. It is important to call this function instead of **read** in macros that are written in Zetalisp but used by the Common Lisp readtable. In particular, this function must be called by macros used in conjunction with the Common Lisp #*n*= and #*n*# syntaxes.

stream is the stream from which to read. This function may be called only from inside a **read**.

For example, this is the reader macro called when the reader sees a quote ('):

```
si:(defun xr-quote-macro (list-so-far stream)
     list-so-far                                    ;not used
     (values (list-in-area read-area
                           'quote (read-recursive stream))
             'list))
```

2.3.16 New optional arguments to read-from-string

read-from-string now takes two new optional arguments for specifying a substring. These are indices for the first character to be read and the character after the last one to be read.

read-from-string *string* &optional (*eof-option* **'si:no-eof-option**) (*start* **0**) *end* *Function*

The characters of *string* are given successively to the reader, and the Lisp object built by the reader is returned. Macro characters and so on will all take effect. If *string* has a fill-pointer it controls how much can be read.

eof-option is what to return if the end of the string is reached, as with other reading functions. *start* is the index in the string of the first character to be read. *end*, if given, is used instead of **(array-active-length** *string***)** as the integer that is one greater than the index of the last character to be read.

read-from-string returns two values: The first is the object read and the second is the index of the first character in the string not read. If the entire string was read, this is the length of the string.

Example:

```
(read-from-string "(a b c)") => (a b c) and 7
```

2.3.17 Changes to prompt-and-read

Changes have been made to many **prompt-and-read** options.

- New options: **:character**, **:date**, **:time-interval-or-never**, **:expression-or-end**, **:pathname-or-nil**, **:string-list**, **:delimited-string**, **:delimited-string-or-nil**, **:host**, **:host-list**, **:pathname-host**, **:keyword-list**, **:font-list**.

- When options accept a single line of text as input, the line can be terminated by RETURN, LINE, or END. These are activation characters, so they can be typed anywhere in the line.

- The **:number** option accepts keyword-value pairs that determine the base in which the number is read and whether or not **nil** can be returned.

- The **:pathname** option accepts a **:default-version** keyword. If not specified, the default version is **:newest**.

prompt-and-read *type* &optional *format-string* &rest *format-args* *Function*

prompt-and-read prompts the user, with *format-string* and its arguments as the prompt. It uses **format** to **query-io** to produce the prompt; it reads from the **query-io** stream, calling the reading function associated with the *type* keyword. If *format-string* is not specified, it generates a prompt appropriate to *type*. The *type* argument can be a list in which the first element is the type keyword and the rest are keyword/value pairs to serve as arguments to the reading function. **prompt-and-read** returns whatever the reading function returns.

This is an appropriate function to call for collecting input from the user. Its main advantages are that it does type checking on the input the user types and that it takes care of redisplaying the prompt at appropriate times (for example, after the screen has been refreshed or after a notification arrives).

```
(prompt-and-read ':number "Please enter a number: ") =>
Please enter a number: 4
4
(prompt-and-read ':string "Please enter a string: ") =>
Please enter a string: 4
"4"
```

It expects to collect input of type *type*, where *type* is a keyword. It handles the following types of input:

Option	*Action*
:eval-form	Reads a Lisp form. Evaluates it and returns the first value. Asks for confirmation of nonconstant values. The Debugger uses this to prompt for a form to evaluate.
:eval-form-or-end	Reads a Lisp form or just END. Evaluates it and returns the first value for a form. Returns two values, **nil** and **:end**, for END. Asks for confirmation of nonconstant values. The Debugger uses this to prompt for a form to evaluate.
:expression	Reads a Lisp expression. (It returns the expression without evaluating it.)
:expression-or-end	Reads a Lisp expression or just END. It returns the expression without evaluating it. If the user just presses END, it returns two values, **nil** and **:end**.
:character	Reads and returns a character. The returned value is a character code (an integer).

(**:number :input-radix** *radix* **:or-nil** *or-nil-value*)
Reads and returns a number, terminated by RETURN, LINE, or END. If **:input-radix** is specified, the number is read in radix *radix*; otherwise, it is read in the current **ibase**. If **:or-nil** is specified with a value of **t**, it returns **nil** if the user just presses RETURN, LINE, or END.

:number-or-nil The same as **(:number :or-nil t)**.

:decimal-number The same as **(:number :input-radix 10.)**.

:decimal-number-or-nil
The same as **(:number :input-radix 10. :or-nil t)**.

(**:date :past-p** *past-p-value* **:never-p** *never-p-value*)
Reads and returns a date, terminated by RETURN, LINE, or END. The returned date is a universal-time integer of the form returned by **time:parse-universal-time**. If **:past-p** is specified with a value of **t**, an ambiguous date is interpreted as being in the past; otherwise, it is interpreted as being in the future. If **:never-p** is specified with a value of **t**, it returns **nil** if the user types "never".

:past-date The same as **(:date :past-p t)**.

:date-or-never The same as **(:date :never-p t)**.

:past-date-or-never
The same as **(:date :past-p t :never-p t)**.

:time-interval-or-never
Reads a time interval, terminated by RETURN, LINE, or END. The interval must be either "never" or alternating numbers and units of time; the units can include seconds, minutes, hours, days, weeks, or years. It returns **nil** if the user types "never". Otherwise, it returns an integer representing the number of seconds in the time interval.

Example:

```
(prompt-and-read ':time-interval-or-never)
Enter a time interval, or "never": 1 day 2 hrs 13 min =>
94380.
```

(**:pathname :default** *defaults* **:default-version** *version*)
Reads a pathname, terminated by RETURN, LINE, or END, merging it with defaults. If **:default** is not specified, the defaults are the value of **fs:*default-pathname-defaults***. If **:default** is specified, its value should be suitable as the second argument to **fs:merge-pathnames**: a pathname, a pathname string, or an alist of hosts and pathnames of the sort that is the value of

fs:*default-pathname-defaults*. If **:default-version** is not specified, the default version is **:newest**. If **:default-version** is specified, its value should be an integer or keyword suitable as the third argument to **fs:merge-pathnames**. It returns the merged pathname.

Example:

```
(prompt-and-read '(:pathname :default ,my-defaults-alist)
                 "Enter a name (default is ~A) "
                 (fs:default-pathname my-defaults-alist))
```

(:pathname-or-nil :default *defaults* **:default-version** *version*)
Like **:pathname**, except that the returned value depends on what the user types:

User types	*Returned value*
A string	Merged pathname
RETURN or LINE only	Default pathname
END only	**nil**

:pathname-list Reads a series of pathnames, separated by commas and terminated by RETURN, LINE, or END. Merges the pathnames with the defaults in **fs:*default-pathname-defaults*** and with a default version of **:newest**. It returns a list of the merged pathnames.

:string Reads a string terminated by RETURN, LINE, or END. It returns the empty string when the string is empty.

:string-or-nil Reads a string terminated by RETURN, LINE, or END. It trims any leading or trailing white space. It returns **nil** when the string is empty.

:string-trim Reads a string terminated by RETURN, LINE, or END. It trims any leading or trailing white space. It returns the empty string when the string is empty.

:string-list Reads a series of strings separated by commas and terminated by RETURN, LINE, or END. It returns a list of the strings.

(:delimited-string :delimiter (*delimiter-1 delimiter-2 ...*) **:buffer-size** *size*)
Reads characters until one of the delimiters is typed. The delimiters are set up as activation characters. If no delimiters are specified, the default is **#\end**. If **:buffer-size** is specified, an initial buffer of size *size* characters is allocated; otherwise, the initial size is 100. characters. It returns the empty string when the string is empty.

(**:delimited-string-or-nil :delimiter** (*delimiter-1 delimiter-2...*) **:buffer-size** *size*)
Like **:delimited-string**, except that it returns **nil** when the string is empty.

(**:host :default** *default*)
Reads the name of a network host, terminated by RETURN, LINE, or END. If **:default** is specified, it should be the name of a host as a symbol or string. If **:default** is specified and the user just presses RETURN, LINE, or END, it returns the host specified by **:default**. Otherwise, it returns the host whose name the user types.

(**:host-list :chaos-only** *chaos-only*)
Reads a series of names of network hosts, separated by spaces or commas, and terminated by RETURN, LINE, or END. If **:chaos-only** is not **nil**, each host must have a Chaos address. It returns a list of the hosts whose names the user types.

(**:pathname-host :default** *default*)
Reads the name of a pathname host, terminated by RETURN, LINE, or END. The name can be "local", "sys", or the name of another logical host as well as the name of a physical host. If **:default** is specified, it should be the name of a pathname host as a string. If **:default** is specified and the user just presses RETURN, LINE, or END, it returns the host specified by **:default**. Otherwise, it returns the host whose name the user types.

:keyword-list Reads a series of names of symbols to be interned in the **keyword** package, separated by spaces or commas, and terminated by RETURN, LINE, or END. The symbol names should not have package prefixes (that is, they should not be preceded by colons). It returns a list of keyword symbols whose names the user types.

:font-list Reads a series of names of fonts, separated by spaces or commas, and terminated by RETURN, LINE, or END. The font names should not have package prefixes (that is, they should not be preceded by **fonts:**), and they must be names of known fonts. It returns a list of fonts whose names the user types.

Streams are permitted to have a handler for **:prompt-and-read** messages. The **prompt-and-read** function first determines whether the **query-io** stream handles the **:prompt-and-read** message. If so, it sends a **:prompt-and-read** message with its own arguments on to the stream. The stream returns several values. The first value the stream returns says

whether or not it wants to handle the interaction with the user itself. It returns **nil** to indicate that it declines to handle the message, in which case the **prompt-and-read** function continues its normal action of prompting the user. When the first value is not **nil**, the **prompt-and-read** function returns the rest of the values to its caller.

2.3.18 Previously Undocumented Feature: Coroutine Streams

Coroutine streams are a means of using output from one function as input to another, and vice versa. Functions are provided that construct two coroutine streams, each associated with a separate stack group but sharing a common I/O buffer.

Functions that produce data as output (output functions) are written in terms of **:tyo** and other output operations. Functions that receive data as input (input functions) are written in terms of **:tyi** and other input operations. Output functions operate on output streams, which handle the **:tyo** message. Input functions operate on input streams, which handle the **:tyi** message. Sometimes it is desirable to view an output function as an input stream, or an input function as an output stream. You can do this with coroutine streams.

Here is a simplified explanation of how coroutine streams work. A coroutine input stream can be built out of an output function. Whenever that stream receives a **:tyi** message, it invokes the output function in a separate stack group so that the function can produce the data that the **:tyi** message returns. A coroutine output stream can be built out of an input function; it works in the opposite fashion. Whenever the output stream receives a **:tyo** message, it invokes the input function in a separate stack group so that the function can receive the data transmitted by the **:tyo** message. It is also possible to connect functions that do both input and output, by using bidirectional coroutine streams. Since you can use coroutine streams to connect two functions, they are the logical inverse of **stream-copy-until-eof**, a function used to connect two streams.

To create a coroutine stream, use one of three functions. If you want to make an input stream from an output function, use **si:make-coroutine-input-stream**. If you want to make an output stream to an input function, use **si:make-coroutine-output-stream**. If you want to make a bidirectional stream for a function that does both input and output, use **si:make-coroutine-bidirectional-stream**.

Following is an example using a coroutine input stream:

```
(setq input-stream
      (si:make-coroutine-input-stream
        #'(lambda (stream) (print-disk-label 0 stream))))

(send input-stream ':line-in) →
  "1645 free, 260499//262144 used (99%)"
```

Following is an example using a coroutine output stream:

```
(setq output-stream
      (si:make-coroutine-output-stream
        #'(lambda (stream) (setq x (read stream)))))

(send output-stream ':string-out "(a b c)")

(send output-stream ':force-output)

x → (A B C)
```

Coroutine streams are implemented as buffered character streams. Each function that makes a coroutine stream actually creates two streams and one new stack group. One stream is associated with the new stack group and the other stream with the stack group that is current when the stream-making function is called. If you use **si:make-coroutine-input-stream** or **si:make-coroutine-output-stream**, one stream is an input stream and the other is an output stream; they share a common buffer. If you use **si:make-coroutine-bidirectional-stream**, both streams are bidirectional; the input buffer of each stream is the output buffer of the other.

With **si:make-coroutine-input-stream**, the output function runs in the new stack group. With **si:make-coroutine-output-stream**, the input function runs in the new stack group. With bidirectional streams, the function that does input or output runs in the new stack group.

In the case of **si:make-coroutine-input-stream**, for example, you typically send **:tyi** messages to the input stream that **si:make-coroutine-input-stream** returns. The input stream is associated with the new stack group. When the input stream receives a **:tyi** message, the new stack group is resumed, and the output function runs in that stack group. The output function typically sends **:tyo** messages to the output stream associated with the stack group from which **si:make-coroutine-input-stream** was called. When the output stream receives a **:tyo** message, the associated stack group is resumed. The data transmitted to the output stream become input to **:tyi** via the buffer that the two streams share. **si:make-coroutine-output-stream** and **si:make-coroutine-bidirectional-stream** work in analogous fashion.

In addition to **:tyi** and **:tyo**, coroutine streams support other standard input and output operations, such as **:line-in** and **:string-out**. Actually, the **:next-input-buffer** method of the input stream and the **:send-output-buffer** method of the output stream resume the new stack group, not the receipt of **:tyi** and **:tyo** messages. Because the streams are buffered, you must send a **:force-output** message to an output stream to cause the new stack group to be resumed.

The instantiable flavors of coroutine streams are **si:coroutine-input-stream**, **si:coroutine-output-stream**, and **si:coroutine-bidirectional-stream**.

Do not confuse coroutine streams with pipes. Coroutine streams are used for intraprocess communication; pipes are used for interprocess communication. The Lisp Machine does not currently support pipes.

si:make-coroutine-input-stream *function* &rest *arguments* *Function*

Creates two coroutine streams, an input stream and an output stream, with a shared buffer. **si:make-coroutine-input-stream** returns the input stream. The input stream is associated with a new stack group and the output stream with the stack group that is current when **si:make-coroutine-input-stream** is called. **:tyi** messages to the input stream cause the new stack group to be resumed and *function* to be called from that stack group. The first argument to *function* is the output stream; any additional arguments come from *arguments*. *function* should send **:tyo** messages to the output stream. These messages resume the stack group in which **si:make-coroutine-input-stream** was called. In this way, output from *function* becomes input to the caller of **si:make-coroutine-input-stream** through the shared buffer.

si:make-coroutine-output-stream *function* &rest *arguments* *Function*

Creates two coroutine streams, an output stream and an input stream, with a shared buffer. **si:make-coroutine-output-stream** returns the output stream. The output stream is associated with a new stack group and the input stream with the stack group that is current when **si:make-coroutine-output-stream** is called. **:tyo** messages to the output stream cause the new stack group to be resumed and *function* to be called from that stack group. The first argument to *function* is the input stream; any additional arguments come from *arguments*. *function* should send **:tyi** messages to the input stream. These messages resume the stack group in which **si:make-coroutine-output-stream** was called. In this way, output from the caller of **si:make-coroutine-output-stream** becomes input to *function* through the shared buffer.

si:make-coroutine-bidirectional-stream *function* &rest *arguments* *Function*

Creates two bidirectional coroutine streams. The input buffer of each stream is the output buffer of the other. One stream is associated with a new stack group and the other with the stack group that is current when **si:make-coroutine-bidirectional-stream** is called. **si:make-coroutine-bidirectional-stream** returns the stream associated with the new stack group.

:tyi and **:tyo** messages to the stream associated with the new stack group cause that stack group to be resumed and *function* to be called from that stack group. The first argument to *function* is the stream associated with the stack group from which **si:make-coroutine-bidirectional-stream** was called. Any additional arguments come from *arguments*. *function* should send **:tyi** or **:tyo** messages to the stream that is its first argument. These

messages resume the stack group in which **si:make-coroutine-output-stream** was called. In this way *function* and the caller of **si:make-coroutine-bidirectional-stream** communicate through the shared buffers; output from one function becomes input to the other.

si:coroutine-input-stream *Flavor*

Coroutine input stream. Defines a **:next-input-buffer** method. Use this to construct an input stream from a function written in terms of output operations.

si:coroutine-output-stream *Flavor*

Coroutine output stream. Defines **:new-output-buffer** and **:send-output-buffer** methods. Use this to construct an output stream to a function written in terms of input operations.

si:coroutine-bidirectional-stream *Flavor*

Bidirectional coroutine stream. Defines **:next-input-buffer, :new-output-buffer,** and **:send-output-buffer** methods. Use this to construct a bidirectional stream to a function written in terms of input and output operations.

2.3.19 format ~\ directives can have package prefixes

Format directives enclosed in backslashes can now have package prefixes. If they have none, they refer to symbols in the **format** package.

See the special form **format:defformat**.

2.3.20 Wildcard Directory Mapping Available

True wildcard mapping of directories is now supported. This facility is used by functions that copy and rename files. It allows you to copy or rename entire subtrees.

The rules for mapping directory components between two wildcard pathnames and a starting instance are parallel to the rules for single names. Directory level components play roughly the roles of characters in the name-translating algorithm. See the section "Wildcard Pathname Mapping".

Consider a directory component as a sequence of directory level components. The levels are separated by level delimiters (> in LMFS). Example: in the pathname >foo>bar>*>mumble*>x>**>y>a.b.3, the directory level components are foo, bar, *, mumble*, x, **, and y. The source and target patterns, as well as the starting instance, are considered as sequences of directory level components, and are matched and translated level by level.

For this purpose, each directory level component may be classified as one of three types:

Type	*Directory representation*
constant	String containing no *'s
wild-inferiors	** in LMFS, ... in VMS
must-match	* or string containing at least one * (but not the string representing wild-inferiors)

The matching and mapping of constant and wild-inferiors levels proceeds in a manner identical to the matching and mapping of constant substrings and *'s for single names. See the section "Wildcard Pathname Mapping". Constant directory level components act as constant substrings in that algorithm, and wild-inferiors levels as *'s. That is, wild-inferiors level components match and, on the target side, carry, zero to any number of constant directory level components.
Examples:

```
Source pattern:      >sys>**>*.*.newest
Target pattern:      >old-systems>release-5>**>*.*.*
Starting instance:   >sys>lmfs>patch>lmfs-33.patch-dir.66
Target instance:     >old-systems>release-5>lmfs>patch>lmfs-33.patch-dir.66

Source pattern:      >a>b>c>**>d>e>**>x.y.*
Target pattern:      >t>u>**>m>**>w>*.*.*
Starting instance:   >a>b>c>p>q>d>e>f>g>x.y.1
Target instance:     >t>u>p>q>m>f>g>w>x.y.1
```

Must-match components are matched with exactly one directory level component, which must be present. They are mapped according to the string-mapping rules in the name-translating algorithm. See the section "Wildcard Pathname Mapping".

Example:

```
Source pattern:      >a>b>c>foo*>d>*>*.*.*
Target pattern:      >x>*bar>y>*man>*.*.*
Starting instance:   >a>b>c>foolish>d>yow>a.lisp.1
Target instance:     >x>lishbar>y>yowman>a.lisp.1
```

You may intersperse constants, must-matches, and wild-inferiors directory level components, as long as the sequence of wildcard types is the same in both patterns.

Example:

```
Source pattern:      >a>*>c>**>*.lisp.*
Target pattern:      >bsg>sub>new-*>q>**>*.*.*
Starting instance:   >a>bb>c>d>e>p1.lisp.6
```

Target instance: >bsg>sub>new-bb>q>d>e>p1.lisp.6

2.3.21 Previously undocumented function: describe-system

describe-system is a useful function for finding information about a system.

describe-system *system-name* &key *(show-files* **t***)* *Function*
(show-transformations **t***)*

Displays useful information about the system named *system-name*. This includes the name of the system source file, the system package default if any, and component systems. For a patchable system, **describe-system** displays the system version and status, a typical patch file name, the sites maintaining the system, and, if the user wants, a listing of patches. If **:show-files** is **t**, it displays the history of the files in the system. Other possible values are **nil** (do not show file history) and **:ask** (ask the user). If **:show-transformations** is **t**, it displays the transformations required to make the system. Other possible values are **nil** (do not display transformations) and **:ask** (ask the user).

2.3.22 Improvements to make-system: error-restart, selective transformations

make-system now has an **error-restart** that reinvokes the **make-system** itself. It probes for files that have changed since the **make-system** was started.

make-system also has a new possible answer, "S" (selective), to its request for confirmation of the list of transformations to be performed. If the user answers "S", **make-system** proceeds as if the **:selective** option had been specified, asking for confirmation of each individual transformation.

See the function **make-system**.

2.3.23 Second argument to si:install-microcode now optional

The second argument to **si:install-microcode**, *to-file-or-version*, is now optional. This argument defaults to a file on FEP:> and rarely needs to be supplied.

si:install-microcode *from-file-or-version* &optional *to-file-or-version* *Function*
(3600 only) Installs microcode from a system file into a file in the FEP file system.

from-file-or-version is a microcode version number (in decimal). The file resides in the logical directory sys:l-ucode;.

to-file-or-version rarely needs to be supplied. It defaults to a file on FEP:> (the root directory of the boot disk) whose name is based on the microcode name and version. If supplied, *to-file-or-version* is either a pathname (string) of a file on FEP:>, or an integer *n*, which stands for the file TMC5-MIC.MIC.n on FEP:>.

2.3.24 Change in argument to process-wait-with-timeout

The *interval* argument to **process-wait-with-timeout** can now be **nil**. If so, **process-wait-with-timeout** waits indefinitely for the application of *function* to *arguments* to return something other than **nil**.

process-wait-with-timeout *whostate time function* &rest *args* *Function*

This is a primitive for waiting. It applies *function* to *arguments* until the function returns something other than **nil** or until the interval times out. *interval* is a time in 60ths of a second. When the process times out, **process-wait-with-timeout** returns **nil**. When the function returns something other than **nil** within the interval, **process-wait-with-timeout** returns **t**.

If *interval* is **nil**, **process-wait-with-timeout** waits indefinitely for the application of *function* to *arguments* to return something other than **nil**. This behavior is the same as that of **process-wait**.

2.3.25 New option for si:sb-on: :mouse (3600 only)

The **:mouse** option for **si:sb-on** on the 3600 causes sequence breaks when the mouse moves. This option is on by default.

si:sb-on &optional *when* *Function*

si:sb-on controls what events cause a sequence break, that is, when rescheduling occurs. The following keywords are names of events that can cause a sequence break.

:clock This event happens periodically based on a clock. The default period is one second. For the 3600, the period is the value of the variable **si:sequence-break-interval**, an integer representing the number of microseconds in the period (default 1000000.). For the LM-2, see the meter **sys:%tv-clock-rate**. This event is enabled by default.

:disk (3600 only) A sequence break happens whenever the disk hardware/firmware decides to wake up the wired disk system. This might occur with every disk I/O operation or after several have been completed. This event is always enabled; you cannot turn it off. However, these sequence breaks do not cause rescheduling.

:mouse (3600 only) Happens when the mouse moves. Sixty times per second it tests the variable **tv:mouse-wakeup**, which is set by the FEP. Causes a sequence break if the value is not **nil**. This event is enabled by default.

:unibus (LM-2 only) Happens when a character is received from the keyboard. Actually, a sequence break happens

whenever input is received from any UNIBUS channel that has a flag bit set. This event is disabled by default.

:keyboard (LM-2 only) Same as **:unibus**.

:chaos (LM-2 only) Happens when a packet is received from the Chaosnet, or transmission of a packet to the Chaosnet is completed. This event is disabled by default.

Since the keyboard and Chaosnet are heavily buffered, there is no particular advantage to enabling the **:keyboard** and **:chaos** events, unless the **:clock** event is disabled.

With no argument, **si:sb-on** returns a list of keywords for the currently enabled events.

With an argument, the set of enabled events is changed. The argument can be a keyword, a list of keywords, **nil** (which disables sequence breaks entirely since it is the empty list), or a number that is the internal mask, not documented here.

2.3.26 New format for trace output

The default format for **trace** output has changed.

Example of the old style:

```
(1 ENTER FACT (4.))
  (2 ENTER FACT (3.))
    (3 ENTER FACT (2.))
      (4 ENTER FACT (1.))
        (5 ENTER FACT (0.))
        (5 EXIT FACT 1.)
      (4 EXIT FACT 1.)
    (3 EXIT FACT 2.)
  (2 EXIT FACT 6.)
(1 EXIT FACT 24.)
```

Example of the new style:

```
1 Enter FACT 4.
| 2 Enter FACT 3.
|   3 Enter FACT 2.
|   | 4 Enter FACT 1.
|   |   5 Enter FACT 0.
|   |   5 Exit FACT 1.
|   | 4 Exit FACT 1.
|   3 Exit FACT 2.
| 2 Exit FACT 6.
1 Exit FACT 24.
```

You can use the variables **si:*trace-columns-per-level***, **si:*trace-bar-p***, **si:*trace-bar-rate***, and **si:*trace-old-style*** to control the format of **trace** output.

si:*trace-columns-per-level* *Variable*

For **trace** output, controls the number of columns of indentation that are added for each level of function call. The value must be an integer. The default is 2.

si:*trace-bar-p* *Variable*

For **trace** output, controls whether columns of vertical bars are printed. If the value is not **nil**, they are printed; otherwise, spaces are printed instead of the vertical bars. The default is **t** (print the bars).

si:*trace-bar-rate* *Variable*

When **si:*trace-bar-p*** is not **nil**, columns of vertical bars are printed in **trace** output for every *n* levels of function call, where *n* is the value. The value must be an integer. The default is 2.

si:*trace-old-style* *Variable*

If not **nil**, the old, Maclisp-compatible form of printing **trace** output is used. The default is **nil** (use the new style).

2.3.27 Recursion in Bound and Default Handlers Eliminated

Condition handlers bound by **condition-bind**, **condition-bind-default**, and related special forms no longer cause infinite recursion when they signal the same condition they are handling.

While a bound or default handler is executing, that handler and all handlers inside it are removed from the list of bound or default handlers. This is to prevent infinite recursion when a handler signals the same condition that it is handling, as in the following simplistic example:

```
(condition-bind ((error '(lambda (x) (ferror "foo"))))
  (ferror "foo"))
```

If you want recursion, the handler should bind its own condition.

2.3.28 :proceed methods can now return nil

It is no longer an error for a **:proceed** method to return **nil** as its first value.

A **:proceed** method can return a first value of **nil** if it declines to proceed from the condition. If a **nil** returned by a **:proceed** method becomes the return value for a **condition-bind** handler, this signifies that the handler has declined to handle the condition, and the condition continues to be signalled. When the **:proceed** message was sent by the Debugger, the Debugger prints a message saying that the condition was not proceeded, and it returns to its command level. This might be used by an interactive **:proceed** method that gives the user the opportunity either to proceed or to abort; if the user aborts, the method returns **nil**. Returning **nil** from a **:proceed** method should not be used as a substitute for detecting earlier (such as

when the condition object is created) that the proceed type is inappropriate for that condition.

2.3.29 New clause for condition-call: :no-error

condition-call and **condition-call-if**, like **condition-case**, can now take a **:no-error** clause as the final clause.

As a special case, *predicate-m* (the last one) can be the special symbol **:no-error**. If *form* is evaluated and no error is signalled during the evaluation, **condition-case** executes the **:no-error** clause instead of returning the values returned by *form*. The variables *vars* are bound to the values produced by *form*, in the style of **multiple-value-bind**, so that they can be accessed by the body of the **:no-error** case. Any extra variables are bound to **nil**.

2.3.30 New message to arithmetic errors: :operands

All arithmetic errors (built on **sys:arithmetic-error**) now handle the **:operands** message. On the 3600, this returns a list of the operands in the operation that caused the error. On the LM-2, this message nearly always returns **nil**.

See the flavor **sys:arithmetic-error**.

2.3.31 Change in Debugger special command for fs:directory-not-found

The condition flavor **fs:directory-not-found** now has two Debugger special commands: **:create-directory**, to create only the lowest level of directory, and **:create-directories-recursively**, to create any missing superiors as well.

fs:directory-not-found *Flavor*

The directory of the file was not found or does not exist. This means that the containing directory was not found. If you are trying to open a directory, and the actual directory you are trying to open is not found, **fs:file-not-found** is signalled. This flavor is built on **fs:file-lookup-error**.

This flavor has two Debugger special commands: **:create-directory**, to create only the lowest level of directory, and **:create-directories-recursively**, to create any missing superiors as well.

2.3.32 New optional argument to gc-immediately

gc-immediately now takes an optional argument, **nil** by default. If it is not **nil**, **gc-immediately** does garbage collection without querying, regardless of how much space is left.

gc-immediately &optional *no-query* *Function*

gc-immediately does nonincremental garbage collection, taking less space and less total time than an incremental gc, but running continuously in the

process calling it, until the garbage collection is complete. The main advantage of this compared to incremental gc is that it requires less free space and hence can succeed where an incremental gc would fail because virtual memory was too full.

If *no-query* is not **nil**, **gc-immediately** commences garbage collection without asking any questions, regardless of how much space is available.

You should call this rather than **si:full-gc** (unless you are compressing a band). The difference is that **gc-immediately** does not lock out other processes, does not run various **full-gc** initializations, and does not affect the static areas.

Suppose garbage collection has already started, that the flip has occurred but not all good data have been copied out of old space. **gc-immediately** then copies the rest of the good data but does not flip again.

2.3.33 New optional arguments to print-notifications

print-notifications now takes optional arguments that allow you to print only part of the notification history.

print-notifications &optional (*from* **0**) (*to* **(1- (length tv:notification-history))**) *Function*

Reprints any notifications that have been received. The difference between notifications and sends is that sends come from other users, while notifications are asynchronous messages from the Lisp Machine system itself. If *from* or *to* is specified, prints only part of the notifications list.

Example: `(print-notifications 0 4)` prints the five most recent notifications.

2.3.34 :draw-filled-in-circle uses same algorithm as :draw-circle

Previously, the **:draw-filled-in-circle** method of **tv:graphics-mixin** used a different algorithm from that of the **:draw-circle** method. This algorithm was slower and sometimes resulted in inaccurate displays when the two methods were used together. **:draw-filled-in-circle** now uses the same algorithm as **:draw-circle**.

:draw-circle *center-x center-y radius* &optional *alu* of **tv:graphics-mixin** *Method*

Draw the outline of a circle specified by its center and radius.

:draw-filled-in-circle *center-x center-y radius* &optional *alu* of **tv:graphics-mixin** *Method*

Draw a filled-in circle specified by its center and radius.

2.3.35 Previously undocumented variables: sys:mouse-x-scale-array and sys:mouse-y-scale-array (LM-2 only)

sys:mouse-x-scale-array and **sys:mouse-y-scale-array** are variables on the LM-2 whose values are arrays used in mouse scaling. These arrays determine the relation between motion of the mouse on the table and motion of the mouse cursor on the screen. That relation can vary with the speed of the mouse. You can use these variables to speed up or slow down the motion of the mouse cursor caused by corresponding motion of the mouse.

sys:mouse-x-scale-array *Variable*

(LM-2 only) The value of this variable is an array that, along with the array that is the value of **sys:mouse-y-scale-array**, can be used to control mouse scaling. These arrays determine the relation between the rates of motion of the mouse on the table and the mouse cursor on the screen. This relation can be nonlinear and can vary with the speed of the mouse. For example, fast mouse motion can move the cursor a distance that is proportionally greater than slow mouse motion.

Scaling is computed as follows. The even-numbered elements of **sys:mouse-x-scale-array** are compared with the value of **sys:mouse-x-speed**, and the even-numbered elements of **sys:mouse-y-scale-array** are compared with the value of **sys:mouse-y-speed**. **sys:mouse-x-speed** and **sys:mouse-y-speed** are the x- and y-components of the mouse speed on the table, typically in units of hundredths of an inch per second.

For each array, the first even array element that is greater than the mouse speed causes its corresponding odd-numbered array element to be multiplied by the mouse motion on the table and then divided by 1024 (decimal). The result is the mouse motion on the screen. Appropriate care is taken to save the fractions for the next computation.

The default array setup code is as follows:

```
;; Set the X scale to 2/3 and the Y scale to 3/5.
;; Disable speed-dependent scaling.
(aset #o37777777 sys:mouse-x-scale-array 0)
(aset (// (lsh 2 10.) 3) sys:mouse-x-scale-array 1)
(aset #o37777777 sys:mouse-y-scale-array 0)
(aset (// (lsh 3 10.) 5) sys:mouse-y-scale-array 1)
```

The following code provides for simple scaling of motion for the Hawley mouse. The microcode knows specially about each array. You may store into each array, but you may not replace it with a new array or use **adjust-array-size** on it.

```
;;; Aids to trying speed-dependent scaling
;;; Specs are scale-factor speed-break
;;; No attempt to treat X and Y differently
;;; Args of (1 80. 2) seem to be about right for the Hawley mouse
(defun mouse-speed-hack (&rest specs)
  (loop for (scale speed) on specs by 'cddr
        for i from 0 by 2
        do (aset (or speed #o37777777) sys:mouse-x-scale-array i)
        (aset (or speed #o37777777) sys:mouse-y-scale-array i)
        (aset (// (fix (* 2 scale 1024.)) 3)
              sys:mouse-x-scale-array (1+ i))
        (aset (// (fix (* 3 scale 1024.)) 5)
              sys:mouse-y-scale-array (1+ i))))

(defun hawley-mouse-hack ()
  (mouse-speed-hack 1 80. 2))
```

The corresponding variables **tv:mouse-x-scale-array** and **tv:mouse-y-scale-array** exist on the 3600, but in this release they have no effect.

sys:mouse-y-scale-array *Variable*

(LM-2 only) The value of this variable is an array that, along with the array that is the value of **sys:mouse-x-scale-array**, can be used to control mouse scaling. See the variable **sys:mouse-x-scale-array**.

2.3.36 New optional argument to tv:mouse-wait

tv:mouse-wait now takes another optional argument, a string to be displayed in the status line while waiting for the status of the mouse to change.

tv:mouse-wait &optional (*old-mouse-x* **tv:mouse-x**) *Function*
(*old-mouse-y* **tv:mouse-y**)
(*old-mouse-buttons* **tv:mouse-last-buttons**)
(*whostate* **"Mouse"**)

This function waits for any of the variables **tv:mouse-x**, **tv:mouse-y**, or **tv:mouse-last-buttons** to become different from the values passed as arguments. While waiting, *whostate* is displayed in the status line. To avoid timing errors, your program should examine the values of the variables, use them, and then pass in the values that it examined as arguments to **tv:mouse-wait** when it is done using the values and wants to wait for them to change again. It is important to do things in this order, or else you might fail to wake up if one of the variables changed while you were using the old values and before you called **tv:mouse-wait**.

2.3.37 New flavors: tv:truncatable-lines-mixin, tv:truncating-lines-mixin

tv:truncatable-lines-mixin causes text to be truncated at the right edge of the window, but only if the window's "truncate line out" flag is set. **tv:truncating-lines-mixin** initializes this flag to on so that truncation actually happens. These flavors replace the obsolete flavor **tv:line-truncating-mixin**, which, despite its name, did not initialize the "truncate line out" flag to on. You can use the new messages **:truncate-line-out** and **:set-truncate-line-out** to read and set this flag. The flavor **tv:truncating-window** is now built on **tv:truncating-lines-mixin**.

tv:truncatable-lines-mixin *Flavor*

If you mix in this flavor and the window's *truncate line out* flag is on, typeout does not wrap around when lines are too long. That is, when the cursor is near the right-hand edge of the window and an attempt is made to type out a character, the character is not typed out; text is truncated at the edge of the window. When the truncate line out flag is turned off, this flavor has no effect.

tv:truncating-lines-mixin *Flavor*

When this flavor is mixed in, lines of output that are too long to fit inside the window do not wrap around but are truncated at the edge of the window. This flavor is built on **tv:truncatable-lines-mixin**. It initializes the window's truncate line out flag to be on.

tv:truncating-window *Flavor*

This flavor is built on **tv:window** with **tv:truncating-lines-mixin** mixed in. If you instantiate a window of this flavor, it will be like regular windows of flavor **tv:window** except that lines will be truncated instead of wrapping around.

:truncate-line-out of **tv:sheet** *Method*

Returns **t** if the window's truncate line out flag is set, or **nil** if it is not.

:set-truncate-line-out *new-value* of **tv:sheet** *Method*

Sets the value of the window's truncate line out flag. If *new-value* is **t** the flag is turned on; if **nil**, it is turned off.

2.3.38 New variable: tv:*mouse-modifying-keystates*

In previous releases you could use the variables **tv:mouse-double-click-time** and **tv:*mouse-incrementing-keystates*** to replace double clicks with shifted clicks.

Now mouse characters — characters with the **%%kbd-mouse** bit set to 1 — can be modified with the modifier keys CONTROL, META, SUPER, and HYPER, just as keyboard characters can. Which of these keys modify mouse characters depends on the value of the variable **tv:*mouse-modifying-keystates***.

You can use **login-forms** in an init file to set the variables **tv:mouse-double-click-time**, **tv:*mouse-incrementing-keystates***, and **tv:*mouse-modifying-keystates*** and customize the behavior of the mouse.

tv:mouse-double-click-time *Variable*

The maximum period of time (in microseconds) between mouse clicks for which the clicks are interpreted as a double click instead of two single clicks. Default: **200000** (decimal). If you set this to **nil**, disabling double clicking entirely, mouse response time improves slightly.

tv:*mouse-incrementing-keystates* *Variable*

A list of names of keys, acceptable to **tv:key-state**. If one or more of these keys are pressed, single mouse clicks are interpreted as double clicks. Default: **(:shift)**.

tv:*mouse-modifying-keystates* *Variable*

A list of names of keys, acceptable to **tv:key-state**. If one or more of these keys are pressed, sets the corresponding modifier bits in the mouse character. Default: **(:control :meta :super :hyper)**. If a key appears as an element of both this list and the list that is the value of **tv:*mouse-incrementing-keystates***, the modifier bit is set and the click is interpreted as a double click.

2.3.39 Shifted Mouse Clicks Can Now Be Used for Editor Commands

Mouse characters can now be modified with modifier keys. See the variable **tv:*mouse-modifying-keystates***. The editor considers each modified mouse click to be a separate command. You can bind commands to particular modified mouse clicks. You can also use Install Mouse Macro (m-X) with modified mouse clicks to increase the number of mouse macros available.

You can put a form in an init file to install a Zwei command on a modified mouse click. Note that in the following example, the mouse click marks the paragraph that surrounds point, not the paragraph under the mouse cursor:

```
(login-forms
  zwei:(set-comtab
         *standard-comtab*                    ;in standard command table
         '(#\Hyper-Mouse-L com-mark-paragraph)
         ))
```

2.3.40 Previously undocumented functions: tv:add-to-system-menu-programs-column, tv:add-to-system-menu-create-menu

tv:add-to-system-menu-programs-column lets you add an entry to the Programs column of the system menu. **tv:add-to-system-menu-create-menu** lets you add an entry to the menu that appears when you click on [Create] in the system menu or in the Edit Screen menu.

tv:add-to-system-menu-programs-column *name form documentation* &optional *after* *Function*

Adds a program to the Programs column of the system menu. *name* is a string, the name to appear in the menu. *form* is a form to evaluate, in its own process, when the program is selected; often this is a call to **tv:select-or-create-window-of-flavor**. *documentation* is mouse documentation for the menu item. *after* determines the position of the new program name in the Programs column:

nil	Bottom of the column
t	Top of the column
string	After the program named *string* that is now in the menu

Example:

```
(tv:add-to-system-menu-programs-column
  "Hardcopy" '(press:hardcopy-via-menus nil t)
  "Print files on the hardcopy printer")
```

tv:add-to-system-menu-create-menu *name flavor documentation* &optional *after* *Function*

Adds an entry to the menu that appears when you click on [Create] in the system menu or in the Edit Screen menu. *name* is a string, the name of the menu item. *flavor*, a flavor name, is the flavor of window that is created when the menu item is selected. *documentation* is mouse documentation for the menu item. *after* determines where in the [Create] menu the item should appear:

nil	Bottom of the menu
t	Top of the menu
string	After the item named *string* that is now in the menu

Example:

```
(tv:add-to-system-menu-create-menu
  "Concept Editor" 'crl:concept-editor
  "Edit the representation of a concept in the CRL system")
```

tv:select-or-create-window-of-flavor *find-flavor* &optional *(create-flavor find-flavor)* *Function*

Selects the most recently selected window of flavor *find-flavor*. If no window of that flavor exists, makes a window of flavor *create-flavor* and selects it.

2.3.41 Argument to :menu type menu items can be a menu or a form

The "argument" to **:menu** type menu items — the specifier for the submenu — can now be a menu or a form. Previously it could be a menu or a symbol.

2.3.42 Clicking Middle Edits Current String in Choose-variable-values Windows

In a choose-variable-values window, clicking middle on a string that is displayed as a value now lets you edit that string.

2.3.43 tv:scroll-maintain-list init function can take arguments

Previously, the init function specified as an argument to **tv:scroll-maintain-list** was itself called with no arguments. Now you can use an optional &rest argument to **tv:scroll-maintain-list** to specify arguments to be passed to the init function at redisplay time.

tv:scroll-maintain-list *init-fun item-fun* &optional *per-element-fun stepper-fun compact-p pre-proc-fun* &rest *init-args* *Function*

Constructs and returns a list item that updates itself when the scroll window is asked to redisplay. Takes the following arguments:

init-fun — The init function that will be called at redisplay time to provide a representation of the set of objects to be displayed.

init-args — Arguments to be passed to *init-fun* when called at redisplay time.

item-fun — The item function, to be applied to each object of yours to produce a display item.

per-element-fun — A function to be put in the list item plist of the list item as the **:function** function.

stepper-fun — The function that is called on the set of objects and all "rest"s of the set. It is expected to return three values: the next element, the "rest" of the set, and **t** if it has returned the last element of the set. If not given, *stepper-fun* defaults to **tv:scroll-maintain-list-stepper**, a function that handles ordinary lists.

compact-p — An optional flag that causes **tv:scroll-maintain-list** to copy the list it builds at each redisplay into a special area for such lists, in order to optimize paging performance. The list so constructed will be stored in compact (that is, cdr-coded) form.

pre-proc-fun — A function to be put in the list item plist of the list item

as the **:pre-process-function** function. If not given, *pre-proc-fun* defaults to **tv:scroll-maintain-list-update-function**.

Following is a simple example:

```
(tv:scroll-maintain-list #'(lambda (instance)     ;The init function
                             (send instance ':value-list))
                         #'(lambda (value)      ;The item function
                             (tv:scroll-parse-item
                               '(:string ,(format nil "~S" value))))
                         nil nil nil nil
                         self)                  ;Argument to init function
```

3. Changes to Networks in Release 5.0

3.1 Incompatible Changes to Networks in Release 5.0

3.1.1 Network Namespace System

Release 5.0 implements a new network database. This database is a collection of objects known to networks, such as hosts, users, networks, printers, sites, and classes of objects. Objects of different classes can have the same names. To eliminate naming conflicts when different sites are linked by long-distance networks, the database is divided into namespaces, or mappings of names of objects to objects.

The database is maintained by database servers. A namespace editor exists to add objects to the database or change their properties. The user interface to the editor is the function **tv:edit-namespace-object**. The system also includes means of defining protocols and media, and defining and invoking network services.

For more information on the changes to networks in Release 5.0: See the section "Network Database". See the section "The Lisp Machine Generic Network System". See the section "Interfacing to the Network System".

3.1.2 chaos:stream, chaos:close, and chaos:finish renamed

The following functions have been renamed in the Chaosnet implementation.

All the known places in the system that use these have been updated. The old function names are still shadowed in the **chaos:** package in order to cause undefined function errors during compilation instead of calling an incompatible function.

Old name	*New name*
chaos:stream	**chaos:make-stream**
chaos:close	**chaos:close-conn**
chaos:finish	**chaos:finish-conn**

3.1.3 neti:reset, neti:enable, and neti:disable replace chaos:reset, chaos:enable, and chaos:disable

The functions **chaos:reset** and **chaos:enable** have been replaced with **neti:reset** and **neti:enable**, which reset and enable the entire network system. If you call **neti:reset** and then want to turn the network back on, you must now call **neti:enable** to do so. Formerly, **chaos:reset** turned the network back on after resetting it.

chaos:disable has been replaced by **neti:disable.**

neti:reset *Function*

Resets the local networks. Disables and then resets the interfaces. After using **neti:reset** you must call **neti:enable** if you want to turn the network back on.

neti:enable *Function*

Enables the local networks and interfaces.

neti:disable *Function*

Disables the local networks and interfaces. If you want to reset the local networks and interfaces and then turn them back on, you should call **neti:reset** and then **neti:enable.**

3.1.4 Changes to chaos:open-stream

In general, applications should use the service mechanism instead of Chaos-specific routines. See the section "The Lisp Machine Generic Network System".

The function **chaos:open-stream** accepts **nil** as the host argument to mean issue a *listen* for the contact name (as opposed to a request at a specific host). The stream that is returned is still in the *RFC Received* state. The following messages can be sent to the stream:

:foreign-host Returns the host object of the host requesting the connection.

:accept Accepts the connection.

:reject &optional *reason*
Rejects the connection with *reason.*

3.1.5 chaos:send-unc-pkt automatically returns the packet to the free pool

chaos:send-unc-pkt does an implicit **chaos:return-pkt,** which returns the packet to the free pool at the appropriate time. The user is not allowed to reuse this packet. This is an incompatible change.

The documentation for **chaos:send-pkt** in the *Chaosnet* document failed to mention that **chaos:send-pkt** automatically returns the packet via **chaos:return-pkt.** The code for this function remains unchanged; this is a clarification.

3.2 New Features in Networks in Release 5.0

3.2.1 New function: chaos:conn-finished-p

chaos:conn-finished-p *conn* *Function*

A predicate that returns something other than **nil** if all data that have been output have been received *and* acknowledged by the foreign side of the connection.

3.2.2 Changes to VMS Chaosnet

The following changes have been made to VMS Chaosnet:

- The NCP writes the CHNCP.GSF global section, which contains the connection database. The SHOWNCP utility displays the data in CHNCP.GSF.
- An NCP internal routing table server allows routing tables to be examined dynamically.
- The FILE server now supports pathname completion and directory creation.
- When transferring binary files to a VMS host, the files are written in the following way: If the file is a "QFASL" file, it is written as an RMS sequential file with variable-length records whose maximum size is 2048 bytes. Note that only the last record can be less than 2048 bytes long. If is not a "QFASL" file, it is written as an RMS sequential file with fixed-length, 512-byte records. A "QFASL" file is any file that has to remember its length in bytes exactly; some examples are Lisp Machine .BIN and .QBN files. An example of a non-"QFASL" binary file is a VMS executable file (.EXE).
- CFTP now supports GET and SEND /BINARY=*byte-size*. It also supports the GET/VAR feature. This tells CFTP, when transferring binary files, to write RMS sequential files with variable-length records whose maximum size is 2048 bytes. Note that only the last record can be less than 2048 bytes long. Without /VAR, CFTP writes RMS sequential files with fixed-length, 512-byte records.
- A TELNET user exists. You can invoke it with the DCL "TELNET" foreign command.

3.2.3 Changes to Serial I/O: Parity Recovery and Xon/Xoff Character Setting

Two features, serial parity recovery and XON/XOFF character setting, have been added to serial I/O. The new parameters (at either initialization or **:put**) are:

:input-error-character

The value is a character to be substituted for any input character in which a parity error is detected. This is independent of the

:check-parity-errors flag. If the value is **nil** (the default), the character is left alone.

:output-xoff-character
The value is a character that is used to control flow of data from the Lisp Machine to the external device. It is used to suspend the flow of data when the **:xon-xoff-protocol** parameter is set. The default is #o023.

:output-xon-character
The value is a character that is used to control flow of data from the Lisp Machine to the external device. It is used to resume the flow of data when the **:xon-xoff-protocol** parameter is set. The default is #o021.

:input-xoff-character
(3600 only) The value is a character that is used to control flow of data from the external device to the Lisp Machine. It is sent by the Lisp Machine to suspend the flow of data when the **:generate-xon-xoff** flag is set. The default is #o023.

:input-xon-character
(3600 only) The value is a character that is used to control flow of data from the external device to the Lisp Machine. It is sent by the Lisp Machine to resume the flow of data when the **:generate-xon-xoff** flag is set. The default is #o021.

3.2.4 Hdlc Serial I/O on the 3600

In Release 5.0, the 3600 supports synchronous serial I/O using HDLC-like bitstuffing protocols. The CCITT-16 CRC polynomial is used.

This facility requires that the 3600 be running with FEP version 14 or later. Also, some older 3600s might require that a special adapter cable be connected to serial port 1.

An HDLC stream is a stream of flavor **si:serial-hdlc-stream**. Use the function **si:make-serial-stream** to make one of these streams. HDLC streams accept **:read-frame** and **:write-frame** messages.

si:serial-hdlc-stream *Flavor*
An HDLC serial I/O stream. This flavor is built on **si:serial-binary-stream** and **si:serial-hdlc-mixin**.

si:make-serial-stream &rest *options* *Function*
This function initializes the serial I/O facility and returns the serial I/O stream.

The *options* argument is an alternating list of keyword symbols naming parameters, and initial values for those parameters. This lets you initialize

parameters when you start using the serial I/O stream. You can change most of them later with the **:put** operation.

make-serial-stream, which accesses a serial line, causes the accessing process to wait if all ports are in use. The command C-M-SUSPEND allows you to invoke a restart handler to close a line that you believe has been left open by mistake.

For documentation of parameters for serial I/O on the 3600: See the section "Parameters: 3600 Serial I/O".

:read-frame *string* &optional (*start* **0**) *end* of **si:serial-hdlc-mixin** *Method*
Reads an HDLC frame into *string*. Returns the length actually read.

:write-frame *string* &optional (*start* **0**) *end* of **si:serial-hdlc-mixin** *Method*
Writes *string* as an HDLC frame. This method never calls **process-wait** and can be used in a simple process. If insufficient buffers are available, it returns a form that evaluates to **t** when buffers become available.

3.2.5 Interface to the Vadic Modem

This is the low-level interface to the Vadic modem. To open a connection with the Vadic modem do:

```
(si:make-serial-stream ':flavor 'si:modem ':phone-number
  number-to-dial other-serial-options)
```

The system uses the autodialer to dial the given number and return a serial stream if it succeeds. If it fails, it signals an error based on **si:modem-error**.

4. Changes to Utilities in Release 5.0

4.1 Incompatible Changes to Utilities in Release 5.0

4.1.1 Default Font Format Now Bfd

The Font Editor now reads and writes BFD format fonts by default. These fonts are stored in files of type BFD, as are the system TV and LGP-1 fonts. You can still read and write font files of other types, such as BIN files on the 3600 and QBIN files on the LM-2. See the section "Changes to Font Editor File Commands".

4.1.2 Changes to Font Editor File Commands

Pressing R in FED translates into "read any font file". Pressing W translates into "write any font file". FED determines what kind of file to read or write based on the (canonical) type of the pathname you type. The default type is BFD. Clicking left on [Read File] is equivalent to pressing R, and clicking left on [Write File] is equivalent to pressing W.

The file type defaults from the (canonical) type of the pathname presented as the default. For example, if you type foo.bfd, you read or write a BFD file, whereas if you type foo.bin, you read or write a BIN file. FED complains if you supply a file type that is not a valid font file type for the machine you are using.

Clicking right on [Read File] or [Write File] gives you a menu of file types to use as the default and the prompt. Holding down the CTRL key while pressing R or W or while clicking left on [Read File] or [Write File] makes the default type BFD. Holding down the META key while using one of these commands makes the default type BIN on the 3600 and QBIN on the LM-2.

4.1.3 Changes to FUNCTION C, FUNCTION M, and FUNCTION Q

FUNCTION C, FUNCTION M, and FUNCTION Q have been changed to provide easier and more consistent control over which windows they affect. The operation of these keystrokes is as follows:

FUNCTION C — Controls the black-on-white state of the entire screen.
Arguments:

None	Toggle
0	Black-on-white
1	White-on-black

FUNCTION c-C — Controls the black-on-white state of the selected window.
Arguments:

	None	Toggle
	0	Same state as main screen's
	1	Opposite of state of main screen

FUNCTION m-C — Controls the black-on-white state of the mouse documentation line. Arguments:

None	Toggle
0	Same state as main screen's
1	Opposite of state of main screen

FUNCTION M — Controls global MORE processing. Arguments:

None	Toggle
0	Turn off MORE processing
1	Turn on MORE processing

FUNCTION c-M — Controls MORE processing for the selected window. Arguments: same as those for FUNCTION M.

FUNCTION Q — Hardcopies the entire screen on the default screen hardcopy device. Arguments: none.

FUNCTION c-Q — Hardcopies the selected window on the default screen hardcopy device. Arguments: none.

FUNCTION m-Q — Hardcopies the entire screen, without the status line and mouse documentation line, on the default screen hardcopy device. Arguments: none.

4.2 New Features in Utilities in Release 5.0

4.2.1 New feature: Flavor Examiner (SELECT X)

The Flavor Examiner is available via SELECT X or the system menu. This is strictly an interim program; it is supported fully in Release 5 but will eventually be incorporated into the Inspector.

Use the HELP command to learn how to use this new feature.

4.2.2 New terminal program (SELECT T)

The new terminal program incorporates the functions of the former Telnet and Supdup programs. It is available via SELECT T. Because it uses the generic network system, it allows access (in the presence of appropriate gateways) via autodialers to dialups, as well as direct Chaosnet and TCP through a gateway.

The prompt is still Connect to host:. To this you simply type the name of any host. (For information on naming of hosts, setting up host databases, declaring host addresses, and supported login services: See the section "Network Database". See the section "The Lisp Machine Generic Network System".)

The network system picks the best login service supported by the host and the optimum route to it. You can no longer specify a particular gateway and special contact name or port using ◊ and /. Such control arguments and new higher-level ones (such as a particular protocol to use, rather than the default) will be added to the terminal program when the system includes a command processor.

Once connected, you can give commands by pressing NETWORK followed by another character. The following commands are available:

NETWORK A	Send an ATTN (in Telnet, a new Telnet "Interrupt Process")
NETWORK D	Disconnect
NETWORK L	Log out of remote host and break the connection
NETWORK Q	Disconnect and deselect this window (Quit)
NETWORK M	Toggle MORE processing

You can issue more complicated commands using the extended command, NETWORK X. This command would use a command processor; in the interim (Release 5.0), this command uses the choose-variable-values facility. With NETWORK X you can change the following:

- the escape character
- whether characters overstrike or erase
- whether MORE processing is enabled
- in the case of Telnet, whether Imlac terminal codes are interpreted in host output

These were all formerly single-letter commands. A facility also exists for logging host output to a wallpaper file.

You can no longer press NETWORK at the Connect to host: prompt to issue a command before connecting. A command processor will fill this role in a future release.

4.2.3 Show Hardcopy Status (m-X) replaces chaos:print-lgp-queue

The Zwei command Show Hardcopy Status (m-X) replaces **chaos:print-lgp-queue** as the means of viewing the print queue.

The command prompts for the name of a printer. You can specify the queue of a particular printer by typing the name of the printer, or you can see the queues for all printers by pressing RETURN.

4.3 Improvements to Utilities in Release 5.0

4.3.1 Font Editor and Inspector use ESCAPE to evaluate forms

The Font Editor (FED) and the Inspector now use the ESCAPE key (on the 3600) and the ALTMODE key (on the LM-2) to evalute a Lisp form. Previously, those programs used the QUOTE key for this function. QUOTE is still accepted for compatibility on the LM-2; this key does not exist on the 3600.

4.3.2 Debugger c-M creates a process

The c-M command in the Debugger has been changed to create a new process. While you are editing the message, you use FUNCTION S to switch back to the Debugger, use its commands to investigate the state of the Lisp environment, and then switch back to the mail process to continue editing the message.

4.3.3 m-SUSPEND selects frame with break read function for Debugger

The behavior of m-SUSPEND in break loops has changed. The **break** read function now intercepts m-SUSPEND itself. When you press m-SUSPEND to enter the Debugger at the beginning of a line in a break loop, the current frame in the Debugger is now the frame that contains the **break** read function. The difference between m-SUSPEND and c-m-SUSPEND is that when you use m-SUSPEND at the beginning of a line, you enter the Debugger with a current frame that is closer to the frame that contains **break**'s caller. This eliminates irrelevant stack frames when, for example, you press c-SUSPEND to interrupt a program and then decide to enter the Debugger. This behavior is likely to change in a future release.

4.3.4 END and c-END swapped in Converse

When using the Converse facility prior to Release 5.0, pressing END sent a message and exited the Converse window; pressing c-END just sent a message, without leaving Converse. Now the reverse is true by default. Pressing END just sends the message, and pressing c-END sends the message and exits from the window.

The new variable **zwei:*converse-end-exits*** controls the behavior of END and c-END in Converse. If its value is **nil**, END sends and remains in Converse, and c-END sends and exits. If its value is not **nil**, c-END sends and remains in Converse, and END sends and exits. The default value is **nil**.

4.3.5 Changes to Converse Notifications

Converse beeps as soon as a message comes in, and if it is supposed to notify the user, it does so without waiting for the main Converse process to wake up. In pop-up mode, when the main Converse process is busy, an incoming message causes Converse to beep but not to display the message. This is necessary since only one message at a time should pop up. While the pop-up window is exposed, an incoming message causes Converse to beep. When the pop-up window is deexposed it is reexposed immediately with the new message in it.

If the main Converse window is exposed, a new message is never shown via notification or a pop-up window. If the main Converse window is exposed but its process is busy (typically, when it is in the Debugger or in an editor command and waiting for typein), Converse beeps but does not display the message. You can display the message with **print-sends** or by clearing the Converse process. You can usually clear the Converse process by pressing ABORT.

5. Changes to the File System in Release 5.0

5.1 Incompatible Changes to the File System in Release 5.0

5.1.1 New Default LMFS Translation Table for Sys Hosts

The default LMFS translation table for the SYS host has been changed to conform to the new rules for translating logical to physical pathnames. See the section "Changes to Logical Pathname Translations".

Following is the new default translation table:

```
'(("FONTS; TV;" ">sys>tv-fonts>")
  ("FONTS; LGP-1;" ">sys>lgp-1-fonts>")
  ("LMFS-PATCH;" ">sys>lmfs>patch>")
  ("*;" ">sys>*>"))
```

Note that logical directory names are followed by semicolons. Note also that translation tables for file systems other than LMFS might need translations for other logical directories as well. For example, because VMS file names cannot contain hyphens, a translation table for a VMS host might need to include specific translations for logical directories whose names contain hyphens, such as PRESS-FONTS;, L-SYS;, L-UCODE;, and L-FEP;.

Conversion tools exist to convert Release 4 to Release 5 site files. These tools take care of the translations for the SYS host. See the document *Software Installation Guide*. Users defining other logical hosts must convert the translation files on their own. See the function **fs:make-logical-pathname-host**.

5.1.2 LMFS Dumper Supports Accordion Wildcards

The LMFS dumper supports the new accordion wildcards. This is an incompatible change in its behavior.

Old Behavior: Previously, the dumper implicitly (and sometimes gratuitously) recursed over all subdirectories, with the wild name *.*.*, of what you asked it to dump. It did this recursion because there was no way to tell it not to. In addition, this recursion was necessary for file system backup.

New Behavior: The dumper no longer recurses over all subdirectories, regardless of the top-level spec. Now, specify exactly those files you wish to dump using wildcards. Wildcards are powerful enough to say exactly what you mean, subsuming all previously possible cases, and allowing for many new ones. If you want recursion, express it via "accordion" wildcards (**).

The default pathname to be dumped has thus been changed from >*.*.* to >**>*.*.*, which is what the previous one was trying to say. If you give the new dumper >*.*.*, it now dumps exactly that, which is to say files directly in the root directory, and not files in any inferiors.

For example, >**>*.*.* specifies all files in the file system, and >lmach>**>*.*.newest gets all of the newest files in >lmach or any of its subdirectories.

Note that there is no operational change if you use the default of the backup dumper, namely, dump the whole file system. It just looks different on the menu.

The dumper lists directories as it goes down encountering them. It is sufficiently clever to avoid simply expanding the wildcard, thereby listing the whole file system at one time. It has its own implementation of **:wild-inferiors** expansion.

5.2 New Features in the File System in Release 5.0

5.2.1 LMFS Now Supports Directory Links

LMFS now supports directory links. A directory link is a link that acts like a directory and points to a directory or another directory link. Directory links must have the type "directory" and the file version 1 in order to be recognized as such. It is impossible to create directory links and directories with conflicting names. It follows that the target of a directory link must be of type "directory" and version 1, although null string type and null version are accepted to mean this, as well.

The maximum length of link-chasing at any level is 10. Full circular-link checking has also been implemented at the file level; it was not previously present. There is no such thing as multilevel circularity.

Example:

```
>abel>foo.bar is a file.
>baker>david.directory.1 is a link to >abel.directory.1
Opening >baker>david>foo.bar in fact opens >abel>foo.bar.
```

Directory links are not marked as such in any way, and are not treated as such unless a directory of that name is being sought. Thus, the setting of link transparencies on a directory link has no effect on its behavior as a directory link.

5.2.2 LMFS Accordion Wildcards

It is now possible to specify wild directories of any arbitrary subtree. The string ** appearing in a LMFS pathname means "any number of intervening levels of directory, including none". This is a form of wildcard, and it creates a wild pathname, which may be used anyplace wild pathnames are used. ** may appear one or many times in the directory component of a filespec, although multiple occurences might be prohibitively expensive. Usually ** is the last directory level; the further away from the last it is, the more expensive it gets.

Note that a directory component of >** means every directory in the file system.

:wild-inferiors represents ** in LMFS directory components. It is no longer

possible to create, from a string, a LMFS pathname with a directory component of **:wild**. Although such pathnames still work, pathnames with directory components of **(:wild-inferiors)** ought to be functionally indistinguishable. Directory components of **:wild** print as though they were **(:wild-inferiors)**.

Pathname	*What it means*
>**>*.lisp	All the newest lisp files on the whole file system.
>**>*>secret>*.*.*	All files in subdirectories (but not top-level directories) named "secret".
>lmach>**>*.*.newest	All the newest files in >lmach and all its subdirectories.

5.2.3 Dumper Restarting and Append-to-tape Default

The LMFS dumper allows you to restart a dump from a given point. The option "restart pathname", when not empty, specifies a pathname that will be compared with the pathname of any file to be dumped or directory to be listed, and processing will be suppressed if the pathname of the object is considered "less" than the restart pathname. This comparison is similar to that done by **fs:pathname-lessp**. All of the dumper's processing is strictly ordered by this predicate.

The LMFS dumper also does not attempt to read tapes before writing on them unless either:

1. It intends to append to that tape, and is at the beginning of the tape.

2. The new site attribute **:validate-lmfs-dump-tapes**, whose default is **nil**, is set to something other than **nil**. The default disabling of this attribute avoids attempts to read blank tape and potentially unbounded tape reading on some TOPS-20s. However, enabling the attribute offers more protection against accidental overwriting of backup tapes.

In Release 5.0, the default for "append to tape" has been changed to "no" if you are running on the 3600. This is because hardware limitations make it impossible to append to a cartridge tape. Of course, you can dump a 3600 file system to a non-3600 remote noncartridge tape server and append.

6. Changes to Zmacs in Release 5.0

6.1 Incompatible Changes to Zmacs in Release 5.0

6.1.1 Both default pathnames for Source Compare (m-X) now use :newest version

Source Compare (m-X) has been changed. The default pathname for the first argument now uses the **:newest** version instead of the **:oldest**. The default for the second argument continues to use **:newest**. The change was made because **:oldest** is usually not the right thing; also, the asymmetry of the two defaults, as well as the fact that **:oldest** is rarely used, caused unexpected behavior.

6.1.2 Changes to Add Patch Changed Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X)

Formerly, these commands queried for each definition to be patched if given a numeric argument, but by default they patched everything without asking. This default behavior was incorrect, because the commands patched things that had been patched before, and they made patches for definitions in the patch buffer itself.

The commands no longer look at their numeric arguments. Add Patch Changed Definitions (m-X) queries for each buffer that contains a definition that might need to be patched; answering N skips the rest of that buffer.

Both commands query for each definition to be patched:

Y	patches it
N	skips it
P	patches it and any additional definitions in the same buffer without asking any more questions

If there are more buffers containing definitions to be patched, the commands ask questions again when they get to the next buffer.

A definition that has not been changed since it was patched is not considered a candidate for patching again, even though it has been changed since the file was read in. Note that patching any region of text lying entirely within a definition (with Add Patch (m-X)) counts as patching that definition.

6.1.3 Set Package (m-X) offers to create a package

Set Package (m-X) used to insist that the package must already exist, although related commands (such as Find File (c-X c-F) and Reparse Attribute List (m-X)) offered to create a package for you. Set Package (m-X) has been changed so that it queries you about whether to create a package that doesn't exist.

6.1.4 Change in numeric arguments to Copy File (m-X)

The numeric arguments to Copy File (m-X) have been reorganized.

The numeric argument controls copying of attributes. With no numeric argument, creation date and author are copied, and the mode (binary or character) of copy is determined by the file being copied. To force mode, or to suppress author or creation date copying, supply a numeric argument created by adding the values corresponding to the descriptions below. For example, c-12 m-X Copy File suppresses author and creation date copying.

1	Force copy in 16-bit binary mode
2	Force copy in character (text) mode
4	Suppress copy of author
8	Suppress copy of creation date

6.2 New Features in Zmacs in Release 5.0

6.2.1 New Zmacs command: Resume Patch (m-X)

Resume Patch (m-X) is a new Zmacs command that allows you to go back to a patch that you were not able to finish in the same session in which you started it.

6.2.2 New Zmacs command: Start Private Patch (m-X)

Start Private Patch (m-X) is similar to Start Patch (m-X), but it does not have anything to do with the world of systems, major and minor version numbers, and official patch directories. Instead of prompting for a system, Start Private Patch prompts for a file name. You can use Add Patch (m-X), Finish Patch (m-X), and Abort Patch (m-X). When you finish the patch it is written out to the specified file.

This command allows you to make a private patch file that you can load, try out, and share with other users before you install a change as a numbered patch that all users automatically get.

6.2.3 New Zmacs command: Source Compare Newest Definition (m-X)

Source Compare Newest Definition (m-X) compares the current definition (the definition that surrounds point) with the newest version in the normal source file for this definition, regardless of patch files.

6.2.4 New Buffer-history Mechanism in Zmacs

A new feature keeps the history list of previous buffers independently for each window.

When you create a new window, it takes its history list initially from the global history list. From then on, as you switch from buffer to buffer within that window, the history of those changes is kept on the list for that window. This affects particularly Select Previous Buffer (c-m-L) and the default for Select Buffer (c-X B).

The global history list still exists and is used for sorting in List Buffers (c-X c-B). Its ordering has not changed; it is maintained exactly as it used to be.

With two windows on the screen, you now have single-key access to four buffers instead of three. This can be helpful when you are working on two buffers at a time, each with its own need for switching to refer to another buffer.

6.2.5 New Zwei command: Comment Out Region (c-X c-;)

Comment Out Region (c-X c-;) comments out each of the lines in the region. When the region ends at the beginning of a line, it does not comment out that line. If any part of the line is part of the region, it does comment out that line.

A numeric argument inverts the meaning of the command, taking the comment indicators away from any commented-out lines in the region. When any part of the line is part of the region, it removes commenting from around that line. This assumes that any comment starting in column 1 is fair game. It stops when it encounters a line that does not begin the way a comment would, even if more lines that have been commented out remain in the region. It does keep the remainder of the region in this case, so that you can resume.

This works correctly for the different comment indicators for different major modes.

6.2.6 New Zwei command: Find Files in Tag Table (m-X)

Find Files in Tag Table (m-X) preloads every file in the selected tag table into the editor. Like Tags Search (m-X), it prompts for a tag table if one doesn't exist.

The prompting allows for specification of a system or all buffers as the tag table or for reading a .TAGS file.

6.2.7 New Zwei commands: Lowercase Code in [Region/Buffer] (m-X), Uppercase Code in [Region/Buffer] (m-X)

Four new Zwei commands allow you to change the case of code in a buffer or region:

- Uppercase Code in Buffer (m-X)
- Uppercase Code in Region (m-X)

- Lowercase Code in Buffer (m-X)
- Lowercase Code in Region (m-X)

These commands allow you to change a program that is typed in Electric Shift Lock Mode into one that is typed entirely in lowercase text, and vice versa. Only code is changed; comments, strings, or quoted characters are not affected. The commands work only where the entire file is typed in the same mode.

Like other Buffer commands, Uppercase Code in Buffer (m-X) and Lowercase Code in Buffer (m-X) query for a buffer name (the default is the current buffer) before operating on that buffer.

Like other Region commands, Uppercase Code in Region (m-X) and Lowercase Code in Region (m-X) operate on the region if there is one; otherwise they operate on the current definition.

The Uppercase commands put into uppercase all code in the associated area. They have the same effect as retyping that code using Electric Shift Lock Mode.

The Lowercase commands put into lowercase all code in the associated area. They have the same effect as retyping that code without using Electric Shift Lock Mode.

6.2.8 New canonical file type: :mss

The canonical file type **:mss** has been added for Scribe manuscript files. The editor treats these files like files of canonical type **:text**.

6.3 Improvements to Zmacs in Release 5.0

6.3.1 Default File Name Changed for Commands in Dired Buffer

In Dired, if you execute an editor command that accepts a file name as an argument, the default is the file name that appears on the line of the Dired buffer containing point.

This change makes it easier to operate on the file selected in Dired. For example, moving point to some line in Dired and doing Source Compare (m-X) causes the Source Compare command (m-X) to default to that file name.

6.3.2 Major-mode-setting Commands Now Query About Updating File Attribute List

Major-mode-setting commands such as Lisp Mode (m-X) now give the standard query about updating the file attribute list: "Set it for the file and attribute list, too?" (This is like the query for Set Package (m-X) and other attribute-setting commands.)

Programmers who call the **zwei:com-*mode*-mode** function directly (to get their own newly created buffer into the correct mode, for example) should bind **zwei:*set-attribute-updates-list*** to **nil** around the call to suppress the query.

6.3.3 Change in Zmacs command Modified Two Windows (c-X 4)

When two windows are being displayed, invoking the Zmacs command Modified Two Windows by typing c-X 4 J *r* is equivalent to invoking the Jump to Saved Position (c-X J *r*) command in the other window. When only one window is displayed, Zmacs goes into two-window mode, with the current buffer in one of the windows, and then jumps to the saved position in the other window.

6.3.4 Internal changes to macros zwei:defmajor and zwei:defminor

The Major and Minor mode system in the editor has been reimplemented using Flavors. For simple applications, the changes that were made are compatible with Release 4.

7. Changes to Zmail in Release 5.0

7.1 Incompatible Changes to Zmail in Release 5.0

7.1.1 Zmail Init File Pathnames Standardized

The names of Zmail init files have been standardized, and init files are now of canonical type **:lisp** for source files and **:bin** and **:qbin** for compiled files. You must change the name of your init file for Release 5.0.

For hosts that support long file names, the standard Zmail init file name is ZMAIL-INIT. Hosts that do not support long file names have conventions peculiar to each system.

Following are the names of Zmail init source files on some hosts:

Host system	*File name*
LMFS/TOPS-20	ZMAIL-INIT.LISP
UNIX	zmail-init.l
VMS	ZMAILINI.LSP
ITS	If user has own directory: ZMAIL >. If user does not have own directory: *USER* ZMAIL.

7.1.2 Babyl files with summary-window-format other than t or nil need to be edited

Any Babyl files with a Babyl file option of **summary-window-format** other than **t** or **nil** must be edited. Previously, Babyl file options were written in the **user** package, which was also where keywords resided. Now that the **keyword** package is separate from **user**, other keyword values (such as **:subject** and **:calendar**) require a colon prefix.

7.1.3 Ramifications of Host Colon Change for Babyl Files

The line MAIL: *HOST*:<*USER*>MESSAGE.TXT;1, which older versions of Zmail might have put in a Babyl file, doesn't work anymore, because it is parsed with *HOST*: interpreted as a device.

You can edit this using Zmail Babyl file options.

The essence of the new scheme is that Babyl files pointing to hosts other than that on which they reside can reside only on Lisp Machines or ITS's, for those are the only systems that natively support *HOST*: as designating hosts.

7.2 New Features in Zmail in Release 5.0

7.2.1 Sorting by Conversations Available

Sorting by conversations is available from the [Sort] command. Use [Sort (R)] to get a menu, then click on [Conversations]. As a result, messages that reference one another within the current sequence are grouped together.

7.2.2 New [Map Over] Menu Item: [Reply]

[Reply] is a new item in the menu that results from [Map Over (R)]. Clicking on this item replies to all messages in the sequence. A line identifying each message is inserted into the reply's In-reply-to field. When composing the reply, yanking (using c-m-Y) yanks all messages into the reply.

7.2.3 New [Map Over] Menu Item: [Select Conversation]

[Select Conversation] is a new item in the menu that results from [Map Over (R)]. You can also perform this operation by Select All Conversations By References (m-X); this is implemented by **zwei:com-zmail-select-all-conversations-by-references**.

[Map Over / Select Conversation] (or its extended command counterpart) selects messages that a message in the sequence refers to, or that refer to a message in the sequence, recursively. It is equivalent to appending together all sequences that result from Select Conversation By References (m-X) for each message in the current sequence. An argument gives a menu of universes to search. The universe defaults to loaded files.

7.3 Improvements to Zmail in Release 5.0

7.3.1 Previously undocumented commands: Delete Conversation By References (m-X), Append Conversation By References (m-X)

Delete Conversation By References (m-X) deletes messages that this message refers to, or that refer to this message, recursively. With an argument, it provides a menu of universes to search. The universe defaults to loaded files.

Append Conversation By References (m-X) appends messages that this message refers to, or that refer to this message, recursively. With an argument, it provides a menu of universes to search. The universe defaults to loaded files.

7.3.2 Rfc822 Domain Addressing Supported

Zmail now understands and, when appropriate, generates RFC822 domain addressing, used by the Arpanet.

Zmail now recognizes Resent (-to, -by, -date, -comments) header fields in addition to Redistributed (-to, -by, -date, -comments) fields.

8. Changes to the FEP in Release 5.0

This chapter describes changes to the FEP software that are concurrent with Release 5.0. FEP software is distributed in its own versions, which are separate from Lisp software releases. Release 5.0 requires FEP version 17 or higher. This chapter describes changes to FEP versions 14 through 17 for which Lisp support exists in Release 5.0. It also describes one change for FEP version 18 for which Lisp support does not exist in Release 5.0. Unless otherwise indicated, each change in any FEP version applies to later versions as well.

8.1 FEP Version 14: New Features

8.1.1 FEP Supports Hdlc Serial I/O

FEP version 14 contains support for synchronous serial I/O using HDLC-like bitstuffing protocols. HDLC serial I/O is available only in Release 5.0 and later releases. See the section "Hdlc Serial I/O on the 3600".

8.2 FEP Version 15: Incompatible Changes

8.2.1 h-c-*upper-left* stops execution of Lisp

As of FEP version 15, if you cannot obtain a Lisp Listener window or if no Lisp Listener is responding to keyboard input, you must press h-c-*upper-left* instead of c-*upper-left* to get to the FEP from Lisp. In earlier FEP versions, either c-*upper-left* or h-c-*upper-left* gets to the FEP from Lisp. (*upper-left* is the key in the upper left corner of the keyboard. It corresponds to LOCAL on old keyboards and FUNCTION on new keyboards.)

However, calling **si:halt** is a better way to stop Lisp than pressing h-c-*upper-left* because h-c-*upper-left* could interrupt disk I/O operations. This can render directories unusable, making all the files in the directories inaccessible until Symbolics personnel recover them.

See the section "Summary of Boot and Halt Operations".

8.2.2 si:halt replaces sys:%halt

On the 3600, **si:halt**, not **sys:%halt**, is now the preferred way to stop execution of Lisp.

si:halt *Function*

On the 3600, **si:halt** stops execution of Lisp and gives control to the FEP. This function stops Lisp without the danger of interrupting disk I/O operations. The function **sys:%halt** should no longer be used.

Interrupting a disk write can cause a fatal ECC error later, because the contents of the disk block are incomplete. This can render directories and other files inaccessible. This is a particular problem when halting the machine while using LMFS.

8.2.3 >Configuration.fep Files Are Now Called >Boot.boot

As of FEP version 15, you should rename >Configuration.fep files to >Boot.boot. The file type .fep is now reserved for files that the user should never modify.

8.2.4 New Defaults for FEP Commands

In FEP version 15, the default file name for each of the FEP commands Boot, Show File, Load Microcode, Load World, and Load Sync-Program is the last file name typed to that command.

The initial default for both the Boot command and the Show File command is now >Boot.boot, not >Configuration.fep. Show File uses and sets the same default string that Boot does. This lets you use Show File to look for an appropriate configuration file and then simply type Boot to cold boot using that file.

The Boot command and Show File command share the default, so that

```
Show File >magic.boot
```

causes Boot's default now to be >magic.boot and vice versa.

When the machine is powered up or the FEP is reset, the defaults are initialized as follows:

Boot/Show File	>Boot.boot
Load Microcode	>Microcode1.mic
Load World	>World1.load
Load Sync-Program	>Sync.sync

8.2.5 Disk Format Command Asks Different Question

When you type the Disk Format command, you are asked a series of questions. One of the questions used to be "*To* cylinder", which expected an *exclusive* upper bound as the answer. As of FEP version 15, Disk Format asks "*Through* cylinder", expecting an *inclusive* upper bound as the answer.

8.3 FEP Version 15: New Features

8.3.1 Loading Sync Programs

As of FEP version 15, files of type .sync contain sync programs for the monitor. A new FEP command, Load Sync-Program *file-name*, loads the specified file (of type .sync) into the sync program memory of the I/O board and causes the screen to clear. This is used for machines with monitors that require different sync programs than the one that is preprogrammed into the FEP.

8.4 FEP Version 15: Improvements

8.4.1 Show Configuration Command Displays More Information

Show Configuration has changed considerably in FEP version 15. It still displays the hardware configuration, but in considerably more detail, telling you part numbers, serial numbers, revisions, manufacture dates and other information peculiar to various parts of the machine (for example, the Ethernet address on the I/O paddle card).

Here is an example of what Show Configuration displays:

```
NanoFEP (P.N. 170018) S.N. 311, manufactured on 83-8-11
  Machine serial number 0.
  Manufactured as rev 1, functions as rev 1, ECO level 0
Datapath (P.N. 170032) S.N. 1192, manufactured on 83-9-9
  Manufactured as rev 3, functions as rev 3, ECO level 0
Sequencer (P.N. 170042) S.N. 186, manufactured on 83-4-1
  Manufactured as rev 4, functions as rev 4, ECO level 0
Memory Control (P.N. 170052) S.N. 1200, manufactured on 83-9-16
  Manufactured as rev 5, functions as rev 5, ECO level 0
Front End (P.N. 170062) S.N. 2040, manufactured on 83-8-19
  Manufactured as rev 5, functions as rev 5, ECO level 0
512K Memory (P.N. 170002) S.N. 515, manufactured on 83-9-20
  Manufactured as rev 2, functions as rev 2, ECO level 0
  LBUS slot 0 (octal base address 0)
512K Memory (P.N. 170002) S.N. 589, manufactured on 83-10-14
  Manufactured as rev 2, functions as rev 2, ECO level 0
  LBUS slot 1 (octal base address 2000000)
IO (P.N. 170082) S.N. 176, manufactured on 83-3-25
  Manufactured as rev 2, functions as rev 2, ECO level 0
  LBUS slot 8 (octal base address 20000000)
FEP Paddle Card (P.N. 170066) S.N. 287, manufactured on 83-7-16
  Manufactured as rev 1, functions as rev 1, ECO level 0
IO Paddle Card (P.N. 170086) S.N. 358, manufactured on 83-9-20
  Ethernet address: 08-00-05-01-30-0B
  Manufactured as rev 1, functions as rev 1, ECO level 0
```

8.4.2 Memory Board Not Needed in Lbus Slot 0

As of FEP version 15 and Release 5.0, it is possible to run Lisp and use the FEP commands Disk Format and Disk Restore without a memory board in LBUS slot 0. You must be running microcode version 238 or higher. This change allows for a 32-bit color display.

8.5 FEP Version 16: New Features

8.5.1 New FEP Commands: Add Disk-type and Clear Disk-types

FEP version 16 contains the new commands Add Disk-type and Clear Disk-types.

Add Disk-type lets you declare an arbitrary disk type to the FEP. You can declare up to four disk types before you have to give the Clear Disk-types command. Add Disk-Type is needed only to format and restore disks. It is not needed for normal operation of any validly formatted disk with a FEP file system.

Add Disk-type has the following arguments, for which it prompts with the argument names in parentheses:

name	The textual name by which this disk type is known
cylinders	The number of cylinders supported by the drive
heads	The number of heads on the drive
sectors	The number of sectors
gap1	The length of "gap1"
gap2	The length of "gap2"
gap3	The length of "gap3"
fast	0 for slower disks, 1 for faster disks

These numbers require careful computation and involve some restrictions of the 3600 hardware. The calculation should be done by Symbolics personnel.

Example:

```
Add Disk-type (name) M2284 (cylinders) 823 (heads) 10
    (sectors) 16 (gap1) 27 (gap2) 31 (gap3) 52 (fast) 0
```

Clear Disk-types clears all disk types declared with the Add Disk-type command.

8.6 FEP Version 16: Improvements

8.6.1 Unplugging Lemo Cables Should Not Halt the FEP

As of FEP Version 16, you should not have to reset the FEP after unplugging the lemo cables from either end. If you find that you must still reset the FEP, please let us know.

Sometimes what appears to be a FEP problem when unplugging lemo cables might be a console problem. You might have to power cycle the console rather than resetting the FEP.

8.6.2 Continue Command Sends an All-keys-up Character to Lisp

As of FEP version 16, the Continue command now sends an all-keys-up character to Lisp. Thus, if you press h-c-*upper-left* and then type Continue, Lisp gets the all-keys-up code and knows that HYPER and CONTROL are up.

The Return-keyboard-to-lisp command also sends an all-keys-up code.

8.6.3 More Information Available on Causes of Crashes

In addition to the information it displays in earlier versions, the Show Status command in FEP version 16 prints a section called "3600 program counters". This section can be useful in diagnosing why your machine crashed. It contains the following information:

Macro PC — The address of the current instruction of compiled Lisp code. This is prefaced with either (Odd) or (Even) since there are two instructions per word.

Next micro PC (NPC) — Usually CPC+1, calculated for pipelining. This item is not displayed in FEP version 17 and later versions.

Current micro PC (CPC) — The address of the current microinstruction.

Old PCs (OPC) — The addresses of the 16 most recently executed microinstructions. OPC+0 was executed most recently, OPC+17 least recently.

Use the function **dbg:decode-micro-pc** to decode the microcode PCs printed by the FEP command Show Status.

dbg:decode-micro-pc *pc* &optional (*name* **sys:%microcode-version**) (*version* **(sys:microcode-version-number sys:%microcode-version**)) *Function*

(3600 only) **dbg:decode-micro-pc** is useful for investigating why a machine crashed. It decodes the octal microinstruction addresses printed by the FEP

command Show Status. To use this function you should first write down the Show Status output. You can then either warm boot the machine using the Start command or call **dbg:decode-micro-pc** on another machine.

pc is an address in the microcode, taken from the CPC or OPC information printed by the Show Status command. Show Status prints these numbers in octal; if your default radix is decimal, precede *pc* by **#o**. Normally the number in the Show Status output with the arrow (→) pointing to it is the relevant number, but it can sometimes be useful to try decoding all of the numbers to get additional clues.

name and *version* are optional; they specify the version of the microcode that was running at the time of the crash. You can omit these arguments if you call **dbg:decode-micro-pc** while using the machine that crashed and while running the same microcode version as at the time of the crash. You can also omit these arguments if you call this function from another machine that has a software *and* hardware configuration that is *identical* to that of the machine that crashed. To find the microcode version name and number that a machine is running, use **(print-herald :verbose t)** or take the name and version number of the microcode file in the machine's boot file (normally fep0:>Boot.boot). Microcode version numbers are decimal; include a period at the end of the number if your default radix is octal.

Example:

```
(dbg:decode-micro-pc #o44552 "tmc5-mic" 253.)
```

dbg:decode-micro-pc prints information that depends on the microinstruction:

Microinstruction	*Information printed*
Halt instruction	The reason it halts the machine. An example is "error in the error handler". These reasons are constant strings in the microcode source program and do not represent any dynamic analysis of the state of the machine.
Signaller of a Lisp error	The internal form of the error message. This is not the same form of error message you would ever see otherwise; normally Lisp software translates these messages into conditions and signals them, and the conditions define more readable error messages. This is useful mainly in decoding OPCs earlier than the one with the arrow, when the machine halted because of "error in the error handler".

Handler for a macroinstruction in compiled Lisp code
: The name of that macroinstruction. A halt here might be caused by running a world together with an incompatible microcode, such as a microcode from an earlier release, that does not implement an instruction used by that world.

If all else fails, the function offers to load the microcode symbol table (from the sys:l-ucode; directory) and then prints the symbolic name of the microinstruction. Loading the microcode symbol table takes a few minutes. Microinstruction symbolic names can sometimes be clues to help in figuring out what the machine was doing at the time it crashed.

Two types of symbolic names exist: those with and without parentheses.

If the name includes parentheses, it is a list of the name of a microcode routine and the path through that routine to reach the microinstruction in question. Beware of a pitfall! These names are not unique; the same microinstruction can be reached by multiple paths from different microcode routines. For example, a microinstruction named (FTN-AR-1 3) might also be part of the microcode for the CAR instruction; you cannot assume too much from the name if it contains parentheses. It is only a clue.

If a symbolic name is just a symbol and has no parentheses, it is unique and names the first microinstruction of a microcode routine.

Beware of assuming too much. If the reason Lisp stopped itself is not "microcode halted", the information that **dbg:decode-micro-pc** prints is not likely to be helpful, though it might be useful to people who understand the hardware.

To decode the macrocode PC printed by the FEP command Show Status, warm boot or go to another machine running identical software and call the function **%find-structure-header** on the number printed by the FEP. This is an octal number; use **#o** if necessary. It should return a compiled-function object, which is the function that was executing at the time. To find the exact place in the function that was executing, note the different between the number printed by the FEP and the address in the printed representation of the compiled-function object. You can use **%pointer-difference** to compute this difference. Multiply this by 2, and add 1 if the FEP said the PC was odd (not even). The result is the instruction number of the current instruction; disassemble the compiled function to see it.

Example:

```
Fep>Show status
...
3600 program counters:
  Macro PC/ (Odd)1244531
  ...
Fep>Start
...
(%find-structure-header #o1244531)
#<DTP-COMPILED-FUNCTION EQUAL 1244530>
(%pointer-difference #o1244531 *)
1
(1+ (* * 2))
3
(disassemble ***)
  0 ENTRY: 2 REQUIRED, 0 OPTIONAL
  1 PUSH-LOCAL FP|0              ;A
  2 PUSH-LOCAL FP|1              ;B
  3 BUILTIN EQL STACK
  ...
```

Instruction 3 (EQL) is the one that halted.

8.7 FEP Version 17: Improvements

8.7.1 Show Status Command Displays More Useful Information

As of FEP version 17, the Show Status command prints more useful information.

The "Sequencer status line", which was confusing and difficult to interpret, has been split into two lines, "Sequencer error status" and "Sequencer miscellaneous status". The former contains only conditions that can stop the machine; it no longer contains reported errors that are not in fact errors. The miscellaneous status line contains the sequencer status bits that may be of occasional interest but are not reasons the machine stopped. For the names of the status bits printed in these two lines: See the section "FEP Show Status Command Output".

The confusing and uninformative "MC status" line has been removed.

8.8 FEP Version 18: Improvements

8.8.1 h-c-*upper-left* waits for Lisp to stop itself

With FEP version 18 and a future release, pressing h-c-*upper-left* does not immediately stop Lisp. Instead, the FEP asks Lisp to stop itself cleanly. If Lisp does stop itself, the FEP prints the message "Lisp stopped itself." If Lisp does not stop itself after about three seconds, the FEP prints, "Waiting for Lisp to stop

itself..." If after another three seconds Lisp does not stop itself, the FEP forcibly stops Lisp and prints, "Halting execution of Lisp."

The Lisp support for this change is not installed in Release 5.0. This means that if you have FEP version 18 and Release 5.0, Lisp never stops itself after you press h-c-*upper-left*. The FEP waits three seconds, prints "Waiting for Lisp to stop itself...", waits another three seconds, then forcibly stops Lisp and prints, "Halting execution of Lisp."

The purpose of this change is to reduce the chance of halting the 3600 during a disk write, which might cause ECC errors. Although the Lisp support for the change is not in Release 5.0, the FEP's delay of about six seconds before halting Lisp reduces (but does not eliminate) the risk of stopping Lisp at a dangerous time. The preferred way of stopping Lisp is still to call **si:halt**; use h-c-*upper-left* only if no Lisp Listener is responding.

9. Release 5.0: Notes and Clarifications

9.1 Clarifications and Corrections for Release 5.0

9.1.1 What happens when you cold boot

Because of the new network system, the procedure the Lisp Machine follows when cold booting has changed. While this does not normally require a different response from you, the process states in the status line are visibly different. This section briefly describes what the machine is doing. For background information: See the section "Network Database". See the section "The Lisp Machine Generic Network System".

The Lisp Machine first needs to determine the current time. Previously, the site file contained a list of hosts on the local network that were time servers. That list no longer exists. Instead, the machine issues a Chaosnet broadcast request (BRD) for the time. While the machine is waiting for the network, the status line displays a process state of "BRD Wait". Any host on the local network can answer this request. This means that in normal operation, if your site has more than one Lisp Machine, it is not necessary to enter the time by hand. However, if no host responds after a short time, the machine asks you to enter the time manually.

Special world loads for distribution no longer exist. A world load produced at one site can be loaded onto a machine at another site and automatically reconfigures itself for the new site. To do this, the machine must find out when booted whether the site has changed. It does this by asking the other machines on the local network, using a Chaosnet broadcast request. The status line again displays a process state of "BRD Wait". If the site has changed, or if no one answers this "Who am I?" query, the machine might ask you whether the site has changed. For information on what to do in this case: See the document *Software Installation Guide*.

Information in the network namespace database is cached in a world load and timestamped. However, namespace information might have changed since the world load was made. To find out about these changes, the machine connects to the namespace server machine. The status line now displays a process state of "Connect *host*". The machine tells the server the timestamp it previously knew for the local namespace. If the world load was saved recently, nothing has changed and the server responds immediately that the timestamp is still current. If some objects in the namespace have changed, the server provides the new timestamp for the local namespace and the names of the changed objects.

Your machine marks as valid any object that the server does *not* say has changed. The old information on the objects that did change still has the old timestamp. When the system needs to access that information, it discovers that the timestamp

no longer matches the timestamp of the local namespace. It then asks the namespace server for the information associated with the current timestamp. While the machine checks the names of old objects, the status line displays a process state of "Net In". For information on the protocol used for this incremental update: See the section "Network Database".

Your machine gets most of the lists of hosts and other network namespace objects that were formerly stored explicitly in the site file by doing a pattern-matching lookup using **net:find-objects-from-property-list**. This information includes, for example, the set of hosts at the local site that can provide store and forward mail service, the set of hardcopy printers at the local site, and the hosts that can spool files to a specific local hardcopy printer. The lists of objects resulting from these lookups are cached in a world load. If the machine does not have a cached list matching the result of a given property list query, or if the incremental update while booting reveals that the cached list is old, the system connects to the namespace server to get the latest list when it is needed.

To avoid the inconvenience of this connection the first time you need to send mail or hardcopy a file, the machine maintains a list of frequently asked questions. After the machine gets the incremental update while booting, a background process called Get Common Property Lists starts. It connects to the namespace server and caches the latest version of these commonly used sets of objects. If the world load is relatively new, this process runs for only a few seconds. The process runs at a lower priority than the Lisp Listener process to avoid interfering with your work.

Booting takes longer with an old world load than a new one because many aspects of the network database have changed. If you have an old world load and find that it takes a long time to boot or that it pages for a long time after booting (while running the Get Common Property Lists process), you might disk-save the world to make a new world load. The new world load caches information with the new namespace timestamp and therefore does not take so long to boot.

A world load that comes from another site takes longer to boot for the same reason: It has no cached information about the local namespace. This is why the installation procedure for new world loads includes booting and disk-saving them.

9.1.2 sort predicate should return nil for equal elements

The predicate for **sort** should return **nil** if its arguments are equal. For example, to sort in the opposite direction from <, use >, not $\geq$. This is because the quicksort algorithm used to sort arrays and cdr-coded lists becomes very much slower when the predicate returns non-**nil** for equal elements while sorting many of them.

See the function **sort**.

9.1.3 store not supported on the 3600

store is not supported on the 3600. **store** existed only for some large programs written in Maclisp, most of which now use more recent array referencing. **store** would require the saving of state in processes, which would slow down all processes. Use **aref** and **aset** (or **setf**) instead.

See the section "Change in Array Referencing".

9.1.4 Using copy-array-portion on the same array

It is safe to use **copy-array-portion** to copy from and to the same array only when the from-index is not less than the to-index, or when the "from" and "to" portions of the array do not overlap.

See the function **copy-array-portion**.

9.1.5 bitblt width from the destination array

The *width* argument to **bitblt** is in units of elements of the destination array. This is important if the source and destination arrays are of different types.

See the function **bitblt**.

9.1.6 Inspecting hash arrays of eq hash tables not permitted

You cannot inspect the hash array of an **eq** hash table. Trying to do so is likely to signal an error.

9.1.7 Known problem: char-upcase and char-downcase undefined for modified characters

char-upcase and **char-downcase** preserve font information. Because font information and control characters use the same character bits, the results of **char-upcase** and **char-downcase** are undefined for characters with modifier bits (CONTROL, META, SUPER, and HYPER).

See the function **char-upcase**. See the function **char-downcase**.

9.1.8 How to use the sys:function-parent declaration

The *Release 4.0 Release Notes* documented the use of the **sys:function-parent** declaration when a definition results from the macro expansion of a source definition. Following is a more comprehensive explanation:

A *definition* is a Lisp expression that appears in a source program file and has a name by which a user would like to refer to it. Definitions come in a variety of types. The main point of definition types is that two definitions with the same name and different types can exist simultaneously, but two definitions with the same

name and the same type redefine each other when evaluated. Some examples of definition type symbols and special forms that define such definitions are:

Type symbol	*Type name in English*	*Special form names*
defun	function	**defun, defmacro, defmethod**
defvar	variable	**defvar, defconst, defconstant**
defflavor	flavor	**defflavor**
defstruct	structure	**defstruct**

Things to note: More than one special form can define a given kind of definition. The name of the most representative special form is typically chosen as the type symbol. This symbol typically has a **si:definition-type-name** property of a string that acts as a prettier form of the name for people to read.
record-source-file-name and related functions take a name and a type symbol as arguments. The editor understands certain definition-making special forms, and knows how to parse them to get out the name and the type. This mechanism has not yet been made user-extensible. Currently the editor assumes that any top-level form it does not know about that starts with "(def" must be defining a function (a definition of type **defun**) and assumes that the cadr of that form is the name of the function. Heuristics appropriate for **defun** are applied to this name if it is a list. In general, a definition whose name is not a symbol and whose type is not **defun** does not work properly. This will be fixed in a future release.

The declaration **sys:function-parent** is of interest to users. The function with the same name is probably not of interest to users; it is part of the mechanism by which the Zmacs command Edit Definition (m-.) figures out what file to look in.

Example:

We have functions called "frobulators" that are stored on the property list of symbols and require some special bindings wrapped around their bodies. Frobulator definitions are not considered function definitions, because the name of the frobulator does not become defined as a Lisp function. Indeed, we could have a frobulator named **list** and Lisp's **list** function would continue to work. Instead we make a new definition type.

```
(defmacro define-frobulator (name arg-list &body body)
  `(progn 'compile
          (add-to-list-of-known-frobulators ',name)
          (record-source-file-name ',name 'define-frobulator)
          (defun (:property ,name frobulator) (self ,@arg-list)
            (declare (sys:function-parent ,name define-frobulator))
            (let (,(make-frobulator-bindings name arg-list))
              ,@body))))

(defprop define-frobulator "Frobulator" si:definition-type-name)
```

Here we would tell the editor how to parse **define-frobulator** if its parser were

user-extensible. Because it is not, we rely on its heuristics to make M-. work adequately for frobulators.

Next we define a frobulator. This is not an interesting definition, for we do not actually know what the word "frobulate" means. We could always recast this example as a symbolic differentiator: We would define the + frobulator to return a list of + and the frobulations of the arguments, the * frobulator to return sums of products of factors and derivatives of factors, and so forth.

```
(define-frobulator list ()
  (frobulate-any-number-of-args self))
```

In **define-frobulator**, we call **record-source-file-name** so that when a file containing frobulator definitions is loaded, we will know what file those definitions came from. Inside the function that is generated, we include a function-parent declaration because no definition of that function is apparent in any source file. The system will take care of doing
(record-source-file-name '(:property list frobulator) 'defun), as it always does when a function definition is loaded. Suppose an error occurs in a frobulator function — in the **list** example above, we might try to call **frobulate-any-number-of-args**, which is not defined — and we use the Debugger C-E command to edit the source. This will be trying to edit **(:property list frobulator)**, the function in which we were executing. The definition that defines this function does not have that name; rather, it is named **list** and has type **define-frobulator**. The **sys:function-parent** declaration enables the editor to know that fact.

If your definition-making special form and your definition type symbol do not have the same name, you should define the special form's **zwei:definition-function-spec** property to be the definition type symbol. This helps the editor parse such special forms.

For another example, more complicated but real, use **mexp** or the Zmacs command Macro Expand Expression (C-sh-M) to look at the macro expansion of:

```
(defstruct (foo :conc-name) one two)
```

The macro **sys:defsubst-with-parent** that it calls is just **defsubst** with a **sys:function-parent** declaration inside. It exists only because of a bug in an old implementation of **defsubst** that made doing it the straightforward way not work.

9.1.9 Use record-source-file-name instead of (remprop *symbol* ':source-file-name)

When redefining functions, some users try to avoid redefinition warnings and queries by using the form **(remprop** *symbol* **':source-file-name)**. The preferred way to do this is to use the form **(record-source-file-name** *function-spec* **'defun t)**. The former method causes the system to forget both the original definition and other definitions for the same symbol (as a variable, flavor, structure, and so forth). **record-source-file-name** lets the system know that the function is defined in two places, and it avoids redefinition warnings and queries.

Of course, if you are redefining something other than a function, use the appropriate definition type symbol instead of **defun** as the second argument to **record-source-file-name**. For example, if you are redefining a flavor, use **defflavor** as the second argument.

9.1.10 Use cdr with locatives returned by locf

When using **locf** to return a locative, you should use **cdr** rather than **car** to access the contents of the cell to which the locative points. This is because **(locf (cdr** *list*)) returns the list itself instead of a locative.

Example:

```
(car (locf (cdr '(a b)))) => a          ;wrong
(cdr (locf (cdr '(a b)))) => (b)        ;right
```

9.1.11 rplaca can be used with stack lists

The *Release 4.0 Release Notes*, section 2.2.30, page 43, stated that **rplaca** could not be used on stack lists. This is incorrect; you can use **rplaca** with stack lists, but not **rplacd**.

See the special form **with-stack-list**. See the special form **with-stack-list***.

9.1.12 FUNCTION 2 W displays current process name in status line

Typing FUNCTION 2 W changes the status line so that it displays the name of the process instead of displaying the name of the user. This also freezes the status line on that process; normally the status line switches to displaying a different process whenever the window selection mechanism tells it to.

If you see an unexpected state in the status line, you can use FUNCTION 2 W to find out what process is in that state (it might be that you are not talking to the process you think you should be.)

FUNCTION 1 W returns the status line to normal.

9.1.13 Known problem with si:gc-reclaim-immediately

The 3600 has a known bug in the garbage collector stimulated by turning on **si:gc-reclaim-immediately**. The typical manifestation is the following error message:

```
>>Trap: The first argument given to SYS:INTERNAL-=, NIL, was not a number.
While in the function SI:SCAVENGE-REGION ← SI:%GC-SCAVENGE ← SI:SCAVENGE-ALL
```

This bug will be fixed in a future release. In the meantime, avoid setting **si:gc-reclaim-immediately** to anything other than **nil**. You might be able to set **si:gc-reclaim-immediately** to **t** if you also set **inhibit-idle-scavenging-flag** to **t**. You must set **inhibit-idle-scavenging-flag** *after* calling **gc-on** and must reset it after warm booting.

9.1.14 tv:set-default-font not supported

The undocumented function **tv:set-default-font** is not supported and should not be used. The purpose of this function is to change the default font used by the system from **fonts:cptfont** to something else. However, this change requires changes in window geometry, and many programs are not prepared to handle the new geometry. Calling **tv:set-default-font** is likely to break these programs. This problem will be addressed in a future release.

9.1.15 Avoid Errors in the Mouse Process

It is not a good idea to perform lengthy or error-prone calculations in the mouse process. An error in the mouse process might make it necessary to reset that process. If you have a complex calculation to perform in a **:mouse-click** method, use **process-run-function** to spin off a new process for the calculation, or send a blip to the window's process and perform the calculation there.

9.1.16 nil not a valid menu item

nil is not a valid menu item. If an item list contains **nil**, it is removed from the list. A menu with no choices at all appears as a small blank menu.

You might be tempted to specify **nil** as a menu item in a form like **(tv:menu-choose '(t nil))**. But this causes a menu with only one choice, **t**, to appear. Instead, you should specify a form like **(tv:menu-choose '(("Yes" . t) ("No" . nil)))**. This presents a menu with two choices: "Yes", which returns **t**, and "No", which returns **nil**. You could also specify **(tv:menu-choose '(("T" . t) ("NIL" . nil)))**. See the document *Window System Choice Facilities*.

10. Release 5.0: Operations and Site Management

10.1 Notes on Operations in Release 5.0

10.1.1 Backup Tape Reliability

To ensure the reliability of backup tapes, it is important to use cartridge tapes of adequate density and to compare the contents of backup tapes with the contents of the original files.

Cartridge tape used on the 3600 should be certified at a minimum of 10000 BPI. Some customers have been using lower density tapes and producing unreliable backups. You can order tapes through your local Symbolics sales office.

It is important to keep the tape heads clean when reading and writing tapes. The heads should be cleaned after every two hours of new cartridge tape use and after every 8 hours of old tape use. "Old tape" is tape that has been written or read more than two hours. Inmac offers a cleaning kit called the "Clean Cycle Kit, Quarter Inch Cartridge Head Cleaner", P/N 7148. The Inmac telephone number is (408) 727-1970.

After writing a backup, press SELECT F, click left on [Reload/Retrieve], and click left on the "Compare" operation in the choose-variable-values window. The "Compare" operation provides a bit-by-bit comparison of the backup tape to the original file. If it reveals discrepancies, you should take a consolidated dump. A consolidated dump dumps all files saved after a specified date. To take a consolidated dump: Click left on [Complete Dump], click left on the "Consolidated" dump type in the choose-variable-values window, and specify the date in the "Consolidate from" field to be the date of your last successful dump.

See the section "Reloading and Retrieving".

10.1.2 Site Configuration for Dialnet

This is the basic information on how to configure your site for Dialnet. It allows the generic network system access to the Vadic autodialer.

Configuring a site:

You should define a network named "DIAL" of type DIAL. This network represents the international dial network. Addresses on this network are the telephone numbers, relative to your site, of the host in question. This means that they should contain any prefixes or area codes that are necessary to dial that number. Add addresses on this network to any hosts you wish to access via telephone.

You must also define what services each host supports. The basic system supports only the LOGIN service. Add the service triple "LOGIN DIAL TTY-LOGIN" to any hosts which support login over dialup lines.

To allow a 3600 or LM-2 to use an autodialing modem with the general network system you must describe its connection with a PERIPHERAL option. For each autodial modem add a peripheral entry of type MODEM which specifies the modem model and the serial port to which it is connected. For example:

```
MODEM MODEL VA3451 UNIT 3
```

Index

#

#

#/ and **#** now identical 19
#:*symbol***-syn-stream** 20
#:syn-stream 20
New reader macro: **#B** 18
#B reader **macro** 18
#/ and **#** now identical 19
Previously undocumented reader macro: **#** and **#** 83

,

,

Use **record-source-file-name** instead of (**remprop** *symbol* **':source-file-name**) 147

*****

*****

****** accordion wildcard specification 120

.

.

.fep file type 134

/

/

/BINARY 109
/VAR 109

0

0

Memory Board Not Needed in Lbus Slot 0 136

1

1

FEP Version 14: New Features 133
FEP Version 15: Improvements 135
FEP Version 15: Incompatible Changes 133
FEP Version 15: New Features 135
FEP Version 16: Improvements 137
FEP Version 16: New Features 136
FEP Version 17: Improvements 140
FEP Version 18: Improvements 140

2

FUNCTION 2 W displays current process name in status line 148
New Microcode in Release 5.0: 270 on 3600, 998 on LM-2 3

3

Hdlc Serial I/O on the 3600 110
New Metering Tools for the 3600 50
store not supported on the 3600 145
New option for **si:sb-on**: **:mouse** (3600 only) 95
Multidimensional Arrays on the 3600 Remember Actual Dimensions 78
3600 select-methods handle **:operation-handled-p** and **:send-if-handles** 81
3600 Supports Ieee Single- and Double-precision Floating Point 48
New Microcode in Release 5.0: 270 on 3600, 998 on LM-2 3

4

Change in Zmacs command Modified Two Windows (c-X 4) 127
Modified Two Windows (c-X 4) Zmacs command 127

5

Changes to Networks in Release 5.0 107
Changes to the FEP in Release 5.0 133
Changes to the File System in Release 5.0 119
Changes to the Lisp Language and Compiler in Release 5.0 5
Changes to Utilities in Release 5.0 113
Changes to Zmacs in Release 5.0 123
Changes to Zmail in Release 5.0 129
Clarifications and Corrections for Release 5.0 143
Improvements to Lisp in Release 5.0 77
Improvements to Utilities in Release 5.0 116
Improvements to Zmacs in Release 5.0 126
Improvements to Zmail in Release 5.0 130
Incompatible Changes to Lisp in Release 5.0 5
Incompatible Changes to Networks in Release 5.0 107
Incompatible Changes to the File System in Release 5.0 119
Incompatible Changes to Utilities in Release 5.0 113
Incompatible Changes to Zmacs in Release 5.0 123
Incompatible Changes to Zmail in Release 5.0 129
New Features in Lisp in Release 5.0 44
New Features in Networks in Release 5.0 108
New Features in the File System in Release 5.0 120
New Features in Utilities in Release 5.0 114
New Features in Zmacs in Release 5.0 124
New Features in Zmail in Release 5.0 130
Notes on Operations in Release 5.0 151
New Microcode in Release 5.0: 270 on 3600, 998 on LM-2 3
Release 5.0: Introduction and Highlights 1
Release 5.0: Notes and Clarifications 143
Release 5.0: Operations and Site Management 151

9

New Microcode in Release 5.0: 270 on 3600, 998 on LM-2 3

<

< oldest version specifier 35

>

> newest version specifier 35
>Configuration.fep Files Are Now Called >Boot.boot 134
>Boot.boot files 134
>Configuration.fep files 134
>Configuration.fep Files Are Now Called >Boot.boot 134

A

NETWORK A command 114
Compiler now warns about implicit **progn**s in **loop**s 82
Major-mode-setting Commands Now Query About Updating File Attribute List 126
Absolute goto 21
Reader Accepts Common Lisp Floating Point Exponents 17
** accordion wildcard specification 120
Accordion wildcards 1
LMFS Accordion Wildcards 120
LMFS Dumper Supports Accordion Wildcards 119
Activation character 65
Activation characters 22, 57
:activation option 57
New input editor options: **:no-input-save**, **:activation, :command, :preemptable** 57
Multidimensional Arrays on the 3600 Remember Actual Dimensions 78
New FEP Commands: Add Disk-type and Clear Disk-types 136
Add Disk-type FEP command 136
Changes to Add Patch Changed Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X) 123
Add Patch Changed Definitions (m-X) Zmacs command 123
Changes to Add Patch Changed Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X) 123
Add Patch Changed Definitions of Buffer (m-X) Zmacs command 123
fs:make-logical-pathname-host replaces **fs:** **add-logical-pathname-host** 37
fs: **add-logical-pathname-host** function 37
Previously undocumented functions: **tv:add-to-system-menu-programs-column, tv:add-to-system-menu-create-menu** 103
tv: **add-to-system-menu-create-menu** function 104
tv: **add-to-system-menu-programs-column** function 104
Previously undocumented functions: **tv:** **add-to-system-menu-programs-column, tv:add-to-system-menu-create-menu** 103
Adding prompt 65
readline and **readline-trim** return additional values 62
RFC822 domain addressing 130
Rfc822 Domain Addressing Supported 130

:adjust-screen-array message to sheets 79
New initialization list: **:after-full-gc** 73
:draw-filled-in-circle uses same algorithm as **:draw-circle** 99
Select All Conversations By References (m-X) Zmail command 130
Compiler Performs Style Checking on All Forms 81
Continue Command Sends an All-keys-up Character to Lisp 137
Deallocating allocated objects of a resource 13
Optimizing disk allocation 25
si: **alphabetic** syntax description 18
ALTMODE key 116
sys:dump-forms-to-file always puts package attribute into binary file 82
New Rules for Reading Ambiguous Tokens 19
floatp returns **t** for any floating-point number 49
:any-tyi method of **tv:stream-mixin** 41, 42
Flavors **tv:** **any-tyi-mixin** and **tv:list-tyi-mixin** obsolete 40
tv: **any-tyi-mixin** flavor 40
:any-tyi-no-hang method of **tv:stream-mixin** 41
Append Conversation By References (m-X) Zmail command 130
:append value of **open** option for **:if-exists** 25
Dumper Restarting and Append-to-tape Default 121
New function: **applyhook** 72
applyhook function 72
applyhook variable 72
>Configuration.lep Files Are Now Called >Boot.boot 134
Keyword Symbols Are Self-evaluating 13
Meaning of argument changed for **fs:parse-pathname** 32
Argument to **:menu** type menu items can be a menu or a form 105
New optional argument to **gc-immediately** 98
Optional argument to **mapatoms-all** and **where-is** eliminated 7
Change in argument to **process-wait-with-timeout** 95
New optional argument to **read** 61
Second argument to **si:install-microcode** now optional 94
New optional argument to **tv:mouse-wait** 101
Hook arguments 73
tv:scroll-maintain-list init function can take arguments 105
Arguments changed for **fs:user-homedir** and **fs:init-file-pathname** 34
Change in numeric arguments to Copy File (m-X) 124
Change in arguments to **print-herald** 39
New optional arguments to **print-notifications** 99
New optional arguments to **read-from-string** 84
New optional arguments to **string-upcase** and **string-downcase** 80
Change in arguments to **unadvise** 39
New message to arithmetic errors: **:operands** 98
sys: **arithmetic-error** flavor 98
bitblt width from the destination array 145
Using **copy-array-portion** on the same array 145
Change in type of array returned by **string-append** 16
Inspecting hash arrays of **eq** hash tables not permitted 145
Multidimensional Arrays on the 3600 Remember Actual Dimensions 78
:as-if-band option for **print-herald** 39
Disk Format Command Asks Different Question 134

Window System Changes
New Features
:validate-lmfs-dump-tapes site
sys:dump-forms-to-file always puts package
Copying file
Vadic
chaos:send-unc-pkt
Sorting by Conversations
Wildcard Directory Mapping
More Information
Associated with Mouse Input 40
Associated with the Input Editor (Rubout Handler) 57
attribute 121
attribute into binary file 82
Major-mode-setting Commands Now Query About Updating File Attribute List 126
attributes 124
autodialer 151
automatically returns the packet to the free pool 108
Available 130
Available 92
Available on Causes of Crashes 137
Avoid Errors in the Mouse Process 149

B

B

B

B exponent identifier 17
Select Buffer (c-X
summary-window-format
Ramifications of Host Colon Change for
B) Zmacs command 125
Babyl file option 129
Babyl Files 129
Babyl files with **summary-window-format** other than **t** or **nil** need to be edited 129
Back-translation 35
Backup Tape Reliability 151
Comparing
Ratios read in current **ibase** and print in current
Default Font Format Now
backup tapes 151
base 19
Bfd 113
:bin canonical type 34
Read rational number in
binary 18
sys:dump-forms-to-file always puts package attribute into binary file 82
Binary left shift 17
bitblt function 145
bitblt width from the destination array 145
Characters with modifier
si:
New descriptions: **si:**
bits 145
bitscale 17
bitscale, si:digitscale, si:non-terminating-macro 17
Blips 41
Receiving
New special forms:
blips 40
block and **tagbody** 46
block special form 46
Invisible
Memory
What happens when you cold
blocks in **progs** and **dos** 78
Board Not Needed in Lbus Slot 0 136
boot 143
Boot FEP command 134
Both default pathnames for Source Compare (m-X) now use **:newest** version 123
Recursion in
Bound and Default Handlers Eliminated 97
Bound handlers 97
Chaosnet sequence
Clock sequence
Disk sequence
Keyboard sequence
Mouse sequence
break 95
break 95
break 95
break 95
break 95
Break loops 116
m-SUSPEND selects frame with
break read function for Debugger 116

si: **break** syntax description 18
Change case of buffer 125
Default File Name Changed for Commands in Dired Buffer 126
Select Previous Buffer (c-m-L) Zmacs command 125
Select Buffer (c-X B) Zmacs command 125
Changes to Add Patch Changed Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X) 123
Add Patch Changed Definitions of Buffer (m-X) Zmacs command 123
Lowercase Code in Buffer (m-X) Zwei command 125
Uppercase Code in Buffer (m-X) Zwei command 125
New Buffer-history Mechanism in Zmacs 125
Streams sharing common buffers 89
List Buffers (c-X c-B) Zmacs command 125
byte function 50
Create a byte specifier 50
Extract size field of a byte specifier 50
New functions: **byte, byte-size, byte-position** 50
New functions: **byte, byte-size, byte-position** 50
byte-position function 50
byte-size function 50
:byte-size option for **copyf** 28
New functions: **byte, byte-size, byte-position** 50
Extract position field of a byte-specifier 50

C C C

FUNCTION C command 113
Changes to FUNCTION C, FUNCTION M, and FUNCTION Q 113
New Zwei command: Comment Out Region (c-X c-;) 125
Comment Out Region (c-X c-;) Zwei command 125
List Buffers (c-X c-B) Zmacs command 125
FUNCTION c-C command 113
c-END Converse command 116
END and c-END swapped in Converse 116
Debugger c-M creates a process 116
c-M Debugger command 116
Select Previous Buffer (c-m-L) Zmacs command 125
c-m-SUSPEND command 116
FUNCTION c-Q command 113
Change in Zmacs command Modified Two Windows (c-X 4) 127
Modified Two Windows (c-X 4) Zmacs command 127
Select Buffer (c-X B) Zmacs command 125
New Zwei command: Comment Out Region (c-X c-;) 125
Comment Out Region (c-X c-;) Zwei command 125
List Buffers (c-X c-B) Zmacs command 125
Jump to Saved Position (c-X J) Zmacs command 127
Unplugging Lemo Cables Should Not Halt the FEP 137
>Configuration.fep Files Are Now Called >Boot.boot 134
New function to be called by reader macros: **si:read-recursive** 83
Argument to **:menu** type menu items can be a menu or a form 105
rplaca can be used with stack lists 148
format ~\ directives can have package prefixes 92
Shifted Mouse Clicks Can Now Be Used for Editor Commands 103
:proceed methods can now return **nil** 97
tv:scroll-maintain-list init function can take arguments 105
Some Methods Can Use Combination Type as Method Type 83
New canonical file type: **:mss** 126
:init canonical pathname type removed 35

:bin canonical type 34
:lisp canonical type 34
:qbin canonical type 34
Carry tape system 1
Change case of buffer 125
Change case of region 125
New special forms **catch** and **throw** replace ***catch** and ***throw** 14
New special forms **catch** and **throw** replace ***catch** and ***throw** 14
catch special form 14
More Information Available on Causes of Crashes 137
Use **cdr** with locatives returned by **locf** 148
CFTP 109
Change case of buffer 125
Change case of region 125
Ramifications of Host Colon Change for Babyl Files 129
Change in argument to **process-wait-with-timeout** 95
Change in arguments to **print-herald** 39
Change in arguments to **unadvise** 39
Change in Debugger special command for **fs:directory-not-found** 98
Change in numeric arguments to Copy File (m-X) 124
Change in type of array returned by **string-append** 16
Change in Zmacs command Modified Two Windows (c-X 4) 127
Changes to Add Patch Changed Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X) 123
Add Patch Changed Definitions (m-X) Zmacs command 123
Changes to Add Patch Changed Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X) 123
Add Patch Changed Definitions of Buffer (m-X) Zmacs command 123
Default File Name Changed for Commands in Dired Buffer 126
Meaning of argument Changed for **fs:parse-pathname** 32
Arguments changed for **fs:user-homedir** and **fs:init-file-pathname** 34
FEP Version 15: Incompatible Changes 133
Window System Changes Associated with Mouse Input 40
Changes to **:mouse-click** method of **tv:essential-mouse** 42
Changes to **:tyi**, **:tyi-no-hang**, **:list-tyi**, :mouse-or-kbd-tyi, and **:mouse-or-kbd-tyi-no-hang** methods of **tv:stream-mixin** 40
Changes to Add Patch Changed Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X) 123
Changes to **chaos:open-stream** 108
Changes to Converse Notifications 117
Changes to Font Editor File Commands 113
Changes to **format:ochar** 21
Changes to FUNCTION C, FUNCTION M, and FUNCTION Q 113
Changes to Host Determination in Pathnames 30
Changes to input editor options **:do-not-echo**, :pass-through, **:prompt**, **:reprompt** 22
Changes to Input Editor User Interface 22

Incompatible Changes to Lisp in Release 5.0 5
Changes to Logical Pathname Translations 35
Changes to Logical Pathnames 35
Changes to **login** 5
Internal changes to macros **zwei:defmajor** and **zwei:defminor** 127
Changes to **make-syn-stream** 20
Changes to Networks in Release 5.0 107
Incompatible Changes to Networks in Release 5.0 107
Changes to **open** 24
Changes to **open** option **:direction** 24
Changes to Packages 6
Changes to **prompt-and-read** 84
Changes to Readtable, Reader, and Printer for Common Lisp 16
Changes to **renamef** and **copyf** 26
Changes to Serial I/O: Parity Recovery and Xon/Xoff Character Setting 109
Changes to the FEP in Release 5.0 133
Changes to the File System in Release 5.0 119
Incompatible Changes to the File System in Release 5.0 119
Incompatible Changes to the Input Editor (Rubout Handler) 21
Changes to the Lisp Language and Compiler in Release 5.0 5
Changes to Utilities in Release 5.0 113
Incompatible Changes to Utilities in Release 5.0 113
Changes to VMS Chaosnet 109
Changes to Zmacs in Release 5.0 123
Incompatible Changes to Zmacs in Release 5.0 123
Changes to Zmail in Release 5.0 129
Incompatible Changes to Zmail in Release 5.0 129
:chaos event 95
:chaos option for **si:sb-on** 95
chaos:close function 107
chaos:stream, **chaos:close**, and **chaos:finish** renamed 107
chaos:close-conn function 107
New function: **chaos:conn-finished-p** 109
chaos:conn-finished-p function 109
neti:reset, **neti:enable**, and **neti:disable** replace **chaos:reset**, **chaos:enable**, and **chaos:disable** 107
chaos:disable function 107
chaos:enable function 107
neti:reset, **neti:enable**, and **neti:disable** replace **chaos:reset**, **chaos:enable**, and **chaos:disable** 107
chaos:finish function 107
chaos:stream, **chaos:close**, and **chaos:finish** renamed 107
chaos:finish-conn function 107
chaos:make-stream function 107
Changes to **chaos:open-stream** 108
chaos:open-stream function 108
Show Hardcopy Status (m-X) replaces **chaos:print-lgp-queue** 115
chaos:print-lpg-queue function 115
chaos:reset function 107
neti:reset, **neti:enable**, and **neti:disable** replace **chaos:reset**, **chaos:enable**, and **chaos:disable** 107
chaos:return-pkt function 108
chaos:send-pkt function 108
chaos:send-unc-pkt automatically returns the packet

to the free pool 108
chaos:send-unc-pkt function 108
chaos:stream function 107
chaos:stream, chaos:close, and **chaos:finish** renamed 107
Changes to VMS Chaosnet 109
Chaosnet sequence break 95
char-downcase function 145
Known problem: **char-upcase** and **char-downcase** undefined for modified characters 145
Known problem: **char-upcase** and **char-downcase** undefined for modified characters 145
char-upcase function 145
Activation character 65
Escape character 114
Macro character 17
Reading and Printing Character Objects 19
:character option for **prompt-and-read** 85
Character output 21
Character quoter 17
Changes to Serial I/O: Parity Recovery and Xon/Xoff Character Setting 109
Character syntax descriptions 17
Continue Command Sends an All-keys-up Character to Lisp 137
Activation characters 22, 57
Floating-point exponent characters 17
Known problem: **char-upcase** and **char-downcase** undefined for modified characters 145
Mouse characters 102
Octal escape for special characters 17
:characters option for **copyf** 28
Characters with modifier bits 145
Circular-link checking 120
Compiler Performs Style Checking on All Forms 81
CHNCP.GSF global section 109
New Choose-variable-values Keywords 76
Clicking Middle Edits Current String in Choose-variable-values Windows 105
si: **circlecross** syntax description 18
Circular-link checking 120
New variable: **cl:*read-default-float-format*** 17
cl:*read-default-float-format* variable 17
cl:double-float format 17
cl:long-float format 17
cl:short-float format 17
cl:single-float format 17
Release 5.0: Notes and Clarifications 143
Clarifications and Corrections for Release 5.0 143
New clause for **condition-call: :no-error** 98
New FEP Commands: Add Disk-type and Clear Disk-types 136
Clear Disk-types FEP command 136
:clear-screen, :clear-eol, and **:clear-eof** messages to windows renamed 44
:clear-eof method of **tv:sheet** 44
:clear-eol method of **tv:sheet** 44
:clear-screen, **:clear-eol,** and **:clear-eof** messages to windows renamed 44
Previously undocumented function: **clear-resource** 78
clear-resource function 78
:clear-rest-of-line method of **tv:sheet** 44
:clear-rest-of-window method of **tv:sheet** 44

:clear-screen method of **tv:sheet** 44
:clear-screen, :clear-eol, and **:clear-eof** messages to windows renamed 44
:clear-window method of **tv:sheet** 44
Clicking Middle Edits Current String in Choose-variable-values Windows 105
Receiving mouse clicks 40
Modified mouse clicks as editor commands 103
Shifted Mouse Clicks Can Now Be Used for Editor Commands 103
:clock event 95
:clock option for **si:sb-on** 95
Clock sequence break 95
chaos: **close** function 107
chaos:stream, chaos: **close,** and **chaos:finish** renamed 107
chaos: **close-conn** function 107
Lowercase Code in Buffer (m-X) Zwei command 125
Uppercase Code in Buffer (m-X) Zwei command 125
Lowercase Code in Region (m-X) Zwei command 125
Uppercase Code in Region (m-X) Zwei command 125
Imlac terminal codes 114
string-length uses same coercion rules as **string** 16
What happens when you cold boot 143
New variable: **tv:** **cold-load-stream-old-selected-window** 74
tv: **cold-load-stream-old-selected-window** variable 74
Ramifications of Host Colon Change for Babyl Files 129
zwei: **com-zmail-select-all-conversations-by-references** function 130
Some Methods Can Use Combination Type as Method Type 83
Add Disk-type FEP command 136
Add Patch Changed Definitions (m-X) Zmacs command 123
Add Patch Changed Definitions of Buffer (m-X) Zmacs command 123
Append Conversation By References (m-X) Zmail command 130
Boot FEP command 134
c-END Converse command 116
c-M Debugger command 116
c-m-SUSPEND command 116
Clear Disk-types FEP command 136
Comment Out Region (c-X c-;) Zwei command 125
Continue FEP command 137
Copy File (m-X) Zmacs command 28, 124
Delete Conversation By References (m-X) Zmail command 130
Disk Format FEP command 134
END Converse command 116
Find Files in Tag Table (m-X)) Zwei command 125
FUCTION m-Q command 113
FUNCTION C command 113
FUNCTION c-C command 113
FUNCTION c-Q command 113
FUNCTION M command 113
FUNCTION m-C command 113
FUNCTION Q command 113
FUNCTION W command 148
h-c-*upper-left* command 133, 140
Jump to Saved Position (c-X J) Zmacs command 127
List Buffers (c-X c-B) Zmacs command 125
Load Microcode FEP command 134
Load Sync-program FEP command 134, 135

Load World FEP command 134
Lowercase Code in Buffer (m-X) Zwei command 125
Lowercase Code in Region (m-X) Zwei command 125
m-SUSPEND command 116
Major-mode-setting command 126
Modified Two Windows (c-X 4) Zmacs command 127
NETWORK A command 114
NETWORK D command 114
NETWORK L command 114
NETWORK M command 114
NETWORK Q command 114
NETWORK X command 114
R Fed command 113
Resume Patch (m-X) Zmacs command 124
Return-keyboard-to-lisp FEP command 137
Select All Conversations By References (m-X) Zmail command 130
Select Buffer (c-X B) Zmacs command 125
Select Conversation By References (m-X) Zmail command 130
Select Previous Buffer (c-m-L) Zmacs command 125
SELECT T command 114
SELECT X command 114
Set Package (m-X) Zmacs command 123
Show Configuration FEP command 135
Show File FEP command 134
Show Hardcopy Status (m-X) Zwei command 115
Show Status FEP command 137, 140
Source Compare (m-X) Zwei command 123
Source Compare Newest Definition (m-X) Zmacs command 124
Start Private Patch (m-X) Zmacs command 124
Uppercase Code in Buffer (m-X) Zwei command 125
Uppercase Code in Region (m-X) Zwei command 125
W Fed command 113
Disk Format Command Asks Different Question 134
Show Configuration Command Displays More Information 135
Show Status Command Displays More Useful Information 140
Change in Debugger special command for **fs:directory-not-found** 98
Change in Zmacs command Modified Two Windows (c-X 4) 127
:command option 58
Continue Command Sends an All-keys-up Character to Lisp 137
New input editor options: **:no-input-save, :activation, :command, :preemptable** 57
New Zwei command: Comment Out Region (c-X c-;) 125
New Zwei command: Find Files in Tag Table (m-X) 125
New Zmacs command: Resume Patch (m-X) 124
New Zmacs command: Source Compare Newest Definition (m-X) 124
New Zmacs command: Start Private Patch (m-X) 124
Changes to Font Editor File Commands 113
Modified mouse clicks as editor commands 103
NETWORK commands 114
New Defaults for FEP Commands 134
Shifted Mouse Clicks Can Now Be Used for Editor Commands 103
Default File Name Changed for Commands in Dired Buffer 126
Major-mode-setting Commands Now Query About Updating File Attribute List 126
New FEP Commands: Add Disk-type and Clear Disk-types 136

New Zwei command: Comment Out Region (c-X c-;) 125
Comment Out Region (c-X c-;) Zwei command 125
Comments for Lisp reader 83
Streams sharing common buffers 89
Common Lisp 1
Changes to Readtable, Reader, and Printer for Common Lisp 16
Reader Accepts Common Lisp Floating Point Exponents 17
Common Lisp readtable 83
Get Common Property Lists process 143
Communication between programs and input editor 63
Both default pathnames for Source Compare (m-X) now use **:newest** version 123
Source Compare (m-X) Zwei command 123
New Zmacs command: Source Compare Newest Definition (m-X) 124
. Source Compare Newest Definition (m-X) Zmacs command 124
Comparing backup tapes 151
Compiler 1
Changes to the Lisp Language and Compiler in Release 5.0 5
Compiler now warns about implicit **progn**s in **loop** 82
Compiler Performs Style Checking on All Forms 81
compiler:style-checker property 81
Complement mouse documentation line 113
Complement screen 113
Complement window 113
Pathname completion on VMS 109
String concatenation 16
:special-command-p method of **condition** 74
New condition flavor: **fs:multiple-file-not-found** 69
New condition flavor: **fs:rename-across-hosts** 70
condition-bind special form 97
condition-call special form 98
condition-call-if special form 98
New clause for **condition-call: :no-error** 98
condition-case special form 98
New message to conditions: **:special-command-p** 74
Show Configuration Command Displays More Information 135
Show Configuration FEP command 135
Site Configuration for Dialnet 151
New function: **chaos: conn-finished-p** 109
chaos: conn-finished-p function 109
String construction 65
Delete contents of window 44
Continue Command Sends an All-keys-up Character to Lisp 137
Continue FEP command 137
Returning control from input editor 57
CONTROL key 102
Controlling typeout style 65
Append Conversation By References (m-X) Zmail command 130
Delete Conversation By References (m-X) Zmail command 130
Select Conversation By References (m-X) Zmail command 130

Sorting by Conversations Available 130
Select All Conversations By References (m-X) Zmail command 130
New [map Over] Menu Item: [select Conversation] 130
[Select Conversation] Map Over menu item 130
END and c-END swapped in Converse 116
c-END Converse command 116
END Converse command 116
Converse facility 116
Changes to Converse Notifications 117
zwei: ***converse-end-exits*** variable 116
Convert number to single-precision floating-point 48
Convert number to small flonum 48
Converting numbers to double-precision floating-point 49
Change in numeric arguments to Copy File (m-X) 124
Copy File (m-X) Zmacs command 28, 124
copy-array-contents function 145
copy-array-contents-and-leader function 145
copy-array-portion function 145
Using **copy-array-portion** on the same array 145
:byte-size option for **copyf** 28
:characters option for **copyf** 28
:create-directories option for **copyf** 28
:report-stream option for **copyf** 28
Changes to **renamef** and **copyf** 26
copyf function 28
Copying file attributes 124
Copying files 28
Previously Undocumented Feature: Coroutine Streams 89
si: **coroutine-bidirectional-stream** flavor 92
si: **coroutine-input-stream** flavor 92
si: **coroutine-output-stream** flavor 92
:correct-input 23, 24
Clarifications and Corrections for Release 5.0 143
sys: **%count-disk-page-read-operations-in-scavenger** meter 52
sys: **%count-disk-page-read-operations-in-transporter** meter 52
Program counter (PC) metering 50
Displaying program counters information 137
Current micro PC (CPC) status information 137
New font: **fonts:** **cptfonti** 75
More Information Available on Causes of Crashes 137
Create a byte specifier 50
Set Package (m-X) offers to create a package 123
Create new logical host 37
:create value of **open** option **:if-does-not-exist** 25
:create-directories option for **copyf** 28
:create-screen-array message to screens 79
Debugger c-M creates a process 116
Creating logical host 38
Directory creation on VMS 109
Ratios read in current **ibase** and print in current **base** 19
Ratios read in current **ibase** and print in current **base** 19
Current micro PC (CPC) status information 137
FUNCTION 2 W displays current process name in status line 148
Clicking Middle Edits Current String in Choose-variable-values

Windows 105
Mouse cursor speed 100

D

NETWORK D command 114
D exponent identifier 17
New data types: **:single-float** and **:double-float** 49
Network database 107
:date option for **prompt-and-read** 85
:date-or-never option for **prompt-and-read** 85
New variable: **dbg:*debug-io-override*** 74
dbg:*debug-io-override* variable 74
dbg:decode-micro-pc function 137
deallocate-whole-resource function 13
Functions moved from the **si** package to **global**: **deallocate-whole-resource, map-resource** 13
Deallocating allocated objects of a resource 13
New variable: **dbg:** ***debug-io-override*** 74
dbg: ***debug-io-override*** variable 74
m-SUSPEND selects frame with **break** read function for Debugger 116
Debugger c-M creates a process 116
c-M Debugger command 116
Change in Debugger special command for **fs:directory-not-found** 98
:decimal-number option for **prompt-and-read** 85
:decimal-number-or-nil option for **prompt-and-read** 85
How to use the **sys:function-parent** declaration 145
sys:function-parent declaration 145
dbg: **decode-micro-pc** function 137
Dumper Restarting and Append-to-tape Default 121
Default File Name Changed for Commands in Dired Buffer 126
Default Font Format Now Bld 113
Default handlers 97
Recursion in Bound and Default Handlers Eliminated 97
New Default LMFS Translation Table for Sys Hosts 119
Both default pathnames for Source Compare (m-X) now use **:newest** version 123
New Default Representations for Newest and Oldest Logical Pathname Versions 35
:default-value option to **make-plane** 80
Font Editor file type defaults 113
New Defaults for FEP Commands 134
New special form: **defconstant** 45
defconstant special form 45
:mixture option for **defflavor** 54
:required-init-keywords option for **defflavor** 54
New option for **defflavor: :mixture** 54
New option for **defflavor: :required-init-keywords** 53
New special form: **defformat** 56
format: **defformat** special form 56
New special form: **define-symbol-macro** 53
define-symbol-macro special form 53
Defining a **format** directive 56
Defining family of flavors 54
Definition 145

New Zmacs command: Source Compare Newest — Definition (m-X) 124
Source Compare Newest — Definition (m-X) Zmacs command 124
Definition types 145
Changes to Add Patch Changed — Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X) 123
Add Patch Changed — Definitions (m-X) Zmacs command 123
Changes to Add Patch Changed — Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X) 123
Add Patch Changed — Definitions of Buffer (m-X) Zmacs command 123
Internal changes to macros **zwei:** — **defmajor** and **zwei:defminor** 127
zwei: — **defmajor** macro 127
Internal changes to macros **zwei:defmajor** and **zwei:defminor** 127
zwei: — **defminor** macro 127
New special form: — **defpackage** 7
defpackage special form 7
sys: — **defsubst-with-parent** macro 145
Delete contents of window 44
Delete Conversation By References (m-X) Zmail command 130
Delete to end of line 44
Delete to end of window 44
:delimited-string option for **prompt-and-read** 85
:delimited-string-or-nil option for **prompt-and-read** 85
:delimiter option for **prompt-and-read** 85
Previously undocumented function: — **describe-system** 94
describe-system function 94
si:alphabetic syntax — description 18
si:break syntax — description 18
si:circlecross syntax — description 18
si:doublequote syntax — description 18
si:macro syntax — description 18
si:single syntax — description 18
si:slash syntax — description 18
si:verticalbar syntax — description 18
si:whitespace syntax — description 18
Character syntax — descriptions 17
New — descriptions: **si:bitscale, si:digitscale, si:non-terminating-macro** 17
bitblt width from the — destination array 145
Previously undocumented special form: — **destructuring-bind** 77
destructuring-bind special form 77
Changes to Host — Determination in Pathnames 30
New function: — **dfloat** 49
dfloat function 49
International — dial network 151
DIAL network type 151
Site Configuration for — Dialnet 151
Disk Format Command Asks — Different Question 134
New descriptions: **si:bitscale, si:** — **digitscale, si:non-terminating-macro** 17
Multidimensional Arrays on the 3600 Remember Actual Dimensions 78
:input value of **open** option — **:direction** 24
:output value of **open** option — **:direction** 24
:probe value of **open** option — **:direction** 24
:probe-directory value of **open** option — **:direction** 24
:probe-link value of **open** option — **:direction** 24

Changes to **open** option **:direction** 24
:direction option 24
Defining a **format** directive 56
~← **format** directive 55
~→ **format** directive 55
~@* **format** directive 21
~@T **format** directive 21
~G **format** directive 21
~X **format** directive 21
format ~\ directives can have package prefixes 92
format directives ~@T and ~@* replace ~X and ~G 21
New **format** directives: ~→ and ~← 55
Home directory 34
Directory creation on VMS 109
LMFS Now Supports Directory Links 120
Wildcard Directory Mapping Available 92
Change in Debugger special command for **fs:** **directory-not-found** 98
fs: **directory-not-found** flavor 98
Dired 126
Default File Name Changed for Commands in Dired Buffer 126
neti:reset, neti:enable, and **neti:disable** replace **chaos:reset, chaos:enable,** and **chaos:disable** 107
chaos: **disable** function 107
neti: **disable** function 107, 108
neti:reset, neti:enable, and **neti:** **disable** replace **chaos:reset, chaos:enable,** and **chaos:disable** 107
Optimizing disk allocation 25
Disk Format Command Asks Different Question 134
Disk Format FEP command 134
:disk option for **si:sb-on** 95
Disk sequence break 95
New FEP Commands: Add Disk-type and Clear Disk-types 136
Add Disk-type FEP command 136
New FEP Commands: Add Disk-type and Clear Disk-types 136
Clear Disk-types FEP command 136
:displaced-conformally option for **make-array** 78
Display process name 148
Displaying program counters information 137
FUNCTION 2 W displays current process name in status line 148
Show Configuration Command Displays More Information 135
Show Status Command Displays More Useful Information 140
:do-not-echo option 22
Changes to input editor options **:do-not-echo, :pass-through, :prompt, :reprompt** 22
Complement mouse documentation line 113
RFC822 domain addressing 130
Rfc822 Domain Addressing Supported 130
Invisible blocks in **progs** and **dos** 78
New data types: **:single-float** and **:double-float** 49
cl: **double-float** format 17
New functions: **sys:single-float-p, sys:** **double-float-p** 49
sys: **double-float-p** function 49
IEEE-standard double-precision 48
3600 Supports Ieee Single- and Double-precision Floating Point 48
Converting numbers to double-precision floating-point 49
si: **doublequote** syntax description 18
:draw-filled-in-circle uses same algorithm as **:draw-circle** 99
:draw-circle method of **tv:graphics-mixin** 99

:draw-filled-in-circle method of **tv:graphics-mixin** 99
:draw-filled-in-circle uses same algorithm as **:draw-circle** 99
sys: **dump-forms-to-file** always puts package attribute into binary file 82
LMFS dumper 121
Dumper Restarting and Append-to-tape Default 121
LMFS Dumper Supports Accordion Wildcards 119

E E E

E exponent identifier 17
tv: **edit-namespace-object** function 107
Babyl files with **summary-window-format** other than **t** or **nil** need to be edited 129
Communication between programs and input editor 63
Input editor 1
Namespace editor 107
Reading function to use input editor 59
Returning control from input editor 57
Incompatible Changes to the Input Editor (Rubout Handler) 21
New Features Associated with the Input Editor (Rubout Handler) 57
Font Editor and Inspector use ESCAPE to evaluate forms 116
Modified mouse clicks as editor commands 103
Shifted Mouse Clicks Can Now Be Used for Editor Commands 103
Changes to Font Editor File Commands 113
Font Editor file type defaults 113
[Read File] Font Editor menu item 113
[Write File] Font Editor menu item 113
Changes to input editor options **:do-not-echo**, **:pass-through**, **:prompt**, **:reprompt** 22
New input editor options: **:no-input-save**, **:activation**, **:command**, **:preemptable** 57
:editor output format style 21
Changes to Input Editor User Interface 22
Using the Input Editor: Examples 65
Clicking Middle Edits Current String in Choose-variable-values Windows 105
sort predicate should return **nil** for equal elements 144
Optional argument to **mapatoms-all** and **where-is** eliminated 7
Recursion in Bound and Default Handlers Eliminated 97
chaos: **enable** function 107
neti: **enable** function 107, 108
neti:reset, **neti:enable**, and **neti:disable** replace **chaos:reset**, **chaos:enable**, and **chaos:disable** 107
neti:reset, **neti:** **enable**, and **neti:disable** replace **chaos:reset**, **chaos:enable**, and **chaos:disable** 107
Enabled events 95
Error encapsulation 65
END and c-END swapped in Converse 116
END Converse command 116
Delete to end of line 44
Erase to end of line 44
Delete to end of window 44
Erase to end of window 44
Inspecting hash arrays of **eq** hash tables not permitted 145

New function: **eql** 44
eql function 44
sort predicate should return **nil** for equal elements 144
Erase to end of line 44
Erase to end of window 44
Erase window 44
Error encapsulation 65
New error flavors: **sys:parse-error** and **sys:parse-ferror** 23
:error value of **open** option **:if-does-not-exist** 25
:error value of **open** option for **:if-exists** 25
Improvements to **make-system:** **error-restart**, selective transformations 94
Syntax errors in read functions 23
Avoid Errors in the Mouse Process 149
New message to arithmetic errors: **:operands** 98
Escape character 114
Octal escape for special characters 17
Font Editor and Inspector use ESCAPE to evaluate forms 116
:mouse-click method of **tv:** **essential-mouse** 42
Changes to **:mouse-click** method of **tv:** **essential-mouse** 42
New **open** option: **:estimated-length** 25
:estimated-length option 25
:eval-form option for **prompt-and-read** 85
:eval-form-or-end option for **prompt-and-read** 85
evalhook function 72, 73
Evaluate a Lisp form 116
Font Editor and Inspector use ESCAPE to evaluate forms 116
:chaos event 95
:clock event 95
:keyboard event 95
Enabled events 95
Flavor Examiner 1
New feature: Flavor Examiner (SELECT X) 114
Using the Input Editor: Examples 65
h-c-*upper-left* stops execution of Lisp 133
meter: **expand-range** function 51
Floating-point exponent characters 17
B exponent identifier 17
D exponent identifier 17
E exponent identifier 17
F exponent identifier 17
L exponent identifier 17
S exponent identifier 17
Reader Accepts Common Lisp Floating Point Exponents 17
:expression option for **prompt-and-read** 85
:expression-or-end option for **prompt-and-read** 85
:extension option to **make-plane** 80
External symbols 6
Extract position field of a byte-specifier 50
Extract size field of a byte specifier 50

F **F** **F**

F exponent identifier 17
Converse facility 116
Defining family of flavors 54
Previously Undocumented Feature: Coroutine Streams 89
New feature: Flavor Examiner (SELECT X) 114
FEP Version 14: New Features 133
FEP Version 15: New Features 135
FEP Version 16: New Features 136
New Features Associated with the Input Editor (Rubout Handler) 57
New Features in Lisp in Release 5.0 44
New Features in Networks in Release 5.0 108
New Features in the File System in Release 5.0 120
New Features in Utilities in Release 5.0 114
New Features in Zmacs in Release 5.0 124
New Features in Zmail in Release 5.0 130
R Fed command 113
W Fed command 113
FEP 1
Unplugging Lemo Cables Should Not Halt the FEP 137
Add Disk-type FEP command 136
Boot FEP command 134
Clear Disk-types FEP command 136
Continue FEP command 137
Disk Format FEP command 134
Load Microcode FEP command 134
Load Sync-program FEP command 134, 135
Load World FEP command 134
Return-keyboard-to-lisp FEP command 137
Show Configuration FEP command 135
Show File FEP command 134
Show Status FEP command 137, 140
New Defaults for FEP Commands 134
New FEP Commands: Add Disk-type and Clear Disk-types 136
Changes to the FEP in Release 5.0 133
FEP Supports Hdlc Serial I/O 133
FEP Version 14: New Features 133
FEP Version 15: Improvements 135
FEP Version 15: Incompatible Changes 133
FEP Version 15: New Features 135
FEP Version 16: Improvements 137
FEP Version 16: New Features 136
FEP Version 17: Improvements 140
FEP Version 18: Improvements 140
Extract size field of a byte specifier 50
Extract position field of a byte-specifier 50
Redistributed- header fields 130
Resent- header fields 130
Init file 34
Logging host output to file 114
Rename file 27
sys:dump-forms-to-file always puts package attribute into binary file 82
Change in numeric arguments to Copy File (m-X) 124
Copy File (m-X) Zmacs command 28, 124
Major-mode-setting Commands Now Query About Updating

File Attribute List 126
Copying file attributes 124
Changes to Font Editor File Commands 113
Show File FEP command 134
Default File Name Changed for Commands in Dired Buffer 126
File opened for input 24
File opened for output 24
Probe file opening 24
summary-window-format Babyl file option 129
Init File Pathnames Standardized 34
Zmail Init File Pathnames Standardized 129
File System 1
Changes to the File System in Release 5.0 119
Incompatible Changes to the File System in Release 5.0 119
New Features in the File System in Release 5.0 120
[Reload/Retrieve] File System Maintenance menu item 151
.fep file type 134
Font Editor file type defaults 113
New canonical file type: **:mss** 126
>Boot.boot files 134
>Configuration.fep files 134
Copying files 28
Ramifications of Host Colon Change for Babyl Files 129
>Configuration.fep Files Are Now Called >Boot.boot 134
New Zwei command: Find Files in Tag Table (m-X) 125
Find Files in Tag Table (m-X)) Zwei command 125
Babyl files with **summary-window-format** other than **t** or **nil** need to be edited 129
[Read File] Font Editor menu item 113
[Write File] Font Editor menu item 113
New Zwei command: Find Files in Tag Table (m-X) 125
Find Files in Tag Table (m-X)) Zwei command 125
net: **find-objects-from-property-list** function 143
chaos: **finish** function 107
chaos:stream, **chaos:close**, and **chaos:** **finish** renamed 107
chaos: **finish-conn** function 107
:finish-typeout method of **tv:stream-mixin** 64
New methods of **tv:stream-mixin**: **:start-typeout**, **:finish-typeout**, **:rescanning-p**, **:force-rescan**, **:replace-input**, **:read-bp** 63
Fixed-width italic font 75
fs:directory-not-found flavor 98
fs:multiple-file-not-found flavor 70
fs:rename-across-hosts flavor 70
fs:undefined-logical-pathname-translation flavor 37
Remove flavor 53
si:coroutine-bidirectional-stream flavor 92
si:coroutine-input-stream flavor 92
si:coroutine-output-stream flavor 92
si:modem flavor 111
si:modem-error flavor 111
si:serial-hdlc-stream flavor 110
sys:arithmetic-error flavor 98
sys:parse-error flavor 23
sys:parse-ferror flavor 24
tv:any-tyi-mixin flavor 40
tv:kbd-mouse-buttons-mixin flavor 42
tv:list-mouse-buttons-mixin flavor 42

tv:list-tyi-mixin flavor 40
tv:margin-space-mixin flavor 75
tv:preemptable-read-any-tyi-mixin flavor 43
tv:truncatable-lines-mixin flavor 102
tv:truncating-lines-mixin flavor 102
tv:truncating-window flavor 102
Flavor Examiner 1
New feature: Flavor Examiner (SELECT X) 114
Flavor **fs:undefined-logical-pathname-translation** replaces **fs:undefined-logical-pathname-directory** 37
Flavor **tv:preemptable-read-any-tyi-mixin** obsolete 43
New condition flavor: **fs:multiple-file-not-found** 69
New condition flavor: **fs:rename-across-hosts** 70
New flavor: **tv:margin-space-mixin** 74
Defining family of flavors 54
Flavors **tv:any-tyi-mixin** and **tv:list-tyi-mixin** obsolete 40
Flavors **tv:list-mouse-buttons-mixin** and tv:kbd-mouse-buttons-mixin obsolete 42
New error flavors: **sys:parse-error** and **sys:parse-ferror** 23
New flavors: **tv:truncatable-lines-mixin**, **tv:truncating-lines-mixin** 102
float function 48
float returns a single-precision number 48
3600 Supports Ieee Single- and Double-precision Floating Point 48
Reader Accepts Common Lisp Floating Point Exponents 17
Floating point numbers 1
Convert number to single-precision floating-point 48
Converting numbers to double-precision floating-point 49
Floating-point exponent characters 17
floatp returns **t** for any floating-point number 49
Floating-point numbers 48, 49
floatp function 49
floatp returns **t** for any floating-point number 49
Flonum 49
Convert number to small flonum 48
Small flonum 49
Fixed-width italic font 75
Font Editor and Inspector use ESCAPE to evaluate forms 116
Changes to Font Editor File Commands 113
Font Editor file type defaults 113
[Read File] Font Editor menu item 113
[Write File] Font Editor menu item 113
Default Font Format Now Bfd 113
:font-list option for **prompt-and-read** 85
New font: **fonts:cptfont** 75
fonts package 6
New font: **fonts:cptfont** 75
:force-rescan method of **tv:stream-mixin** 64
New methods of **tv:stream-mixin**: **:start-typeout**, **:finish-typeout**, **:rescanning-p**, **:force-rescan**, **:replace-input**, **:read-bp** 63
Forgetting objects remembered by a resource 78
Argument to **:menu** type menu items can be a menu or a form 105
block special form 46

catch special form 14
condition-bind special form 97
condition-call special form 98
condition-call-if special form 98
condition-case special form 98
defconstant special form 45
define-symbol-macro special form 53
defpackage special form 7
destructuring-bind special form 77
Evaluate a Lisp form 116
format:defformat special form 56
multiple-value-call special form 47
multiple-value-prog1 special form 48
store special form 145
tagbody special form 47
throw special form 15
trace special form 96
unadvise special form 40
with-stack-list special form 148
with-stack-list* special form 148
Nonkeyword form of **make-array** is obsolete 16
New special form: **defconstant** 45
New special form: **define-symbol-macro** 53
New special form: **defpackage** 7
Previously undocumented special form: **destructuring-bind** 77
New special form: **format:defformat** 56
cl:double-float format 17
cl:long-float format 17
cl:short-float format 17
cl:single-float format 17
Disk Format Command Asks Different Question 134
Defining a **format** directive 56
~← **format** directive 55
~→ **format** directive 55
~@* **format** directive 21
~@T **format** directive 21
~G **format** directive 21
~X **format** directive 21
format directives ~@T and ~@* replace ~X and ~G 21
New **format** directives: ~→ and ~← 55
Disk Format FEP command 134
New format for **trace** output 96
Default Font Format Now Bfd 113
:editor output format style 21
:read output format style 21
:sail output format style 21
format ~\ directives can have package prefixes 92
:format-args 24
format: ***format-output*** variable 56
:format-string 24
format:*format-output* variable 56
New special form: **format:defformat** 56
format:defformat special form 56
Changes to **format:ochar** 21
format:ochar function 21
Compiler Performs Style Checking on All Forms 81
Font Editor and Inspector use ESCAPE to evaluate forms 116

New special forms **catch** and **throw** replace ***catch** and ***throw** 14
New special forms: **block** and **tagbody** 46
New special forms: **multiple-value-call** and **multiple-value-prog1** 47
m-SUSPEND selects frame with **break** read function for Debugger 116
chaos:send-unc-pkt automatically returns the packet to the free pool 108
New variable: **fs:*remember-passwords*** 70
fs:*remember-passwords* variable 70
fs:make-logical-pathname-host replaces **fs:add-logical-pathname-host** 37
fs:add-logical-pathname-host function 37
Change in Debugger special command for **fs:directory-not-found** 98
fs:directory-not-found flavor 98
Arguments changed for **fs:user-homedir** and **fs:init-file-pathname** 34
fs:init-file-pathname function 34
fs:make-logical-pathname-host function 37
fs:make-logical-pathname-host replaces **fs:add-logical-pathname-host** 37
New condition flavor: **fs:multiple-file-not-found** 69
fs:multiple-file-not-found flavor 70
Meaning of argument changed for **fs:parse-pathname** 32
fs:parse-pathname function 33
New condition flavor: **fs:rename-across-hosts** 70
fs:rename-across-hosts flavor 70
Previously undocumented function: **fs:set-logical-pathname-host** 38
fs:set-logical-pathname-host function 38
Flavor **fs:undefined-logical-pathname-translation** replaces **fs:undefined-logical-pathname-directory** 37
fs:undefined-logical-pathname-translation flavor 37
Flavor **fs:undefined-logical-pathname-translation** replaces **fs:undefined-logical-pathname-directory** 37
Arguments changed for **fs:user-homedir** and **fs:init-file-pathname** 34
fs:user-homedir function 34
FUCTION m-Q command 113
si: **full-gc** function 73
applyhook function 72
bitblt function 145
byte function 50
byte-position function 50
byte-size function 50
chaos:close function 107
chaos:close-conn function 107
chaos:conn-finished-p function 109
chaos:disable function 107
chaos:enable function 107
chaos:finish function 107
chaos:finish-conn function 107
chaos:make-stream function 107
chaos:open-stream function 108
chaos:print-lpg-queue function 115
chaos:reset function 107
chaos:return-pkt function 108
chaos:send-pkt function 108
chaos:send-unc-pkt function 108
chaos:stream function 107
char-downcase function 145

char-upcase function 145
clear-resource function 78
copy-array-contents function 145
copy-array-contents-and-leader function 145
copy-array-portion function 145
copyf function 28
dbg:decode-micro-pc function 137
deallocate-whole-resource function 13
describe-system function 94
dfloat function 49
eql function 44
evalhook function 72, 73
float function 48
floatp function 49
format:ochar function 21
fs:add-logical-pathname-host function 37
fs:init-file-pathname function 34
fs:make-logical-pathname-host function 37
fs:parse-pathname function 33
fs:set-logical-pathname-host function 38
fs:user-homedir function 34
gc-immediately function 98
intern function 7
intern-local function 7
intern-local-soft function 7
intern-soft function 7
load-file-list function 39
login function 5
make-array function 16
make-package function 7
make-plane function 80
make-syn-stream function 20
map-resource function 14
mapatoms-all function 7
meter:expand-range function 51
meter:function-name-with-escapes function 52
meter:function-range function 52
meter:list-functions-in-bucket function 51
meter:make-pc-array function 51
meter:map-over-functions-in-bucket function 52
meter:monitor-all-functions function 51
meter:print-functions-in-bucket function 51
meter:range-of-bucket function 52
meter:report function 51
meter:start-monitor function 51
meter:stop-monitor function 51
mod function 49
net:find-objects-from-property-list function 143
neti:disable function 107, 108
neti:enable function 107, 108
neti:reset function 107, 108
package-used-by-list function 7
pkg-create-package function 7
print-herald function 39
print-notifications function 99
process-wait-with-timeout function 95
prompt-and-read function 85
read function 61

read-delimited-string function 60
read-from-string function 84
read-or-end function 62
readline function 62
readline-or-nil function 63
readline-trim function 62
record-source-file-name function 147
remprop function 147
renamef function 27
rplaca function 148
rplacd function 148
set-syntax-from-description function 18
si:full-gc function 73
si:halt function 133
si:install-microcode function 94
si:make-coroutine-bidirectional-stream function 91
si:make-coroutine-input-stream function 91
si:make-coroutine-output-stream function 91
si:make-process-queue function 71
si:make-serial-stream function 110, 111
si:patch-loaded-p function 70
si:process-dequeue function 71
si:process-enqueue function 71
si:process-queue-locker function 71
si:read-recursive function 83
si:reset-process-queue function 71
si:sb-on function 95
sort function 144
string function 16
string-append function 16
string-compare function 81
string-downcase function 80
string-length function 16
string-upcase function 80
sys:%halt function 133
sys:double-float-p function 49
sys:parse-ferror function 24
sys:single-float-p function 49
tv:add-to-system-menu-create-menu function 104
tv:add-to-system-menu-programs-column function 104
tv:edit-namespace-object function 107
tv:menu-choose function 149
tv:mouse-wait function 101
tv:scroll-maintain-list function 105
tv:select-or-create-window-of-flavor function 104
tv:set-default-font function 149
undefflavor function 53
where-is function 7
zwei:com-zmail-select-all-conversations-by-references function 130
Changes to FUNCTION 2 W displays current process name in status line 148
Changes to FUNCTION C command 113
Changes to FUNCTION C, FUNCTION M, and FUNCTION Q 113
Changes to FUNCTION c-C command 113
Changes to FUNCTION c-Q command 113
tv:scroll-maintain-list init function can take arguments 105
m-SUSPEND selects frame with **break** read function for Debugger 116

FUNCTION key 133
FUNCTION M command 113
Changes to FUNCTION C, FUNCTION M, and FUNCTION Q 113
FUNCTION m-C command 113
Changes to FUNCTION C, FUNCTION M, and FUNCTION Q 113
FUNCTION Q command 113
New function to be called by reader macros: **si:read-recursive** 83
Reading function to use input editor 59
FUNCTION W command 148
meter: **function-name-with-escapes** function 52
How to use the **sys:** **function-parent** declaration 145
sys: **function-parent** declaration 145
meter: **function-range** function 52
New function: **applyhook** 72
New function: **chaos:conn-finished-p** 109
Previously undocumented function: **clear-resource** 78
Previously undocumented function: **describe-system** 94
New function: **dfloat** 49
New function: **eql** 44
Previously undocumented function: **fs:set-logical-pathname-host** 38
New function: **make-package** 7
New function: **mod** 49
New function: **read-delimited-string** 60
New function: **readline-or-nil** 63
New function: **si:patch-loaded-p** 70
New function: **si:read-or-end** 62
Previously undocumented function: **string-compare** 81
New function: **sys:parse-ferror** 24
New function: **undefflavor** 53
Redefining functions 147
Syntax errors in read functions 23
Functions moved from the **si** package to **global**: **deallocate-whole-resource**, **map-resource** 13
New functions: **byte**, **byte-size**, **byte-position** 50
New functions: **si:make-process-queue**, si:process-enqueue, **si:process-dequeue**, si:process-queue-locker, **si:reset-process-queue** 71
New functions: **sys:single-float-p**, **sys:double-float-p** 49
Previously undocumented functions: tv:add-to-system-menu-programs-column, **tv:add-to-system-menu-create-menu** 103

G G G

New optional argument to **gc-immediately** 98
gc-immediately function 98
New variable: **gc-on** 73
gc-on variable 73
Known problem with **si:** **gc-reclaim-immediately** 148
si: **gc-reclaim-immediately** variable 148
GET 109
Get Common Property Lists process 143
Symbols in **global** and **keyword** packages with the same names 7
Global history 125
Global kill history 22

global package 1, 6
Symbols moved to or from **global** package 9
global package symbols 9
CHNCP.GSF global section 109
Functions moved from the **si** package to **global: deallocate-whole-resource, map-resource** 13
Absolute goto 21
:draw-circle method of **tv:** **graphics-mixin** 99
:draw-filled-in-circle method of **tv:** **graphics-mixin** 99

H

h-c-*upper-left* command 133, 140
h-c-*upper-left* stops execution of Lisp 133
h-c-*upper-left* waits for Lisp to stop itself 140
si:halt replaces **sys:** **%halt** 133
si: **halt** function 133
sys: **%halt** function 133
si: **halt** replaces **sys:%halt** 133
Unplugging Lemo Cables Should Not Halt the FEP 137
3600 select-methods handle **:operation-handled-p** and **:send-if-handles** 81
Rubout handler 1, 22
Incompatible Changes to the Input Editor (Rubout Handler) 21
New Features Associated with the Input Editor (Rubout Handler) 57
Bound handlers 97
Default handlers 97
Recursion in Bound and Default Handlers Eliminated 97
What happens when you cold boot 143
Hardcopy 113
Show Hardcopy Status (m-X) replaces **chaos:print-lgp-queue** 115
Show Hardcopy Status (m-X) Zwei command 115
Inspecting hash arrays of **eq** hash tables not permitted 145
Inspecting hash arrays of **eq** hash tables not permitted 145
format ~\ directives can have package prefixes 92
FEP Supports Hdlc Serial I/O 133
Hdlc Serial I/O on the 3600 110
Redistributed- header fields 130
Resent- header fields 130
Logical Pathnames Now Hierarchical 35
Release 5.0: Introduction and Highlights 1
Global history 125
Global kill history 22
Input history 22
Home directory 34
Hook arguments 73
Create new logical host 37
Creating logical host 38
Ramifications of Host Colon Change for Babyl Files 129
Changes to Host Determination in Pathnames 30
:host option for **prompt-and-read** 85
Logging host output to file 114
:host-list option for **prompt-and-read** 85
New Default LMFS Translation Table for Sys Hosts 119
How to use the **sys:function-parent** declaration 145

HYPER key 102

FEP Supports Hdlc Serial I/O 133
Hdlc Serial I/O on the 3600 110
:input-error-character serial I/O parameter 109
:input-xoff-character serial I/O parameter 109
:input-xon-character serial I/O parameter 109
:output-xoff-character serial I/O parameter 109
:output-xon-character serial I/O parameter 109
Changes to Serial I/O: Parity Recovery and Xon/Xoff Character Setting 109

Ratios read in current **ibase** and print in current **base** 19
#/ and **#** now identical 19
B exponent identifier 17
D exponent identifier 17
E exponent identifier 17
F exponent identifier 17
L exponent identifier 17
S exponent identifier 17
3600 Supports leee Single- and Double-precision Floating Point 48
IEEE-standard double-precision 48
IEEE-standard single-precision 48

:create value of **open** option **:if-does-not-exist** 25
:error value of **open** option **:if-does-not-exist** 25
New **open** options: **:if-exists** and **:if-does-not-exist** 25
nil value of **open** option **:if-does-not-exist** 25
:if-does-not-exist option 26
:if-does-not-exist option for **open** 25

:append value of **open** option for **:if-exists** 25
:error value of **open** option for **:if-exists** 25
:new-version value of **open** option for **:if-exists** 25
:overwrite value of **open** option for **:if-exists** 25
:rename value of **open** option for **:if-exists** 25
:rename-and-delete value of **open** option for **:if-exists** 25
:supersede value of **open** option for **:if-exists** 25
:truncate value of **open** option for **:if-exists** 25
New **open** options: **:if-exists** and **:if-does-not-exist** 25
:if-exists option 25
:if-exists option for **open** 25
Imlac terminal codes 114

Compiler now warns about implicit **progns** in **loops** 82
FEP Version 15: Improvements 135
FEP Version 16: Improvements 137
FEP Version 17: Improvements 140
FEP Version 18: Improvements 140
Improvements to Lisp in Release 5.0 77
Improvements to **make-system**: **error-restart**, selective transformations 94
Improvements to Utilities in Release 5.0 116
Improvements to Zmacs in Release 5.0 126
Improvements to Zmail in Release 5.0 130
FEP Version 15: Incompatible Changes 133
Incompatible Changes to Lisp in Release 5.0 5
Incompatible Changes to Networks in Release 5.0 107
Incompatible Changes to the File System in Release

5.0 119
Incompatible Changes to the Input Editor (Rubout Handler) 21
Incompatible Changes to Utilities in Release 5.0 113
Incompatible Changes to Zmacs in Release 5.0 123
Incompatible Changes to Zmail in Release 5.0 129
Current micro PC (CPC) status information 137
Displaying program counters information 137
Macro PC status information 137
Old PCs (OPC) status information 137
Show Configuration Command Displays More Information 135
Show Status Command Displays More Useful Information 140
More Information Available on Causes of Crashes 137
inhibit-idle-scavenging-flag variable 148
:init canonical pathname type removed 35
Init file 34
Init File Pathnames Standardized 34
Zmail Init File Pathnames Standardized 129
tv:scroll-maintain-list init function can take arguments 105
:space init option for **tv:margin-space-mixin** 75
Arguments changed for **fs:user-homedir** and **fs:** **init-file-pathname** 34
fs: **init-file-pathname** function 34
:initial-dimensions option to **make-plane** 80
New options for **make-plane:** **:initial-dimensions, :initial-origins** 80
New options for **make-plane: :initial-dimensions,** **:initial-origins** 80
:initial-origins option to **make-plane** 80
New initialization list: **:after-full-gc** 73
File opened for input 24
Mouse input 42
Window System Changes Associated with Mouse Input 40
Yanking previous input 22
Input editor 1
Communication between programs and input editor 63
Reading function to use input editor 59
Returning control from input editor 57
Incompatible Changes to the Input Editor (Rubout Handler) 21
New Features Associated with the Input Editor (Rubout Handler) 57
Changes to input editor options **:do-not-echo, :pass-through, :prompt, :reprompt** 22
New input editor options: **:no-input-save, :activation, :command, :preemptable** 57
Changes to Input Editor User Interface 22
Using the Input Editor: Examples 65
Prompting for input from user 85
Input history 22
Yanking input in Zwei 1
:input value of **open** option **:direction** 24
:input-error-character serial I/O parameter 109
:input-xoff-character serial I/O parameter 109
:input-xon-character serial I/O parameter 109
Inspecting hash arrays of **eq** hash tables not permitted 145
Font Editor and Inspector use ESCAPE to evaluate forms 116
si: **install-microcode** function 94
Second argument to **si:** **install-microcode** now optional 94
Use **record-source-file-name** instead of **(remprop** *symbol* **':source-file-name)** 147
Changes to Input Editor User Interface 22

Interface to the Vadic Modem 111
intern function 7
intern, intern-local, intern-soft, and intern-local-soft return two values 7
intern-local function 7
intern, **intern-local, intern-soft,** and **intern-local-soft** return two values 7
intern-local-soft function 7
intern, intern-local, intern-soft, and **intern-local-soft** return two values 7
intern-soft function 7
intern, intern-local, **intern-soft,** and **intern-local-soft** return two values 7
Internal changes to macros **zwei:defmajor** and **zwei:defminor** 127
Internal symbols 6
International dial network 151
Release 5.0: Introduction and Highlights 1
Invisible blocks in **progs** and **dos** 78
Nonkeyword form of **make-array** is obsolete 16
Fixed-width Italic font 75
New [map Over] Menu Item: [reply] 130
New [map Over] Menu Item: [select Conversation] 130
Argument to **:menu** type menu items can be a menu or a form 105
ITS 34
h-c-*upper-left* waits for Lisp to stop itself 140

J

J J

Jump to Saved Position (c-X J) Zmacs command 127
Jump to Saved Position (c-X J) Zmacs command 127

K

K K

tv: **kbd-mouse-buttons-mixin** flavor 42
Flavors **tv:list-mouse-buttons-mixin** and **tv:** **kbd-mouse-buttons-mixin** obsolete 42
ALTMODE key 116
CONTROL key 102
FUNCTION key 133
HYPER key 102
LOCAL key 133
META key 102
QUOTE key 116
SUPER key 102
:keyboard event 95
Keyboard keys 1
:keyboard option for **si:sb-on** 95
Keyboard sequence break 95
Keyboard keys 1
keyword package 1, 6
keyword package symbols 7
Symbols in **global** and **keyword** packages with the same names 7
Keyword Symbols Are Self-evaluating 13
:keyword-list option for **prompt-and-read** 85
New Choose-variable-values Keywords 76
Global kill history 22
Known problem with

si:gc-reclaim-immediately 148
Known problem: **char-upcase** and **char-downcase** undefined for modified characters 145

L

NETWORK L command 114
L exponent identifier 17
Lisp language 1
Changes to the Lisp Language and Compiler in Release 5.0 5
Memory Board Not Needed in Lbus Slot 0 136
Trim leading and trailing white space 62
Binary left shift 17
Unplugging Lemo Cables Should Not Halt the FEP 137
Patch level 70
Complement mouse documentation line 113
Delete to end of line 44
Erase to end of line 44
FUNCTION 2 W displays current process name in status line 148
Truncating lines 102
LMFS Now Supports Directory Links 120
Changes to Readtable, Reader, and Printer for Common Lisp 16
Common Lisp 1
Continue Command Sends an All-keys-up Character to Lisp 137
h-c-*upper-left* stops execution of Lisp 133
:lisp canonical type 34
Reader Accepts Common Lisp Floating Point Exponents 17
Evaluate a Lisp form 116
Improvements to Lisp in Release 5.0 77
Incompatible Changes to Lisp in Release 5.0 5
New Features in Lisp in Release 5.0 44
Lisp language 1
Changes to the Lisp Language and Compiler in Release 5.0 5
Comments for Lisp reader 83
Common Lisp readtable 83
h-c-*upper-left* waits for Lisp to stop itself 140
Major-mode-setting Commands Now Query About Updating File Attribute List 126
List Buffers (c-X c-B) Zmacs command 125
meter: **list-functions-in-bucket** function 51
Flavors **tv:** **list-mouse-buttons-mixin** and tv:kbd-mouse-buttons-mixin obsolete 42
tv: **list-mouse-buttons-mixin** flavor 42
:list-tyi method of **tv:stream-mixin** 41
Changes to **:tyi, :tyi-no-hang,** **:list-tyi, :mouse-or-kbd-tyi,** and :mouse-or-kbd-tyi-no-hang methods of **tv:stream-mixin** 40
tv: **list-tyi-mixin** flavor 40
Flavors **tv:any-tyi-mixin** and **tv:** **list-tyi-mixin** obsolete 40
New initialization list: **:after-full-gc** 73
rplaca can be used with stack lists 148
Stack lists 148
Get Common Property Lists process 143
New Meters for the LM-2 52
New Microcode in Release 5.0: 270 on 3600, 998 on LM-2 3
Previously undocumented variables: **sys:mouse-x-scale-array** and **sys:mouse-y-scale-array**

(LM-2 only) 100
LMFS 34
LMFS Accordion Wildcards 120
LMFS dumper 121
LMFS Dumper Supports Accordion Wildcards 119
LMFS Now Supports Directory Links 120
New Default LMFS Translation Table for Sys Hosts 119
Load Microcode FEP command 134
Load Sync-program FEP command 134, 135
Load World FEP command 134
load-file-list function 39
load-file-list obsolete 39
Loading Sync Programs 135
LOCAL key 133
Use **cdr** with locatives returned by **locf** 148
Use **cdr** with locatives returned by **locf** 148
locf macro 148
Lock queue 71
Round-robin locking 71
Logging host output to file 114
Logging in 1, 5
Create new logical host 37
Creating logical host 38
Logical Pathname Name, Type, and Version Now Separated by Periods 35
Changes to Logical Pathname Translations 35
New Default Representations for Newest and Oldest Logical Pathname Versions 35
Changes to Logical Pathnames 35
Logical Pathnames Now Hierarchical 35
Changes to **login** 5
login function 5
cl: **long-float** format 17
loop macro 82
Break loops 116
Compiler now warns about implicit **progn**s in **loops** 82
Lowercase Code in Buffer (m-X) Zwei command 125
Lowercase Code in Region (m-X) Zwei command 125

M M M

FUNCTION M command 113
NETWORK M command 114
Changes to FUNCTION C, FUNCTION M, and FUNCTION Q 113
FUNCTION m-C command 113
FUCTION m-Q command 113
m-SUSPEND command 116
m-SUSPEND selects frame with **break** read function for Debugger 116
Change in numeric arguments to Copy File (m-X) 124
Changes to Add Patch Changed Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X) 123
New Zmacs command: Resume Patch (m-X) 124
New Zmacs command: Source Compare Newest Definition (m-X) 124
New Zmacs command: Start Private Patch (m-X) 124
New Zwei command: Find Files in Tag Table (m-X) 125
Changes to Add Patch Changed Definitions (m-X) and Add Patch Changed Definitions of Buffer

(m-X) 123
Both default pathnames for Source Compare (m-X) now use **:newest** version 123
Set Package (m-X) offers to create a package 123
Show Hardcopy Status (m-X) replaces **chaos:print-lgp-queue** 115
Add Patch Changed Definitions (m-X) Zmacs command 123
Add Patch Changed Definitions of Buffer (m-X) Zmacs command 123
Copy File (m-X) Zmacs command 28, 124
Resume Patch (m-X) Zmacs command 124
Set Package (m-X) Zmacs command 123
Source Compare Newest Definition (m-X) Zmacs command 124
Start Private Patch (m-X) Zmacs command 124
Append Conversation By References (m-X) Zmail command 130
Delete Conversation By References (m-X) Zmail command 130
Select All Conversations By References (m-X) Zmail command 130
Select Conversation By References (m-X) Zmail command 130
Lowercase Code in Buffer (m-X) Zwei command 125
Lowercase Code in Region (m-X) Zwei command 125
Show Hardcopy Status (m-X) Zwei command 115
Source Compare (m-X) Zwei command 123
Uppercase Code in Buffer (m-X) Zwei command 125
Uppercase Code in Region (m-X) Zwei command 125
Find Files in Tag Table (m-X)) Zwei command 125
#B reader **macro** 18
locf macro 148
loop macro 82
meter:with-monitoring macro 52
package-declare macro 7
swapf macro 82
sys:defsubst-with-parent macro 145
sys:with-open-file-search macro 69
tv:with-mouse-grabbed-on-sheet macro 74
with-input-editing macro 59
zwei:defmajor macro 127
zwei:defminor macro 127
Macro character 17
Macro PC status information 137
si: **macro** syntax description 18
New reader macro: **#B** 18
Previously undocumented reader macro: **#** and **#** 83
Previously undocumented macro: **swapf** 82
New macro: **sys:with-open-file-search** 69
New macro: **tv:with-mouse-grabbed-on-sheet** 74
New macro: **with-input-editing** 59
Internal changes to macros **zwei:defmajor** and **zwei:defminor** 127
New function to be called by reader macros: **si:read-recursive** 83
[Reload/Retrieve] File System Maintenance menu item 151
Major and Minor mode system 127
Major-mode-setting command 126
Major-mode-setting Commands Now Query About Updating File Attribute List 126
:displaced-conformally option for **make-array** 78
make-array function 16
Nonkeyword form of **make-array** is obsolete 16
si: **make-coroutine-bidirectional-stream** function 91
si: **make-coroutine-input-stream** function 91
si: **make-coroutine-output-stream** function 91
fs: **make-logical-pathname-host** function 37
fs: **make-logical-pathname-host** replaces

fs:add-logical-pathname-host 37
New function: **make-package** 7
make-package function 7
meter: **make-pc-array** function 51
:default-value option to **make-plane** 80
:extension option to **make-plane** 80
:initial-dimensions option to **make-plane** 80
:initial-origins option to **make-plane** 80
:type option to **make-plane** 80
make-plane function 80
New options for **make-plane: :initial-dimensions, :initial-origins** 80
si: **make-process-queue** function 71
New functions: **si:** **make-process-queue, si:process-enqueue, si:process-dequeue,** si:process-queue-locker, **si:reset-process-queue** 71
si: **make-serial-stream** function 110, 111
chaos: **make-stream** function 107
Changes to **make-syn-stream** 20
make-syn-stream function 20
Improvements to **make-system: error-restart,** selective transformations 94
Release 5.0: Operations and Site Management 151
[Reply] Map Over menu item 130
[Select Conversation] Map Over menu item 130
New [map Over] Menu Item: [reply] 130
New [map Over] Menu Item: [select Conversation] 130
meter: **map-over-functions-in-bucket** function 52
Functions moved from the **si** package to **global: deallocate-whole-resource, map-resource** 13
map-resource function 14
Optional argument to **mapatoms-all** and **where-is** eliminated 7
mapatoms-all function 7
Wildcard Directory Mapping Available 92
:set-space method of **tv:** **margin-space-mixin** 75
:space init option for **tv:** **margin-space-mixin** 75
:space method of **tv:** **margin-space-mixin** 75
New flavor: **tv:** **margin-space-mixin** 74
tv: **margin-space-mixin** flavor 75
Meaning of argument changed for **fs:parse-pathname** 32
New Buffer-history Mechanism in Zmacs 125
Memory Board Not Needed in Lbus Slot 0 136
nil not a valid menu item 149
[Read File] Font Editor menu item 113
[Reload/Retrieve] File System Maintenance menu item 151
[Reply] Map Over menu item 130
[Select Conversation] Map Over menu item 130
[Sort] Zmail menu item 130
[Write File] Font Editor menu item 113
New [map Over] Menu Item: [reply] 130
New [map Over] Menu Item: [select Conversation] 130
Argument to **:menu** type menu items can be a menu or a form 105
Argument to **:menu** type menu items can be a menu or a form 105
Argument to **:menu** type menu items can be a menu or a form 105
tv: **menu-choose** function 149
New message to arithmetic errors: **:operands** 98

New message to conditions: **:special-command-p** 74
:rename message to pathnames 27
:create-screen-array message to screens 79
:redirect-screen-array message to screens 79
:adjust-screen-array message to sheets 79
:rename message to streams 27
:prompt-and-read messages to streams 85
:clear-screen, **:clear-eol**, and **:clear-eof** messages to windows renamed 44
META key 102
sys:%count-disk-page-read-operations-in-scavenger meter 52
sys:%count-disk-page-read-operations-in-transporter meter 52
sys:%scavenger-run-time meter 52
sys:%transporter-run-time meter 52
sys:%tv-clock-counter meter 52
meter:expand-range function 51
meter:function-name-with-escapes function 52
meter:function-range function 52
meter:list-functions-in-bucket function 51
meter:make-pc-array function 51
meter:map-over-functions-in-bucket function 52
meter:monitor-all-functions function 51
meter:print-functions-in-bucket function 51
meter:range-of-bucket function 52
meter:report function 51
meter:start-monitor function 51
meter:stop-monitor function 51
meter:with-monitoring macro 52
Program counter (PC) metering 50
New Metering Tools for the 3600 50
New Meters for the LM-2 52
:mouse-click method 149
:special-command-p method of **condition** 74
:read-frame method of **si:serial-hdlc-mixin** 111
:write-frame method of **si:serial-hdlc-mixin** 111
:mouse-click method of **tv:essential-mouse** 42
Changes to **:mouse-click** method of **tv:essential-mouse** 42
:draw-circle method of **tv:graphics-mixin** 99
:draw-filled-in-circle method of **tv:graphics-mixin** 99
:set-space method of **tv:margin-space-mixin** 75
:space method of **tv:margin-space-mixin** 75
:clear-eof method of **tv:sheet** 44
:clear-eol method of **tv:sheet** 44
:clear-rest-of-line method of **tv:sheet** 44
:clear-rest-of-window method of **tv:sheet** 44
:clear-screen method of **tv:sheet** 44
:clear-window method of **tv:sheet** 44
:set-truncate-line-out method of **tv:sheet** 102
:truncate-line-out method of **tv:sheet** 102
:any-tyi method of **tv:stream-mixin** 41, 42
:any-tyi-no-hang method of **tv:stream-mixin** 41
:finish-typeout method of **tv:stream-mixin** 64
:force-rescan method of **tv:stream-mixin** 64
:list-tyi method of **tv:stream-mixin** 41
:mouse-or-kbd-tyi method of **tv:stream-mixin** 42
:mouse-or-kbd-tyi-no-hang method of **tv:stream-mixin** 42
:read-bp method of **tv:stream-mixin** 65

:replace-input method of **tv:stream-mixin** 65
:rescanning-p method of **tv:stream-mixin** 64
:start-typeout method of **tv:stream-mixin** 63
:tyi method of **tv:stream-mixin** 41
:tyi-no-hang method of **tv:stream-mixin** 41
Some Methods Can Use Combination Type as Method Type 83
Removing methods 53
:proceed methods can now return **nil** 97
Some Methods Can Use Combination Type as Method Type 83
Changes to **:tyi**, **:tyi-no-hang**, **:list-tyi**, **:mouse-or-kbd-tyi**, and **:mouse-or-kbd-tyi-no-hang** methods of **tv:stream-mixin** 40
New methods of **tv:stream-mixin**: **:start-typeout**, :finish-typeout, **:rescanning-p**, **:force-rescan**, **:replace-input**, **:read-bp** 63
Current micro PC (CPC) status information 137
Load Microcode FEP command 134
New Microcode in Release 5.0: 270 on 3600, 998 on LM-2 3
Clicking Middle Edits Current String in Choose-variable-values Windows 105
Major and Minor mode system 127
New option for **defflavor**: **:mixture** 54
:mixture option 54
:mixture option for **defflavor** 54
New function: **mod** 49
mod function 49
Major and Minor mode system 127
Interface to the Vadic Modem 111
si: **modem** flavor 111
si: **modem-error** flavor 111
Known problem: **char-upcase** and **char-downcase** undefined for modified characters 145
Modified mouse clicks as editor commands 103
Change in Zmacs command Modified Two Windows (c-X 4) 127
Modified Two Windows (c-X 4) Zmacs command 127
Characters with modifier bits 145
meter: **monitor-all-functions** function 51
Show Configuration Command Displays More Information 135
More Information Available on Causes of Crashes 137
MORE processing 113, 114
Show Status Command Displays More Useful Information 140
New option for **si:sb-on**: **:mouse** (3600 only) 95
Mouse characters 102
Receiving mouse clicks 40
Modified mouse clicks as editor commands 103
Shifted Mouse Clicks Can Now Be Used for Editor Commands 103
Mouse cursor speed 100
Complement mouse documentation line 113
Mouse input 42
Window System Changes Associated with Mouse Input 40
:mouse option for **si:sb-on** 95
Avoid Errors in the Mouse Process 149
Mouse sequence break 95
:mouse symbol 42

:mouse-click method 149
:mouse-click method of **tv:essential-mouse** 42
Changes to **:mouse-click** method of **tv:essential-mouse** 42
tv: **mouse-double-click-time** variable 103
tv: ***mouse-incrementing-keystates*** variable 103
New variable: **tv:** ***mouse-modifying-keystates*** 102
tv: ***mouse-modifying-keystates*** variable 103
:mouse-or-kbd-tyi method of **tv:stream-mixin** 42
Changes to **:tyi**, **:tyi-no-hang**, **:list-tyi**, **:mouse-or-kbd-tyi**, and **:mouse-or-kbd-tyi-no-hang** methods of **tv:stream-mixin** 40
:mouse-or-kbd-tyi-no-hang method of **tv:stream-mixin** 42
Changes to **:tyi**, **:tyi-no-hang**, **:list-tyi**, **:mouse-or-kbd-tyi**, and **:mouse-or-kbd-tyi-no-hang** methods of **tv:stream-mixin** 40
New optional argument to **tv:** **mouse-wait** 101
tv: **mouse-wait** function 101
Previously undocumented variables: **sys:** **mouse-x-scale-array** and **sys:mouse-y-scale-array** (LM-2 only) 100
sys: **mouse-x-scale-array** variable 100
Previously undocumented variables: **sys:mouse-x-scale-array** and **sys:** **mouse-y-scale-array** (LM-2 only) 100
sys: **mouse-y-scale-array** variable 101
Functions moved from the **si** package to **global**: deallocate-whole-resource, **map-resource** 13
Symbols moved to or from **global** package 9
New canonical file type: **:mss** 126
Multidimensional Arrays on the 3600 Remember Actual Dimensions 78
New condition flavor: **fs:** **multiple-file-not-found** 69
fs: **multiple-file-not-found** flavor 70
New special forms: **multiple-value-call** and **multiple-value-prog1** 47
multiple-value-call special form 47
New special forms: **multiple-value-call** and **multiple-value-prog1** 47
multiple-value-prog1 special form 48

N N N

Display process name 148
Default File Name Changed for Commands in Dired Buffer 126
FUNCTION 2 W displays current process name in status line 148
Logical Pathname Name, Type, and Version Now Separated by Periods 35
Symbols in **global** and **keyword** packages with the same names 7
Network namespace 143
Namespace editor 107
Network namespace system 1, 107
NCP 109
Babyl files with **summary-window-format** other than **t** or **nil** need to be edited 129
Memory Board Not Needed in Lbus Slot 0 136
net:find-objects-from-property-list function 143
neti:disable function 107, 108
neti:reset, **neti:enable**, and **neti:disable** replace **chaos:reset**, **chaos:enable**, and **chaos:disable** 107
neti:enable function 107, 108
neti:reset, **neti:enable**, and **neti:disable** replace **chaos:reset**,

chaos:enable, and **chaos:disable** 107
neti:reset function 107, 108
neti:reset, **neti:enable**, and **neti:disable** replace **chaos:reset**, **chaos:enable**, and **chaos:disable** 107
International dial network 151
NETWORK A command 114
NETWORK commands 114
NETWORK D command 114
Network database 107
NETWORK L command 114
NETWORK M command 114
Network namespace 143
Network namespace system 1, 107
NETWORK Q command 114
DIAL network type 151
NETWORK X command 114
Networks 1
Changes to Networks in Release 5.0 107
Incompatible Changes to Networks in Release 5.0 107
New Features in Networks in Release 5.0 108
New Buffer-history Mechanism in Zmacs 125
New canonical file type: **:mss** 126
New Choose-variable-values Keywords 76
New clause for **condition-call**: **:no-error** 98
New condition flavor: **fs:multiple-file-not-found** 69
New condition flavor: **fs:rename-across-hosts** 70
New data types: **:single-float** and **:double-float** 49
New Default LMFS Translation Table for Sys Hosts 119
New Default Representations for Newest and Oldest Logical Pathname Versions 35
New Defaults for FEP Commands 134
New descriptions: **si:bitscale**, **si:digitscale**, **si:non-terminating-macro** 17
New error flavors: **sys:parse-error** and **sys:parse-ferror** 23
New feature: Flavor Examiner (SELECT X) 114
FEP Version 14: New Features 133
FEP Version 15: New Features 135
FEP Version 16: New Features 136
New Features Associated with the Input Editor (Rubout Handler) 57
New Features in Lisp in Release 5.0 44
New Features in Networks in Release 5.0 108
New Features in the File System in Release 5.0 120
New Features in Utilities in Release 5.0 114
New Features in Zmacs in Release 5.0 124
New Features in Zmail in Release 5.0 130
New FEP Commands: Add Disk-type and Clear Disk-types 136
New flavor: **tv:margin-space-mixin** 74
New flavors: **tv:truncatable-lines-mixin**, **tv:truncating-lines-mixin** 102
New font: **fonts:cptfonti** 75
New **format** directives: ~→ and ~← 55
New format for **trace** output 96

New function to be called by reader macros: **si:read-recursive** 83
New function: **applyhook** 72
New function: **chaos:conn-finished-p** 109
New function: **dfloat** 49
New function: **eql** 44
New function: **make-package** 7
New function: **mod** 49
New function: **read-delimited-string** 60
New function: **readline-or-nil** 63
New function: **si:patch-loaded-p** 70
New function: **si:read-or-end** 62
New function: **sys:parse-ferror** 24
New function: **undefflavor** 53
New functions: **byte, byte-size, byte-position** 50
New functions: **si:make-process-queue**, si:process-enqueue, **si:process-dequeue**, si:process-queue-locker, **si:reset-process-queue** 71
New functions: **sys:single-float-p, sys:double-float-p** 49
New initialization list: **:after-full-gc** 73
New input editor options: **:no-input-save, :activation, :command, :preemptable** 57
Create new logical host 37
New macro: **sys:with-open-file-search** 69
New macro: **tv:with-mouse-grabbed-on-sheet** 74
New macro: **with-input-editing** 59
New message to arithmetic errors: **:operands** 98
New message to conditions: **:special-command-p** 74
New Metering Tools for the 3600 50
New Meters for the LM-2 52
New methods of **tv:stream-mixin**: **:start-typeout, :finish-typeout, :rescanning-p**, :force-rescan, **:replace-input, :read-bp** 63
New Microcode in Release 5.0: 270 on 3600, 998 on LM-2 3
New **open** option: **:estimated-length** 25
New **open** options: **:if-exists** and **:if-does-not-exist** 25
New option for **defflavor**: **:mixture** 54
New option for **defflavor**: **:required-init-keywords** 53
New option for **si:sb-on**: **:mouse** (3600 only) 95
New optional argument to **gc-immediately** 98
New optional argument to **read** 61
New optional argument to **tv:mouse-wait** 101
New optional arguments to **print-notifications** 99
New optional arguments to **read-from-string** 84
New optional arguments to **string-upcase** and **string-downcase** 80
New options for **make-plane**: **:initial-dimensions, :initial-origins** 80
New reader macro: **#B** 18
New Rules for Reading Ambiguous Tokens 19
New special form: **defconstant** 45
New special form: **define-symbol-macro** 53
New special form: **defpackage** 7

New special form: **format:defformat** 56
New special forms **catch** and **throw** replace ***catch** and ***throw** 14
New special forms: **block** and **tagbody** 46
New special forms: **multiple-value-call** and **multiple-value-prog1** 47
New terminal program (SELECT T) 114
New variable: **cl:*read-default-float-format*** 17
New variable: **dbg:*debug-io-override*** 74
New variable: **fs:*remember-passwords*** 70
New variable: **gc-on** 73
New variable: **tv:*mouse-modifying-keystates*** 102
New variable: **tv:cold-load-stream-old-selected-window** 74
New variable: **tv:rh-typeout-default** 65
New Zmacs command: Resume Patch (m-X) 124
New Zmacs command: Source Compare Newest Definition (m-X) 124
New Zmacs command: Start Private Patch (m-X) 124
New Zwei command: Comment Out Region (c-X c-;) 125
New Zwei command: Find Files in Tag Table (m-X) 125
New [map Over] Menu Item: [reply] 130
New [map Over] Menu Item: [select Conversation] 130
:new-version value of **open** option for **:if-exists** 25
New Default Representations for Newest and Oldest Logical Pathname Versions 35
New Zmacs command: Source Compare Newest Definition (m-X) 124
Source Compare Newest Definition (m-X) Zmacs command 124
Both default pathnames for Source Compare (m-X) now use **:newest** version 123
> newest version specifier 35
:proceed methods can now return **nil** 97
sort predicate should return **nil** for equal elements 144
Babyl files with **summary-window-format** other than **t** or **nil** need to be edited 129
nil not a valid menu item 149
nil value of **open** option **:if-does-not-exist** 25
New clause for **condition-call:** **:no-error** 98
:no-input-save option 57
New input editor options: **:no-input-save**, **:activation**, **:command**, **:preemptable** 57
New descriptions: **si:bitscale**, **si:digitscale**, **si:** **non-terminating-macro** 17
Nonkeyword form of **make-array** is obsolete 16
nil not a valid menu item 149
Unplugging Lemo Cables Should Not Halt the FEP 137
Memory Board Not Needed in Lbus Slot 0 136
Inspecting hash arrays of **eq** hash tables not permitted 145
tv:set-default-font not supported 149
store not supported on the 3600 145
Release 5.0: Notes and Clarifications 143
Notes on Operations in Release 5.0 151
Changes to Converse Notifications 117
Reprints notifications 99
Shifted Mouse Clicks Can Now Be Used for Editor Commands 103

Default Font Format Now Bfd 113
)Configuration.fep Files Are Now Called)Boot.boot 134
Logical Pathnames Now Hierarchical 35
#/ and **#** now identical 19
Second argument to **si:install-microcode** now optional 94
Major-mode-setting Commands Now Query About Updating File Attribute List 126
:proceed methods can now return **nil** 97
Logical Pathname Name, Type, and Version Now Separated by Periods 35
LMFS Now Supports Directory Links 120
Both default pathnames for Source Compare (m-X) now use **:newest** version 123
Compiler now warns about implicit **progn**s in **loop**s 82
float returns a single-precision number 48
floatp returns **t** for any floating-point number 49
Read rational number in binary 18
:number option for **prompt-and-read** 85
Convert number to single-precision floating-point 48
Convert number to small flonum 48
:number-or-nil option for **prompt-and-read** 85
Floating point numbers 1
Floating-point numbers 48, 49
Converting numbers to double-precision floating-point 49
Change in numeric arguments to Copy File (m-X) 124

O O O

Reading and Printing Character Objects 19
Deallocating allocated objects of a resource 13
Forgetting objects remembered by a resource 78
Flavor **tv:preemptable-read-any-tyi-mixin** obsolete 43
Flavors **tv:any-tyi-mixin** and **tv:list-tyi-mixin** obsolete 40
Flavors **tv:list-mouse-buttons-mixin** and **tv:kbd-mouse-buttons-mixin** obsolete 42
load-file-list obsolete 39
Nonkeyword form of **make-array** is obsolete 16
Changes to **format:** **ochar** 21
format: **ochar** function 21
Octal escape for special characters 17
Set Package (m-X) offers to create a package 123
Old PCs (OPC) status information 137
New Default Representations for Newest and Oldest Logical Pathname Versions 35
< oldest version specifier 35
New option for **si:sb-on**: **:mouse** (3600 only) 95
Previously undocumented variables: **sys:mouse-x-scale-array** and **sys:mouse-y-scale-array** (LM-2 only) 100
Old PCs (OPC) status information 137
:if-does-not-exist option for **open** 25
:if-exists option for **open** 25
Changes to **open** 24
:input value of **open** option **:direction** 24
:output value of **open** option **:direction** 24
:probe value of **open** option **:direction** 24
:probe-directory value of **open** option **:direction** 24
:probe-link value of **open** option **:direction** 24
Changes to **open** option **:direction** 24
:create value of **open** option **:if-does-not-exist** 25
:error value of **open** option **:if-does-not-exist** 25
nil value of **open** option **:if-does-not-exist** 25
:append value of **open** option for **:if-exists** 25

:error value of **open** option for **:if-exists** 25
:new-version value of **open** option for **:if-exists** 25
:overwrite value of **open** option for **:if-exists** 25
:rename value of **open** option for **:if-exists** 25
:rename-and-delete value of **open** option for **:if-exists** 25
:supersede value of **open** option for **:if-exists** 25
:truncate value of **open** option for **:if-exists** 25
New **open** option: **:estimated-length** 25
open options 24, 25
New **open** options: **:if-exists** and **:if-does-not-exist** 25
Changes to **chaos:** **open-stream** 108
chaos: **open-stream** function 108
File opened for input 24
File opened for output 24
Probe file opening 24
Opening pathname 24
New message to arithmetic errors: **:operands** 98
3600 select-methods handle **:operation-handled-p** and **:send-if-handles** 81
Release 5.0: Operations and Site Management 151
Notes on Operations in Release 5.0 151
Optimizing disk allocation 25
:activation option 57
:command option 58
:direction option 24
:do-not-echo option 22
:estimated-length option 25
:if-does-not-exist option 26
:if-exists option 25
:mixture option 54
:no-input-save option 57
:pass-through option 22
:preemptable option 43, 58
:prompt option 23
:reprompt option 23
:required-init-keywords option 54
summary-window-format Babyl file option 129
:input value of **open** option **:direction** 24
:output value of **open** option **:direction** 24
:probe value of **open** option **:direction** 24
:probe-directory value of **open** option **:direction** 24
:probe-link value of **open** option **:direction** 24
Changes to **open** option **:direction** 24
:create value of **open** option **:if-does-not-exist** 25
:error value of **open** option **:if-does-not-exist** 25
nil value of **open** option **:if-does-not-exist** 25
:append value of **open** option for **:if-exists** 25
:error value of **open** option for **:if-exists** 25
:new-version value of **open** option for **:if-exists** 25
:overwrite value of **open** option for **:if-exists** 25
:rename value of **open** option for **:if-exists** 25
:rename-and-delete value of **open** option for **:if-exists** 25
:supersede value of **open** option for **:if-exists** 25
:truncate value of **open** option for **:if-exists** 25
:byte-size option for **copyf** 28
:characters option for **copyf** 28
:create-directories option for **copyf** 28
:report-stream option for **copyf** 28
:mixture option for **defflavor** 54

:required-init-keywords option for **defflavor** 54
New option for **defflavor**: **:mixture** 54
New option for **defflavor**: **:required-init-keywords** 53
:displaced-conformally option for **make-array** 78
:if-does-not-exist option for **open** 25
:if-exists option for **open** 25
:as-if-band option for **print-herald** 39
:verbose option for **print-herald** 39
:character option for **prompt-and-read** 85
:date option for **prompt-and-read** 85
:date-or-never option for **prompt-and-read** 85
:decimal-number option for **prompt-and-read** 85
:decimal-number-or-nil option for **prompt-and-read** 85
:delimited-string option for **prompt-and-read** 85
:delimited-string-or-nil option for **prompt-and-read** 85
:delimiter option for **prompt-and-read** 85
:eval-form option for **prompt-and-read** 85
:eval-form-or-end option for **prompt-and-read** 85
:expression option for **prompt-and-read** 85
:expression-or-end option for **prompt-and-read** 85
:font-list option for **prompt-and-read** 85
:host option for **prompt-and-read** 85
:host-list option for **prompt-and-read** 85
:keyword-list option for **prompt-and-read** 85
:number option for **prompt-and-read** 85
:number-or-nil option for **prompt-and-read** 85
:past-date option for **prompt-and-read** 85
:past-date-or-never option for **prompt-and-read** 85
:pathname option for **prompt-and-read** 85
:pathname-host option for **prompt-and-read** 85
:pathname-list option for **prompt-and-read** 85
:pathname-or-nil option for **prompt-and-read** 85
:string option for **prompt-and-read** 85
:string-list option for **prompt-and-read** 85
:string-or-nil option for **prompt-and-read** 85
:string-trim option for **prompt-and-read** 85
:time-interval-or-never option for **prompt-and-read** 85
:chaos option for **si:sb-on** 95
:clock option for **si:sb-on** 95
:disk option for **si:sb-on** 95
:keyboard option for **si:sb-on** 95
:mouse option for **si:sb-on** 95
:unibus option for **si:sb-on** 95
New option for **si:sb-on**: **:mouse** (3600 only) 95
:space init option for **tv:margin-space-mixin** 75
:default-value option to **make-plane** 80
:extension option to **make-plane** 80
:initial-dimensions option to **make-plane** 80
:initial-origins option to **make-plane** 80
:type option to **make-plane** 80
New **open** option: **:estimated-length** 25
Second argument to **si:install-microcode** now optional 94
New optional argument to **gc-immediately** 98
Optional argument to **mapatoms-all** and **where-is** eliminated 7
New optional argument to **read** 61
New optional argument to **tv:mouse-wait** 101
New optional arguments to **print-notifications** 99

New optional arguments to **read-from-string** 84
New optional arguments to **string-upcase** and **string-downcase** 80
open options 24, 25
Changes to input editor options **:do-not-echo**, **:pass-through**, **:prompt**, **:reprompt** 22
New options for **make-plane**: **:initial-dimensions**, **:initial-origins** 80
New **open** options: **:if-exists** and **:if-does-not-exist** 25
New input editor options: **:no-input-save**, **:activation**, **:command**, **:preemptable** 57
Babyl files with **summary-window-format** other than **t** or **nil** need to be edited 129
New Zwei command: Comment Out Region (c-X c-;) 125
Comment Out Region (c-X c-;) Zwei command 125
Character output 21
File opened for output 24
New format for **trace** output 96
:editor output format style 21
:read output format style 21
:sail output format style 21
Output space 21
Logging host output to file 114
:output value of **open** option **:direction** 24
:output-xoff-character serial I/O parameter 109
:output-xon-character serial I/O parameter 109
Overstrike 114
:overwrite value of **open** option for **:if-exists** 25
New [map Over] Menu Item: [reply] 130
New [map Over] Menu Item: [select Conversation] 130

P P P

fonts package 6
global package 1, 6
keyword package 1, 6
Set Package (m-X) offers to create a package 123
Symbols moved to or from **global** package 9
user package 1, 6
Set Package (m-X) offers to create a package 123
Set Package (m-X) Zmacs command 123
sys:dump-forms-to-file always puts package attribute into binary file 82
format ~\ directives can have package prefixes 92
global package symbols 9
keyword package symbols 7
Functions moved from the **si** package to **global**: **deallocate-whole-resource**, **map-resource** 13
package-declare macro 7
package-used-by-list function 7
Packages 1
Changes to Packages 6
Symbols in **global** and **keyword** packages with the same names 7
chaos:send-unc-pkt automatically returns the packet to the free pool 108
:input-error-character serial I/O parameter 109
:input-xoff-character serial I/O parameter 109
:input-xon-character serial I/O parameter 109
:output-xoff-character serial I/O parameter 109
:output-xon-character serial I/O parameter 109
Stream parameter 65

Changes to Serial I/O: Parity Recovery and Xon/Xoff Character Setting 109
New error flavors: **sys:** **parse-error** and **sys:parse-ferror** 23
sys: **parse-error** flavor 23
New error flavors: **sys:parse-error** and **sys:** **parse-ferror** 23
New function: **sys:** **parse-ferror** 24
sys: **parse-ferror** flavor 24
sys: **parse-ferror** function 24
Meaning of argument changed for **fs:** **parse-pathname** 32
fs: **parse-pathname** function 33
Parsing pathnames 33
:pass-through option 22
Changes to input editor options **:do-not-echo,** **:pass-through, :prompt, :reprompt** 22
Suppress prompting for passwords 70
:past-date option for **prompt-and-read** 85
:past-date-or-never option for **prompt-and-read** 85
New Zmacs command: Resume Patch (m-X) 124
New Zmacs command: Start Private Patch (m-X) 124
Resume Patch (m-X) Zmacs command 124
Start Private Patch (m-X) Zmacs command 124
Changes to Add Patch Changed Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X) 123
Add Patch Changed Definitions (m-X) Zmacs command 123
Changes to Add Patch Changed Definitions (m-X) and Add Patch Changed Definitions of Buffer (m-X) 123
Add Patch Changed Definitions of Buffer (m-X) Zmacs command 123
Patch level 70
New function: **si:** **patch-loaded-p** 70
si: **patch-loaded-p** function 70
Patches 123
Opening pathname 24
Source pathname 26
Target pathname 26
Pathname completion on VMS 109
Logical Pathname Name, Type, and Version Now Separated by Periods 35
:pathname option for **prompt-and-read** 85
Reversible wild pathname translation 35
Wild pathname translation 35
Changes to Logical Pathname Translations 35
Physical pathname translations 35
:init canonical pathname type removed 35
New Default Representations for Newest and Oldest Logical Pathname Versions 35
:pathname-host option for **prompt-and-read** 85
:pathname-list option for **prompt-and-read** 85
:pathname-or-nil option for **prompt-and-read** 85
Pathnames 1
:rename message to pathnames 27
Changes to Host Determination in Pathnames 30
Changes to Logical Pathnames 35
Parsing pathnames 33
Both default pathnames for Source Compare (m-X) now use **:newest** version 123
Logical Pathnames Now Hierarchical 35
Init File Pathnames Standardized 34
Zmail Init File Pathnames Standardized 129

Current micro PC (CPC) status information 137
Macro PC status information 137
Program counter (PC) metering 50
Old PCs (OPC) status information 137
Compiler Performs Style Checking on All Forms 81
Logical Pathname Name, Type, and Version Now Separated by Periods 35
Inspecting hash arrays of **eq** hash tables not permitted 145
Physical pathname translations 35
pkg-create-package function 7
3600 Supports Ieee Single- and Double-precision Floating Point 48
Reader Accepts Common Lisp Floating Point Exponents 17
Floating point numbers 1
chaos:send-unc-pkt automatically returns the packet to the free pool 108
Jump to Saved Position (c-X J) Zmacs command 127
Extract position field of a byte-specifier 50
sort predicate should return **nil** for equal elements 144
New input editor options: **:no-input-save**, **:activation**, **:command**, **:preemptable** 57
:preemptable option 43, 58
tv: **preemptable-read-any-tyi-mixin** flavor 43
Flavor **tv:** **preemptable-read-any-tyi-mixin** obsolete 43
format ~\ directives can have package prefixes 92
Select Previous Buffer (c-m-L) Zmacs command 125
Yanking previous input 22
Previously Undocumented Feature: Coroutine Streams 89
Previously undocumented function: **clear-resource** 78
Previously undocumented function: **describe-system** 94
Previously undocumented function: **fs:set-logical-pathname-host** 38
Previously undocumented function: **string-compare** 81
Previously undocumented functions: tv:add-to-system-menu-programs-column, **tv:add-to-system-menu-create-menu** 103
Previously undocumented macro: **swapf** 82
Previously undocumented reader macro: **#** and **#** 83
Previously undocumented special form: **destructuring-bind** 77
Previously undocumented variables: sys:mouse-x-scale-array and **sys:mouse-y-scale-array** (LM-2 only) 100
Ratios read in current **ibase** and print in current **base** 19
Print queue 115
meter: **print-functions-in-bucket** function 51
:as-if-band option for **print-herald** 39
:verbose option for **print-herald** 39
Change in arguments to **print-herald** 39
print-herald function 39
Show Hardcopy Status (m-X) replaces **chaos:** **print-lgp-queue** 115
chaos: **print-lpg-queue** function 115
Symbol print-name quoter 17

New optional arguments to **print-notifications** 99
print-notifications function 99
Changes to Readtable, Reader, and Printer for Common Lisp 16
Reading and Printing Character Objects 19
Printing Uninterned Symbols 19
New Zmacs command: Start Private Patch (m-X) 124
Start Private Patch (m-X) Zmacs command 124
Probe file opening 24
:probe value of **open** option **:direction** 24
:probe-directory value of **open** option **:direction** 24
:probe-link value of **open** option **:direction** 24
Known problem with **si:gc-reclaim-immediately** 148
Known problem: **char-upcase** and **char-downcase** undefined for modified characters 145
:proceed methods can now return **nil** 97
Avoid Errors in the Mouse Process 149
Debugger c-M creates a process 116
Get Common Property Lists process 143
Display process name 148
FUNCTION 2 W displays current process name in status line 148
si: **process-dequeue** function 71
New functions: **si:make-process-queue, si:process-enqueue, si:process-dequeue, si:process-queue-locker, si:reset-process-queue** 71
si: **process-enqueue** function 71
New functions: **si:make-process-queue, si:** **process-enqueue, si:process-dequeue**, si:process-queue-locker, **si:reset-process-queue** 71
si: **process-queue-locker** function 71
New functions: **si:make-process-queue, si:process-enqueue, si:process-dequeue, si:process-queue-locker, si:reset-process-queue** 71
Change in argument to **process-wait-with-timeout** 95
process-wait-with-timeout function 95
MORE processing 113, 114
Compiler now warns about implicit **progns** in **loops** 82
Terminal program 1
New terminal program (SELECT T) 114
Program counter (PC) metering 50
Displaying program counters information 137
Loading Sync Programs 135
Communication between programs and input editor 63
Invisible blocks in **progs** and **dos** 78
Adding prompt 65
:prompt option 23
Changes to input editor options **:do-not-echo, :pass-through, :prompt, :reprompt** 22
:character option for **prompt-and-read** 85
:date option for **prompt-and-read** 85
:date-or-never option for **prompt-and-read** 85
:decimal-number option for **prompt-and-read** 85
:decimal-number-or-nil option for **prompt-and-read** 85
:delimited-string option for **prompt-and-read** 85
:delimited-string-or-nil option for **prompt-and-read** 85
:delimiter option for **prompt-and-read** 85
:eval-form option for **prompt-and-read** 85
:eval-form-or-end option for **prompt-and-read** 85
:expression option for **prompt-and-read** 85

:expression-or-end option for **prompt-and-read** 85
:font-list option for **prompt-and-read** 85
:host option for **prompt-and-read** 85
:host-list option for **prompt-and-read** 85
:keyword-list option for **prompt-and-read** 85
:number option for **prompt-and-read** 85
:number-or-nil option for **prompt-and-read** 85
:past-date option for **prompt-and-read** 85
:past-date-or-never option for **prompt-and-read** 85
:pathname option for **prompt-and-read** 85
:pathname-host option for **prompt-and-read** 85
:pathname-list option for **prompt-and-read** 85
:pathname-or-nil option for **prompt-and-read** 85
:string option for **prompt-and-read** 85
:string-list option for **prompt-and-read** 85
:string-or-nil option for **prompt-and-read** 85
:string-trim option for **prompt-and-read** 85
:time-interval-or-never option for **prompt-and-read** 85
Changes to **prompt-and-read** 84
prompt-and-read function 85
:prompt-and-read messages to streams 85
Prompting for input from user 85
Suppress prompting for passwords 70
compiler:style-checker property 81
Get Common Property Lists process 143
sys:dump-forms-to-file always puts package attribute into binary file 82

Q

Changes to FUNCTION C, FUNCTION M, and FUNCTION Q 113
FUNCTION Q command 113
NETWORK Q command 114
:qbin canonical type 34
Major-mode-setting Commands Now Query About Updating File Attribute List 126
Disk Format Command Asks Different Question 134
Lock queue 71
Print queue 115
Unlock queue 71
QUOTE key 116
Character quoter 17
String quoter 17
Symbol print-name quoter 17

R

R Fed command 113
Ramifications of Host Colon Change for Babyl Files 129
meter: **range-of-bucket** function 52
Read rational number in binary 18
Ratios read in current **ibase** and print in current **base** 19
New optional argument to **read** 61
[Read File] Font Editor menu item 113
read function 61
m-SUSPEND selects frame with **break** read function for Debugger 116

Syntax errors in read functions 23
Ratios read in current **ibase** and print in current **base** 19
:read output format style 21
Read rational number in binary 18
New methods of **tv:stream-mixin**: **:start-typeout**, **:finish-typeout**, **:rescanning-p**, **:force-rescan**, :replace-input,
:read-bp 63
:read-bp method of **tv:stream-mixin** 65
New variable: **cl:** ***read-default-float-format*** 17
cl: ***read-default-float-format*** variable 17
New function: **read-delimited-string** 60
read-delimited-string function 60
si: ***read-extended-ibase-signed-number*** variable 20
si: ***read-extended-ibase-unsigned-number*** variable 19
:read-frame method of **si:serial-hdlc-mixin** 111
New optional arguments to **read-from-string** 84
read-from-string function 84
New function: **si:** **read-or-end** 62
read-or-end function 62
New function to be called by reader macros: **si:** **read-recursive** 83
si: **read-recursive** function 83
Comments for Lisp reader 83
Reader Accepts Common Lisp Floating Point Exponents 17
#B reader **macro** 18
New reader macro: **#B** 18
Previously undocumented reader macro: **#** and **#** 83
New function to be called by reader macros: **si:read-recursive** 83
Changes to Readtable, Reader, and Printer for Common Lisp 16
New Rules for Reading Ambiguous Tokens 19
Reading and Printing Character Objects 19
Reading from streams 65
Reading function to use input editor 59
readline and **readline-trim** return additional values 62
readline function 62
New function: **readline-or-nil** 63
readline-or-nil function 63
readline-trim function 62
readline and **readline-trim** return additional values 62
Common Lisp readtable 83
Changes to Readtable, Reader, and Printer for Common Lisp 16
Receiving blips 40
Receiving mouse clicks 40
record-source-file-name function 147
Use **record-source-file-name** instead of (**remprop** *symbol* **':source-file-name**) 147
Changes to Serial I/O: Parity Recovery and Xon/Xoff Character Setting 109
Recursion in Bound and Default Handlers Eliminated 97
Redefining functions 147
:redirect-screen-array message to screens 79
Redistributed- header fields 130
Append Conversation By References (m-X) Zmail command 130
Delete Conversation By References (m-X) Zmail command 130
Select All Conversations By References (m-X) Zmail command 130
Select Conversation By References (m-X) Zmail command 130

Change case of region 125
New Zwei command: Comment Out Region (c-X c-;) 125
Comment Out Region (c-X c-;) Zwei command 125
Lowercase Code in Region (m-X) Zwei command 125
Uppercase Code in Region (m-X) Zwei command 125
Changes to Networks in Release 5.0 107
Changes to the FEP in Release 5.0 133
Changes to the File System in Release 5.0 119
Changes to the Lisp Language and Compiler in Release 5.0 5
Changes to Utilities in Release 5.0 113
Changes to Zmacs in Release 5.0 123
Changes to Zmail in Release 5.0 129
Clarifications and Corrections for Release 5.0 143
Improvements to Lisp in Release 5.0 77
Improvements to Utilities in Release 5.0 116
Improvements to Zmacs in Release 5.0 126
Improvements to Zmail in Release 5.0 130
Incompatible Changes to Lisp in Release 5.0 5
Incompatible Changes to Networks in Release 5.0 107
Incompatible Changes to the File System in Release 5.0 119
Incompatible Changes to Utilities in Release 5.0 113
Incompatible Changes to Zmacs in Release 5.0 123
Incompatible Changes to Zmail in Release 5.0 129
New Features in Lisp in Release 5.0 44
New Features in Networks in Release 5.0 108
New Features in the File System in Release 5.0 120
New Features in Utilities in Release 5.0 114
New Features in Zmacs in Release 5.0 124
New Features in Zmail in Release 5.0 130
Notes on Operations in Release 5.0 151
New Microcode in Release 5.0: 270 on 3600, 998 on LM-2 3
Release 5.0: Introduction and Highlights 1
Release 5.0: Notes and Clarifications 143
Release 5.0: Operations and Site Management 151
Backup Tape Reliability 151
[Reload/Retrieve] File System Maintenance menu item 151
Multidimensional Arrays on the 3600 Remember Actual Dimensions 78
New variable: **fs:** ***remember-passwords*** 70
fs: ***remember-passwords*** variable 70
Forgetting objects remembered by a resource 78
Remove flavor 53
:init canonical pathname type removed 35
Removing methods 53
remprop function 147
Use **record-source-file-name** instead of (**remprop** *symbol* '**:source-file-name**) 147
Rename file 27
:rename message to pathnames 27
:rename message to streams 27
:rename value of **open** option for **:if-exists** 25
New condition flavor: **fs:** **rename-across-hosts** 70
fs: **rename-across-hosts** flavor 70
:rename-and-delete value of **open** option for **:if-exists** 25
:clear-screen, **:clear-eol**, and **:clear-eof** messages to windows renamed 44
chaos:stream, **chaos:close**, and **chaos:finish** renamed 107
Changes to **renamef** and **copyf** 26

renamef function 27
New special forms **catch** and **throw** replace ***catch** and ***throw** 14
neti:reset, **neti:enable**, and **neti:disable** replace **chaos:reset**, **chaos:enable**, and **chaos:disable** 107
format directives ~@T and ~@* replace ~X and ~G 21
:replace-input method of **tv:stream-mixin** 65
New methods of **tv:stream-mixin**: **:start-typeout**, **:finish-typeout**, **:rescanning-p**, **:force-rescan**, **:replace-input**, **:read-bp** 63
Show Hardcopy Status (m-X) replaces **chaos:print-lgp-queue** 115
fs:make-logical-pathname-host replaces **fs:add-logical-pathname-host** 37
Flavor **fs:undefined-logical-pathname-translation** replaces **fs:undefined-logical-pathname-directory** 37
si:halt replaces **sys:%halt** 133
New [map Over] Menu Item: [reply] 130
[Reply] Map Over menu item 130
meter: **report** function 51
:report-stream option for **copyf** 28
New Default Representations for Newest and Oldest Logical Pathname Versions 35
Reprints notifications 99
Changes to input editor options **:do-not-echo**, **:pass-through**, **:prompt**, **:reprompt** 22
:reprompt option 23
New option for **defflavor**: **:required-init-keywords** 53
:required-init-keywords option 54
:required-init-keywords option for **defflavor** 54
:rescanning-p method of **tv:stream-mixin** 64
New methods of **tv:stream-mixin**: **:start-typeout**, **:finish-typeout**, **:rescanning-p**, **:force-rescan**, **:replace-input**, **:read-bp** 63
Resent- header fields 130
chaos: **reset** function 107
neti: **reset** function 107, 108
neti:reset, **neti:enable**, and **neti:disable** replace **chaos:reset**, **chaos:enable**, and **chaos:disable** 107
neti: **reset**, **neti:enable**, and **neti:disable** replace **chaos:reset**, **chaos:enable**, and **chaos:disable** 107
New functions: **si:make-process-queue**, **si:process-enqueue**, **si:process-dequeue**, si:process-queue-locker, **si:reset-process-queue** 71
si: **reset-process-queue** function 71
Deallocating allocated objects of a resource 13
Forgetting objects remembered by a resource 78
Dumper Restarting and Append-to-tape Default 121
New Zmacs command: Resume Patch (m-X) 124
Resume Patch (m-X) Zmacs command 124
readline and **readline-trim** return additional values 62
:proceed methods can now return **nil** 97
sort predicate should return **nil** for equal elements 144
intern, **intern-local**, **intern-soft**, and **intern-local-soft** return two values 7
Return-keyboard-to-lisp FEP command 137
chaos: **return-pkt** function 108
Use **cdr** with locatives returned by **locf** 148
Change in type of array returned by **string-append** 16
Returning control from input editor 57
float returns a single-precision number 48

floatp returns **t** for any floating-point number 49
chaos:send-unc-pkt automatically returns the packet to the free pool 108
Reversible wild pathname translation 35
RFC822 domain addressing 130
Rfc822 Domain Addressing Supported 130
New variable: **tv:** **rh-typeout-default** 65
tv: **rh-typeout-default** variable 65
Round-robin locking 71
rplaca can be used with stack lists 148
rplaca function 148
rplacd function 148
Rubout handler 1, 22
Incompatible Changes to the Input Editor (Rubout Handler) 21
New Features Associated with the Input Editor (Rubout Handler) 57
string-length uses same coercion rules as **string** 16
New Rules for Reading Ambiguous Tokens 19

S

S

S

S exponent identifier 17
:sail output format style 21
:draw-filled-in-circle uses same algorithm as **:draw-circle** 99
Using **copy-array-portion** on the same array 145
string-length uses same coercion rules as **string** 16
Symbols in **global** and **keyword** packages with the same names 7
Jump to Saved Position (c-X J) Zmacs command 127
:chaos option for **si:** **sb-on** 95
:clock option for **si:** **sb-on** 95
:disk option for **si:** **sb-on** 95
:keyboard option for **si:** **sb-on** 95
:mouse option for **si:** **sb-on** 95
:unibus option for **si:** **sb-on** 95
si: **sb-on** function 95
New option for **si:** **sb-on**: **:mouse** (3600 only) 95
sys: **%scavenger-run-time** meter 52
Complement screen 113
:create-screen-array message to screens 79
:redirect-screen-array message to screens 79
tv: **scroll-maintain-list** function 105
tv: **scroll-maintain-list** init function can take arguments 105
Second argument to **si:install-microcode** now optional 94
CHNCP.GSF global section 109
Select All Conversations By References (m-X) Zmail command 130
Select Buffer (c-X B) Zmacs command 125
Select Conversation By References (m-X) Zmail command 130
New [map Over] Menu item: [select Conversation] 130
[Select Conversation] Map Over menu item 130
Select Previous Buffer (c-m-L) Zmacs command 125
SELECT T command 114
New terminal program (SELECT T) 114
SELECT X command 114
New feature: Flavor Examiner (SELECT X) 114
3600 select-methods handle **:operation-handled-p** and

:send-if-handles 81
tv: **select-or-create-window-of-flavor** function 104
Improvements to **make-system**: **error-restart**, selective transformations 94
m-SUSPEND selects frame with **break** read function for Debugger 116
Keyword Symbols Are Self-evaluating 13
SEND 109
3600 select-methods handle **:operation-handled-p** and **:send-if-handles** 81
chaos: **send-pkt** function 108
chaos: **send-unc-pkt** automatically returns the packet to the free pool 108
chaos: **send-unc-pkt** function 108
Continue Command Sends an All-keys-up Character to Lisp 137
Logical Pathname Name, Type, and Version Now Separated by Periods 35
Token separators 17
Chaosnet sequence break 95
Clock sequence break 95
Disk sequence break 95
Keyboard sequence break 95
Mouse sequence break 95
FEP Supports Hdlc Serial I/O 133
Hdlc Serial I/O on the 3600 110
:input-error-character serial I/O parameter 109
:input-xoff-character serial I/O parameter 109
:input-xon-character serial I/O parameter 109
:output-xoff-character serial I/O parameter 109
:output-xon-character serial I/O parameter 109
Changes to Serial I/O: Parity Recovery and Xon/Xoff Character Setting 109
:read-frame method of **si:** **serial-hdlc-mixin** 111
:write-frame method of **si:** **serial-hdlc-mixin** 111
si: **serial-hdlc-stream** flavor 110
Set Package (m-X) offers to create a package 123
Set Package (m-X) Zmacs command 123
zwei: ***set-attribute-updates-list*** variable 126
tv: **set-default-font** function 149
tv: **set-default-font** not supported 149
Previously undocumented function: **fs:** **set-logical-pathname-host** 38
fs: **set-logical-pathname-host** function 38
:set-space method of **tv:margin-space-mixin** 75
set-syntax-from-description function 18
:set-truncate-line-out method of **tv:sheet** 102
Changes to Serial I/O: Parity Recovery and Xon/Xoff Character Setting 109
Streams sharing common buffers 89
:clear-eof method of **tv:** **sheet** 44
:clear-eol method of **tv:** **sheet** 44
:clear-rest-of-line method of **tv:** **sheet** 44
:clear-rest-of-window method of **tv:** **sheet** 44
:clear-screen method of **tv:** **sheet** 44
:clear-window method of **tv:** **sheet** 44
:set-truncate-line-out method of **tv:** **sheet** 102
:truncate-line-out method of **tv:** **sheet** 102
:adjust-screen-array message to sheets 79
Binary left shift 17
Shifted Mouse Clicks Can Now Be Used for Editor Commands 103

cl: **short-float** format 17
Unplugging Lemo Cables Should Not Halt the FEP 137
sort predicate should return **nil** for equal elements 144
Show Configuration Command Displays More Information 135
Show Configuration FEP command 135
Show File FEP command 134
Show Hardcopy Status (m-X) replaces **chaos:print-lgp-queue** 115
Show Hardcopy Status (m-X) Zwei command 115
Show Status Command Displays More Useful Information 140
Show Status FEP command 137, 140
Functions moved from the **si** package to **global: deallocate-whole-resource, map-resource** 13
si:*read-extended-ibase-signed-number* variable 20
si:*read-extended-ibase-unsigned-number* variable 19
si:*trace-bar-p* variable 97
si:*trace-bar-rate* variable 97
si:*trace-columns-per-level* variable 97
si:*trace-old-style* variable 97
si:alphabetic syntax description 18
si:bitscale 17
New descriptions: **si:bitscale, si:digitscale, si:non-terminating-macro** 17
si:break syntax description 18
si:circlecross syntax description 18
si:coroutine-bidirectional-stream flavor 92
si:coroutine-input-stream flavor 92
si:coroutine-output-stream flavor 92
New descriptions: **si:bitscale, si:digitscale, si:non-terminating-macro** 17
si:doublequote syntax description 18
si:full-gc function 73
Known problem with **si:gc-reclaim-immediately** 148
si:gc-reclaim-immediately variable 148
si:halt function 133
si:halt replaces **sys:%halt** 133
si:install-microcode function 94
Second argument to **si:install-microcode** now optional 94
si:macro syntax description 18
si:make-coroutine-bidirectional-stream function 91
si:make-coroutine-input-stream function 91
si:make-coroutine-output-stream function 91
si:make-process-queue function 71
New functions: **si:make-process-queue, si:process-enqueue, si:process-dequeue,** si:process-queue-locker, **si:reset-process-queue** 71
si:make-serial-stream function 110, 111
si:modem flavor 111
si:modem-error flavor 111
New descriptions: **si:bitscale, si:digitscale, si:non-terminating-macro** 17
New function: **si:patch-loaded-p** 70
si:patch-loaded-p function 70
si:process-dequeue function 71
New functions: **si:make-process-queue, si:process-enqueue,**

si:process-dequeue, si:process-queue-locker, **si:reset-process-queue** 71
si:process-enqueue function 71
New functions: **si:make-process-queue**, **si:process-enqueue**, **si:process-dequeue**, **si:process-queue-locker**, **si:reset-process-queue** 71
si:process-queue-locker function 71
New functions: **si:make-process-queue**, **si:process-enqueue**, **si:process-dequeue**, **si:process-queue-locker**, **si:reset-process-queue** 71
New function: **si:read-or-end** 62
New function to be called by reader macros: **si:read-recursive** 83
si:read-recursive function 83
New functions: **si:make-process-queue**, **si:process-enqueue**, **si:process-dequeue**, **si:process-queue-locker**, **si:reset-process-queue** 71
si:reset-process-queue function 71
:chaos option for **si:sb-on** 95
:clock option for **si:sb-on** 95
:disk option for **si:sb-on** 95
:keyboard option for **si:sb-on** 95
:mouse option for **si:sb-on** 95
:unibus option for **si:sb-on** 95
si:sb-on function 95
New option for **si:sb-on**: **:mouse** (3600 only) 95
:read-frame method of **si:serial-hdlc-mixin** 111
:write-frame method of **si:serial-hdlc-mixin** 111
si:serial-hdlc-stream flavor 110
si:single syntax description 18
si:slash syntax description 18
si:verticalbar syntax description 18
si:whitespace syntax description 18
si: **single** syntax description 18
3600 Supports Ieee Single- and Double-precision Floating Point 48
Single-character symbol 18
New data types: **:single-float** and **:double-float** 49
cl: **single-float** format 17
sys: **single-float-p** function 49
New functions: **sys:** **single-float-p**, **sys:double-float-p** 49
IEEE-standard single-precision 48
Convert number to single-precision floating-point 48
float returns a single-precision number 48
:validate-lmfs-dump-tapes site attribute 121
Site Configuration for Dialnet 151
Release 5.0: Operations and Site Management 151
Extract size field of a byte specifier 50
si: **slash** syntax description 18
Memory Board Not Needed in Lbus Slot 0 136
Small flonum 49
Convert number to small flonum 48
Some Methods Can Use Combination Type as Method Type 83
sort function 144
sort predicate should return **nil** for equal elements 144
Sorting by Conversations Available 130
[Sort] Zmail menu item 130
Both default pathnames for Source Compare (m-X) now use **:newest**

version 123
Source Compare (m-X) Zwei command 123
New Zmacs command: Source Compare Newest Definition (m-X) 124
Source Compare Newest Definition (m-X) Zmacs command 124
Source pathname 26
Use **record-source-file-name** instead of **(remprop** *symbol* ' **:source-file-name)** 147
Output space 21
Trim leading and trailing white space 62
:space init option for **tv:margin-space-mixin** 75
:space method of **tv:margin-space-mixin** 75
Octal escape for special characters 17
Change in Debugger special command for **fs:directory-not-found** 98
block special form 46
catch special form 14
condition-bind special form 97
condition-call special form 98
condition-call-if special form 98
condition-case special form 98
defconstant special form 45
define-symbol-macro special form 53
defpackage special form 7
destructuring-bind special form 77
format:defformat special form 56
multiple-value-call special form 47
multiple-value-prog1 special form 48
store special form 145
tagbody special form 47
throw special form 15
trace special form 96
unadvise special form 40
with-stack-list special form 148
with-stack-list* special form 148
New special form: **defconstant** 45
New special form: **define-symbol-macro** 53
New special form: **defpackage** 7
Previously undocumented special form: **destructuring-bind** 77
New special form: **format:defformat** 56
New special forms **catch** and **throw** replace ***catch** and ***throw** 14
New special forms: **block** and **tagbody** 46
New special forms: **multiple-value-call** and **multiple-value-prog1** 47
New message to conditions: **:special-command-p** 74
:special-command-p method of **condition** 74
** accordion wildcard specification 120
< oldest version specifier 35
> newest version specifier 35
Create a byte specifier 50
Extract size field of a byte specifier 50
Mouse cursor speed 100
Stack lists 148
rplaca can be used with stack lists 148
Init File Pathnames Standardized 34
Zmail Init File Pathnames Standardized 129
New Zmacs command: Start Private Patch (m-X) 124
Start Private Patch (m-X) Zmacs command 124

meter: **start-monitor** function 51
:start-typeout method of **tv:stream-mixin** 63
New methods of **tv:stream-mixin:** **:start-typeout**, **:finish-typeout**, **:rescanning-p**, **:force-rescan**, **:replace-input**, **:read-bp** 63
Show Hardcopy Status (m-X) replaces **chaos:print-lgp-queue** 115
Show Hardcopy Status (m-X) Zwei command 115
Show Status Command Displays More Useful Information 140
Show Status FEP command 137, 140
Current micro PC (CPC) status information 137
Macro PC status information 137
Old PCs (OPC) status information 137
FUNCTION 2 W displays current process name in status line 148
h-c-*upper-left* waits for Lisp to stop itself 140
meter: **stop-monitor** function 51
h-c-*upper-left* stops execution of Lisp 133
store not supported on the 3600 145
store special form 145
Synonym stream 20
chaos: **stream** function 107
Stream parameter 65
chaos: **stream**, **chaos:close**, and **chaos:finish** renamed 107
:any-tyi method of **tv:** **stream-mixin** 41, 42
:any-tyi-no-hang method of **tv:** **stream-mixin** 41
:finish-typeout method of **tv:** **stream-mixin** 64
:force-rescan method of **tv:** **stream-mixin** 64
:list-tyi method of **tv:** **stream-mixin** 41
:mouse-or-kbd-tyi method of **tv:** **stream-mixin** 42
:mouse-or-kbd-tyi-no-hang method of **tv:** **stream-mixin** 42
:read-bp method of **tv:** **stream-mixin** 65
:replace-input method of **tv:** **stream-mixin** 65
:rescanning-p method of **tv:** **stream-mixin** 64
:start-typeout method of **tv:** **stream-mixin** 63
:tyi method of **tv:** **stream-mixin** 41
:tyi-no-hang method of **tv:** **stream-mixin** 41
Changes to **:tyi**, **:tyi-no-hang**, **:list-tyi**, **:mouse-or-kbd-tyi**, and **:mouse-or-kbd-tyi-no-hang** methods of **tv:** **stream-mixin** 40
New methods of **tv:** **stream-mixin**: **:start-typeout**, **:finish-typeout**, **:rescanning-p**, **:force-rescan**, **:replace-input**, **:read-bp** 63
:prompt-and-read messages to streams 85
:rename message to streams 27
Previously Undocumented Feature: Coroutine Streams 89
Reading from streams 65
Streams sharing common buffers 89
string-length uses same coercion rules as **string** 16
String concatenation 16
String construction 65
string function 16
Clicking Middle Edits Current String in Choose-variable-values Windows 105
:string option for **prompt-and-read** 85
String quoter 17
Change in type of array returned by **string-append** 16
string-append function 16
Previously undocumented function: **string-compare** 81
string-compare function 81

New optional arguments to **string-upcase** and **string-downcase** 80
string-downcase function 80
string-length function 16
string-length uses same coercion rules as **string** 16
:string-list option for **prompt-and-read** 85
:string-or-nil option for **prompt-and-read** 85
:string-trim option for **prompt-and-read** 85
New optional arguments to **string-upcase** and **string-downcase** 80
string-upcase function 80
:editor output format style 21
:read output format style 21
:sail output format style 21
Controlling typeout style 65
Compiler Performs Style Checking on All Forms 81
compiler: **style-checker** property 81
Submenu 105
Subpackages 6
summary-window-format Babyl file option 129
Babyl files with **summary-window-format** other than **t** or **nil** need to be edited 129
Supdup 1
SUPER key 102
Superpackages 6
:supersede value of **open** option for **:if-exists** 25
Rfc822 Domain Addressing Supported 130
tv:set-default-font not supported 149
store not supported on the 3600 145
LMFS Dumper Supports Accordion Wildcards 119
LMFS Now Supports Directory Links 120
FEP Supports Hdlc Serial I/O 133
3600 Supports Ieee Single- and Double-precision Floating Point 48
Suppress prompting for passwords 70
Previously undocumented macro: **swapf** 82
swapf macro 82
END and c-END swapped in Converse 116
:mouse symbol 42
Single-character symbol 18
Use **record-source-file-name** instead of (**remprop** *symbol* **':source-file-name**) 147
Symbol print-name quoter 17
#: *symbol*-**syn-stream** 20
External symbols 6
global package symbols 9
Internal symbols 6
keyword package symbols 7
Printing Uninterned Symbols 19
Uninterned symbols 19
Keyword Symbols Are Self-evaluating 13
Symbols in **global** and **keyword** packages with the same names 7
Symbols moved to or from **global** package 9
#: **syn-stream** 20
Loading Sync Programs 135
Load Sync-program FEP command 134, 135
Synonym stream 20
si:alphabetic syntax description 18
si:break syntax description 18

si:circlecross syntax description 18
si:doublequote syntax description 18
si:macro syntax description 18
si:single syntax description 18
si:slash syntax description 18
si:verticalbar syntax description 18
si:whitespace syntax description 18
Character syntax descriptions 17
Syntax errors in read functions 23
New Default LMFS Translation Table for Sys Hosts 119
sys:%count-disk-page-read-operations-in-scavenger meter 52
sys:%count-disk-page-read-operations-in-transporter meter 52
si:halt replaces **sys:%halt** 133
sys:%halt function 133
sys:%scavenger-run-time meter 52
sys:%transporter-run-time meter 52
sys:%tv-clock-counter meter 52
sys:arithmetic-error flavor 98
sys:defsubst-with-parent macro 145
New functions: **sys:single-float-p**, **sys:double-float-p** 49
sys:double-float-p function 49
sys:dump-forms-to-file always puts package attribute into binary file 82
sys:function-parent declaration 145
How to use the **sys:function-parent** declaration 145
Previously undocumented variables: **sys:mouse-x-scale-array** and sys:mouse-y-scale-array (LM-2 only) 100
sys:mouse-x-scale-array variable 100
Previously undocumented variables: **sys:mouse-x-scale-array** and **sys:mouse-y-scale-array** (LM-2 only) 100
sys:mouse-y-scale-array variable 101
New error flavors: **sys:parse-error** and **sys:parse-ferror** 23
sys:parse-error flavor 23
New error flavors: **sys:parse-error** and **sys:parse-ferror** 23
New function: **sys:parse-ferror** 24
sys:parse-ferror flavor 24
sys:parse-ferror function 24
sys:single-float-p function 49
New functions: **sys:single-float-p**, **sys:double-float-p** 49
New macro: **sys:with-open-file-search** 69
sys:with-open-file-search macro 69
Carry tape system 1
File System 1
Major and Minor mode system 127
Network namespace system 1, 107
Window System Changes Associated with Mouse Input 40
Changes to the File System in Release 5.0 119
Incompatible Changes to the File System in Release 5.0 119
New Features in the File System in Release 5.0 120
[Reload/Retrieve] File System Maintenance menu item 151

T

SELECT T command 114
floatp returns **t** for any floating-point number 49
Babyl files with **summary-window-format** other than **t** or **nil** need to be edited 129
New terminal program (SELECT T) 114
New Zwei command: Find Files in Tag Table (m-X) 125
Find Files in Tag Table (m-X) Zwei command 125
New Default LMFS Translation Table for Sys Hosts 119
Inspecting hash arrays of **eq** hash tables not permitted 145
New Zwei command: Find Files in Tag Table (m-X) 125
Find Files in Tag Table (m-X) Zwei command 125
New special forms: **block** and **tagbody** 46
tagbody special form 47
tv:scroll-maintain-list init function can take arguments 105
Backup Tape Reliability 151
Carry tape system 1
Comparing backup tapes 151
Target pathname 26
Telnet 1
Imlac terminal codes 114
Terminal program 1
New terminal program (SELECT T) 114
Babyl files with **summary-window-format** other than **t** or **nil** need to be edited 129
New special forms **catch** and **throw** replace ***catch** and ***throw** 14
New special forms **catch** and **throw** replace ***catch** and ***throw** 14
throw special form 15
:time-interval-or-never option for **prompt-and-read** 85
Token separators 17
New Rules for Reading Ambiguous Tokens 19
New Metering Tools for the 3600 50
TOPS-20 34
New format for **trace** output 96
trace special form 96
si: ***trace-bar-p*** variable 97
si: ***trace-bar-rate*** variable 97
si: ***trace-columns-per-level*** variable 97
si: ***trace-old-style*** variable 97
Trim leading and trailing white space 62
Improvements to **make-system**: **error-restart**, selective transformations 94
Reversible wild pathname translation 35
Wild pathname translation 35
New Default LMFS Translation Table for Sys Hosts 119
Changes to Logical Pathname Translations 35
Physical pathname translations 35
sys: **%transporter-run-time** meter 52
Trim leading and trailing white space 62
tv: **truncatable-lines-mixin** flavor 102
New flavors: **tv:** **truncatable-lines-mixin**, **tv:truncating-lines-mixin** 102
:truncate value of **open** option for **:if-exists** 25
:truncate-line-out method of **tv:sheet** 102
Truncating lines 102
New flavors: **tv:truncatable-lines-mixin**, **tv:** **truncating-lines-mixin** 102
tv: **truncating-lines-mixin** flavor 102
tv: **truncating-window** flavor 102

sys: **%tv-clock-counter** meter 52
tv:*mouse-incrementing-keystates* variable 103
New variable: **tv:*mouse-modifying-keystates*** 102
tv:*mouse-modifying-keystates* variable 103
Previously undocumented functions: **tv:add-to-system-menu-programs-column**, **tv:add-to-system-menu-create-menu** 103
tv:add-to-system-menu-create-menu function 104
tv:add-to-system-menu-programs-column function 104
Previously undocumented functions: **tv:add-to-system-menu-programs-column**, **tv:add-to-system-menu-create-menu** 103
Flavors **tv:any-tyi-mixin** and **tv:list-tyi-mixin** obsolete 40
tv:any-tyi-mixin flavor 40
New variable: **tv:cold-load-stream-old-selected-window** 74
tv:cold-load-stream-old-selected-window variable 74
tv:edit-namespace-object function 107
:mouse-click method of **tv:essential-mouse** 42
Changes to **:mouse-click** method of **tv:essential-mouse** 42
:draw-circle method of **tv:graphics-mixin** 99
:draw-filled-in-circle method of **tv:graphics-mixin** 99
tv:kbd-mouse-buttons-mixin flavor 42
Flavors **tv:list-mouse-buttons-mixin** and **tv:kbd-mouse-buttons-mixin** obsolete 42
Flavors **tv:list-mouse-buttons-mixin** and tv:kbd-mouse-buttons-mixin obsolete 42
tv:list-mouse-buttons-mixin flavor 42
tv:list-tyi-mixin flavor 40
Flavors **tv:any-tyi-mixin** and **tv:list-tyi-mixin** obsolete 40
:set-space method of **tv:margin-space-mixin** 75
:space init option for **tv:margin-space-mixin** 75
:space method of **tv:margin-space-mixin** 75
New flavor: **tv:margin-space-mixin** 74
tv:margin-space-mixin flavor 75
tv:menu-choose function 149
tv:mouse-double-click-time variable 103
New optional argument to **tv:mouse-wait** 101
tv:mouse-wait function 101
tv:preemptable-read-any-tyi-mixin flavor 43
Flavor **tv:preemptable-read-any-tyi-mixin** obsolete 43
New variable: **tv:rh-typeout-default** 65
tv:rh-typeout-default variable 65
tv:scroll-maintain-list function 105
tv:scroll-maintain-list init function can take arguments 105
tv:select-or-create-window-of-flavor function 104
tv:set-default-font function 149
tv:set-default-font not supported 149
:clear-eof method of **tv:sheet** 44
:clear-eol method of **tv:sheet** 44
:clear-rest-of-line method of **tv:sheet** 44
:clear-rest-of-window method of **tv:sheet** 44
:clear-screen method of **tv:sheet** 44
:clear-window method of **tv:sheet** 44
:set-truncate-line-out method of **tv:sheet** 102
:truncate-line-out method of **tv:sheet** 102
:any-tyi method of **tv:stream-mixin** 41, 42
:any-tyi-no-hang method of **tv:stream-mixin** 41
:finish-typeout method of **tv:stream-mixin** 64

:force-rescan method of **tv:stream-mixin** 64
:list-tyi method of **tv:stream-mixin** 41
:mouse-or-kbd-tyi method of **tv:stream-mixin** 42
:mouse-or-kbd-tyi-no-hang method of **tv:stream-mixin** 42
:read-bp method of **tv:stream-mixin** 65
:replace-input method of **tv:stream-mixin** 65
:rescanning-p method of **tv:stream-mixin** 64
:start-typeout method of **tv:stream-mixin** 63
:tyi method of **tv:stream-mixin** 41
:tyi-no-hang method of **tv:stream-mixin** 41
Changes to **:tyi**, **:tyi-no-hang**, **:list-tyi**, **:mouse-or-kbd-tyi**, and **:mouse-or-kbd-tyi-no-hang** methods of **tv:stream-mixin** 40
New methods of **tv:stream-mixin**: **:start-typeout**, **:finish-typeout**, **:rescanning-p**, **:force-rescan**, **:replace-input**, **:read-bp** 63
tv:truncatable-lines-mixin flavor 102
New flavors: **tv:truncatable-lines-mixin**, **tv:truncating-lines-mixin** 102
New flavors: **tv:truncatable-lines-mixin**, **tv:truncating-lines-mixin** 102
tv:truncating-lines-mixin flavor 102
tv:truncating-window flavor 102
New macro: **tv:with-mouse-grabbed-on-sheet** 74
tv:with-mouse-grabbed-on-sheet macro 74
intern, **intern-local**, **intern-soft**, and **intern-local-soft** return two values 7
Change in Zmacs command Modified Two Windows (c-X 4) 127
Modified Two Windows (c-X 4) Zmacs command 127
:tyi method of **tv:stream-mixin** 41
Changes to **:tyi**, **:tyi-no-hang**, **:list-tyi**, **:mouse-or-kbd-tyi**, and **:mouse-or-kbd-tyi-no-hang** methods of **tv:stream-mixin** 40
:tyi-no-hang method of **tv:stream-mixin** 41
Changes to **:tyi**, **:tyi-no-hang**, **:list-tyi**, **:mouse-or-kbd-tyi**, and **:mouse-or-kbd-tyi-no-hang** methods of **tv:stream-mixin** 40
.fep file type 134
:bin canonical type 34
:lisp canonical type 34
:qbin canonical type 34
DIAL network type 151
Some Methods Can Use Combination Type as Method Type 83
Some Methods Can Use Combination Type as Method Type 83
Font Editor file type defaults 113
Argument to **:menu** type menu items can be a menu or a form 105
Change in type of array returned by **string-append** 16
:type option to **make-plane** 80
:init canonical pathname type removed 35
Logical Pathname Name, Type, and Version Now Separated by Periods 35
New canonical file type: **:mss** 126
Controlling typeout style 65
Definition types 145
New data types: **:single-float** and **:double-float** 49

U U U

Change in arguments to **unadvise** 39
unadvise special form 40
New function: **undefflavor** 53
undefflavor function 53
Known problem: **char-upcase** and **char-downcase** undefined for modified characters 145
Flavor **fs:undefined-logical-pathname-translation** replaces **fs:undefined-logical-pathname-directory** 37
fs: **undefined-logical-pathname-translation** flavor 37
Flavor **fs:** **undefined-logical-pathname-translation** replaces **fs:undefined-logical-pathname-directory** 37
Previously Undocumented Feature: Coroutine Streams 89
Previously undocumented function: **clear-resource** 78
Previously undocumented function: **describe-system** 94
Previously undocumented function: **fs:set-logical-pathname-host** 38
Previously undocumented function: **string-compare** 81
Previously undocumented functions: tv:add-to-system-menu-programs-column, **tv:add-to-system-menu-create-menu** 103
Previously undocumented macro: **swapf** 82
Previously undocumented reader macro: **#** and **#** 83
Previously undocumented special form: **destructuring-bind** 77
Previously undocumented variables: **sys:mouse-x-scale-array** and **sys:mouse-y-scale-array** (LM-2 only) 100
:unibus option for **si:sb-on** 95
Uninterned symbols 19
Printing Uninterned Symbols 19
UNIX 34
Unlock queue 71
Unplugging Lemo Cables Should Not Halt the FEP 137
Major-mode-setting Commands Now Query About Updating File Attribute List 126
Uppercase Code in Buffer (m-X) Zwei command 125
Uppercase Code in Region (m-X) Zwei command 125
Shifted Mouse Clicks Can Now Be Used for Editor Commands 103
rplaca can be used with stack lists 148
Show Status Command Displays More Useful Information 140
Prompting for input from user 85
Changes to Input Editor User Interface 22
user package 1, 6
Arguments changed for **fs:** **user-homedir** and **fs:init-file-pathname** 34
fs: **user-homedir** function 34
:draw-filled-in-circle uses same algorithm as **:draw-circle** 99
string-length uses same coercion rules as **string** 16
Utilities 1
Changes to Utilities in Release 5.0 113
Improvements to Utilities in Release 5.0 116
Incompatible Changes to Utilities in Release 5.0 113
New Features in Utilities in Release 5.0 114

V V V

Vadic autodialer 151
Interface to the Vadic Modem 111
nil not a valid menu item 149
:validate-lmfs-dump-tapes site attribute 121
:input value of **open** option **:direction** 24
:output value of **open** option **:direction** 24
:probe value of **open** option **:direction** 24
:probe-directory value of **open** option **:direction** 24
:probe-link value of **open** option **:direction** 24
:create value of **open** option **:if-does-not-exist** 25
:error value of **open** option **:if-does-not-exist** 25
nil value of **open** option **:if-does-not-exist** 25
:append value of **open** option for **:if-exists** 25
:error value of **open** option for **:if-exists** 25
:new-version value of **open** option for **:if-exists** 25
:overwrite value of **open** option for **:if-exists** 25
:rename value of **open** option for **:if-exists** 25
:rename-and-delete value of **open** option for **:if-exists** 25
:supersede value of **open** option for **:if-exists** 25
:truncate value of **open** option for **:if-exists** 25
intern, intern-local, intern-soft, and **intern-local-soft** return two values 7
readline and **readline-trim** return additional values 62
applyhook variable 72
ci:*read-default-float-format* variable 17
dbg:*debug-io-override* variable 74
format:*format-output* variable 56
fs:*remember-passwords* variable 70
gc-on variable 73
inhibit-idle-scavenging-flag variable 148
si:*read-extended-ibase-signed-number* variable 20
si:*read-extended-ibase-unsigned-number* variable 19
si:*trace-bar-p* variable 97
si:*trace-bar-rate* variable 97
si:*trace-columns-per-level* variable 97
si:*trace-old-style* variable 97
si:gc-reclaim-immediately variable 148
sys:mouse-x-scale-array variable 100
sys:mouse-y-scale-array variable 101
tv:*mouse-incrementing-keystates* variable 103
tv:*mouse-modifying-keystates* variable 103
tv:cold-load-stream-old-selected-window variable 74
tv:mouse-double-click-time variable 103
tv:rh-typeout-default variable 65
zwei:*converse-end-exits* variable 116
zwei:*set-attribute-updates-list* variable 126
New variable: **ci:*read-default-float-format*** 17
New variable: **dbg:*debug-io-override*** 74
New variable: **fs:*remember-passwords*** 70
New variable: **gc-on** 73
New variable: **tv:*mouse-modifying-keystates*** 102
New variable: **tv:cold-load-stream-old-selected-window** 74
New variable: **tv:rh-typeout-default** 65
Previously undocumented variables: **sys:mouse-x-scale-array** and sys:mouse-y-scale-array (LM-2 only) 100
:verbose option for **print-herald** 39

Both default pathnames for Source Compare (m-X) now use **:newest** version 123
FEP Version 14: New Features 133
FEP Version 15: Improvements 135
FEP Version 15: Incompatible Changes 133
FEP Version 15: New Features 135
FEP Version 16: Improvements 137
FEP Version 16: New Features 136
FEP Version 17: Improvements 140
FEP Version 18: Improvements 140
Logical Pathname Name, Type, and Version Now Separated by Periods 35
< oldest version specifier 35
> newest version specifier 35
New Default Representations for Newest and Oldest Logical Pathname Versions 35
si: **verticalbar** syntax description 18
VMS 34
Directory creation on VMS 109
Pathname completion on VMS 109
Changes to VMS Chaosnet 109

W W W

FUNCTION W command 148
FUNCTION 2 W displays current process name in status line 148
W Fed command 113
h-c-*upper-left* waits for Lisp to stop itself 140
Compiler now warns about implicit **progn**s in **loop**s 82
What happens when you cold boot 143
What happens when you cold boot 143
Optional argument to **mapatoms-all** and **where-is** eliminated 7
where-is function 7
Trim leading and trailing white space 62
si: **whitespace** syntax description 18
bitblt width from the destination array 145
Wild pathname translation 35
Reversible wild pathname translation 35
:wild-inferiors 120
Wildcard Directory Mapping Available 92
** accordion wildcard specification 120
Accordion wildcards 1
LMFS Accordion Wildcards 120
LMFS Dumper Supports Accordion Wildcards 119
Complement window 113
Delete contents of window 44
Delete to end of window 44
Erase window 44
Erase to end of window 44
Window System Changes Associated with Mouse Input 40
Clicking Middle Edits Current String in Choose-variable-values Windows 105
Change in Zmacs command Modified Two Windows (c-X 4) 127
Modified Two Windows (c-X 4) Zmacs command 127
:clear-screen, **:clear-eol**, and **:clear-eof** messages to windows renamed 44
New macro: **with-input-editing** 59
with-input-editing macro 59

meter: **with-monitoring** macro 52
New macro: **tv:** **with-mouse-grabbed-on-sheet** 74
tv: **with-mouse-grabbed-on-sheet** macro 74
New macro: **sys:** **with-open-file-search** 69
sys: **with-open-file-search** macro 69
with-stack-list special form 148
with-stack-list* special form 148
Load World FEP command 134
[Write File] Font Editor menu item 113
:write-frame method of **si:serial-hdlc-mixin** 111

X

X X

NETWORK X command 114
SELECT X command 114
New feature: Flavor Examiner (SELECT X) 114
Changes to Serial I/O: Parity Recovery and Xon/Xoff Character Setting 109

Y

Y Y

Yanking input in Zwei 1
Yanking previous input 22
What happens when you cold boot 143

Z

Z Z

Zmacs 1
New Buffer-history Mechanism in Zmacs 125
Add Patch Changed Definitions (m-X) Zmacs command 123
Add Patch Changed Definitions of Buffer (m-X) Zmacs command 123
Copy File (m-X) Zmacs command 28, 124
Jump to Saved Position (c-X J) Zmacs command 127
List Buffers (c-X c-B) Zmacs command 125
Modified Two Windows (c-X 4) Zmacs command 127
Resume Patch (m-X) Zmacs command 124
Select Buffer (c-X B) Zmacs command 125
Select Previous Buffer (c-m-L) Zmacs command 125
Set Package (m-X) Zmacs command 123
Source Compare Newest Definition (m-X) Zmacs command 124
Start Private Patch (m-X) Zmacs command 124
Change in Zmacs command Modified Two Windows (c-X 4) 127
New Zmacs command: Resume Patch (m-X) 124
New Zmacs command: Source Compare Newest Definition (m-X) 124
New Zmacs command: Start Private Patch (m-X) 124
Changes to Zmacs in Release 5.0 123
Improvements to Zmacs in Release 5.0 126
Incompatible Changes to Zmacs in Release 5.0 123
New Features in Zmacs in Release 5.0 124
Zmail 1
Append Conversation By References (m-X) Zmail command 130
Delete Conversation By References (m-X) Zmail command 130
Select All Conversations By References (m-X) Zmail command 130
Select Conversation By References (m-X) Zmail command 130
Changes to Zmail in Release 5.0 129
Improvements to Zmail in Release 5.0 130

Incompatible Changes to Zmail in Release 5.0 129
New Features in Zmail in Release 5.0 130
Zmail Init File Pathnames Standardized 129
[Sort] Zmail menu item 130
Yanking input in Zwei 1
Comment Out Region (c-X c-;) Zwei command 125
Find Files in Tag Table (m-X)) Zwei command 125
Lowercase Code in Buffer (m-X) Zwei command 125
Lowercase Code in Region (m-X) Zwei command 125
Show Hardcopy Status (m-X) Zwei command 115
Source Compare (m-X) Zwei command 123
Uppercase Code in Buffer (m-X) Zwei command 125
Uppercase Code in Region (m-X) Zwei command 125
New Zwei command: Comment Out Region (c-X c-;) 125
New Zwei command: Find Files in Tag Table (m-X) 125
zwei:*converse-end-exits* variable 116
zwei:*set-attribute-updates-list* variable 126
zwei:com-zmail-select-all-conversations-by-references function 130
Internal changes to macros **zwei:defmajor** and **zwei:defminor** 127
zwei:defmajor macro 127
Internal changes to macros **zwei:defmajor** and **zwei:defminor** 127
zwei:defminor macro 127

Previously undocumented reader macro: **#** and **#** 83

New **format** directives: ~→ and ~← 55
~← **format** directive 55
New **format** directives: ~→ and ~← 55
~→ **format** directive 55
~@* **format** directive 21
format directives ~@T and ~@* replace ~**X** and ~**G** 21
format directives ~@T and ~@* replace ~**X** and ~**G** 21
~@T **format** directive 21
format directives ~@T and ~@* replace ~**X** and ~**G** 21
~**G** 21
~**G format** directive 21
format directives ~@T and ~@* replace ~**X** and ~**G** 21
~**X format** directive 21
format ~\ directives can have package prefixes 92

symbolics™

RN Release Notes/ Patch Notes

Release 5.1 Patch Notes

Cambridge, Massachusetts

Release 5.1 Patch Notes

 # 998011

June 1984

This document corresponds to Release 5.1.

This document was prepared by the Documentation Group of Symbolics, Inc.

The software described in this document is furnished only under license, and may be used or copied only in accordance with the terms of such license.

Nothing contained in this document should be construed to imply the granting of a license to make, use, or sell any of the equipment or software described herein.

Symbolics, Inc. makes no representation that the connection of its products in the manner described in this document will not infringe existing or future patent rights.

The information in this book is subject to change without notice, and should not be construed to imply any representation or commitment by Symbolics, Inc.

Symbolics, Symbolics 3600, Symbolics 3670, Zetalisp, and Macsyma are trademarks of Symbolics, Inc.

This document may not be reproduced in whole or in part without the prior written consent of Symbolics, Inc.

Printed in the USA.

Copyright © 1984
Symbolics, Inc.
All Rights Reserved

Printing year and number: 87 86 85 84 9 8 7 6 5 4 3 2 1

Table of Contents

Page

1. Introduction **1**

1.1 New microcode: 998 on LM-2, 292 on 3600 1

2. Improvements **2**

2.1 c-ABORT can now be typed while entering Debugger (3600 only) 2
2.2 LMFS Backups (LM-2, 3600) 2
2.3 Serial I/O (3600 only) 2

3. Installation Instructions (3600 Only) **3**

Index **5**

1. Introduction

These release notes accompany Release 5.1. They describe changes made since Release 5.0. This patch release fixes some problems and improves performance in some areas. Only the visible changes are mentioned here.

This document contains information relevant to both the LM-2 and the 3600. However, some sections apply to only one or the other. Section headings indicate such cases.

1.1 New microcode: 998 on LM-2, 292 on 3600

Release 5.1 uses microcode version 998 on the LM-2 and 292 on the 3600.

2. Improvements

2.1 c-ABORT can now be typed while entering Debugger (3600 only)

The cover letter sent out with Release 5.0 (dated 1 May 1984) stated:

> 3. Do not type c-ABORT while your machine is entering the debugger or Lisp may halt the next time that you enter the debugger. This will be fixed in a future release.

As of Release 5.1, this restriction no longer applies.

2.2 LMFS Backups (LM-2, 3600)

In Release 5.0, backups, especially to remote tapes, might not work. Some of the data on the tapes might be damaged even though the files could be backed up and restored. Release 5.1 fixes this bug.

2.3 Serial I/O (3600 only)

Sometimes, serial I/O would hang or would not complete. This has been fixed in Release 5.1.

3. Installation Instructions (3600 Only)

The 5.1 software tape contains patch files, the merged sources for the patched definitions, new microcode, and additional examples that were not in the 5.0 distribution tape.

To load the distribution tape, follow these instructions.

1. At a Lisp Listener, evaluate the following two forms:

```
(fs:define-canonical-type :sync-program "sync"
  (:unix "sn" "sync")
  (:unix42 "sync" "sn")
  (:vms "syn"))

(defprop :sync-program 8 :binary-file-byte-size)
```

2. Place the distribution tape in a cartridge tape drive of a 3600.

3. Load the contents of the Release 5.1 tape by typing the following form to a Lisp Listener:

```
(dis:load-distribution-tape)
```

   "Where to get distribution tape" menu pops up; indicate tape host (if remote) and click on [Do It].

   "Items to be loaded" menu pops up; click on [Do It]. The files start loading.

4. After the contents of the tape have been loaded onto your sys host, type:

```
(si:install-microcode 292.)
```

5. Shut your machine down by typing the following forms:

```
(logout)
(si:halt)
```

6. Boot the machine using microcode 292.

7. Evaluate these two forms:

```
(fs:define-canonical-type :sync-program "sync"
  (:unix "sn" "sync")
  (:unix42 "sync" "sn")
  (:vms "syn"))

(defprop :sync-program 8 :binary-file-byte-size)
```

8. Type the following form:

```
(load-and-save-patches)
```

This will cause the machine to log in as user LISP-MACHINE, load the patches, and disk-save the world. You will need a world load file approximately 20% larger than your current world.

Note: Please be sure to **(load-and-save-patches)** into all worlds that are built on Release 5.0. If you have a Release 5.0 Macsyma world, for example, be sure to **load-and-save-patches** in that world as well as in any other 5.0-based worlds you are using.

Release 5.2 Bulletin

This document may not be reproduced in whole or in part without the prior written consent of Symbolics, Inc.

Printed in the USA.

Copyright © 1984
Symbolics, Inc.
All Rights Reserved

Release 5.2 Bulletin
#

September 1984

This document corresponds to Release 5.2.

This document was prepared by the Home Office Software Support Group of Symbolics, Inc.

The software described in this document is furnished only under license, and may be used or copied only in accordance with the terms of such license.

Nothing contained in this document should be construed to imply the granting of a license to make, use, or sell any of the equipment or software described herein.

Symbolics, Inc. makes no representation that the connection of its products in the manner described in this document will not infringe existing or future patent rights.

The information in this book is subject to change without notice, and should not be construed to imply any representation or commitment by Symbolics, Inc.

Symbolics, Symbolics 3600, Symbolics 3640, Symbolics 3670, Zetalisp, and MACSYMA are trademarks of Symbolics, Inc.

This document may not be reproduced in whole or in part without the prior written consent of Symbolics, Inc.

Printed in the USA.

Copyright © 1984
Symbolics, Inc.
All Rights Reserved

Printing year and number: 87 86 85 84 9 8 7 6 5 4 3 2 1

Table of Contents

		Page
1	Patch for Fast **draw-string** Microcode in 5.2	1
2	Patch for **make-system** in 5.2	2
3	Patch for **si:ascii-translating-input-stream-mixin** whopper in 5.2	2
4	Use of Simple ASCII Printers with 5.2	2
5	New Microcode Types	3
6	Disk Configurations with Release 5.2	4
	6.1 Systems with 280 Mbytes or More	5
	6.2 Systems with 167-Mbyte Disks	5
	6.3 Systems with 140-Mbyte Disks (Model 3640s)	5
7	Running out of Room	5
	7.1 LMFS vs FEP File System	5
	7.2 World Loads	6
	7.3 Local File Space	6
	7.4 Paging Space	6
8	Keeping track of bug reports at the customer site	7
	8.1 Instructions for the Lisp Machine user	8
	8.2 Instructions for the site administrator	9

Release 5.2 Bulletin

NOTE: To add these patches to your world, enter them into an editor buffer, making sure that you specify the correct package for each, compile the file, boot a clean world, and load the file before disk-saving.

1 Patch for Fast draw-string Microcode in 5.2

The new fast **draw-string** microcode in Release 5.2 contains an error which causes the system to go into an infinite loop when displaying certain multi-font lines in the editor. Compiling the following patch will cure this problem. You must compile this, rather than evaluate it.

```
tv:(deff %draw-string-internal-that-doesnt-work
        (function tv:%draw-string-internal))

tv:
(defun %draw-string-internal (array alu x y string font index limit xlim)
  (multiple-value-bind (new-x new-index done-p)
      (%draw-string-internal-that-doesnt-work
        array
        alu
        x
        y
        string
        font
        index
        limit
        xlim)
    (when (and (not done-p)
               (< new-index limit)
               (= (+ new-x (font-char-width font)) xlim))
      (%draw-char-internal font
                           (ldb %%ch-char
                                (aref string new-index))
                           new-x
                           y
                           alu
                           array)
      (setq new-x xlim)
      (incf new-index)
      (setq done-p (= new-index limit)))
    (values new-x new-index done-p)))
```

2 Patch for make-system in 5.2

If a system in 5.2 is defined using logical pathnames, *make-system* is incapable of detecting whether or not they have changed since they were loaded last. Consequently, it will load all the binary files of the system every time. To cure this problem, compile the following form into the system:

```
si:
(defun system-get-loaded-id (file)
  (let ((info (get-file-loaded-id file *force-package*)))
    (when info
      (cons (send (car info) :translated-pathname)
            (cdr info)))))
```

3 Patch for si:ascii-translating-input-stream-mixin whopper in 5.2

In 5.2 the following method is defined to accept no arguments. This causes an error to occur when it receives one from a calling function. Enter the function with the following changes and then compile it into your world to avoid this error.

```
si:
(defwhopper (ascii-translating-input-stream-mixin :tyi) (&optional eof)
  (let((ch (continue-whopper eof)))
    (selectq ch
      (10 #\bs)
      (11 #\tab)
      (12 #\line)
      (14 #\ff)
      (15 (let ((ch1 (continue-whopper eof)))
            (unless (or (null ch1) (= ch1 12)) (send self :untyi ch1)))
          #\cr)
      (177 #\rubout)
      (t ch))))
```

4 Use of Simple ASCII Printers with 5.2

With 5.2, the use of simple ASCII printers is supported. To hook up your ASCII printer for use with the hardcopy system, do the following:

1. Physically connect your ASCII printer to your Lisp Machine through the serial port, making sure that the RS-232 connection is correct. Both the printer and the computer are DTE, making the use of a "null modem" necessary. This connector switches several signals , allowing bidirectional communication. Note that the Lisp Machine must also receive DTR from the printer. (See page 23-24 of the *NETIO* document for RS-232 pin assignments.)

2. Create a printer object in your site namespace with type ASCII and interface SERIAL. Below is an example of some of the information shown by **tv:edit-namespace-object** for an ASCII printer. The printer's name is Erie.

```
Printer: ERIE
Type: ASCII
Site: FISHERY
Pretty Name: "Lake Erie"
Interface: SERIAL
Host: TENNESSEE
Interface Options: Set: Pair: UNIT 1 Pair: BAUD 300 Pair: NO-BANNER-PAGE T
Dplt Logo: SYMBOLICS
```

Some of these attributes are optional. See page 13 of the *PROT* document for an explanation of printer attributes.

3. Edit the host object of the Lisp Machine to which you have attached the printer and add the following attributes: (NOTE: Use "LGP" as shown, even though the printer is not an Laser Graphics Printer.)

```
Service: Set: HARDCOPY-STATUS CHAOS LGP-QUEUE
Service: Set: HARDCOPY CHAOS LGP
Spooled Printer: Pair: ERIE
```

-where ERIE is the name of your printer as defined above. You can also edit the *Default Printer* attribute to make your ASCII printer the default printer for your site, if you wish. Save out this new namespace information.

4. Create a top-level directory on the host machine for the printer called *Print-spooler*. Using the above example, the directory would be *Tennessee:>print-spooler>*.

5. Now boot a clean world and type the following to a Lisp listener:

```
(si:login-to-sys-host)
(make-system 'print)
```

and then **disk-save**. When you boot this new world, your ASCII printer should be accessible.

5 New Microcode Types

The logical directory *sys: l-ucode;* now includes multiple types of microcode for each version number. The correct microcode to install depends upon the particular hardware configuration of your machine. When your machine is shipped, the default microcode filename is correct, but if your machine is upgraded (for example, an FPA board is installed) you might need to override the default used by

si:install-microcode to get the correct type for your configuration. Below is an example of how you would get the microcode for a 3600 running 5.2, with no console upgrade but an FPA board installed:

(si:install-microcode "tmc5-fpa-mic.mic.296")

The correct microcode types for each system and hardware configuration are shown below. In this table, *n* indicates the version of microcode that is required; the version number must be followed by a period (.). Microcode version 296. is required for Release 5.2.

Machine Type:	**3600**	**3670**	**3640**
-standard configuration	tmc5-mic.mic.n	tmc5-io4-mic.mic.n	tmc5-io4-st506-mic.mic.n
-with console upgrade (I/O rev. 6)	tmc5-io4-mic.mic.n	not applicable	not applicable
-with FPA board	tmc5-fpa-mic.mic.n	tmc5-io4-fpa-mic.mic.n	tmc5-io4-st506-fpa-mic.mic.n
-with both console upgrade and FPA board	tmc5-io4-fpa-mic.mic.n	not applicable	not applicable

If you use the wrong microcode for your configuration, your machine will not boot, except in the case where your system has an FPA and you use a non-FPA microcode. In this case, the machine will function normally, but will not make use of the FPA at all.

6 Disk Configurations with Release 5.2

Symbolics 3600-family systems shipped with Release 5.2 on their disks have new disk configurations. The disk configuration with which the system is shipped depends on the amount of disk space available on that particular machine. There are three basic disk configurations:

1. Systems that have at least 280 Mbytes of disk storage
2. Systems with a 167-Mbyte disk drive
3. Systems with a 140-Mbyte disk drive (currently, 3640 systems only)

6.1 Systems with 280 Mbytes or More

These systems contain a 49K-block file system partition (>LMFS.FILE), a 60K-block paging file, and a 30K-block world load file. They have a complete set of sources and online documentation on the disk, so no tapes are provided unless they are ordered with the machine.

6.2 Systems with 167-Mbyte Disks

These systems contain a 10K-block file system partition (>LMFS.FILE), a 60K-block paging file, and a 30K-block world load file. These systems have a minimal set of sources, which are sufficient to permit you to operate in a stand-alone configuration, with a modest amount of disk space available for user files. The full system sources and online documentation are not available on the disk but are included on tapes shipped with the system. These tapes cannot be loaded successfully onto the 3640. Instead, they should be loaded onto a file server that has room for the 41 Mbytes of information they contain. *There is not enough disk space on these systems to support full sources and documentation while maintaining two large world loads.*

6.3 Systems with 140-Mbyte Disks (Model 3640s)

These systems contain a 30K-block world load, a 45K-block paging partition, and a 30K-block auxiliary paging partition, which is sometimes used as a world load file. These paging files are used differently than on previous machines. In particular, the procedure for transferring bands and for creating new worlds with **disk-save** is very different. Please see section 1.3 of the *Release 5.2 Patch Notes* for further details. These systems are shipped with source and documentation tapes, which can be loaded onto a file server with space for the 41 Mbytes of information they contain. These tapes cannot be successfully loaded onto the system itself.

7 Running out of Room

7.1 LMFS vs FEP File System

There are two file systems available on the Lisp Machine: the Lisp Machine File System (LMFS) and the FEP File System (FEP FS). LMFS is a general purpose, highly flexible file system, suitable for everyday use. Currently, only the Lisp Machine processor understands how to operate on LMFS files. The FEP FS is a simple, basic file system that both the Lisp Machine and Front End processors understand how to access. The FEP FS is used mainly to store world loads, microcode loads, paging files, boot files, and file system partitions that LMFS uses to store its structure and data. The FEP FS is not a good place for users to store their files; that is what LMFS is for.

7.2 World Loads

Sometimes **disk-save** or **si:receive-band** might inform you that you have run out of FEP file system space. For 280- and 167-Mbyte systems, you should delete and expunge old, unneeded world loads, and then resume from the **disk-save** "out of room" error or retry the **si:receive-band** operation. You should not delete any world loads from a 140-Mbyte system. See section 1.3 of the *Release 5.2 Patch Notes* for details.

It is wise to keep a large (25K-30K), noncritical world load on the Lisp Machine's disk, where it is available for **disk-restore** to use in case all world loads become nonfunctional. This will also help reserve space for the installation of Release 6.0. If you do not reserve that space, you may end up overwriting valuable world loads in order to install Release 6.0.

7.3 Local File Space

Sometimes, writing a file out to a Lisp Machine File System (LMFS) will produce an "out of room" error. This means that the present allocation of that particular LMFS is not large enough to accommodate your request for space. It might help to expunge directories with deleted files in them, and it may help to delete extra, unneeded versions of files, using the Zmacs command Dired (m-X).

If you still don't have enough space after you have deleted and expunged the unnecessary files, you might consider creating an auxiliary file partition. You should only consider doing so on systems that have at least 280 Mbytes of storage. There is no room in the FEP file system to allocate an auxiliary file partition for 140-Mbyte systems, and allocating an auxiliary file partition on a 167-Mbyte system might inhibit the creation of large world loads or even prevent the installation of Release 6.0.

Even for 280-Mbyte systems, you are trading off world load space for file space when you create auxiliary partitions. Be sure to reserve enough FEP file system space for two large world loads (about 60K blocks): the world you are currently running from and a spare world load for **disk-restore** to use.

See the section "Multiple Partitions" in the document *Files* for details on how to create auxiliary file partitions. Once you have created an auxiliary file partition, you should never delete it. Deleting it would lose all the data contained in that partition and make the entire Lisp Machine File System unusable.

7.4 Paging Space

Programs that use large amounts of virtual memory may require you to allocate additional paging space, to perform better or to perform at all. Only systems with at least 280 Mbytes of disk storage really have enough room to permit additional paging files to be allocated without adversely affecting the maintenence of worlds on the

machine. In order to add an additional paging file to your virtual memory set, you must first create a FEP file. Below is a function that creates a FEP file of a given length.

```
(defun create-fep-file (name length)
  (with-open-file (stream name :direction :block :if-exists :error)
    (send stream :allocate length)))
```

The code below creates an additional 20K-block paging file on unit zero, using the above function:

```
(create-fep-file "fep0:>page1.page" 20000.)
```

After creating the extra paging file, any boot files should be modified to use this new paging partition. A typical boot file might look something like this:

```
clear machine
load microcode >tmc5-mic.mic.292
load world >Release-5-1.load
set chaos 401
start
```

After creating the new paging partition, boot files should be edited to look something like this:

```
clear machine
load microcode >tmc5-mic.mic.292
load world >Release-5-1.load
clear paging
add paging >page.page
add paging >page1.page
set chaos 401
start
```

It is safe to delete extra paging partitions, but only if they are not in active use. Be sure to cold boot by hand, and *do not* type the Add Paging command for the extra paging partition you intend to delete.

8 Keeping track of bug reports at the customer site

Note: This pertains to all Release 5.0, 5.1, 5.2, and later installations.

Symbolics is currently developing a means for automatically forwarding bug reports from the local site to Symbolics customer service via a variety of networks. Although this facility is not yet fully in place, we would like to institute a procedure for the user to collect and submit bug reports that will both accomodate present needs and provide for later enhancement.

Included below are instructions for both the user and the site administrator on how to report bugs, and how to set up the framework for collecting these bug reports and transmitting them to Symbolics.

8.1 Instructions for the Lisp Machine user

For the purposes of this discussion, Lisp Machine bugs have been divided into two categories: those that cause the Lisp Machine to enter the debugger and those that indicate that something "just didn't work".

Suppose you entered a form and were thrown into the debugger with the following error message and prompt:

> *>>Trap: The argument given to the CAR instruction, :STRAWBERRY, was not a list, a locative, or NIL.*
>
> [possibly followed by some proceed options, and then the debugger prompt, which is a right-pointing arrow.]

At the debugger prompt, you should type the debugger command c-M. This inserts a stack history and some information about the loaded software environment into an editor buffer and prompts for a description of what you were doing at the time. You should type in the description of the problem and a phone number where you can be reached. When done, press the END key. This sends the bug report, and you are placed back in the debugger. You can then inspect the stack, return from some frame, abort the computation, view arguments or perform a variety of operations. For more information on the debugger, see the *DEBUG* document.

Here's another (fictitious) example of a bug where something "just doesn't work":

Suppose you typed **(print-disk-label)** and the function immediately returned NIL without having printed out any information on the disk label at all.

First, jot down as much as you can remember in the way of unusual circumstances. Had the machine been warm-booted? Were you using a locally improved version of **print-disk-label**?

Secondly, see if the bug is reproducible. Unless, of course, it's something like "the robot arm attachment went through the wall."

If it seems that the bug is worth reporting, then use the Zmacs M-X BUG command to send mail about the problem. This will prompt you to enter a completion of the mailing list name "BUG-". Since LISPM is the default, just hit a carriage return to send it to BUG-LISPM. This will put you at the top of an editor buffer, which contains information about your system. You should enter your description of what you were doing when the error occurred and any information about your configuration which may be pertinant (ex. the type of disk you have, if it is a disk problem). Also enter your name and where you can be reached. When you feel comfortable with the bug report, hit END to send it. You may be prompted for a message subject, should be a one-line quick description of the problem, for example, "print-disk-label doesn't print anything at all."

At this point you are back in the editor, and can go back to whatever you were doing before noticing the error.

8.2 Instructions for the site administrator

If the Lisp Machine(s) at the site are networked to a timesharing system that provides mail service, set up a mailbox on the timesharing service to receive bug reports. Name the mailbox BUG-LISPM and have it route mail to a file, which can then be printed, dumped to tape, or whatever you chose to do with it.

Suppose the name of the timeshared host at site FISHERY is Pepper and that it is the only mail server at the site. You should then edit the site object for the FISHERY site, using the namespace editor:

```
(tv:edit-namespace-object :site "FISHERY")
```

You would then click left on the Token field following the Host-for-bug-reports field in the object, type `Pepper`, press RETURN, and then click on the Save menu item to save your change. Finally, after the change has been saved, leave the namespace editor by clicking on the Quit menu item.

The site administrator should periodically use the mail program on the mail serving machine to study the contents of the BUG-LISPM mailbox. Bug reports that are related to a site-developed program should be answered or forwarded to the appropriate mail address within the site. Questions that can be answered by the site administrator should be so answered. All of the other messages in the file should be conveyed to Symbolics by either printed or magnetic media. If there is some doubt about whether a given bug exists in a site-developed program or in Symbolics software, send the bug report to Symbolics.

The exact mechanism by which the bug reports are sent from the customer to Symbolics Software Support is something to be negotiated by the customer and the Symbolics contact.

symbolics™

RN Release Notes/ Patch Notes

Cambridge, Massachusetts

Release 5.2 Patch Notes
 # 998019

August 1984

This document corresponds to Release 5.2.

This document was prepared by the Documentation Group of Symbolics, Inc.

The software described in this document is furnished only under license, and may be used or copied only in accordance with the terms of such license.

Nothing contained in this document should be construed to imply the granting of a license to make, use, or sell any of the equipment or software described herein.

Symbolics, Inc. makes no representation that the connection of its products in the manner described in this document will not infringe existing or future patent rights.

The information in this book is subject to change without notice, and should not be construed to imply any representation or commitment by Symbolics, Inc.

Symbolics, Symbolics 3600, Symbolics 3640, Symbolics 3670, Zetalisp, and MACSYMA are trademarks of Symbolics, Inc.

UNIX is a trademark of Bell Laboratories, Inc.
VAX, TOPS-20, DECwriter, VT100, and VMS are trademarks of Digital Equipment Corporation.
TENEX is a registered trademark of Bolt Beranek and Newman Inc.
Ann Arbor Ambassador is a trademark of Ann Arbor Terminals, Inc.
TD-80 is a trademark of Cipher Data Products, Inc.
Maxtor is a trademark of Maxtor Corporation
Philips is a trademark of Philips Electronics Ltd
Moniterm is a trademark of Moniterm Corporation

This document may not be reproduced in whole or in part without the prior written consent of Symbolics, Inc.

Printed in the USA.

Copyright © 1984
Symbolics, Inc.
All Rights Reserved

Printing year and number: 87 86 85 84 9 8 7 6 5 4 3 2 1

Table of Contents

Page

1. Installation Instructions for Release 5.2 1

1.1 Introduction to Installation Instructions 1
1.2 Installation Instructions (3600 and 3670 Only) 1
1.3 Installation Instructions (3640 Only) 3
1.3.1 Customizing and Saving the World 3
1.3.2 Saving Subsequent Worlds 5

2. Release 5.2: Introduction and Highlights 7

2.1 New Microcode in Release 5.2: 296 on 3600 8

3. Changes to the Lisp Language in Release 5.2 9

3.1 Incompatible Changes to Lisp in Release 5.2 9
3.1.1 Symbols Added to the **global** Package in Release 5.2 9
3.2 New Features in Lisp in Release 5.2 9
3.2.1 New Feature: Floating Point Accelerator 9
3.2.2 New Function: **tan** 9
3.2.3 New Functions: **char-standard** and **char-code** 10
3.2.4 New functions: **char-flipcase** and **string-flipcase** 11
3.2.5 New Flavor Functions: **send-if-handles**, **lexpr-send-if-handles**, and **operation-handled-p** 12
3.3 Improvements to Lisp in Release 5.2 12
3.3.1 **defconstant**'s Query Offers Three Choices 12

4. Changes to Zmacs in Release 5.2 15

4.1 New Features in Zmacs in Release 5.2 15
4.1.1 Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) 15
4.1.2 Two New Commands: Select Patch (m-X) and View Patch (m-X) 15
4.1.3 New Command: Select Some Buffers As Tag Table (m-X) 15
4.1.4 New Hardcopy Option to Kill or Save Buffers (m-X) 15
4.1.5 New Dired Command, L, Loads a File 15

5. Changes to Utilities in Release 5.2 17

5.1 New Features in Utilities in Release 5.2 17
5.1.1 New Compiler Special Form: **compiler:make-message-obsolete** 17

5.2 Improvements to Utilities in Release 5.2 17
5.2.1 Optional Argument to **fed** Can Now be a BFD Object 17
5.2.2 Garbage Collection of Compiled Functions 17
5.2.3 Patch Facility Supports Multiple Patches In-progress 18
5.2.4 Inspector Displays Hash Tables 19

6. Changes to the User Interface in Release 5.2 21

6.1 Incompatible Changes to the User Interface in Release 5.2 21
6.1.1 **tv:add-system-key** renamed to **tv:add-select-key**; **tv:add-escape-key** renamed to **tv:add-function-key** 21
6.1.2 Time Parser No Longer Accepts Dates in European Format 21
6.2 New Features in the User Interface in Release 5.2 22
6.2.1 New special form: **define-prompt-and-read-type** 22
6.2.2 Previously Undocumented Feature: the 3600-family Calendar Clock 23
6.2.3 Previously Undocumented Feature: Audio Level and Screen Brightness Controls on 3670 and 3640 Consoles 23
6.3 Improvements to the User Interface in Release 5.2 24
6.3.1 New type keyword for **prompt-and-read** and **tv:choose-variable-values**: **:host-or-local** 24
6.3.2 Time Parser Accepts Dates in ISO Standard Format 24

7. Changes to Zmail in Release 5.2 27

7.1 Show Draft Disposition (m-X) Zmail Command 27
7.2 Start Background Save (m-X) Zmail Command 27
7.3 c-m-L is a Synonym for [Select] 27
7.4 Moving a Message by Filters Now Uses All Filters 27
7.5 New Numeric Arguments for c-m-Y 27
7.6 Internet Domain Addressing 28

8. Changes to the File System in Release 5.2 29

8.1 Incompatible Changes to the File System in Release 5.2 29
8.1.1 :clear-screen and :clear-eol Obsolete Names 29
8.2 New Features in the File System in Release 5.2 29
8.2.1 LMFS Tape Spec Prompting 29
8.2.2 Direct Access File Streams 30
8.2.3 New keyword options to **open** 32
8.3 Improvements to the File System in Release 5.2 32
8.3.1 Faster Creation of Logical Pathname Hosts 32
8.3.2 **:finish** is supported by some output file streams 34

9. Changes to Networks in Release 5.2 35

9.1 Incompatible Changes to Networks in Release 5.2 35
9.1.1 Size of Packet Buffers 35

9.2 New Features in Networks in Release 5.2 35
9.2.1 Using the Terminal Program with Hosts Connected to the Serial Line 35
9.2.2 New Variable to Hold Packet Size 37
9.3 Improvements to Networks in Release 5.2 37
9.3.1 Improvements to **:error-disposition** Option of **net:define-server** 37
9.3.2 Improvements to **tv:edit-namespace-object** 37

10. Changes to the FEP in Release 5.2 39

10.1 FEP Version, Serial I/O, and Model Number Compatibility 39
10.2 FEP Version 18: Improvements 39
10.2.1 Changes in FEP Fault Lights on the 3670 39
10.3 FEP Version 18: New Features 40
10.3.1 **si:halt** has new *fep-commands* argument 40
10.3.2 h-c-*upper-left* waits for Lisp to stop itself 41
10.4 FEP Version 22: Incompatible Changes 41
10.4.1 FEP Halt Command Differences in 3600s and 3670s 41
10.4.2 FEP Shutdown Command Replaces Halt Command in FEP Version 22 41
10.5 FEP Version 22: New Features 42
10.5.1 New FEP Command: Set Display-string 42
10.5.2 New FEP Command: Set Monitor-type 42
10.6 FEP Version 22: Improvements 42
10.6.1 Changes in FEP Fault Lights on the 3600 42
10.7 FEP Version 23: Incompatible Changes 43
10.7.1 New Default for FEP Add Paging-file Command 43
10.8 Additional Information About the FEP in Release 5.2 43
10.8.1 Previously Undocumented FEP Command: Load FEP 43
10.8.2 Previously Undocumented Topic: Debugging in the FEP 43
10.8.3 Using FEP Clear Paging-files Command Before Add Paging-file 44
10.8.4 Previously Undocumented Tape Utility for Writing FEP Files to Tape 44

11. Changes to Tape and Disks in Release 5.2 45

11.1 Incompatible Changes to Tape and Disks in Release 5.2 45
11.1.1 Must Specify Dumper Tape Drive Identifier 45
11.2 New Features in Tape and Disks in Release 5.2 45
11.2.1 New Feature: Carry Tape System 45
11.2.2 New Feature: Maxtor XT-1140 Support 50
11.2.3 New Feature: TD-80 Tape Drive Support 50
11.3 Improvements to Tape and Disks in Release 5.2 50
11.3.1 Distribution Tape Program Changes 50
11.3.2 Improvements to Distribution Tape Loader 51

11.3.3 Previously Undocumented Utility: Writing FEP Files to Tape 52
11.4 Tape Spec Prompting 53
11.4.1 Tape Specs 53
11.4.2 Tape Spec Parameters 54
11.4.3 How Tape Specs Are Merged 55
11.4.4 Tape Spec Syntax 56
11.4.5 Tape Spec Merging Examples 58

12. Release 5.2: Operations and Site Management 59

12.1 Notes on Operations in Release 5.2 59
12.1.1 Installing a Tape Drive 59
12.1.2 Installing an ASCII Print Server on VMS 59
12.1.3 Installing an ASCII Print Server on UNIX 60
12.1.4 Sending *who-am-i* Requests Directly to Namespace Servers 61
12.1.5 3600 **disk-save** Improvement 62

13. Release 5.2: Notes and Clarifications 63

13.1 Clarifications and Corrections for Release 5.2 63
13.1.1 Clarification of use of **dbg:with-erring-frame** 63
13.1.2 Clarification of Use of **:no-increment-patch** Option to **make-system** 63
13.1.3 Clarification of Use of **:nowarn** Option to **load-patches** 63
13.1.4 Effect of Character Set Translation on Direct Access File Streams 63

14. New Audio System in Release 5.2 65

14.1 Introduction to the Digital Audio Facilities 65
14.2 Microcode Support for the Digital Audio Facilities 65
14.2.1 The Audio Microtask 65
14.2.2 Sample Format 66
14.2.3 Audio Command Format 67
14.2.4 The Polyphony Feature 69
14.2.5 Notes on Wired Structures 71
14.3 Lisp Primitives for the Digital Audio Facilities 73
14.3.1 Functions, Variables, and Macros for Digital Audio 73
14.3.2 Digital Audio Parameters 73
14.3.3 Testing for the Existence of Audio 74
14.3.4 The Audio Wrapping Form 74
14.3.5 Building Audio Command Lists 74
14.3.6 Storing Samples 76
14.3.7 Looping Through Audio Command Lists 77
14.3.8 Synchronization Flags 78
14.3.9 Starting and Stopping the Audio Microtask 79
14.3.10 Conversions Between Sample Formats 79
14.3.11 Conversions for the Polyphony Feature 80

14.3.12 Computing Polyphonic Increments 81
14.4 Examples of Using the Audio Facilities 81
14.4.1 Sine Wave Example 81
14.4.2 Sawtooth Wave Example 83
14.4.3 Square Wave Example 83
14.4.4 Beep Example 84
14.4.5 Non-real-time Synthesis Example 85
14.4.6 Playing Large Pieces Example 86
14.4.7 Polyphony Example 89

Index **93**

1. Installation Instructions for Release 5.2

1.1 Introduction to Installation Instructions

This section describes installation instructions in two parts: the first part explains procedures for installing Release 5.2 on 3600s and 3670s, and the second part explains procedures for installing Release 5.2 on 3640s.

Before beginning any new site installation, you must have registered your site in a global site registry administered by Symbolics. To register your site, please contact Symbolics Software Support at 617-577-7500 (effective July 1984).

1.2 Installation Instructions (3600 and 3670 Only)

The 5.2 software tape contains patch files, the merged sources for the patched definitions, new source files, and new microcode.

Adding the 5.2 patches increases the size of your world by about 8000 blocks, so before you load these patches you might want to edit your FEP directories to make sure you have enough room to disk save the resulting world.

To load the distribution tape, follow these instructions.

1. Place the distribution tape in a cartridge tape drive of a 3600 or 3670.

2. Load the contents of the Release 5.2 tape by typing the following form to a Lisp Listener:

   ```
   (dis:load-distribution-tape)
   ```

   `Where to get distribution tape` menu pops up; indicate tape host (if remote) and click on [Do It].

   `Items to be loaded` menu pops up; click on [Do It]. The files start loading.

3. After the contents of the tape have been loaded onto your sys host, type:

   ```
   (si:install-microcode 296.)
   ```

   Be sure to type the decimal point after the `296`.

4. To install the software, you must first adjust the prewired virtual address space of the world load that you wish to update to Release 5.2. At a Lisp Listener, type:

```
(load "sys:patch;fixup-world")
Loading SYS: PATCH; FIXUP-WORLD.LISP.NEWEST into package SYSTEM-INTERNALS
World load file for wired space adjustment (default = FEP0:>current world):
```

Specify the name of the world load to alter. The default is the current world (the one that you booted from). When this patch finishes loading, the prewired virtual address space has been adjusted.

5. Shut your machine down by typing the following forms:

```
(logout)
(si:halt)
```

6. Boot the altered world using microcode 296. Perhaps the easiest way to do this is to look at the contents of your boot file with the FEP Show File command (which defaults to >Boot.boot). Manually enter the same boot file commands, replacing the old microcode pathname with the new. For example, instead of entering the command:

```
Load microcode >tmc5-mic.mic.292
```

Enter the following command:

```
Load microcode >tmc5-mic.mic.296
```

7. Log in as user LISP-MACHINE:

```
(si:login-to-sys-host)
```

8. Load patches (which causes the world to grow about 8000 blocks):

```
(load-patches :noselective)
```

If the resulting world is too large, you might want to use the form **(si:full-gc :gc-compiled-functions t)** , which should reduce the size of the world load by about 8000 blocks. This might result in a small loss of performance on machines with only one memory board (due to loss of locality).

9. Save the world:

```
(disk-save)
```

Note: Please be sure to alter the prewired space, load patches, optionally do a full garbage collection (use **si:full-gc**), and disk save in all worlds that you wish to upgrade to Release 5.2. If you have a Release 5.1 MACSYMA world, for example, be sure to alter the prewired space, load patches, and disk save in that world as well as in any other 5.1-based worlds you are using.

1.3 Installation Instructions (3640 Only)

Since the 140 megabyte disk drive of the 3640 contains a smaller paging partition than the 3600 or 3670, you must manage your 3640 FEP file system differently. For a complete description of paging partitions: See the section "Disk Partitions".

This section describes the different procedures that you follow to manipulate paging space when you are doing the following things:

- Loading the world
- Customizing and saving that world load
- Saving future world loads

The disk of your 3640 already contains Release 5.2, so you do not have to load the distribution tape.

The Release 5.2 distribution disk contains the 5.2 distribution world, a large paging file (called Page.page), and an auxiliary file that is the same size as the 5.2 distribution world. You use the auxiliary file in one way for normal operation and in another way when putting a new world on the disk.

In normal operation, you boot a world load file and use both Page.page and the auxiliary file for paging. In this case, you call the auxiliary file Aux.Page.

When you want to create a new world or transfer a new world to the disk, you boot your world load file and use only Page.page for paging. Instead of using the auxiliary file for paging, you rename it and use it as the target of your **disk-save** or **receive-band**. Once you have successfully created the new world, you rename the old world load file to Aux.page and use it as your auxiliary paging file.

The auxiliary file is always actively in use, either as:

- A paging file (in normal operation)
- The target file for new world load

1.3.1 Customizing and Saving the World

The shipped configuration assumes the auxiliary file to be the receptacle for your site's customized world load and so contains just one paging file:

```
Release-5-2-DIST.LOAD.1      30,000
Aux.Page.1                   30,000
Page.Page.1                  45,000
```

A customized, normal configuration uses the auxiliary file as a paging file and so contains two paging files:

Current-world.load	30,000
Aux.Page.1	30,000
Page.Page.1	45,000

To create your customized world, follow these instructions:

1. Boot the distribution world using microcode 296. Use only Page.Page.1 for paging and reserve the auxiliary file. It is best to initially boot by hand rather than to use the boot file so that you can set your Chaos address:

   ```
   Clear Machine
   Load Microcode microcode-file-name
   Load World world-load-file-name
   Clear Paging
   Add Paging >Page.Page
   Set Chaos-Address this-machine's-chaos-address
   Start
   ```

2. Rename the auxiliary file to whatever name you wish, for example:

   ```
   (renamef "FEP:>Aux.Page.1" "FEP:>new-world.load.1")
   ```

3. Customize the booted world and then save it into your new world load file:

   ```
   (disk-save "FEP:new-world.load.1")
   ```

   Since you are asking to save the world into an existing file, you are prompted for an action with which to proceed. The correct answer is Overwrite. You are then asked if you want to update the boot file. Answer yes. The `Set Chaos` line that you typed is added to the boot file at this time.

4. Rename the distribution world to be the auxiliary file:

   ```
   (renamef "FEP:>Release-5-2-DIST.LOAD.1" "FEP:>Aux.Page.1")
   ```

5. At this point, you should edit the boot file, FEP:>Boot.boot, to add the auxiliary file as an additional paging file. After the line `Add Paging >Page.Page`, add:

   ```
   Add Paging >Aux.Page
   ```

   Your edited boot file should look like this:

   ```
   Clear Machine
   Load Microcode microcode-file-name
   Load World world-load-file-name
   Clear Paging
   Add Paging >Page.Page
   Add Paging >Aux.Page
   Set Chaos-Address this-machine's-chaos-address
   Start
   ```

   Save the edited version.

6. Log out and halt the machine.

7. Boot the new world using the boot file.

1.3.2 Saving Subsequent Worlds

Whenever you wish to create a new world on your 3640 disk, you must follow a similar procedure to that shown above.

1. Boot manually, and do *not* type the `Add Paging >Aux.Page` command:

   ```
   Clear Machine
   Load Microcode microcode-file-name
   Load World world-load-file-name
   Clear Paging
   Add Paging >Page.Page
   Set Chaos-Address this-machine's-chaos-address
   Start
   ```

2. Rename the auxiliary file to whatever name you wish, for example:

   ```
   (renamef "FEP:>Aux.Page.1" "FEP:>Newer-world.load.1")
   ```

3. Either customize the booted world and then save it into your new world load file, or else transfer the world from some other machine:

   ```
   (disk-save "FEP:>Newer-world.load.1")
   ```

   or:

   ```
   (si:receive-band "other-machine-name"
           "remote-band-name" "FEP:>Newer-world.load.1")
   ```

   Since you are asking to save the world into an existing file, you are prompted for an action with which to proceed. The correct answer is Overwrite. Then you are asked if you want to update the boot file. Answer yes.

4. Rename the old world to the auxiliary file:

   ```
   (renamef "FEP:>Old-world.LOAD.1" "FEP:>Aux.Page.1")
   ```

5. Log out and halt the machine.

6. Boot the machine using the new boot file.

2. Release 5.2: Introduction and Highlights

These notes accompany the release of Release 5.2. They describe changes made since Release 5.0. The changes are organized in the following sections. Within each section, the material is organized into incompatible changes, new features, and improvements.

Changes to the Lisp Language in Release 5.2
This section describes changes relevant to the Lisp language.

Changes to Zmacs in Release 5.2
This section describes changes in the Zmacs editor.

Changes to Utilities in Release 5.2
This section describes changes in what any other computer would call the operating system and utilities. This includes the Debugger, the Inspector, the garbage collector, network support, and various system keyboard features.

Changes to the User Interface in Release 5.2
This section describes changes to the user interface, including the window system.

Changes to Zmail in Release 5.2
This section describes changes in the Zmail mail reading and sending program.

Changes to the File System in Release 5.2
This section describes changes in the Lisp Machine file system.

Changes to Networks in Release 5.2
This section describes changes in network implementation, interface, and protocols.

Changes to the FEP in Release 5.2
This section describes changes in the FEP. Release 5.2 requires FEP version number 18 or higher.

Changes to Tape and Disks in Release 5.2
This section describes changes in the distribution tape program and the addition of the carry tape system and tape spec prompting.

Release 5.2: Operations and Site Management
This section describes changes to the system and site configuration features of the system. These changes are important to the people who are responsible for the software at each site.

Release 5.2: Notes and Clarifications
This section contains explanations and clarifications of items that people found confusing in previous releases and documentation.

New Audio System in Releases 5.2
This section describes the new audio system available on 3640s and 3670s with I/O Rev. 4 or later installed.

You can find all the incompatible changes by reading the first part of each section. A complete list of changes appears in the Table of Contents.

2.1 New Microcode in Release 5.2: 296 on 3600

Release 5.2 world loads must be run with microcode version 296 on the 3600. The old world loads do not work with the new microcode, and the new world loads do not work with the old microcode.

Please note that microcode version 998 is still run on the LM-2.

3. Changes to the Lisp Language in Release 5.2

3.1 Incompatible Changes to Lisp in Release 5.2

3.1.1 Symbols Added to the global Package in Release 5.2

The following symbols have been added to the **global** package:

char-code
char-flipcase
char-standard
define-prompt-and-read-type
lexpr-send-if-handles
operation-handled-p
read-or-end
send-if-handles
string-flipcase
tan

3.2 New Features in Lisp in Release 5.2

3.2.1 New Feature: Floating Point Accelerator

Release 5.2 supports the Floating Point Accelerator (FPA) product. This is a circuit board and supporting microcode that speed up single-precision floating-point operations. It does not affect double-precision operations.

The FPA operates only when your machine has an FPA board and is running with FPA microcode. The microcode is distributed with Release 5.2. If your machine has an FPA board but is running without FPA microcode, the machine operates normally but without the FPA. However, your machine *cannot* run FPA microcode without an FPA board. The machine is likely to crash under these circumstances.

Contact your Symbolics customer service representative for information about installing the FPA board.

3.2.2 New Function: tan

tan *x* *Function*

Returns the tangent of *x*, where *x* is expressed in radians.

For example:

```
(tan 1.0d0) => 1.5574077246549023d0
```

3.2.3 New Functions: char-standard and char-code

Zetalisp has a limited facility for defining multiple character sets. The standard Lisp Machine character set is discussed in another section. See the section "The Character Set". Unlike standard characters in Common Lisp, Zetalisp standard characters can be in fonts other than font 0. You can have both standard and nonstandard character sets in multiple fonts.

Two functions, **char-standard** and **char-code**, provide support for nonstandard character sets that do not have the usual interpretations of case and font.

char-standard *char* *Function*

Returns **t** if *char* is a standard character, with the usual interpretations of case and font. By default, **char-standard** always returns **t**. You can redefine this function to introduce multiple character sets.

char-code *char* *Function*

Returns the character code for *char*, ignoring font. By default, the character code is the **%%ch-char** field. You can redefine this function to introduce multiple character sets.

Always use **char-code** instead of **(ldb %%ch-char** *char***)** to determine the character code so that your programs can run without modification when the Common Lisp character set is introduced.

char-standard and **char-code** are hooks. On the 3600, you can redefine these functions to examine the value of the **%%ch-font** field of their argument and to use this in computing their result. You cannot redefine them on the LM-2.

Make sure that replacement definitions for **char-standard** and **char-code** are thoroughly debugged using different names before redefining them. Defective versions of these functions can cause the system to crash.

By redefining these functions, you can control the behavior of **char-equal**, **char-lessp**, **char-upcase**, **string-equal**, **string-search**, and other system functions that ignore font information or that only make sense for the standard character set.

If **char-standard** is redefined and you evaluate the form **(neq (char-standard c1) (char-standard c2))**, then **(char-equal c1 c2)** returns **nil**. **char-code** usually returns a number between 0 and **(dpb -1 %%ch-char 0)**, inclusive. If it is redefined, it can return numbers greater than **(dpb -1 %%ch-char 0)**.

For example:

Suppose you have three different Greek fonts and you want to define a Greek character set. Characters in fonts 1, 2, and 3 are assumed to be Greek. Characters in font 0 and in fonts 4 through 255 are assumed to be in the standard character set. Characters in font 0 must always be in the standard character set.

Suppose that the values of the **%%ch-char** field of the characters α, β, and γ are the same as that of the characters A, B, and C. We want **(char-equal #/A #/α)** and **(string-equal "ABC" "$\alpha\beta\gamma$")** to return **nil**. But we want **(string-equal "$\alpha\beta\gamma$" "$\alpha\beta\gamma$")** to return **t**, even if the first string and the second string are in two different fonts.

The system provides these definitions of **char-standard** and **char-code**:

```
(defun char-standard (ignore) t)

(defun char-code (char) (ldb %%ch-char char))
```

You can define a Greek character set, allowing three fonts using font numbers 1, 2, and 3, by using these definitions instead:

```
(defun char-standard (char)
     (let ((font (ldb %%ch-font char)))
       (or (zerop font) (≥ font 4))))

(defun char-code (char)
  (let ((font (ldb %%ch-font char))
        (code (ldb %%ch-char font)))
    (if (or (zerop font) (≥ font 4))
        code
        (dpb 1 %%ch-font code))))
```

You can define multiple character sets in a similar manner.

3.2.4 New functions: char-flipcase and string-flipcase

char-flipcase *char* *Function*

If *char*, which must be an integer, is a lowercase alphabetic character in the standard character set its uppercase form is returned. If *char* is an uppercase alphabetic character in the standard character set its lowercase form is returned. Otherwise, it returns *char*. If font information is present it is preserved. The result of **char-flipcase** is undefined for characters with modifier bits.

string-flipcase *string* &optional (*from* **0**) *to* (*copy-p* **t**) *Function*

If *copy-p* is not **nil**, returns a copy of *string*, with uppercase alphabetic characters replaced by the corresponding lowercase characters, and with lowercase alphabetic characters replaced by the corresponding uppercase characters. If *copy-p* is **nil**, exchanges the case of characters in *string* itself and then returns the modified *string*. *from* is the index in *string* at which to begin exchanging the case of characters. If *to* is supplied, it is used in place of **(array-active-length** *string***)** as the index one greater than the last character whose case is to be exchanged. Characters not in the standard character set are unchanged.

3.2.5 New Flavor Functions: send-if-handles, lexpr-send-if-handles, and operation-handled-p

Three new functions (**send-if-handles**, **lexpr-send-if-handles**, and **operation-handled-p**)have been added.

Currently, **send-if-handles**, **lexpr-send-if-handles**, and **operation-handled-p** work by sending **:operation-handled-p** and **:send-if-handles** messages. For example, (**send-if-handles** *object message arguments*) sends *message* to *object* with the argument *arguments*.

In a later release, these functions might be redefined so that they perform the operation directly, thus eliminating control over how the operation is executed. If you need this type of control, you should use
(**send** *object* **:operation-handled-p** *message*) rather than
(**operation-handled-p** *object message*).

send-if-handles *object message-name* &rest *arguments* *Function*
Sends the message named *message-name* to *object* if the flavor associated with *object* has a method defined for *message-name*. If it does not have a method defined, **nil** is returned. *message-name* is a message name and *arguments* is a list of arguments for that message.

lexpr-send-if-handles *object message-name* &rest *arguments* *Function*
Sends the message named *message-name* to *object* if the flavor associated with *object* has a method defined for *message-name*. *message-name* is a message name and *arguments* is a list of arguments for that message. If *object* does not have a method defined, **nil** is returned.

The difference between **lexpr-send-if-handles** and **send-if-handles** is that for **lexpr-send-if-handles**, the last element of arguments should be a list; all the elements of that list are passed as arguments. **lexpr-send-if-handles** is to **send-if-handles** as **lexpr-send** is to **send**.

operation-handled-p *object message-name* &rest *arguments* *Function*
Returns **t** if the flavor associated with *object* has a method defined for *message-name* and **nil** if a method is not defined for *message-name*.

3.3 Improvements to Lisp in Release 5.2

3.3.1 defconstant's Query Offers Three Choices

defconstant's query when the value of a constant is being changed now offers three choices: Y, N, and P.

- The Y answer changes the value.

- The N answer does not change the value.
- The P answer changes the value and, when you change any future value, it prints a warning rather than a query.

The P answer sets **inhibit-fdefine-warnings** to **:just-warn**. **defconstant** obeys that variable, just as **query-about-redefinition** does. Use the following form to revert to the querying mode:

```
(setq inhibit-fdefine-warnings nil)
```

When the value of a constant is changed by a patch file, a warning is now printed. Formerly no warning was printed. See the special form **defconstant**.

4. Changes to Zmacs in Release 5.2

4.1 New Features in Zmacs in Release 5.2

4.1.1 Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X)

Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) compare a definition against that in the newest version of the source file rather than the version from which the definition was most recently loaded. The two new commands never look in patch files; they only look in "original source files." If the definition was added by a patch (so that no original source file was recorded), the command cannot find the name of the source file. However, if you have already read the source file into the editor it finds the definition in the editor buffer. See the section "Comparing/Merging Current/Newest Versions: Compare/Merge Commands for Definitions: Zmacs Manual".

4.1.2 Two New Commands: Select Patch (m-X) and View Patch (m-X)

Zmacs now supports several patches in progress at once, via the Select Patch (m-X) and View Patch (m-X) commands. For a detailed description of these patch facility improvements: See the section "Patch Facility Supports Multiple Patches In-progress".

4.1.3 New Command: Select Some Buffers As Tag Table (m-X)

The new command Select Some Buffers As Tag Table (m-X) offers another way to specify a tag table. See the section "Tags Tables and Search Domains: Searching, Replacing, and Sorting: Zmacs Manual".

4.1.4 New Hardcopy Option to Kill or Save Buffers (m-X)

The Kill or Save Buffers (m-X) command now offers a hardcopy option on its menu. Choices are: Save, Kill, Unmodify, and Hardcopy. See the section "Destroying Buffers: Buffer Commands: Zmacs Manual".

4.1.5 New Dired Command, L, Loads a File

A new command to Dired allows you to load a file merely by typing L, just as you can edit a file by typing E.

5. Changes to Utilities in Release 5.2

5.1 New Features in Utilities in Release 5.2

5.1.1 New Compiler Special Form: compiler:make-message-obsolete

A new compiler special form, **compiler:make-message-obsolete**, allows you to declare that a message name is obsolete (or will be obsolete in a future release) and to inform users of the valid replacement.

compiler:make-message-obsolete *message-name format-string* *Special Form*

Allows you to generate compiler warnings about obsolete message names. The first argument, *message-name*, is the obsolete message name. The second argument, *format-string*, is the warning to be printed. If the string contains the ~S format directive, it will be replaced by the object that was sent the message.

Example:

```
(compiler:make-message-obsolete :clear-screen
        "You have sent the message :CLEAR-SCREEN to the object ~S.
        This name is obsolete.  The new name for this message is
        :CLEAR-WINDOW.  Please update your code.")
```

5.2 Improvements to Utilities in Release 5.2

5.2.1 Optional Argument to fed Can Now be a BFD Object

The optional *font* argument to **fed** can now be a BFD object in addition to a string or a symbol. See the section "Entering and Leaving FED".

5.2.2 Garbage Collection of Compiled Functions

The function **si:full-gc** has an argument **:gc-compiled-functions** that garbage-collects old compiled function objects.

si:full-gc &key *system-release gc-compiled-functions* *Function*

si:full-gc does an immediate, complete, nonincremental garbage collection. **si:full-gc** runs the forms on the **full-gc** initialization list and then does any garbage collection without multiprocessing (inside a **without-interrupts** form), so the machine essentially "freezes" and does nothing but garbage collection for the duration. This operation takes about 20 minutes. After the garbage collection is completed and before it reenables scheduling and returns, **si:full-gc** runs the forms on the **after-full-gc** initialization list.

(si:full-gc :gc-compiled-functions t) garbage-collects the **compiled-function-area**, eliminating old compiled function objects. It works only on the 3600 family. It can significantly reduce the size of a frequently patched world if done before a **disk-save**. As a side-effect, it removes all **:previous-definition** properties. It implies **:system-release t**.

Normally, you would not use this function to perform nonincremental garbage collection; use **gc-immediately** instead. **si:full-gc** is mainly used by Symbolics staff just before releasing a world load.

5.2.3 Patch Facility Supports Multiple Patches In-progress

In Release 5.2 the patch facility allows you to have several patches in progress at once. Thus you can patch several different systems or several different minor versions of the same system during one patch session. (Note: please be very careful to create your patches in the correct order when code in one patch depends on code in another. Where doubtful, make one patch at a time.)

To keep track of your various patches, the patch facility now considers patches to be either active or inactive and in one of the following states:

- Inital
- In-progress
- Aborted
- Finished

Inactive patches are in an aborted or finished state. *Active patches* are in an initial or in-progress state. *Initial* means that the patch buffer has been initialized but as yet no definitions have been added to the buffer. *In-progress* means that definitions have been added to the buffer. A new Zmacs command, View Patches (m-X), displays the state of all your patches.

Release 5.2 introduces the concept of the *current patch*. If more than one patch is in progress, one of them is known as the current patch. The commands that add patches, like Add Patch (m-X), add *only* to the current patch. Another new Zmacs command, Select Patch (m-X), allows you to select another patch as the current one from a menu of active patches.

The add-patch commands now prompt you for comments. Just pressing END means "no comment". Finish Patch (m-X) lets you edit the patch comments and offers to send mail about the patch. The initial contents of the mail buffer include the name of the patch file and your patch comments. Formerly Finish Patch prompted you for comments and did not offer to send mail.

If you do not wish to be queried about sending mail, change the value of **zwei:*send-mail-about-patch*** from **:ask**, the default, to **t** or **nil**. **t** opens a Zmacs mail buffer without querying, and **nil** takes no action regarding the sending of patch mail.

Aside from the two new Zmacs commands, View Patches (m-X) and Select Patch (m-X), all other patching commands from previous releases remain in place. See the section "Making Patches".

5.2.4 Inspector Displays Hash Tables

The Inspector pane now displays these components of hash tables: flavor of the hash table, the method table, the names and values of the instance-variable slots followed by the key/value pairs for the entries of the hash table. The value for a given key is modifiable. See the section "The Inspector Inspection Pane".

6. Changes to the User Interface in Release 5.2

6.1 Incompatible Changes to the User Interface in Release 5.2

6.1.1 tv:add-system-key renamed to tv:add-select-key; tv:add-escape-key renamed to tv:add-function-key

tv:add-system-key has been renamed to **tv:add-select-key**, and **tv:add-escape-key** has been renamed to **tv:add-function-key**. These changes were made for consistency with key names in the 3600 family. For compatibility, in this release **tv:add-system-key** is synonymous with **tv:add-select-key**, and **tv:add-escape-key** is synonymous with **tv:add-function-key**.

See the function **tv:add-select-key**. See the function **tv:add-function-key**.

6.1.2 Time Parser No Longer Accepts Dates in European Format

Strings of the kind used as the first argument to **time:parse** and **time:parse-universàl-time** can no longer contain dates in European format. In such strings, the first integer is always parsed as the month and the second integer as the day. For example, `"3//4//85"` or `"3-4-85"` is always the same as `"March 4, 1985"`, never `"April 3, 1985"`. A string like `"15//3//85"` is an error.

Formerly, if either of the first two integers was greater than 12., that integer was parsed as the day and the other integer as the month. If both integers were less than 13., the site's timezone determined whether the first integer was parsed as the month or the day. In European timezones the first integer was parsed as the day.

For example, formerly:

`"3//15//85"` was equivalent to `"March 15, 1985"` in all timezones
`"15//3//85"` was equivalent to `"March 15, 1985"` in all timezones
`"3//4//85"` was equivalent to `"March 4, 1985"` in non-European timezones
`"3//4//85"` was equivalent to `"April 3, 1985"` in European timezones

As of Release 5.2:

`"3//15//85"` is equivalent to `"March 15, 1985"` in all timezones
`"15//3//85"` is an error
`"3//4//85"` is equivalent to `"March 4, 1985"` in all timezones

See the section "Reading Dates and Times".

6.2 New Features in the User Interface in Release 5.2

6.2.1 New special form: define-prompt-and-read-type

define-prompt-and-read-type *keyword parameter-list description* &body *body* *Special Form*

Defines a new type keyword for **prompt-and-read**. Defines a dispatch function to be called to get input from the user when **prompt-and-read** is called with a type keyword of *keyword*. The dispatch function is defined as the **prompt-and-read** property of *keyword*. Its parameter list is *parameter-list*, and its body is *body*. **prompt-and-read** returns whatever the dispatch function returns.

The dispatch function is called with two or more arguments:

- The first argument is the stream to read from, usually **query-io**.
- The second argument is a list of input editor options. The dispatch function can add further options to this list.
- If the first argument to **prompt-and-read** is just *keyword*, the dispatch function is called with no more arguments. If the first argument to **prompt-and-read** is (*keyword* . *type-args*), the remaining arguments to the dispatch function are the elements of *type-args*. By convention, this is an alternating series of keywords and values.

parameter-list usually looks something like this:

```
(stream options &key type-arg-1 type-arg-2 ...)
```

If this **prompt-and-read** type does not allow any *type-args*, *parameter-list* might look like `(stream options)`.

description is a string that describes the type of input the user should enter. This string is displayed as part of the default **prompt-and-read** prompt. It should be a suitable argument for the **format** control string `"Enter ~A: "`. For example, if *description* is `"a number"`, the default **prompt-and-read** prompt is `"Enter a number: "`.

body is the body of the dispatch function. Often the body is a call to a more primitive reading function, such as **read** or **readline**. It is the responsibility of the body or a function it calls to provide input editing if needed.

Example:

```
(define-prompt-and-read-type :delimited-string "a string delimited by <End>"
                             (stream options &key delimiter buffer-size)
  (read-delimited-string (or delimiter #\end) stream nil options buffer-size))
```

6.2.2 Previously Undocumented Feature: the 3600-family Calendar Clock

Machines in the 3600 family have a calendar clock that operates independently of the other Lisp Machine timers. When you cold boot and the machine fails to get the time from the network, it asks you to type in the time. If the calendar clock has been set, it uses the calendar clock reading as the default for the time you specify. If the calendar clock has not been set, it offers to set it to the time you type in. See the function **time:initialize-timebase**.

You can also set the calendar clock yourself using **time:set-calendar-clock** and read it using **time:read-calendar-clock**.

time:set-calendar-clock *new-time* *Function*

(3600 family only) Sets the calendar clock to *new-time*, which must be either a universal time or a suitable argument to **time:parse**. Returns **t** if the calendar clock is set successfully, otherwise **nil**.

time:read-calendar-clock &optional *even-if-bad* *Function*

(3600 family only) Attempts to read the calendar clock. If the attempt is unsuccessful, returns **nil**. If the attempt is successful and the time appears to be valid, returns the time in universal time form. If the attempt is successful but the time appears to be invalid, takes action depending on the value of *even-if-bad*:

nil or unspecified	Returns **nil**
Not **nil**	Attempts to convert the internal format to universal time. If the conversion is successful, returns the time in universal time form. Otherwise, signals an error.

6.2.3 Previously Undocumented Feature: Audio Level and Screen Brightness Controls on 3670 and 3640 Consoles

The audio level on 3670 and 3640 consoles can be changed by using the following keystrokes:

LOCAL-L makes the sound louder

LOCAL-Q makes the sound quieter

This control affects the console hardware and does not interact with the digital audio feature software.

Screen brightness can be adjusted via the keyboard as well.

LOCAL-B makes the picture brighter

LOCAL-D makes the picture dimmer

The LOCAL key is used like a shift key for these commands.

6.3 Improvements to the User Interface in Release 5.2

6.3.1 New type keyword for prompt-and-read and tv:choose-variable-values: :host-or-local

prompt-and-read and **tv:choose-variable-values** now accept the **:host-or-local** type keyword. **:host-or-local** is like **:host** except that it returns the local host if the user types "local".

Following is the syntax for **:host-or-local** as the first argument to **prompt-and-read**:

(**:host-or-local :default** *default*)
: Reads the name of a network host or the string "local", terminated by RETURN, LINE, or END. If **:default** is specified, it should be the name of a host as a symbol or string. If *default* is the string "local" or the symbol **:local**, the default is the local host. If **:default** is specified and the user just presses RETURN, LINE, or END, it returns the host specified by **:default**. If the user types "local", it returns the local host. Otherwise, it returns the host whose name the user types.

Following is the meaning of **:host-or-local** as a **tv:choose-variable-values** keyword:

:host-or-local
: The value is a network host. It is read as the name of a host or the string "local" to represent the local host. If the host is the local host, it is printed as "Local"; otherwise, it is printed as the name of the host.

See the function **prompt-and-read**. See the section "The Choose Variable Values Facility".

6.3.2 Time Parser Accepts Dates in ISO Standard Format

Strings of the kind used as the first argument to **time:parse** and **time:parse-universal-time** can now contain dates in ISO standard format. These strings are of the form "yyyy-mm-dd", where:

yyyy
: Four digits representing the year

mm
: The name of the month, an abbreviation for the month, or one or two digits representing the month

dd
: One or two digits representing the day

See the section "Reading Dates and Times".

7. Changes to Zmail in Release 5.2

7.1 Show Draft Disposition (m-X) Zmail Command

After you have sent a message, you can use Show Draft Dispositions (m-X) to find out which mail server sent your message, to whom, at what time.

See the section "Leaving Mail Mode in Zmail".

7.2 Start Background Save (m-X) Zmail Command

Start Background Save (m-X) suppresses background mail checks and starts a save in the background. This allows you to compose and send mail messages while the save is being done. See the section "Saving, Expunging, Killing, and Renaming Zmail Messages".

7.3 c-m-L is a Synonym for [Select]

Using [Select] returns you to the previously selected sequence.

c-m-L is like [Select]. With an argument of 0 it works like [Select (R)]. With an argument of 1 or greater, it works as in Zmacs and selects from the stack of previously selected sequences. See the section "Changing Buffers: Buffer Commands: Zmacs Manual". See the section "Selecting Zmail Mail Buffers and Files".

7.4 Moving a Message by Filters Now Uses All Filters

When you use [Move (M)] to move a message to a file, all the filters in the filter-mail file alist are checked and the message is moved to the files corresponding to the filters in the list satisfied by the message. See the section "Copying a Zmail Message to Another Buffer".

7.5 New Numeric Arguments for c-m-Y

c-m-Y yanks the current message into a reply buffer. Unless an argument is given, the message is indented four spaces to set it off from the reply. The variable **zwei:*prune-headers-after-yanking***, settable in your profile, controls the automatic pruning of message headers yanked into a reply. The default is to not prune headers.

The new numeric arguments to c-m-Y allow you to choose indentation and header pruning of the yanked message.

The arguments to c-m-Y control the indentation and the pruning of headers, as follows:

Argument	*Options*
none	Indentation, pruning per **zwei:*prune-headers-after-yanking***
1	No indentation, pruning per **zwei:*prune-headers-after-yanking***
2	Indentation, pruning per reverse of **zwei:*prune-headers-after-yanking***
3	No indentation, pruning per reverse of **zwei:*prune-headers-after-yanking***

See the section "Other Zmail Commands".

7.6 Internet Domain Addressing

Zmail supports the Internet RFC822 domain-addressing formats, for the purpose of parsing and replying to messages with domain-format addresses in their headers. If the machine name is registered in the ARPA network host table, that name is used in the address. If the machine name is not registered, the network address isg used, in the form [*address*], where *address* is the four-integer numbers that specify the site in Internet addressing.

8. Changes to the File System in Release 5.2

8.1 Incompatible Changes to the File System in Release 5.2

8.1.1 :clear-screen and :clear-eol Obsolete Names

The following messages for output streams have been renamed:

- **:clear-screen** is now **:clear-window**
- **:clear-eol** is now **:clear-rest-of-line**

In release 5.2, the old names are still accepted for compatibility, but their support is not guaranteed indefinitely. We suggest that you use the new names in new programs.

:clear-window *Message*
Erases the window on which this stream displays. Non-window streams don't support this operation.

:clear-rest-of-line *Message*
Erases from the current position to the end of the current line.

8.2 New Features in the File System in Release 5.2

8.2.1 LMFS Tape Spec Prompting

The new interface for the tape specification for such utilities as the LMFS Dumper/Reloader improves and refines specification techniques.

The fields "Tape host" and "Tape drive" are removed from the LMFS dumper and reloader (and tape comparer, lister, and all other forms of the reloader) menus. In their place is now a new field, "Tape spec", which allows control of all tape parameters, including host, drive, density, and more subtle parameters. See the section "Tape Spec Prompting: Changes to Tape and Disks in Release 5.2".

The next-tape prompting has been streamlined to take advantage of the new tape-prompting system.

It was not previously possible to control density for these programs.

Note: Because Choose Variable Values menus provide no way to control defaults for fields that have explicit merging, it is often wise to click middle on "Tape spec" to yank the typed-out default and edit it.

8.2.2 Direct Access File Streams

Direct access file streams are supported by the file system. The only file system currently supporting them is LMFS. Direct access file streams are designed to facilitate reading and writing data from many different points in a file. They are typically used to construct files organized into discrete extents or *records*, whose positions within a file are known by programs that access them in nonsequential order. Although this could be done with the **:set-pointer** message to input file streams, the direct access facility provides the following additional functions:

- Direct access to output files.
- Bidirectional file streams, which allow interspersed reading and writing of data to and from varied locations in a file.
- No use of network connections or file buffers during the time between data reading and the next call to position. In contrast, using the **:set-pointer** message with ordinary ("sequential") input file streams incurs a significant network and data transfer overhead if the program repeatedly positions, reads several bytes, and then computes for a time.

8.2.2.1 Stream Messages

The following new messages and usages of old messages are relevant to direct access file streams.

:read-bytes *n-bytes file-position* *Message*

Sent to a direct access input or bidirectional file stream, this requests the transfer of *n-bytes* bytes from position *file-position* of the file. The message itself does not return any data to the caller. It causes the stream to be positioned to that point in the file, and the transfer of *n-bytes* bytes to begin. An EOF is sent following the requested bytes. The bytes can then be read using **:tyi**, **:string-in**, or any of the standard input messages or functions.

The stream enforces the byte limit, and presents an EOF if you attempt to read bytes beyond that limit. You must actually read all the bytes and read past (that is, consume from the stream) the EOF for best performance.

It is also possible, before all the bytes have been read, to perform stream operations other than reading bytes. For example, an application might read several records at a time, to optimize transfer and buffering, and decide, after reading the first record, to position somewhere else. Direct access file streams handle this properly. Nevertheless, network and buffering resources allocated to the stream (both on the local machine and server machine) are not freed unless all the requested bytes (of the last **:read-bytes** request) and the EOF following them are read.

If you request more bytes than remain in the file, you receive the remaining bytes followed by EOF.

```
(with-open-file (f "saratoga:>foo.lisp" :direct t
        :direction :io))        ; Opens a file for direct I/O

(send f :read-bytes 10. 25.)    ; Requests 10 bytes, starting at byte 25.

(send f :tyi)                   ; Gets one byte from the stream.

(defvar record (make-array 10. :type art-string))       ; a record buffer

(send f :string-in eof-option record)   ;; Reads 10-character record.
(send f :tyi)                           ;; Consumes the following EOF.
```

Direct Access Output File Streams

The consumption of network and buffering resources by direct access output streams is not currently optimized.

You create direct access output to output and bidirectional direct access file streams by sending a **:set-pointer** message to the stream, and beginning to write bytes using standard messages, such as **:tyo, :string-out,** and so forth. The bytes are written to the file starting at the location requested, at successive file positions. Although you can extend the file in this manner, you cannot do a **:set-pointer** to *beyond* the current end of the file.

Direct access output, therefore, consists of sequences of **:set-pointer** messages and data output. Data is not guaranteed to actually appear in the file until either the stream is closed or a **:finish** message is sent to the stream. See the message **:finish**.

Direct Access Bidirectional File Streams

Bidirectional direct access file streams combine the features of direct access input and output file streams. Sequences of **:read-bytes** messages and reading data can be interspersed with sequences of **:set-pointer** messages and writing data. The stream is effectively switched between "input" and "output" states by the **:read-bytes** and **:set-pointer** messages. You cannot read data with **:tyi** or similar messages if a **:set-pointer** message has been sent to the stream since the last **:read-bytes** message. Similarly, you cannot write data with **:tyo** or similar messages unless a **:set-pointer** message has been sent to the stream since the last **:read-bytes** or **:tyi** messages, or similar operation.

When the EOF of a byte sequence requested with a **:read-bytes** message has been read for a bidirectional stream, the system frees network and buffering resources.

8.2.3 New keyword options to open

There are two new keywords to **open** that control the new direct access file stream feature:

:direct The default is **nil**. t specifies a direct access stream.

:direction :io The file is being opened for intermixed input and output. Bidirectionality is supported only if the stream is to be a direct stream, that is, **:direct t** is given as well.

8.3 Improvements to the File System in Release 5.2

8.3.1 Faster Creation of Logical Pathname Hosts

The functions **fs:set-logical-pathname-host** and **fs:make-logical-pathname-host** can now avoid querying the namespace server to determine whether a physical host already exists with the same name.

Caution: This saves some time, although it could cause confusion later, if a physical host is added with the same name.

fs:make-logical-pathname-host *name* &key *Function*
no-search-for-shadowed-physical

Defines *name*, which should be a string or symbol, to be the name of a logical pathname host. *name* should not conflict with the name of any existing host, logical or physical.

This function loads the file sys:site;*name*.translations. This file should contain a single form: a call to **fs:set-logical-pathname-host**. The file is always loaded into the **file-system** package. See the function **fs:set-logical-pathname-host**.

fs:make-logical-pathname-host not only loads this file but also arranges for the same file to be reloaded in the future. **load-patches** checks the translations file for each logical host that is defined in the current world; if any file has been changed it is reloaded. **load-patches** does this if and only if no specific systems are specified in its arguments.

fs:make-logical-pathname-host alters the **logical-pathnames-translation-files** system so that it contains the translations files for all logical hosts defined in the current world. **load-patches** loads updated translations files by calling **make-system** on this system.

When a world load is taken to a new site, the translation file for each logical host that is defined in the current world is reloaded from the new site's site directory. This changes all logical pathnames to map into the new set of physical pathnames defined by the new site.

An **fs:make-logical-pathname-host** form often appears in the file sys:site;*system-name*.system. **make-system** looks for this file when given the name of an unknown system. In addition to a call to **fs:make-logical-pathname-host**, this file should contain a call to **si:set-system-source-file**, which specifies the logical pathname of the file containing the **defsystem** form.

The argument *no-search-for-shadowed-physical* (default **nil**) means to look only in the existing pathname hosts for a host with the same name as the logical host. This saves time by not asking the namespace server whether the name of the newly defined logical host conflicts with the names of any physical hosts, but it prevents you from seeing the following warnings:

```
Warning: the host ~A must now be referred to as ~A: in pathnames,
                        since ~A is now a logical pathname host.
                        This affects ~[no~:;~:*~D~] extant pathnames.

Warning: the nickname ~A: for the physical host ~A
                          will now refer instead to the
                          logical pathname host ~A.
                          Use ~A: in pathnames.
```

Example:

Following are the contents of the file sys:site;cube.system:

```
;;; -*-  Mode: LISP; Package: USER -*-

(fs:make-logical-pathname-host "cube")
(si:set-system-source-file "cube" "cube: cube; cubpkg")
```

fs:set-logical-pathname-host *logical-host* &key *physical-host* *translations no-translate* *no-search-for-shadowed-physical* *Function*

fs:set-logical-pathname-host creates a logical host named *logical-host* if it does not already exist. It then establishes translations of logical directories on *logical-host* to physical directories on *physical-host*. *translations* is a "translations" list of two-element lists of strings representing associated logical directories (source patterns) and physical directories (target patterns). For the format of the lists and the translation rules: See the section "Logical Pathname Translation".

Logical directory names should be terminated by semicolons, but **fs:set-logical-pathname-host** accepts names without semicolons. Host names can appear in the strings in the translations list, but each logical host in a string must refer to the same host as *logical-host*, and each physical host in a string must refer to the same host as *physical-host*. If the physical pathname is on a TOPS-20 or VMS device, you must include the host name (either explicitly or implicitly, with an initial colon) so that the device is not taken to be the host.

If *no-translate* is **nil** or unsupplied, the translation of every interned logical pathname is checked. Properties are copied from the old physical pathname to the the new one, and logical pathnames that now have no corresponding physical pathnames are uninterned. If *no-translate* is not **nil** this mapping is suppressed, and some physical pathnames might not get the properties of the logical pathname. The consequences of this are unknown.

A call to **fs:set-logical-pathname-host** is usually the only form in the file sys:site;*logical-host*.translations. This file is loaded by **fs:make-logical-pathname-host** (always in the **file-system** package), which also arranges for it to be reloaded in the future. **load-patches** checks this file for all logical hosts in the current world and reloads the file if it has changed.

The argument *no-search-for-shadowed-physical* (default **nil**) means to look only in the existing pathname hosts for a host with the same name as the logical host. This saves time by not asking the namespace server whether the name of the newly defined logical host conflicts with the names of any physical hosts, but it prevents you from seeing the following warnings:

```
Warning: the host ~A must now be referred to as ~A: in pathnames,
                        since ~A is now a logical pathname host.
                        This affects ~[no~:;~:*~D~] extant pathnames.

Warning: the nickname ~A: for the physical host ~A
                          will now refer instead to the logical pathname host ~A.
                          Use ~A: in pathnames.
```

Example:

Following is the contents of the file sys:site;cube.translations:

```
;;; -*- Mode: LISP; Package: FILE-SYSTEM -*-

(set-logical-pathname-host "cube"
  ':physical-host "pointer"
  ':translations '(("cube;" ">cube>")))
```

8.3.2 :finish is supported by some output file streams

The **:finish** message is supported by some output file streams (currently, remote and local LMFS).

:finish finalizes file content. It ensures that all data have actually been written to the file, and sets byte count. It converts ordinary output openings (that is, openings that are not for direct access) into append openings. It allows other users to access the data that have been written before the **:finish** message was sent.

9. Changes to Networks in Release 5.2

9.1 Incompatible Changes to Networks in Release 5.2

9.1.1 Size of Packet Buffers

The current size of a packet buffer on the 3600 family is 1498 bytes, which is different than in previous releases. Note, however, that the size of packet buffers is not guaranteed to be the same from one release to another.

9.2 New Features in Networks in Release 5.2

9.2.1 Using the Terminal Program with Hosts Connected to the Serial Line

You can connect a 3600 family machine to another host via the serial line. Specifically, you can use the terminal program to communicate with another host when the 3600 serial line is connected to a terminal port on the other host.

The network system treats the set of hosts connected to the serial lines of a 3600 as a special network, a *pseudonet*. Before you can use the terminal program to talk to another host over the serial line, you must use **tv:edit-namespace-object** to create this network and assign an address on that network to the 3600. You might want to create or modify the remote host as well.

1. Create the network. Give it a **name** attribute associated with the 3600 and a **type** attribute of **serial-pseudonet**.

   In the following example, Merrimack is the name of the 3600:

   ```
   NETWORK MERRIMACK-SERIAL
   TYPE SERIAL-PSEUDONET
   ```

2. Add an entry to the **address** attribute of the 3600 to specify that the 3600 is connected to the new network. Each **address** entry is usually a pair of the form (*network address*). By convention, the 3600 is assigned address 0 on a serial pseudonet. Following is an example of a new **address** entry for the 3600 Merrimack:

   ```
   ADDRESS MERRIMACK-SERIAL 0
   ```

3. If the line rate of the serial line is other than 9600 baud, supply a **peripheral** entry for the 3600 giving the correct baud rate. The peripheral type is **serial-pseudonet**, and the **unit** attribute is the unit number of the serial line. Following is an example of a **peripheral** entry for the 3600:

   ```
   PERIPHERAL SERIAL-PSEUDONET UNIT 2 BAUD 4800
   ```

4. If you want the terminal program to start out simulating one of the supported terminal types, add a **terminal-type** attribute to the peripheral. Currently supported terminal types are the VT100 and Ann Arbor Ambassador. For example, to make the terminal program simulate an Ambassador, add to the 3600 a **peripheral** entry of this form:

```
PERIPHERAL SERIAL-PSEUDONET UNIT 2 BAUD 9600 TERMINAL-TYPE Ambassador
```

You can now use the terminal program to connect to the remote host. At the "Connect to host:" prompt, you must supply an address of the form MERRIMACK-SERIAL|2. If you want to type a name or nickname of the remote host instead, use **tv:edit-namespace-object** to add **address** and **service** entries for the remote host. If the remote host does not exist in the network database, use **tv:edit-namespace-object** to create it.

For the **address** entry, specify the serial pseudonet and an address that corresponds to the unit number of the serial line to which the host is connected. The **service** entry is a triple of the form (*service medium protocol*). For the regular host login server, *service* is **login**, *medium* is **serial-pseudonet**, and *protocol* is **tty-login**. Following is an example of **address** and **service** entries for the remote host Blue connected to the 3600 Merrimack:

```
HOST BLUE
SYSTEM-TYPE TENEX
ADDRESS MERRIMACK-SERIAL 2
SERVICE LOGIN SERIAL-PSEUDONET TTY-LOGIN
```

You can also use the serial line to connect to servers other than normal login on a remote host. You must add a **service** entry for the remote host to specify the kind of service, the **serial-pseudonet** medium, and the protocol that the remote host uses. You must also add an **address** entry on the serial pseudonet for the remote host. In the **address** entry, specify the address in the form *protocol=unit* instead of just *unit*. Following are examples of **address** and **service** entries for a file server using protocol **myftp** on remote host Blue:

```
HOST BLUE
SYSTEM-TYPE TENEX
ADDRESS MERRIMACK-SERIAL MYFTP=2
SERVICE FILE SERIAL-PSEUDONET MYFTP
```

For information on the terminal program: See the section "New terminal program SELECT T".

For information on network and host attributes: See the section "Object Definitions: Network Database".

For information on services, media, and protocols: See the section "The Lisp Machine Generic Network System".

9.2.2 New Variable to Hold Packet Size

neti:raw-packet-buffer-size *Variable*

The variable stores the number of bytes in the array returned by **neti:allocate-packet-buffer**. This is the maximum number of bytes that any packet can have. The value depends on the architecture of the machine and, to a lesser extent, on the particular system release. It is not guaranteed to be the same from one release to another. Nevertheless, since packet buffers can be used as temporary storage, knowing their size can be important.

9.3 Improvements to Networks in Release 5.2

9.3.1 Improvements to :error-disposition Option of net:define-server

The **:ignore** and **:notify** keywords to the **:error-disposition** option of **net:define-server** now take optional values, as follows: **:error-disposition** (**:notify** *error-flavor-1 error-flavor-2 ...*) and **:error-disposition** (**:ignore** *error-flavor-1 error-flavor-2 ...*). The ability to specify one or more error flavors allows finer control over error notification. See the special form **net:define-server**.

9.3.2 Improvements to tv:edit-namespace-object

When editing an object that lives in more than one namespace (for example, a host that is on both the local Chaosnet and the Arpanet), a pop-up window appears that lists the name of each namespace and asks which "view" (that is, which namespace) you want to edit. For a complete description of **tv:edit-namespace-object**: See the section "User Interface to the Network Database".

10. Changes to the FEP in Release 5.2

This chapter describes changes to the FEP software that are concurrent with Release 5.2. FEP software is distributed in its own versions, which are separate from Lisp software releases. Release 5.2 requires FEP version 17 or higher. Unless otherwise indicated, each change in any FEP version applies to later versions as well.

10.1 FEP Version, Serial I/O, and Model Number Compatibility

This section describes which FEP proms are appropriate for the Lisp Machine, based on the model number of the machine (3600, 3640, 3670). In general, the higher-numbered versions are preferable.

On all models, FEP prom version 22 or 24 is required for full functionality of the three serial lines attached to the Lisp Machine.

For 3600s, you can use FEP versions 17, 18, 22, and 24.

For 3640s, you must use FEP version 24.

For 3670s, you can use either FEP version 22 or 24.

10.2 FEP Version 18: Improvements

10.2.1 Changes in FEP Fault Lights on the 3670

Beginning with FEP version 18, 3670s use the following rules for turning on and off the lights on the processor front panel:

- When the FEP starts up, it clears all lights on the processor front panel.
- When the FEP starts Lisp, the FEP turns the Fault light off and the Run light on.
- Whenever Lisp stops for any reason, the FEP turns the Run light off and the Fault light on.
- When the FEP executes the Clear Machine, Load Microcode, or Load World command, it turns the Fault light off on the assumption that you are fixing the cause of the fault.

10.3 FEP Version 18: New Features

10.3.1 si:halt has new *fep-commands* argument

In FEP version 18, **si:halt** has a new optional argument *fep-commands* that can be used to invoke FEP commands.

si:halt &optional *fep-commands* *Function*

On the 3600, **si:halt** stops execution of Lisp and gives control to the FEP. This function does not interrupt disk I/O operations when it stops Lisp.

The function **sys:%halt** should no longer be used as it can halt disk operations. Interrupting a disk write can cause a fatal ECC error later, because the contents of the disk block are incomplete. This can render directories and other files inaccessible and is a particular problem when halting the machine while using LMFS.

If the optional *fep-commands* are not supplied or if the argument is **nil**, **si:halt** places you at the FEP level. If you supply a string for *fep-commands*, Lisp causes the FEP to interpret the string as if the string were typed in from the keyboard or read from a command file. It is important to use a new line for each command that is part of the string.

The following form halts Lisp, passes control to the FEP and causes the Show Version and Continue commands to be executed.

```
(si:halt "Show Version
Continue
")
```

This example generates the following display:

```
(si:halt "Show Version
Continue
")
Lisp stopped itself.
Closing command file.
Fep>
Fep>Show Version
Fep version 22 running in prom.
NIL
```

si:halt with optional *fep-commands* can be included in function definitions to define booting and loading procedures, as shown in the following examples:

```
(defun boot-macsyma ()                          ;boots using >Macsyma.boot
  (si:halt "Boot >Macsyma.boot
"))

(defun reboot ()                                ;boots using the default
  (si:halt (format nil "Boot~%")))              ;boot file

(defun reload-world ()                          ;loads the default world
  (si:halt (format nil "Load World~%Start~%"))) ;and starts Lisp

(defun reload-microcode ()                      ;loads the default
  (si:halt (format nil "Load Microcode~%Start~%"))) ;microcode and starts
```

10.3.2 h-c-*upper-left* waits for Lisp to stop itself

With FEP version 18, pressing h-c-*upper-left* does not immediately stop Lisp. Instead, the FEP asks Lisp to stop itself cleanly. If Lisp does stop itself, the FEP prints the message "Lisp stopped itself." If Lisp does not stop itself after about three seconds, the FEP prints, "Waiting for Lisp to stop itself..." If after another three seconds Lisp does not stop itself, the FEP forcibly stops Lisp and prints, "Halting execution of Lisp." The purpose of this behavior is to reduce the chance of halting the 3600 during a disk write, which might cause ECC errors. The preferred way of stopping Lisp is still to call **si:halt**; use h-c-*upper-left* only if no Lisp Listener is responding.

10.4 FEP Version 22: Incompatible Changes

10.4.1 FEP Halt Command Differences in 3600s and 3670s

The FEP's Halt command behaves differently on a 3600 than on a 3670. On the 3600, the Halt command asks the same question as it did previously — "Do you really want to halt the FEP?" — but additionally displays the message "FEP Halted" in the nanofep display. On the 3670, the Halt command instead asks the question, "Do you really want to power down the 3600?", because it is possible to power down the 3670 from the FEP. Additionally, the fault light on the switch panel is lit.

10.4.2 FEP Shutdown Command Replaces Halt Command in FEP Version 22

In FEP version 22, the FEP Shutdown command replaces the Halt command. Although the Halt command still exists for compatibility, it might disappear in the future. See the section "Less Common FEP Commands".

10.5 FEP Version 22: New Features

10.5.1 New FEP Command: Set Display-string

In FEP version 22, a new command, Set Display-string, has been added for 3600s. This command displays the string in the nanofep display of machines that have a nanofep display. The length of the string is limited to 12 characters, the number of characters in the nanofep display. If more characters are used, the string is truncated. This command can be used in a .boot file.

See the section "Less Common FEP Commands".

10.5.2 New FEP Command: Set Monitor-type

FEP version 22 has the new Set Monitor-type command which provides a method for specifying the monitor type. The Set Monitor-type command ensures that the sync program used is for the monitor type requested. *monitor-type* can be either **Moniterm** or **Philips**; the types can be abbreviated to their first letter, **m** for **Moniterm** and **p** for **Philips**.

Set Monitor-type is used if the monitor is changed at a site and the ID prom is not changed accordingly. This command can be used in a boot file.

See the section "Less Common FEP Commands".

10.6 FEP Version 22: Improvements

10.6.1 Changes in FEP Fault Lights on the 3600

Beginning with FEP version 22, 3600s use the following rules for turning on and off the lights on the processor front panel:

- When the FEP starts up, it clears all lights on the processor front panel.
- When the FEP starts Lisp, the FEP turns the Fault light off and the Run light on.
- Whenever Lisp stops for any reason, the FEP turns the Run light off and the Fault light on.
- When the FEP executes the Clear Machine, Load Microcode, or Load World command, it turns the Fault light off on the assumption that you are fixing the cause of the fault.

10.7 FEP Version 23: Incompatible Changes

10.7.1 New Default for FEP Add Paging-file Command

Beginning with FEP Version 23 proms, >reserve.page is the default for Add Paging-file. For earlier proms, >page.page is the default.

See the section "Less Common FEP Commands".

10.8 Additional Information About the FEP in Release 5.2

10.8.1 Previously Undocumented FEP Command: Load FEP

The Load Fep command loads and starts loadable FEP programs. The syntax is

```
Load Fep file
```

The names of the FEP programs are usually of the form Vxx-name, where xx is the number of the FEP version on which the program runs and name is the name of the program. For example, the file name FEP:>V22-debug.flod would indicate that the program ran on FEP version 22 and was used for debugging Lisp crashes.

See the section "Less Common FEP Commands". See the section "FEP File Types".

10.8.2 Previously Undocumented Topic: Debugging in the FEP

Files were provided in Release 5.0 that can be used for debugging in the FEP for FEP versions 17, 18, and 22.

10.8.2.1 Debugging in the FEP

The Release 5.0 tapes included some files that were provided as an extra debugging aide. These files can be used to enter a debugging mode in the FEP. This is especially useful for problems that cause control to return to the FEP, thus making it impossible to use the debugging methods normally used in Lisp.

These files are:

- v17-debug.flod
- v18-debug.flod
- v22-debug.flod

The file name indicates the version of the FEP software for which the file can be used. The files should now reside on your sys host in the directory with the logical pathname sys: l-fep; v*nn*-debug flod where *nn* indicates the version of FEP software.

To use these files, you should copy the appropriate file to the FEP filesystem *before* you need to use it. To copy the file, first find out which version of FEP software is

installed in your machine. You can do this by either typing the Lisp function **print-herald** to Lisp or by typing the Show Version command at the FEP level. Your FEP version should be either 17, 18, or 22. This corresponds to the number in one of the filenames above. To copy this file to your FEP file system, use the Zmacs command Copy File (m-X).

For example, if you are using FEP version 22 software, you would use the following command to copy the .flod file to the FEP file system:

```
Copy File (m-X) sys:l-fep;v22-debug.flod fep0:>v22-debug.flod
```

The .flod file cannot be used on any other FEP version; trying to use one on a different FEP version will have no effect.

After you have copied the file to the FEP file system, you can enter the debugging mode by loading the file with the Load Fep command, as shown in the following example:

```
Fep>Load Fep >v22-debug.flod
```

This will put you into a debugging mode very similar to the Debugger under Lisp, whereby you can move up and down the stack to examine the state of the machine and determine the source of the problem. The HELP key lists the commands that are available.

One particularly useful command when the 3600 has crashed while paging is c-m-S. This command allows you to switch between the auxiliary stack (where paging code runs) and the normal stack (where user code runs). If the 3600 crashed while executing on the auxiliary stack, user stack frames will not be found until c-m-S is executed.

10.8.3 Using FEP Clear Paging-files Command Before Add Paging-file

You should use the Clear Paging-files command before using the Add Paging-file command in two situations:

- If you are typing in the Add Paging-file command from the keyboard.
- If you are adding more than one paging file. If you are adding more than one paging file, use Clear Paging-files before adding the set of paging files.

Neither the FEP nor Lisp has any error checking to detect doubly allocated disk blocks, which might occur in either of these two situations. Not clearing the paging files will eventually cause surprising errors to occur after booting.

See the section "Less Common FEP Commands".

10.8.4 Previously Undocumented Tape Utility for Writing FEP Files to Tape

In Release 5.0, the **tape:write-fep-files-to-tape** function was added. This make it easier to write large (requiring more than one cartridge tape) FEP files to tape using a local tape drive. This is very useful for large world loads. This is described in more detail in the chapter *Changes to Tape and Disks in Release 5.2.*

11. Changes to Tape and Disks in Release 5.2

11.1 Incompatible Changes to Tape and Disks in Release 5.2

11.1.1 Must Specify Dumper Tape Drive Identifier

Lisp machine tape servers running release 5.2 now interpret tape drive identifiers; they had not in the past. Therefore, when you invoke **distribution:write-distribution-tape** and are presented with the Choose Variable Values Menu (labeled *Distribution Dump Options*), it is necessary to unambiguously specify the tape drive you intend to use. Leaving this field blank, as in the past, will no longer work for Lisp Machine tape servers. See the section "Tape Spec Prompting: Changes to Tape and Disks in Release 5.2". See the section "The Dumper Choose Variable Values Menu".

11.2 New Features in Tape and Disks in Release 5.2

11.2.1 New Feature: Carry Tape System

A new tape portability tool, the carry tape system, is now supported.

The carry tape system provides a means of dumping selected files or sets of files to magnetic tape (cartridge or industry-compatible) and loading them at a later time, at a potentially different site. Files can be dumped from any host or set of hosts, and reloaded to any place on any host.

The carry tape system is meant to provide a standard, system-independent interchange medium for exchanging single programs and files between sites. It is meant to fill in a gap between backup dumpers and the distribution tape system. Its interface is easy to use, and requires no files or declarative forms to be prepared in advance.

The following are the three components of the carry tape system:

- The carry dumper
- The carry loader
- The carry tape lister

11.2.1.1 The Dumper

tape:carry-dump *file-or-files* &key *tape-host density reel* (*report* **t**) *Function*

Dumps a file or set of files to a carry tape. Any type of file can be dumped. Character files are dumped and reloaded using the Lisp Machine character

set as an interchange medium. Binary files are dumped and reloaded with the proper byte size as long as either of the following is true:

- The file is of one of the system's known canonical types.
- The operating system on which the file resides knows and can supply the byte size.

file-or-files a pathname, filespec, or list of pathnames and/or filespecs. Wildcard pathnames or filespecs may be used. Recursive ("accordion") wildcards may be used to dump subtrees on those hosts that support them.

tape-host a host object or the name of a host object to use for tape access. **:local** specifies the local tape drive. If not specified, the standard tape host prompt and defaulting mechanism is used.

density a fixnum, specifying tape density, which may be used when the applicable default is not appropriate.

reel can be a string, specifying tape reel name for tape servers that need this information (none of the currently supported ones do).

report tells the carry dumper to report its progress as it dumps files. A value of **nil** tells it not to. The default is to report to **standard-output**. Any value besides **nil** or **t** is expected to be a stream to which the reports will be written.

Currently, carry dumps must fit on one tape.

The carry dumper commences by finding out all available information about the files to be dumped, verifying their existence. It then asks for confirmation, and proceeds to dump all the files specified without intervention.

```
(tape:carry-dump "swanee:>minerals>*.d*")
To be dumped:
swanee:>minerals>*.d*: 7 files
Is this right? (Y or N) Yes.
Type name of tape host (default (CR) = POINTER): scrc
Tape mounted on drive mta0:.
Dumping swanee:>minerals>abel.data.3 (5-bit bytes)
Dumping swanee:>minerals>abel.directory.7
.....
End of dump.
```

11.2.1.2 The Loader

The carry loader loads files from a carry tape. The carry system makes no attempt to copy any file properties, including author and creation date. The carry system copies only file contents, and the loader provides reasonable defaults for the target file name.

tape:carry-load &key *host density reel* (*report* **t**) *Function*

host	a host object or the name of a host object to use for tape access. **:local** specifies the local tape drive. If not specified, the standard tape host prompt and defaulting mechanism is used.
density	a fixnum, specifying tape density, which may be used when the applicable default is not appropriate.
reel	can be a string, specifying tape reel name for tape servers that need this information (none of the currently supported ones do).
report	tells the carry loader to report its progress as it loads files. A value of **nil** tells it not to. The default is to report to **standard-output**. Any value besides **nil** or **t** is expected to be a stream, to which the reports will be written.

These arguments are rarely needed.

The carry loader begins its operation by reporting the filespecs given to the dumper, and asks if you wish to load all of the files dumped. If only one filespec or pathname was given, it is assumed that you want to load it all, and no question is asked:

```
(tape:carry-load)
Type name of tape host (default (CR) = APSO): beta
Tape mounted on drive mta0:.
Carry dump made by DCF.
Dump taken at 11/13/82 09:05:22.
Dumped on machine EAGLE.
Dumped: e:>trees>apple.orchard
```

The set of files dumped as a result of each filespec given to the dumper is called a *group*. If many groups were dumped, the loader lists the filespecs of each group at the start of its operation, and asks for instructions about which groups are to be loaded (selectively) and which groups are to be skipped:

```
The following groups of files were dumped:
e:>trees>apple.orchard
e:>animals>whales>tails.tales
e:>baseball>runs>foul.*
-----
Load all these files? (ABORT to get out) (Y, Q, or M)
```

The possible responses are:

Y	Ignore distinctions of group, and proceed as described below.
Q	Query about each individual group, and proceed as below for those groups that are selected for loading.
M	Same as Q, but present a multiple-choice menu instead of querying for each group.

If you do not want to load anything, ABORT will work at any time.

The carry loader can either query for the target location of each file to be loaded, or proceed in semi-automatic mode, in which the host and directory from which each file was dumped are used as a key to target loading of subsequent files from that host and directory. The name, type, and version of each file to be loaded are developed automatically from the name, type, and version of the file that was dumped, by means of the same mechanism used by ordinary file copying.

The normal action of the carry loader is to query for each file, with a query of the following form:

```
Load SWANEE:>minerals>rock5.data.6 into BULLWINKLE:/usr2/jones/rock5.data?
       (Y, N, O, or A)?
```

The following responses apply:

Y, SPACE	Load the file into the place specified. The host and directory shown remain the default target directory for all files from this host and directory at the sending site.
N	Do not load this file at all. The host and directory shown remain the default target directory for all files from this host and directory at the sending site, in spite of this.
O	Prompt for another place in which to put this file. The host and directory into which this file is then loaded become the default for all subsequent files from the same host and directory at the sending site. You are queried again at the time such files are encountered.
A	Load the file into the place specified. All further files from the same host and directory at the sending site are then automatically loaded into the same host and directory as this file without querying you.

11.2.1.3 The Lister

The carry lister describes what is on a carry tape. Once started, it does not interact in any way.

tape:carry-list &key *host density reel* (*report* **t**) *Function*

host	a host object or the name of a host object to use for tape access. **:local** specifies the local tape drive. If not specified, the standard tape host prompt and defaulting mechanism is used.
density	a fixnum, specifying tape density, which may be used when the applicable default is not appropriate.
reel	can be a string, specifying tape reel name for tape servers that need this information (none of the currently supported ones do).
report	tells the carry lister to report its progress as it lists files. A value of **nil** tells it not to. The default is to report to **standard-output**. Any value besides **nil** or **t** is expected to be a stream, to which the reports will be written.

These arguments are rarely needed.

11.2.1.4 What Tape Tool is Appropriate?

Symbolics supports at least five different tape formats, all different and incompatible. Each format is specific to one Lisp Machine tool. Many people wonder which tool is appropriate for which application.

The *LMFS dumper and reloader* are used for snapshotting files on a local Lisp Machine file system and reloading those files on the same local Lisp Machine file system, in the same place at a later date. The intended use of LMFS backup is to reload files onto the same machine from which they were dumped.

The *Distribution dumper and loader* are intended to distribute transportable systems and libraries, defined by system declarations and pseudosystem declarations, on logical hosts, from one site to another. The distribution system specializes in finding appropriate source and object versions of systems, appropriate patch files, and so forth. It is cumbersome to use it for transporting miscellaneous files between sites or hosts.

TAPEX is a format for transferring character (source) files (only) between hosts. It is the only one of these formats which can be read or written by other than a Lisp Machine. TAPEX programs exist for TOPS-20 and Multics, as well as for the Lisp Machine (**tape:tapex**). It cannot deal with any type of file except character files, and has no features to speak of. Each individual dumping or loading requires an interaction.

Carry tape is the most general tool, and is meant to dump individual files and sets of

files given by wildcard filespec, and load them at any site. Recent improvements to the wildcard facility make this a very powerful and easy-to-use tool.

IFS tape is a 3600-specific format readable by the 3600 FEP. It is used to transport and distribute 3600 FEP files and file systems, including 3600 world loads and microcode.

11.2.2 New Feature: Maxtor XT-1140 Support

The Maxtor XT-1140 disk is supplied with Symbolics 3640 computers. Its unformatted capacity is 140 megabytes.

11.2.3 New Feature: TD-80 Tape Drive Support

Effective with Release 5.2, the 3600 supports the TD-80 industry-compatible tape drive. TD-80 drives on 3600 systems are denoted by drive identifiers 0, 1, and so forth. The cartridge drive, when present, is denoted by the drive identifier "cart". See the section "Tape Spec Prompting: Changes to Tape and Disks in Release 5.2".

11.3 Improvements to Tape and Disks in Release 5.2

11.3.1 Distribution Tape Program Changes

11.3.1.1 Dumper

If a tape error is encountered during dumping, the dump can be restarted on another tape without respecifying the files to be dumped by means of a restart option which appears on the restart option list at the time of the error. If you invoke the restart option, it prompts (using the new tape-prompting system) for a (potentially new) tape spec and restarts the dump with a new tape attachment and the same dump systems and options. See the section "Tape Spec Prompting: Changes to Tape and Disks in Release 5.2".

There is no more support for old `Net Listen` tape protocol.

11.3.1.2 The Dump Menu

When doing a distribution dump, you can change the tape host without using the mouse by invoking the new keyboard command `H`. `Local` is an acceptable input.

The following dump menu line no longer exists:

```
Local Remote Net Listen Band
```

On the LM-2 it is now:

```
Tape Band
```

`Band` means the same as it did previously, and `Tape` replaces `Local` and `Remote`. On the 3600, the only option is tape.

The line:

```
Tape Host
```

now accepts the name of any host, or Local, although, on the LM-2, it will be ignored if `Band` is selected.

11.3.1.3 New Tape Spec Prompting: Distribution Tape Program Changes

The new tape specification system provides correct defaults for host and drive. See the section "Tape Spec Prompting: Changes to Tape and Disks in Release 5.2".

Reloader/mapper/copier tape prompting: On the 3600 (again, because `Band` is not a possibility), there is no more "where to get the distribution tape" menu. You get the canonical new tape-prompting system prompt (which is customized, as it can be).

On the LM-2, there is still the "where to get the distribution tape" menu, but the default host and drive are chosen with the help of the new tape-prompting system.

11.3.2 Improvements to Distribution Tape Loader

11.3.2.1 Specifying a New Logical Host Name

When loading an ordinary distribution tape, the distribution tape loader now always asks you to confirm the use of a logical host before using it, giving the opportunity to load the tape onto a different logical host than the one it was dumped from (sys). For example, you can use this feature to load a sys: distribution onto experimental-sys:, or you might simply prefer a different logical host name because logical host names differ between sites. This feature does not work for patch files, however, since they do not use logical pathnames.

The first time the loader sees a logical pathname host during any load operation, it asks whether sys is the correct logical host at this site. Previously it did not ask this question for hosts that were already defined, especially sys. The possible answers to the question vary depending on whether the logical pathname host is already defined. A new feature allows you to type `U` instead and load the tape into a logical host other than sys. The loader lets you define this logical host according to the following answers:

`Y, E, SPACE`	Existing Logical Host is Okay
`R`	Define logical host by reading <filename on sys:site;>
`D`	Define Logical Host
`U`	Use a different logical host
`S`	Skip this file
`P`	Punt this host for the rest of the tape

Of the first three answers above, one and only one will be available.

A logical host can now have the same name as a physical host. When this happens,

a warning is printed. If the name of the logical host is the same as the currently preferred name of the physical host:

- The system chooses a new preferred name for pathnames on the physical host.
- The warning includes the old and new names.
- All cached occurrences of the old name in pathnames are removed.

You can always choose a new name, since you can use a namespace-qualified name (containing a vertical bar). (Logical host names are not registered in the namespace system and never contain vertical bars, whitespace characters, or colons.)

Examples of the possible warnings are:

```
Warning: the host QUABBIN must now be referred to as QUABBIN: in
pathnames,  since Q is now a logical pathname host.  This affects 3104
extant pathnames.

Warning: the nickname F: for the physical host POINTER  will now
refer instead to the logical pathname host F.    Use P: in pathnames.

Warning: the host QUABBIN must now be referred to as SCRC|Q: in
pathnames,  since QUABBIN is now a logical pathname host.  This affects
3104 extant pathnames.
```

11.3.3 Previously Undocumented Utility: Writing FEP Files to Tape

In Release 5.0, the **tape:write-fep-files-to-tape** function was added. This make it easier to write large (requiring more than one cartridge tape) FEP files to tape using a local tape drive. This is very useful for large world loads.

11.3.3.1 Writing FEP Files to Tape

You can write files to tape using a local tape drive with the **tape:write-fep-files-to-tape** function. This can be used for large (requiring more than one cartridge tape) FEP files and is very useful with large world loads. To do this, you first get access to the necessary software by making a fep-tape system. You can use the **:silent** and **:noconfirm** options, as shown in the following example:

```
(make-system 'fep-tape :silent :noconfirm)
```

The next step is to use the function **tape:write-fep-files-to-tape** to write the FEP files to tape. This can be used to write both microcode and world load files.

To restore these files from tape, use the FEP command Disk Restore. For more information about restoring files from tape, refer to the Release 5.0 Software Installation Guide.

When the end of tape is encountered, the machine will return to the FEP. You then put the second tape into the tape drive, and use another Disk Restore command using the same destination filename. This appends the data from the second tape onto the designated file.

tape:write-fep-files-to-tape &optional *mic-name* *Function*

Writes FEP files to tape. *mic-name* is the name of file-format microcode that precedes the microcode and world load files on distribution tapes.

When an argument is supplied within the form, the function assumes that the argument is the file-format microcode and uses stream format. When an argument is not supplied, you are prompted for a file name, which is assumed to be a microcode or world load file and which is then written out in distribution format. Thus, supplying a file-format microcode name should be used only when writing an initial microcode file to tape.

You will be prompted as to whether the first tape is in place. Put the tape in the local tape drive and then answer "Y". You will then be prompted as to whether you wish to write a file to tape; you should answer "Y". Next, enter the filename of the world load. You will also be prompted for file and restoration comments. As the file is written out, the number of blocks will be printed on the screen. When the end of the tape is reached, the following message is printed:

```
"starting a new tape"
```

and you will be prompted as to whether a new tape is in place. Put a fresh tape in the drive and type "Y" to continue. This will continue writing the file on the second tape.

tape:write-fep-files-to-tape is in a separate system named the fep-tape system. To use **tape:write-fep-files-to-tape**, you must first make a fep-tape system by typing **(make-system 'fep-tape)**.

11.4 Tape Spec Prompting

11.4.1 Tape Specs

The concept called a *tape spec* provides a means for describing, passing around, and defaulting tape host, drive, and parameter selection.

The basic idea is that **tape:make-stream**, which is called by all tape-using programs to create tape streams, is provided by programs with partially complete information, sometimes none at all. If this information is insufficient to open a tape stream, **tape:make-stream** prompts for more information.

Only one such prompt is issued by **tape:make-stream**. In this prompt, a complete specification (a *tape spec*) for all variable parameters of the tape stream to be opened is presented as a *default*. This default is made by merging previously used parameters, that is, the last time since boot that you created a tape stream, with ones explicitly provided by the tape-using program that called **tape:make-stream**. The meaning of this default, and its use in processing your response to the prompt, is very similar to the parallel concepts in pathname merging. If you press RETURN,

the default is used as is to provide all parameters to **tape:make-stream**. Otherwise, you can specify individual parameters, which are *merged*, as described below, with the default, to produce parameters to **tape:make-stream**. The merged specification is saved to produce the new default, for next time.

The default default values (the defaults after boot) are chosen from the available hardware on the local machine and the declared tape servers at the site.

For many parameters, a "null value" can be specified, which applies whatever the host chooses by default.

The following is a typical prompt:

```
Type tape host or spec (default (Return) = Local: 0, reel=joe, den=1600):
```

The defaults in this example are tape on the local machine, drive 0 (which is either LM-2 or 3600 (TD-80) half-inch (industry compatible)) tape, a reel named "joe", and density of 1600 bits per inch.

11.4.2 Tape Spec Parameters

The following are the supported changeable parameters with allowable synonyms:

host (machine) The *tape host*, that is, the host you have mounted or intend to mount the tape on. The string "local" is acceptable to specify the local host. Any name known to the namespace system may be used to specify a host.

device (dev, unit) The tape drive on that host that will be used. For Lisp Machines, numeric values of 0 to 7 specify the designated "reel-to-reel" (half-inch industry-compatible) drives, when present, and "cart" specifies 3600 cartridge tape. For other tape hosts, you must know the host's drive specification format, if the tape host does not choose the drive you expect by default.

reel (volume, vol) The name of the tape reel. Whether or not this parameter is necessary depends on both the program and the tape host being used. For instance, the distribution dumper requires a reel name to compute the name of the log file. Few other Lisp Machine programs do. Some tape hosts (none currently supported) might require a reel name in order to inform their operators to locate and mount the reel you want.

density (dens, den)
The tape density in bits per inch. Some tape controllers on some hosts utilize this information, others ignore it. The default, if the program does not specify a density, is 1600 for non-cartridge tape.

number-of-buffers (buffers, bufs, buffs, n-buffers, n-bufs, n-buffs)
This is a performance-control parameter only applicable to local tape on a Lisp Machine. It controls the number of read-

ahead/write-behind buffers used by both the industry-compatible tape stream implementations and the 3600 cartridge tape stream implementation.

record-length (length, len, reclen, recsize)
The length of tape records, in bytes. Normally, programs specify this parameter explicitly, such that another program with which it is communicating via tape, can read what it has written or write what the other program can read. Nevertheless, it is sometimes useful to change this parameter at tape spec merging time, in order to experiment with unknown tapes, and so forth. This parameter should not be altered casually.

granularity (gran) This specifies a number *n*, such that all tapes written by the stream being created are padded with zeros (the pad character is program-changeable) to be multiples of *n* in length. It is ignored in the case of cartridge tape. The default for this parameter depends on the tape host and its stream and hardware. For 3600 industry-compatible tape, it is 1. For LM-2 industry-compatible tape, it is 4. Other hosts have various defaults, often not 1. Although supplied Lisp Machine tape applications specify a granularity of 4, it would be wrong for **tape:make-stream** to prohibit you from writing records of any length if you have a tape host not so restricted. However, tape hosts can, in general, not read tapes at granularities not multiples of their own granularity, and this causes great problems moving tapes from site to site. This parameter can be used to adjust granularity when writing tape, to help remedy this problem in some cases.

minimum-record-length (minimum, minimum-record, minimum-length, minrec, minreclen)
This specifies a number *n*, such that all tapes written by the stream being created will be padded with zeros (the pad character is program-changeable) to be at least *n* bytes in length. The default is 64. As with granularity, the goal here is to ensure compatibility between systems. Some can write records shorter than others can read, and some can even write records shorter than they themselves can read. Again, this parameter is usually managed by tape-using programs, but might need to be adjusted by hand in some cases where writing records smaller than 64 bytes is required, but modifying the program is impossible or undesirable.

11.4.3 How Tape Specs Are Merged

When **tape:make-stream** prompts you with a default tape spec, you type a (perhaps empty) tape spec. It merges the two to produce the final tape spec.

The tape spec you type can specify one of three types of values for each parameter:

Explicit value	If you specify the value of some parameter, the value you specified, assuming it is legitimate, is used.
Merge default	By not saying anything at all about a parameter, the default from the default shown to you is used.
Null value	By supplying a *null value* for a parameter, the tape host uses its own default. This is how you cancel a displayed default value without supplying an explicit value. A side-effect of this is that there is not an explicit value for this parameter in the next default. For the **host** parameter, a null value is the same as not saying anything.

Here are the merging rules:

Each explicit or null value given in the typed tape spec replaces the corresponding element of the default to give the new tape spec. Components not specified are carried over from the default.

However, if the typed **host** is not the **host** in the default, and the typed tape spec does not specify a **device**, the **device** from the default will *not* be used, but a null value assumed.

See the section "Tape Spec Merging Examples".

11.4.4 Tape Spec Syntax

The general form of a tape spec is a comma-separated list of equal-sign pairs:

```
host=LM-6, reel=foo, density=800
```

The order of parameters is not significant. Each *parameter specification* is of the form

```
parameter = value
```

The specifications are separated by commas. Spaces and tabs around the equal sign or commas are not significant. Spaces and tabs in values are considered significant parts of the value.

Values that contain commas, equal signs, quotation marks (" "), or colons can be quoted with quotation marks to preserve their integrity during parsing. Reader string slashification is operative in such strings.

A null value can be specified by any of the following:

```
reel =
reel = ()
reel = ""
```

In the first example, the equal sign is followed by the end of the line.

Some special allowances are made for specification of **host** and **device** in a less-general syntax, more like file pathname syntax.

- A tape spec consisting of only a single token is interpreted as specifying the **host** parameter by that token. The following are valid, equivalent tape specs:

```
host = lm-6
LM-6
Lisp-Machine-6
"LM-6"
```

- If a token followed by a colon, or even an empty token followed by a colon appears where a parameter specification is expected, that token is interpreted as specifying the value of the **host** parameter. A colon thus used "replaces" the comma that would have been there (although the syntax "lm-6:" is permitted for "host=lm6"). The following are valid, equivalent tape specs:

```
lm-6: reel=foo, device = "mta0:"
reel= foo, host= lm-6, device = "mta0:"
reel=foo, lm-6: device = "mta0"
reel=foo, lm-6: device = mta0
```

- A null token before a colon specifies the merge default for the **host** parameter. The utility of this is explained below.

- A single token appearing where a parameter specification is expected, that is, followed by the end of the spec or a comma, is interpreted as specifying a **device**.

  Thus, the following are equivalent:

```
device=cart
:cart
```

  The following, too, are equivalent:

```
LM-6: cart, reel = foo
LM-6: reel = foo, device = cart
cart, reel = foo, host = LM-6
device = cart, reel = foo, host = LM-6
```

Note again that a tape spec consisting of a single token, followed by either colon or no delimiter at all, is (in the latter case) a special case, and designates the value of the **host** parameter, not the **device** parameter.

The syntax of tape specs is quite general and flexible. It is also, in most common cases, trivial. Far and away, the most common answer to the **tape:make-stream** prompt is simply RETURN.

In most other cases, the only relevant information that **tape:make-stream** is

lacking is the name of the host, which can simply be typed alone as a valid tape spec. It is for this reason that the prompt says the following:

```
Type tape host or spec
```

The next most common case is specifying a device. A single leading colon differentiates this from a host. The following example specifies both host and device:

```
LM-6: cart
```

11.4.5 Tape Spec Merging Examples

The following are examples of merging tape specs. It should be noted that tape specs are abstract quantities, not strings. The merging is performed on the abstract tape specs, not on strings.

```
old default:            Local: 0
spec:                   :cart
merged result:          Local: cart

old default:            Quabbin: 0
spec:                   den=3200
merged result:          Quabbin: 0, den=3200

old default:            Vixen: 0
spec:                   Stony
merged result:          Stony

old default:            Quabbin: 0, den=3200
spec:                   den=
merged result:          Quabbin: 0
```

12. Release 5.2: Operations and Site Management

12.1 Notes on Operations in Release 5.2

The *Software Installation Guide* describes procedures for installing a tape drive for a Lisp Machine host, using the **tv:edit-namespace-object** function.

12.1.1 Installing a Tape Drive

You install a tape drive for a Lisp Machine host with the **tv:edit-namespace-object** function, specifying `tape chaos rtape` as the last `Service: Set:` entry in the host object.

To enable a tape drive for a Lisp Machine host, you must add a **service** attribute in the host object. See the section "Host Objects: Network Database".

tv:edit-namespace-object edits objects in the namespace database. For a complete description of the namespace database: See the section "Network Database".

For example:

```
(tv:edit-namespace-object :host "Janis")
```

pops up namespace menu that displays all the attributes of the host Janis. You add the tape service by clicking on `Set:` of the last `Service: Set:` entry, then type:

```
tape chaos rtape RETURN
```

12.1.2 Installing an ASCII Print Server on VMS

The ASCII print server allows someone on a Lisp Machine to hardcopy a file on an ASCII printer, such as a line printer or a DECwriter, from some VMS system.

To bring up an ASCII printer named Stump-Pond on host Comet:

1. Create the printer object named Stump-Pond (using **tv:edit-namespace-object**) and give it the following attributes:

   ```
   Type ASCII
   Format ASCII
   Default-font FAKE9
   Header-font FAKE9
   host COMET
   ```

   See the function **tv:edit-namespace-object**.

2. Edit the host object Comet and add the attributes:

   ```
   Service HARDCOPY CHAOS LGP
   Printer STUMP-POND
   ```

3. Read the tape containing the VMS ASCII print server software into your VMS system using BACKUP. The tape was written with this command:

```
$ BACKUP CHAOS$DISK:[CHAOS...]MT:ASCLGP.BCK/DENSITY=1600
```

   The following commands retrieve the software:

```
$ MOUNT/FOREIGN MTAn
$ BACKUP MT:ASCLGP.BCK disk:[*...]*.*.*
```

4. Edit the file SYS$MANAGER:CHSTART.COM to include the print server in the INSTALL commands. Insert the line:

```
CHAOS$RFC:LGP/OPEN/SHARED/HEADER/PRIV=(TMPMBX,SYSPRV)
```

   after the command that runs the INSTALL utility.

5. You should also add the following commands to the SYS$MANAGER:SYSTARTUP.COM file:

```
$ ASSIGN/SYSTEM <spooled device> <printer name>
$ SET/DEVICE=<spooled device> <spooled device>
$ INIT/QUEUE <spooled device>
$ START/QUEUE <spooled device>
```

   where <spooled device> is the name of the device that will be a spooled printer and <printer name> is the name of the network printer object.

If you plan to use an ASCII printer regularly from a Lisp Machine, you should probably include the following in your lispm-init.l file:

```
(login-forms
  (setq press:*user-hardcopy-page-headings* nil))
```

12.1.3 Installing an ASCII Print Server on UNIX

The ASCII print server allows someone on a Lisp Machine to hardcopy a file on an ASCII printer, such as a line printer or a DECwriter, on some UNIX system.

To bring up an ASCII printer named Stump-Pond on host Cupid:

1. Create the printer object named Stump-Pond (using **tv:edit-namespace-object**) and give it the following attributes:

```
Type ASCII
Format ASCII
Default-font FAKE9
Header-font FAKE9
host CUPID
```

   See the function **tv:edit-namespace-object**.

2. Edit the host object Cupid and add the attributes:

```
Service HARDCOPY CHAOS LGP
Printer STUMP-POND
```

On the UNIX (BSD) 4.2 network tape, there is a server which answers the "LGP" contact name on the Chaosnet. This server simply gets some information for the banner page and then calls upon the "lpr" spooling system to do the work with the following line:

```
execl("/usr/ucb/lpr", "LGP", "-l", "-T", nargs[FILE_NAME], 0);
```

Refer to *4.2BSD Line Printer Spooler Manual* (in the /usr/doc documents directory) and manual pages for lpr and printcap for background information on how this all works.

As explained in the above documents, there are several ways for a system manager to configure the lpr system. The execl command above uses the -l argument to allow binary interpretation by the spooler. This is appropriate for an LGP printer. The -T gives the spooler the name to use on the banner page. Because there is no -P specification, lpr will use its default device. In the distribution system, this uses /dev/lp as the device and spools in the directory /usr/spool/lp. It is easy to modify the LPR configuration to use a different default device and/or a different spooling directory.

To get the LGP server to use a different printer (here, "myprinter"), first install a printcap entry for the printer. This includes specifying a spooling directory for the printer, so make that directory if it doesn't already exist. Then change the execl statement to the following:

```
execl("/usr/ucb/lpr", "LGP", "-l", "-P", "myprinter","-T",
        nargs[FILE_NAME], 0);
```

Be sure to link /dev/myprinter to the appropriate hardware device. For example, by linking /dev/console to /dev/myprinter, making the directory /usr/spool/myprinter, and installing the following entry in /etc/printcap one can use the console DECwriter as a remote printer with the above change to execl:

```
myprinter|local line printer:\
        :lp=/dev/myprinter:sd=/usr/spool/myprinter:lf=/usr/adm/lpd-errs:
```

If you plan to use an ASCII printer regularly from a Lisp Machine, you should probably include the following in your lispm-init.l file:

```
(login-forms
  (setq press:*user-hardcopy-page-headings* nil))
```

12.1.4 Sending *who-am-i* Requests Directly to Namespace Servers

As of Release 5.2, you can speed up the boot process for the machines at your site and cut down on some of the site-wide overhead of this process by telling machines at your site to send their *who-am-i* requests directly to the site's namespace servers.

The *who-am-i* request is part of the procedure Lisp Machines use at boot time to determine if the saved Lisp environment matches the current site. If the information returned in response to the *who-am-i* request differs from the information that was saved away in the booted world load, the machine adjusts itself to the new site. (This adjustment takes a few minutes.)

In release 5.0, the Lisp Machine always broadcast a *who-am-i* request over the local network at boot time. Any other machine receiving this request would then return a response containing the just-booted machine's identity.

Since the initial *who-am-i* request was broadcast to all machines on the local network, the booting of a single machine might have incurred the overhead of running the *who-am-i* server at many machines at the site, when only one server need do so. This would have used resources unnecessarily, and might have slowed down machines unexpectedly when other machines were booted.

You now have the option of specifying that requests for *who-am-i* service first be directed to the namespace servers of the (presumed unchanged) site; no broadcast message is sent unless these direct queries fail. To specify this, add the following service triple to namespace server host objects (using the namespace editor):

```
(WHO-AM-I CHAOS-SIMPLE WHO-AM-I)
```

12.1.5 3600 disk-save Improvement

The **disk-save** function gives two new options to the choices that it offers when you try to save without having enough room in the FEP file system. Previously, you could either run Dired on the directory or expunge the directory. Now you can also do the following:

L	List all files in the root directory of the FEP file system of the unit on which you tried to save.
S	Selectively delete load files. It goes into a loop asking about each ".load" file in the root directory of the FEP file system of the unit on which you tried to save (except the file you are currently running). It queries you about deletion of each one, expunging after each successful deletion until you have enough room.

This is probably the only option you need to use.

Also, if you try to run Dired, **disk-save** now explains why it is a bad idea and asks for confirmation.

13. Release 5.2: Notes and Clarifications

13.1 Clarifications and Corrections for Release 5.2

13.1.1 Clarification of use of dbg:with-erring-frame

dbg:with-erring-frame *(var object)* *Macro*

dbg:with-erring-frame sets up an environment with appropriate bindings for using the rest of the functions that examine the stack. It binds *var* with the frame pointer to the stack frame that signalled the error. *var* is always a pointer to an interesting stack frame. *object* is the condition object for the error, which was the first argument given to the **condition-bind** handler.

```
(defun my-handler (condition-object)
  (dbg:with-erring-frame (frame-ptr condition-object)
    body...))
```

Inside *body*, the variable **frame-ptr** is bound to the frame pointer of the frame that got the error.

Sometimes, you might want to use the special variable **dbg:*current-frame*** as *var* because some functions expect this special variable to be bound to the stack frame that signalled the error.

You would use this special variable if you are sending the **:bug-report-description** message to the condition object, which calls stack-examination routines that depend on the idea of a current frame, in addition to the other things that **dbg:with-erring-frame** sets up.
:bug-report-description is the message that generates the text that the Debugger c-M command puts in the mail composition window. See the message **:bug-report-description**.

13.1.2 Clarification of Use of :no-increment-patch Option to make-system

Please note that if you specify the **:no-increment-patch** option to **make-system**, it must follow, not precede, the **:compile** option in the **make-system** declaration.

13.1.3 Clarification of Use of :nowarn Option to load-patches

:nowarn does not imply **:noselective**. It suppresses only the warnings generated while a patch file is being loaded and the subsequent query to the user.

13.1.4 Effect of Character Set Translation on Direct Access File Streams

The Lisp Machine generic file access protocol was designed to provide access to ASCII-based file systems for Lisp Machines. Lisp Machines support 8-bit characters and have 256 characters in their character set. This results in difficulties when communicating with ASCII machines, which have 7-bit characters.

The file server, on machines not using the Lisp Machine character set, is required to perform character translations for any character (not binary) opening. Some Lisp Machine characters expand to more than one ASCII character. Thus, for character files, when we speak of a given position in a file or the length of a file, we must specify whether we are speaking in *Lisp Machine units* or *server units*.

This causes major problems in file position reckoning. It is useless for the Lisp Machine (or other user side) to carefully monitor file position, counting characters, during output, when character translation is in effect. This is because the operating system interface for "position to point *x* in a file", which the server must use, operates in server units, but the Lisp Machine (or other user end) has counted in Lisp Machine units. The user end cannot try to second-guess the translation-counting process without losing host independence.

Since direct access file streams are designed for organized file position management, they are particularly susceptible to this problem. As with other file streams, it is only a problem when character files are used.

You can avoid this problem by always using binary files. If you must use character files, consider doing one of the following:

- Know the expansions of the Lisp Machine, that is, characters such as Return that do not expand into single host characters. Note that this sacrifices host independence.

- Do not use these characters. See the section "Qfile Character Set Translation". This section explains which characters are expanded on the Lisp Machine.

14. New Audio System in Release 5.2

14.1 Introduction to the Digital Audio Facilities

The 3600 audio facilities consist of two 16-bit digital audio channels and supporting microcode. The facilities read arrays of samples from 3600 memory and feed them to the console at a rate of 50,000 samples per second. The channels are fed data in parallel; it is a "stereo" facility. When active, the audio microcode reads a pair of samples from 3600 main memory every 20 microseconds, supplying one 16-bit value to each channel. Although both channels appear digitally at the console, currently only one channel has an output device supplied. A digital-to-analog converter (DAC) in the console feeds a small speaker and 8-ohm headphone jack, as well as a low-level analog output compatible with standard "auxiliary" inputs to consumer audio equipment.

The 3600 audio microcode also supports a *polyphony feature*. The polyphony feature allows the use of the audio facility for the performance of music, obviating the need to generate samples for an entire performance. The polyphony feature is experimental in this release (5.2); it may be radically altered or removed in future releases.

Use the online tools described elsewhere to find out more about a given object in the audio facility: See the document *Program Development Tools and Techniques*. For practical examples of programming with flavors: See the document *Window System Program Examples*.

The digital audio facilities are demonstrated through several code examples. See the section "Examples of Using the Audio Facilities".

Note: the digital audio facility works only on 3600-family Lisp Machines running System 5.2 (or later), with the Revision 6 (or later) I/O board (IO-REV.6) installed.

14.2 Microcode Support for the Digital Audio Facilities

14.2.1 The Audio Microtask

This discussion covers the microcode interface, that is, the formats of commands and samples interpreted by the audio microcode. This is the lowest-level interface to this facility, and only the barest primitives are described here. The formats and commands given here may change in future version of the hardware, microcode, and software.

The audio microcode runs in its own *microtask* and thus operates parallel with the execution of Lisp. The audio microtask is either *active* or *stopped* at any time. Since the microtask scheduler works according to a priority queue, when the audio

task is active, it "wakes up" every 20 microseconds, and executes, preempting Lisp, until it either outputs an audio sample pair or stops. The generation of audio samples is not affected by the behavior of Lisp programs, including the masking of interrupts, and so forth.

When active, the audio microtask follows a *command list*, or program of its own, consisting of *audio commands*, stored by the programmer in main memory before the audio microcode is started. The command list is stored in sequential *physical* memory locations (although it may contain "jumps"). Each command occupies one or more 3600 words. The words are expected to be fixnums. The 32 data bits of each fixnum contain the data interpreted by the audio microtask. The commands include directives to control the flow of the command list as well as directives to output data to the console DAC. The audio microcode also maintains a *repeat counter* to facilitate generation of repetitive or continuous waveforms. See the section "Looping Through Audio Command Lists".

The audio microtask is started by the execution of the **%audio-start** instruction by Lisp; the evaluation of the form **(sys:%audio-start)** effects this. When this instruction is executed, the audio microtask will fetch the physical address of the beginning of the command list from the variable **sys:%audio-command-pointer**. Therefore, this variable must be set to the physical address of the beginning of the command list *prior* to the execution of the form **(sys:%audio-start)**. The audio microcode stops when it encounters an explicit command to this effect in its command list.

The audio microtask is coded for real-time performance; it does no validity checking, and issues no diagnostics. If you program the audio microtask via the techniques described in this document, it is your responsibility, as always, to create valid programs. In the case of the digital audio facilities, however, the result of an invalid program could be a machine halt or destruction of the integrity of virtual memory, or both. If certain bit patterns are interpreted as audio commands, they can modify storage locations. Save your editor buffers often when debugging code for the audio microcode.

14.2.2 Sample Format

Each sample pair is expected to be a fixnum. The 32 data bits of each fixnum include two samples, one for each channel. The sample pair is read by the audio microtask in one operation, and the samples are sent to each channel in parallel. Each sample is a 16-bit unsigned integer, one in the lower (bits 0-15) half word (channel 0), and one in the upper (bits 16-31) half word (channel 1).

A sample value of 0 produces the lowest analog output voltage, and a sample value of all 1s (65535, octal 177777) produces the highest. A voltage of zero is represented by the midpoint value, 32768 (octal 100000).

Channel 0 is currently supplied with analog output hardware in the console; Channel 1 is not. The digital-to-analog converter in the console is only of 12-bit precision, and thus, it ignores the low 4 bits of Channel 0 samples.

14.2.3 Audio Command Format

Audio commands occupy one or more words of sequential physical memory. The command words are expected to be fixnums. The fixnum data (32 bits) for each command is described in this section.

The format of the first word of each command is as follows, described by byte specifiers in the **sys** package:

%%audio-command-op
A 4-bit *opcode* selecting the action to be performed by the audio microcode. Each of the currently assigned opcodes is described elsewhere. See the section "Audio Command Opcodes". See the section "Polyphony Command Opcodes".

%%audio-command-arg
A 28-bit quantity, whose meaning differs for each opcode. When the contents of this field, known as the *operand*, is described as an *address*, it must be a physical address. The usual way to obtain such a physical address is via the function **si:%vma-to-pma** (which does a virtual-to-physical translation). This function is given a fixnum virtual memory address. The usual way to derive such addresses, which are usually references to array element cells, is via the **%pointer** and **aloc** functions. A physical address computed from a virtual address in this way cannot be validly used unless the relevant virtual address has been wired in advance. See the section "Notes on Wired Structures".

14.2.3.1 Audio Command Opcodes

These are the valid opcodes of audio commands, with the exception of those commands associated with the polyphony feature. See the section "The Polyphony Feature". The descriptions tell what action is performed by the audio microtask when a command having this opcode is encountered by the microtask. The opcodes are listed under the the name of the system constant (also in the **sys** package) that gives the opcode value.

%audio-command-stop
Causes the audio microtask to halt execution. No more commands will be fetched, or samples sent to the console, until the next execution of the **sys:%audio-start** instruction. The operand is ignored.

%audio-command-jump
Causes the audio microtask to fetch its next instruction not from the next sequential location, but from the physical address that is the value of the operand. Sequential execution of commands continues at that physical address.

%audio-command-load-repeat

Loads the repeat register with the value of the operand. The operand is an unsigned 28-bit number to be loaded into the repeat register, not an address. See the description of the **%audio-command-loop** opcode for the use of this register.

%audio-command-loop

Decrements the repeat register by 1. If the result is greater than zero, the operand is interpreted as a jump address, and execution of commands continues at that address, as with **%audio-command-jump**. Otherwise, if the result is less than or equal to zero, command execution continues with the next sequential command.

%audio-command-samples

Designates a vector of sample pairs to be sent to the console. The operand is the physical address of the first sample pair; the remaining samples are fetched from successive words of physical memory. The word in the command stream after the **%audio-command-samples** command contains a fixnum that is the count of the number of sample pairs to be fetched and sent to the console before the execution of **%audio-command-samples** terminates, and the microtask proceeds to the next sequential command. The **%audio-command-samples** command is thus a two-word command.

%audio-command-zero

A synchronization primitive. The operand is the *physical* address of a cell, usually an array element. The audio microcode stores a fixnum zero in that cell as the result of executing the command having the opcode **%audio-command-zero**. The software may use this facility to test if the audio microtask has passed a given point in its command list. This enables the software to ascertain when it is safe to unwire or reuse data structures containing audio commands and/or samples. It is important to remember that the audio task, when active, locks out Lisp execution until it either sends a sample or goes idle. For example, if **%audio-command-zero** is immediately followed by **%audio-command-stop**, the observation of the zeroed cell by Lisp software implies that the microtask has already read, interpreted, and executed the **%audio-command-stop**.

%audio-command-immediate

Designates a vector of sample pairs to be sent to the console. Unlike **%audio-command-samples**, the sample pairs appear in the command list, in consecutive physical memory locations immediately following the the **%audio-command-immediate** command word. The operand of **%audio-command-immediate** is a number, which is the count of sample pairs. That number of

sample pairs is fetched from the command list and sent to the console, one every 20 microseconds (at a 50 KHz sampling rate). Execution of the command list proceeds with the next command after the vector of sample pairs, after all samples have been sent to the console.

It is critically important that the operand is equal to the number of samples provided, lest commands be interpreted as samples or vice versa.

14.2.4 The Polyphony Feature

Note: The polyphony feature is experimental in Release 5.2. It may be radically altered in function and/or interface in future releases, or may be removed entirely.

The polyphony feature of the 3600 audio microcode provides a way to generate polyphonic music in real time. There is no need to precompute the samples and store them before playback from disk. The polyphony feature can produce six *voices*, where a voice is a rhythmically independent sequence of musical notes. Each voice can be assigned a predefined, programmer-specified waveform, which determines the spectrum and the amplitude of the notes that appear in that voice, regardless of their pitch (frequency). The waveform specification determines the shape and amplitude of *one cycle* only of the waveform. This waveform is repeated at different frequencies to produce musical tones.

The polyphony feature is not intended as a general-purpose music synthesis facility. For example, no control over the amplitude envelopes (attack, decay, and so forth) of the sounds produced is provided. The polyphony feature is intended for use in music system prototyping, that is, composition research, music editing programs, and so forth. Nevertheless, the square-envelope notes it produces are not very different from those produced by some electronic organs. When properly programmed and amplified, the digital audio facility is capable of reasonably authentic performance of much of the organ literature.

14.2.4.1 Operation of Polyphony

The basic function of the polyphony feature is to generate, in parallel, six separate wave signals, usually of different frequencies, and sum them, at the sampling times of the audio facility. The audio microcode accomplishes this by maintaining, for each voice, a *wavetable*, a *wavetable cursor*, and an *increment*.

The wavetable for each voice consists of 256 fixnums stored in consecutive locations in physical memory, defining the *waveform* for notes in that voice. (Note: the size of the wavetables may change in a future release.) The fixnums constitute *wave values*, which digitally describe the waveform of the voice.

The detailed interpretation of the wave values is as follows: Each fixnum wavetable element is interpreted as the algebraic sum of the wave values for the channels 0 and 1, channel 1 having been shifted 16 bits left. In detail, the value for channel 0

is a 32-bit signed (31 bits and sign, 2's complement) value between -2**15 and 2**15-1, inclusive. The value for channel 1, also in the range -2**15 to 2**15-1, is shifted left 16 bits and added algebraically to the value for channel 0. The resulting number (which will always be a fixnum) is the value of the wavetable entry. Note that this is not the same format as that of audio samples used by other parts of the audio facility.

When polyphony is running (that is, when the audio microtask is interpreting the command **%audio-command-polyphony**), one value from each of the six tables is extracted, and these values are added algebraically. The resulting value is then offset by 2**15 *in each halfword*, and the resulting two halfwords are sent as audio samples to the two audio channels.

You must ensure that the sum of the values from each table will never exceed the range -2**15 to 2**15-1 for either channel. The audio microcode will clip or overflow into the other channel if this range is exceeded.

Associated with each voice is also a counter/pointer called the *wavetable cursor*. This quantity is a 32-bit unsigned number. The high-order eight bits of the wavetable cursor for each voice constitute an index, which selects the entry of its wavetable to be summed into the audio sample to be produced. The low bits are used to measure the passage of time, overflowing into the high bits 256 times per cycle of that voice.

Also associated with each voice is a quantity called an *increment*. The increment is a 32-bit fixnum. It controls the frequency, or pitch, of the note in each voice, by controlling the rate of incrementing of the wavetable cursor for that voice. When the command **%audio-command-polyphony** is being interpreted by the audio microtask, the increment for each voice is added to the wavetable cursor for that voice, and the resulting quantity is made the new wavetable cursor. (This addition is performed *after* the wavetable sample is extracted). Thus, when this repeated addition produces enough change in the value of the wavetable cursor such that the top eight bits are affected, a different wavetable entry for that voice will be fetched at the next sampling time. Note that continued incrementing in this manner "wraps around". In this way, the wavetable cursor is way reset to the beginning of the wavetable, after the last entry in the wavetable has been used.

The following function (available in the **audio** package) computes the increment for a voice from the frequency:

```
(defun frequency-polyphonic-increment (frequency)
     (round (* frequency (float 1_32.)) audio:*sample-rate*))
```

You simultaneously establish the increment and wavetable location for a voice by the audio command **%audio-command-load-voice**. You instruct the polyphony facility to output samples by the audio command **%audio-command-polyphony**. This command uses all of the wavetables and increments previously established by **%audio-command-load-voice**, and outputs as many samples as requested, one every 20 microseconds, generated by summing entries from the six wavetables, incrementing the six wavetable cursors by the six associated increments as each sample is generated.

Note: changing the wavetable and/or increment for a voice does not affect any other voice in any way. Since the audio microtask is awakened by an external timer, and runs until it either outputs a sample pair or stops, no discontinuity in notes played by other voices is observed when **%audio-command-load-voice** is interpreted to change the note in one voice.

Polyphony Command Opcodes

%audio-command-load-voice

Establishes a wavetable and increment for one voice of the polyphony feature The operand is the physical address of the base of the wavetable for the voice. The word in the command stream after **%audio-command-load-voice** is, in its 32 data bits, the increment for the voice. The low three (that is, the least significant) bits of this increment are the binary number of the voice whose wavetable and increment are to be established. **%audio-command-load-voice** is effectively a two-word command.

When polyphony is being performed, the audio microcode will, for each voice, use the wavetable and increment established for that voice. There is no way to assert that a voice does not exist, or has no wavetable, or no increment. A valid wavetable and increment must be established for each of the polyphonic voices before **%audio-command-polyphony** is executed by the audio microcode, regardless of whether that voice is needed for the performance of the particular composition.

%audio-command-load-voice does not affect the value of the wavetable cursor for the voice involved.

%audio-command-polyphony

The operand is an unsigned 28-bit number. The audio microcode sends out that many samples, one each 20 microseconds, generated from the currently established wavetables of the polyphony feature. The wavetable cursors of each voice used by the polyphony feature are incremented by the increment established for that voice as each sample is sent out. The values of the increments and the wavetable cursors are not reset in any way by either the start of **%audio-command-polyphony**, or its completion.

14.2.5 Notes on Wired Structures

The audio microtask fetches commands from sequential locations of physical memory. Branch addresses in the command list are physical addresses. Audio sample data pointed to by the command list are also described by physical address. Wavetables used by the polyphony feature are also described and accessed by physical address.

The audio microtask does not perform virtual address translation. Thus, the

command list and sample data must be stored in data structures *wired*, or locked, in main memory. That is, they must be prevented from being paged out or moved by the Lisp Machine operating system. As a digital audio programmer, you must therefore be aware of page boundaries.

Audio command lists and sample vectors must be stored in wired pages consecutive in main memory, or scattered throughout main memory. If commands are stored in pages scattered throughout main memory, jumps must be programmed at the end of each page, to send the audio microcode on to the next page. If sample vectors are stored in pages scattered throughout main memory, you must use a separate **%audio-command-samples** command to describe the samples on each page. Wavetables for the polyphony feature must be in consecutive locations in main memory.

It is conventional to use Lisp arrays as the data structure containing audio commands, samples, and wavetables. Any type of array is usable for this purpose. **art-q** arrays allow one audio command or sample pair per element, and are also the only type of array whose elements may validly be addressed by the **aloc** function.

14.2.5.1 Lisp Primitives for Wiring Memory

The relevant Lisp primitives to wire data structures for the digital audio facility are **si:wire-structure**, **si:wire-words**, and **si:wire-consecutive-words**. **si:wire-words** wires any extent of virtual memory into physical memory, although the page frames into which successive pages are wired may not be contiguous. **si:wire-consecutive-words** also wires any extent of virtual memory into physical memory, but successive pages are guaranteed to be stored in successive page frames in physical memory. **si:wire-structure** wires an entire structure (a convenience device to avoid having to calculate the location and extent of the virtual memory occupied by a structure) in the manner of **si:wire-words**.

Since commands must be stored in consecutive locations in physical memory, **si:wire-consecutive-words** suggests itself as the natural primitive for this application. However, success of this primitive depends on the availability of consecutive page frames of main memory not already containing wired pages, and it is thus less likely to succeed as more pages are wired. Use of **si:wire-structure** and **si:wire-words** for audio data does not encounter this problem, but requires explicit programmer handling of page boundaries, as outlined previously.

%find-structure-header and **%structure-total-size** are used to find the virtual memory location and extent of whole arrays or other structures to be wired. **si:page-array-calculate-bounds** can be used to calculate the virtual memory location and extent of portions of array that are to be be wired, when **si:wire-words** or **si:wire-consecutive-words** is used. **%pointer-difference** can also be used to determine the length of the extent, in words, between two addresses obtained via these primitives or the **aloc** function.

Structures, or portions thereof, wired by any of these primitives, should be unwired

by **si:unwire-structure** or **si:unwire-words** (as appropriate) only after it has been ensured (via the techniques described) that the audio microtask is not fetching commands or samples from these structures.

14.3 Lisp Primitives for the Digital Audio Facilities

14.3.1 Functions, Variables, and Macros for Digital Audio

This section describes the functions, variables, and macros available to the Lisp Machine programmer to aid in programming the 3600 Digital Audio Facilities. All of these objects are tools for programming the audio microtask. Therefore, this section assumes that you already understand the microcode capabilities. See the section "Microcode Support for the Digital Audio Facilities".

All of the digital audio functions, variables, and macros appear in the **audio** package. Several comprehensive examples of their use are provided in the file sys:examples;audio-examples.lisp. See the section "Examples of Using the Audio Facilities".

These Lisp tools assume the existence of an audio *command array*, in which audio microtask commands are placed, and out of which they are executed by the audio microtask. A macro (**audio:with-audio**) manages the wiring and unwiring of command arrays within the scope of a program.

A default audio command array is provided as part of these audio support primitives. All of these primitives, however, allow the specification of any suitable user-provided array as a command array. Such an array must be a nonindirect, single-dimensional **art-q** array, with a fill pointer, allocated in a static area (such as **audio:audio-area**).

Command arrays, as all arrays, are finite in extent. Carefully planned synchronization techniques must be utilized to allow uninterrupted sound to be produced from a single command array that is being serially reused for sequences of audio commands. See the section "Examples of Using the Audio Facilities".

14.3.2 Digital Audio Parameters

These are the critical constants of the audio facility. In programs these constants should be used instead of the numbers that are their current values in order to accommodate future modification of the audio facility.

audio:*sample-rate* *Variable*

The number of times per second that an audio sample is output when the audio microtask is active. This is a single-precision floating-point number. Its current value is **50e3**, as determined by the hardware.

audio:*number-of-polyphonic-voices* *Variable*

The number of polyphonic voices defined by the (experimental) polyphony feature. See the section "The Polyphony Feature". This is a fixnum, and its current value is 6.

14.3.3 Testing for the Existence of Audio

audio:audio-exists *Variable*

This variable has a value of other than **nil** if and only if the machine on which it is evaluated has an operational audio facility.

14.3.4 The Audio Wrapping Form

audio:with-audio &optional *command-array* &body *body* *Macro*

Encases code that generates audio commands. It prepares a command array for use by wiring it in an appropriate fashion and unwires it when the body of the form is exited. When exited, it also unconditionally halts the audio microtask, silencing the audio output.

If *command-array* is given as **nil**, the default command array is used.

When the scope of **audio:with-audio** is entered, it also zeroes the fill pointer of the supplied command array. The various interface functions described later utilize the fill pointer of the command array to keep track of the current position in the audio command list being built.
audio:with-audio also globally binds scheduler parameters to allow the process generating audio commands to gain control when necessary and more rapidly than usual.

14.3.5 Building Audio Command Lists

The functions listed in this section prepare arguments for, build, and store audio commands in a command array. They assume that the fill pointer of the array describes the next available location in the array, and they update the fill pointer as needed. The array must be wired, as some of these functions will compute and store physical addresses of locations in the command array. Calling these functions does not produce sound. Sound is produced when the audio facility is directed (via **audio:audio-start**) to a command list produced by calling these functions.

The fill pointer of the array defines a logical pointer called the *audio index*. The function **audio:audio-index** (which defines a location accessible with **setf**) is used to access this index (for example, for use as an argument to a later function call).

The current implementation uses command arrays that are wired into successive, contiguous page frames of physical memory. (Note: This may change in the future.) The exclusive use of these primitives hides this implementation detail. In order to accommodate future changes in this strategy, do not perform calculations on audio

indices. Instead, request them whenever needed via **audio:audio-index**, and use them only as arguments to the primitives provided.

Use of the macro **audio:with-audio** is the recommended way to establish the proper context in which these functions may be validly used. Each of them takes an optional argument, which specifies the command array in question. This argument always defaults to the facility's default command array.

audio:audio-index &optional *command-array* *Function*
This function returns the audio index for the next command to be stored in the command array in question. The form **(audio:audio-index)** is suitable for use as the first operand of a **setf** form.

audio:audio-room &optional *command-array* *Function*
This function returns the amount of available (unallocated) space, in single words, in the current command array.

audio:audio-limit &optional *command-array* *Function*
Returns a number one greater than the audio index of the last usable location in the command array.

audio:audio-push-audio-stop &optional *command-array* *Function*
Pushes a **%audio-command-stop** onto the command list in the command array. ("Push", as used in the names of these interfaces, means "add to the end of, at the current audio index, and increment the audio index appropriately.").

audio:push-audio-jump *target-index* &optional *command-array* *Function*
Pushes a **%audio-command-jump** onto the command list in the command array. The argument *target-index* is expected to be an audio index into the same command array, obtained previously from **audio:audio-index**.

audio:push-audio-zero-flag *flag-index* &optional *command-array* *Function*
Pushes a **%audio-command-zero** onto the command list in the command array. The argument *flag-index* is expected to be an audio index, into the same command array, of a "flag". Such flags are allocated, and their indices returned, by **audio:reserve-audio-flags**.

audio:push-audio-load-voice *voice-number wave-array* *Function*
wave-array-start-time
wave-array-index-increment &optional
command-array
Pushes a **%audio-command-load-voice** onto the command list in the command array. *voice-number* is a number, zero or greater, below the value of **audio:*number-of-polyphonic-voices***, that specifies which polyphonic voice is to have its wavetable and increment loaded by the command to be built and stored. *wave-array-index-increment* is the value of that increment,

which may be computed from the frequency of the tone desired by use of the function **audio:frequency-polyphonic-increment**. The wavetable for the voice is expected to be in the **art-q** array *wave-array*. The argument *wave-array-start-index* is the index into that array where the 256-word, wired, contiguous in physical memory, wavetable begins.

audio:push-audio-polyphony *number-of-samples* &optional *command-array* *Function*

Pushes a **%audio-command-polyphony** onto the command list in the command array. The argument *number-of-samples* specifies the sample count for the command to be built and pushed.

audio:modify-audio-command-arg *new-arg arg-type command-index* &optional *command-array* *Function*

Modifies an audio command that has already been pushed in the command array specified. This function must be used with extreme care: it can easily create invalid audio programs, which can destroy machine integrity. It modifies the 28-bit argument in the first word of the command whose index into the command array (*command-index*) is given. To be sure that this command may be validly used, read the description of the format of the individual audio command. See the section "Microcode Support for the Digital Audio Facilities". *new-arg* is the new value of the command whose index is given. The argument *arg-type* describes how it is converted to a 28-bit value for insertion in the existing command:

:immediate
No processing is done. *new-arg* is expected to be a non-negative fixnum, which must be a count.

:index
The argument is an audio index into the command array specified. The location of the corresponding array cell is computed, verified to be wired, and the physical address of that location stored in the command.

:location
The argument is a locative into a wired array of audio commands. The fact that this location is wired is verified, and the corresponding physical address stored in the command.

14.3.6 Storing Samples

The functions and macros described in this section place audio sample pairs into the command program. These commands can be either immediate (**%audio-command-immediate**) or stored elsewhere (**%audio-command-samples**).

audio:push-array-of-audio-samples *array* &optional *from to command-array immediate-p* *Function*

Pushes appropriate commands onto the command list in the command array specified, to output all the sample pairs in the array *array* between indices *from* and (up to but not including) *to*. *from* defaults to 0, and *to* to the active length of *array*. *array* must be an **art-q** array containing precomputed sample pairs.

If *immediate-p* is non-**nil**, the data will be copied into the command array, and output by means of **%audio-command-immediate**.

If *immediate-p* is **nil**, *array* is assumed (and checked) to be wired, and as many **%audio-command-samples** commands as necessary to describe the data to be output are built and pushed. *array* need not be wired in contiguous page frames.

audio:computing-immediate-audio-samples (*count* &optional *command-array*) &body *body* *Macro*

Facilitates the storing of immediate audio sample pairs. The code it wraps, *body*, is responsible for generating immediate audio sample pairs: it does so by calling the macro **audio:push-immediate-audio-sample**, within the scope of the use of **audio:computing-immediate-audio-samples**. Each use of **audio:push-immediate-audio-sample** stores one sample. The macro **audio:computing-immediate-audio-samples** arranges for an appropriate **%audio-command-immediate** to be constructed to describe all the samples stored. If the argument *count* is non-**nil** (at run time), it is expected to be a fixnum, which is the number of values to be stored. **audio:computing-immediate-audio-samples** will check, when it is exited, that that is the actual number of values stored, and signal an error if not. If *count* is **nil**, no checking is done, and **audio:computing-immediate-audio-samples** assumes that the number of samples that have been pushed is the correct number, and modifies the commands it builds appropriately.

audio:push-immediate-audio-sample *sample* *Macro*

Stores one audio sample pair, which is the value of its argument. This macro can be used validly within the scope of **audio:computing-immediate-audio-samples**.

14.3.7 Looping Through Audio Command Lists

These two macros facilitate the use of **%audio-command-loop** to create loops in audio command lists. Keep in mind that the audio microcode does not support nested loops.

audio:audio-loop (*repeat-count-or-nil* &optional *command-array*) *Macro*
&body *body*

This macro builds a loop (with **%audio-command-loop** and **%audio-command-load-repeat**) in the audio command list in the command array specified. The code, *body*, which is wrapped by this macro pushes commands for the body of the loop. The macro generates the audio command to loop back at the time its scope is exited. The argument *repeat-count-or-nil*, when non-**nil**, specifies how many times the loop is to be executed by the audio microtask. That is the number which will be loaded into the repeat register. If *repeat-count-or-nil* is **nil** (at run time), the wrapped code must compute the number of loop repetitions, and invoke the macro **audio:set-audio-repeat-count**, whose argument is that number, some time before the scope of **audio:audio-loop** is exited. A diagnostic is issued (at run time) if the macro's scope is exited without the repeat count having been specified by one of these two means.

audio:set-audio-repeat-count *count* *Macro*

Sets the value *count* as the repeat count for an audio command list loop that is currently being built by **audio:audio-loop**. This macro may only be validly used within the scope of **audio:audio-loop**.

14.3.8 Synchronization Flags

These functions allocate, in the command array specified, locations to be used as synchronization flags (for **%audio-command-zero**), and allow the flags to be waited for and reset. The "reset", or "normal", state of these flags, is non-zero. The audio microcode "sets" them, by setting them to zero, when a **%audio-command-zero** is executed. By means of these flags, the real-time progress of the audio microtask can be monitored.

audio:reserve-audio-flags *count* &optional *command-array* *Function*

Allocates, in the command list currently being built in the command array specified, *count* locations to be used as audio flags. The flags are reset. A **%audio-command-jump** is inserted in the command list being constructed, so that the audio microtask will jump around the locations being used as flags. The return value of this function is the index, in the command array given, of the first of the flags allocated. You can assume, if more than one flag was allocated by a call to **audio:reserve-audio-flags**, that the indices of flags other than the first are the sequential integers above the value returned.

audio:wait-for-audio-flag *flag-index* &optional *who-state audio* *Function*
reset-flag t command-array

Waits for the audio flag specified by *flag-index*, in the command array specified, to be set. Normally, it is the audio microtask that sets these flags, by means of **%audio-command-zero**. *whostate* is the state to be displayed

in the status line. If *reset-flag* is given as **nil** (this is *not* the default), the flag is not reset. The resetting, when requested, is performed *after* the flag has been observed to be set. The indices given to **audio:wait-for-audio-flag** should be those obtained from **audio:reserve-audio-flags**.

14.3.9 Starting and Stopping the Audio Microtask

These functions are used to start and stop the audio microtask.

audio:audio-start *index* &optional *command-array* *Function*
Starts the audio microtask, via the instruction **sys:%audio-start**, at the audio command specified by *index* in the command array specified. The array must be wired, and contain a valid, wired, audio command list.

audio:audio-stop &optional *command-array* *Function*
Stops the audio microtask immediately, causing immediate silence. **audio:audio-stop** accomplishes this by storing a **%audio-command-stop** instruction at location zero (0) of the command array given, and issuing **audio:audio-start** at that command. Thus, **audio:audio-stop** is destructive to the command array, and requires that it be wired.

14.3.10 Conversions Between Sample Formats

The following functions encode and decode sample pairs. They are provided to hide the internal representation of sample pairs. Some of these "functions" are actually implemented as macros to help make code that prepares audio samples as fast as possible.

These functions convert between three formats of samples, *float*, *fixnum*, and *sample*. Float and fixnum formats describe channel values. Sample format is the actual format of sample pairs stored in command arrays and sample arrays.

Fixnum format consists of integers in the range $-1^{**}15 \leq x < 1^{**}15$. Float format consists of floating numbers and float channels are in the range $-1.0 \leq x < 1.0$. You must ensure that a float format value is never +1.0.

audio:float-channel-fix *float* *Function*
Converts a float format value to fixnum format.

audio:fix-channel-float *fix* *Function*
Converts a fixnum format value to float format.

audio:fix-sample *right* &optional *left right* *Function*
Takes one or two fixnum format values for the two channels and returns a sample pair in sample format containing those two values.

audio:float-sample *right* &optional *left right* *Function*

Takes one or two float format values for the two channels and returns a sample pair in sample format containing those two values.

audio:sample-channels *sample* *Function*

Takes a sample pair in sample format and returns two values, the right and left channel values of that sample, respectively, in fixnum format.

audio:sample-add-fix *sample right-increment* &optional *left-increment right-increment* *Function*

Takes a sample pair and one or two increments, which are expected to be in fixnum format. The two channels of the sample pair are incremented by the two increments, and a new sample pair so constructed is returned. If the right channel goes out of range, it will overflow into the left channel instead of clipping.

audio:sample-add-float *sample right-increment* &optional *left-increment right-increment* *Function*

Takes a sample pair and one or two increments, which are expected to be in float format. The two channels of the sample pair are incremented by the two increments, and a new sample pair so constructed is returned. If the right channel goes out of range, it will overflow into the left channel instead of clipping.

audio:sample-add-sample *sample1 sample2* *Function*

Takes two sample pairs, in sample format, and produces a new sample pair by adding them. The operation performed is the addition of the fixnum format values corresponding to the channel values in the sample pairs. In other words, it is as if **audio:sample-add-sample** extracted the sample values from the sample pairs using **audio:sample-channels**, then added the channel values and reconstructed a sample pair using **audio:fix-sample**. The actual operation of **audio:sample-add-sample** is considerably more efficient.

14.3.11 Conversions for the Polyphony Feature

These functions convert between fixnum and float format channel values and the values stored in wavetables used by the polyphony feature. See the section "The Polyphony Feature".

audio:fix-polyphonic-wave-table-entry *right* &optional *left right* *Function*

Takes one or two channel values in fixnum format and returns a fixnum representing those two values, in the format used in wavetables. This is not the same as sample format.

audio:float-polyphonic-wave-table-entry *right* &optional *left right* *Function*
Takes one or two channel values in float format and returns a fixnum representing those two values, in the format used in wavetables. This is not the same as sample format.

audio:polyphonic-wave-table-entry-channels *entry* *Function*
Takes as an argument an *entry* from a polyphonic wavetable, and returns two values in fixnum format, the right and left channel values encoded therein, respectively.

14.3.12 Computing Polyphonic Increments

This function computes the appropriate wavetable increment to specify the frequencies in polyphonic textures.

audio:frequency-polyphonic-increment *frequency* *Function*
Computes an increment value suitable for use with **%audio-command-load-voice**. The increment produced corresponds to a frequency of *frequency*. That is, the increment returned causes the wavetable for the voice with which it is used to be scanned *frequency* times per second.

14.4 Examples of Using the Audio Facilities

This chapter presents seven program examples that use the digital audio facilities, in both real-time and non-real-time synthesis applications.

14.4.1 Sine Wave Example

This example generates a sine wave at a specified frequency.

```
(defun sine-wave (frequency)
  (audio:with-audio ()                  ;Set up the audio environment
    (let* ((start (audio:audio-index))  ;Get the current (starting) index
           (samples-per-cycle (sys:round audio:*sample-rate* frequency))
           ;; Spread out several cycles to get a more accurate
           ;; frequency.  Extra factor of 2 makes sure there is room.
           (number-of-cycles (max 1 (// (audio:audio-limit) samples-per-cycle 2)))
           ;; Actual number of samples we are going to produce
           (number-of-samples (* samples-per-cycle number-of-cycles)))
      ;; Make sure we have room to play this frequency
      (when (> (+ number-of-samples 2) (audio:audio-limit))
        (ferror "Frequency too low"))
      ;; This form allows us to compute number-of-samples inline
      ;; (as opposed to computing them in a separate array). If we
      ;; didn't know how many samples we were going to produce we could
      ;; supply NIL for number-of-samples and the form will keep track
      ;; and adjust the command array when the form is exited.  Since we
      ;; do supply the number of samples, the form will check to make
      ;; sure we supply exactly that many.  This helps us to avoid writing
      ;; incorrect audio programs.
      (audio:computing-immediate-audio-samples (number-of-samples)
        (loop for sample-number below number-of-samples
              as phase =
                 ;; This is the phase (angle) that is passed to sin
                 ;; to get the sine wave.  (This will cons double-floats in
                 ;; systems where si:pi is a double-float.)
                 (// (* 2 si:pi sample-number number-of-cycles)
                     number-of-samples)
              as sample =
                 ;; Take the sin of the phase.  Also multiply it by
                 ;; something less than 1 so we never get a value of 1.0
                 ;; (a restriction, see documentation).  Take the
                 ;; resulting floating point number in the range [-1.0,
                 ;; +1.0) and create a 'sample.'
                 (audio:float-sample (* (sin phase) 0.9))
```

```
do ;; Now actually push the sample into the command array.
      (audio:push-immediate-audio-sample sample)))
      ;; All of the samples are computed and an appropriate command has
      ;; been generated to output them.  Now we cause a jump back to the
      ;; beginning to keep the sound going.
      (audio:push-audio-jump start)
      ;; The program is complete, we can now start the audio facility.
      (audio:audio-start start)
      ;; When you've heard enough, just type anything.  with-audio
      ;; supplies code to turn off the audio facility when exited and do
      ;; other bookkeeping.
      (tyi))))
```

14.4.2 Sawtooth Wave Example

This is roughly the same as sine wave, but instead produces a sawtooth and only generates one cycle for it.

```
(defun saw-wave (frequency)
  (audio:with-audio ()
    (let* ((start (audio:audio-index))
           (samples-per-cycle (sys:round audio:*sample-rate* frequency)))
      (audio:computing-immediate-audio-samples (samples-per-cycle)
        (loop for sample-number below samples-per-cycle
              as value =
                 ;; create a sawtooth value in the range [-1.0,1.0).
                 ;; Note this can never be exactly 1.0 since
                 ;; sample-number never quite gets as large as
                 ;; samples-per-cycle.
                 (- (// (* 2.0 sample-number) samples-per-cycle) 1.0)
              do (audio:push-immediate-audio-sample (audio:float-sample value)))
        (audio:push-audio-jump start)
        (audio:audio-start start)
        (tyi)))))
```

14.4.3 Square Wave Example

This example demonstrates yet another type of waveform: a square wave. The **audio-loop** form is also exemplified.

```
(defun square-wave (frequency)
  (audio:with-audio ()
    (let* ((start (audio:audio-index))
           (samples-per-cycle (sys:round audio:*sample-rate* frequency))
           ;; Compute the number of samples for the high value and
           ;; low value.  Divide them as evenly as possible.
           (samples-first-half (// samples-per-cycle 2))
           (samples-second-half (- samples-per-cycle samples-first-half)))
      ;; Create a loop that will repeat samples-first-half times.  If we
      ;; weren't sure how many times we want to repeat, we could specify
      ;; NIL and then use set-audio-repeat-count to set the count.
        (audio:audio-loop  (samples-first-half)
        ;; Compute 1 value (the high value) for output.
        (audio:computing-immediate-audio-samples (1)
          (audio:push-immediate-audio-sample (audio:float-sample 0.9))))
      ;; Do the same for the second half.
      (audio:audio-loop (samples-second-half)
        (audio:computing-immediate-audio-samples (1)
          (audio:push-immediate-audio-sample (audio:float-sample -0.9))))
      ;; Jump back to the beginning so we get more than one cycle.
      (audio:push-audio-jump start)
      (audio:audio-start start)
      (tyi))))
```

14.4.4 Beep Example

This is basically a modified square-wave.

```
(defun %beep-ignoring-most-issues (frequency duration)
  (audio:with-audio ()
    (let* ((start (audio:audio-index))
           (samples-per-cycle (sys:round audio:*sample-rate* frequency))
           (samples-first-half (// samples-per-cycle 2))
           (samples-second-half (- samples-per-cycle samples-first-half)))
      ;; Can't nest loops, so we have to do the outer loop with a jump
      ;; and bash the location when time has elapsed.
      (audio:audio-loop (samples-first-half)
        (audio:computing-immediate-audio-samples (1)
          (audio:push-immediate-audio-sample (audio:float-sample 0.9))))
      (audio:audio-loop (samples-second-half)
        (audio:computing-immediate-audio-samples (1)
          (audio:push-immediate-audio-sample (audio:float-sample -0.9))))
```

```
;; This is the tricky part.  We need to put a jump to the
     ;; beginning, but we need to know where it is so we can cause it
     ;; to fall through.  We also need a flag so we know when the audio
     ;; has stopped so we can exit.  If we simply exited without
     ;; waiting, the with-audio form could turn off the sound prematurely.
     (let* (;; get the index that we will eventually bash and put in a
            ;; jump back to the start.
            (jump-index (prog1 (audio:audio-index) (audio:push-audio-jump start)))
            ;; reserve (and reset) an audio flag.
            (flag-index (audio:reserve-audio-flags 1))
            ;; reserve-audio-flags puts in a jump command around the
            ;; flags it reserves, so we could have gotten the
            ;; fall-through index after pushing the jump command.
            ;; Anyway, get the index of the fall-through location.
            (fall-through-index (audio:audio-index)))
       ;; When we bash the jump command the microcode will jump to here
       ;; instead, which will cause the flag to get zeroed and the
       ;; audio facility to stop.  Both events happen atomically as far
       ;; as Lisp can tell because no samples are output in the
       ;; intervening time.
       (audio:push-audio-zero-flag flag-index)
       (audio:push-audio-stop)
       ;; Start the audio
       (audio:audio-start start)
       ;; Wait the appropriate number of microseconds.
       (loop with start-time = (sys:%microsecond-clock)
             until
             (≥ (%32-bit-difference (sys:%microsecond-clock) start-time) duration))
       ;; Here is where we bash the argument of the jump command to
       ;; instead jump to the fall-through code.
       (audio:modify-audio-command-arg fall-through-index :index jump-index)
       ;; Wait for the microcode to get to the flag and stop before we exit.
       (audio:wait-for-audio-flag flag-index "%BEEP")))))
```

14.4.5 Non-real-time Synthesis Example

Certain kinds of very high quality sound cannot be generated in real time (one sample computed every 20 microseconds). Small pieces (pieces that can fit in physical memory) can be computed and then played later.

```
(defun play-audio-sample-array
  (array &optional (from 0) (to (array-active-length array)))
  (audio:with-audio ()
    ;; with-wired-structure wires the structure on entry
    ;;  and unwires on exit. External sample arrays must be wired.
    (si:with-wired-structure array
      (let* ((flag-index (audio:reserve-audio-flags 1))
             (start (audio:audio-index)))
        ;; Cause the samples to be played.  If we supplied a non-NIL
        ;; immediate-p argument, we wouldn't have to wire the
        ;; structure, since the samples would be put in the command
        ;; array which is already wired.  However, most command arrays
        ;; are not very large and probably couldn't hold all the
        ;; samples.  It's a tradeoff.
        (audio:push-array-of-audio-samples array from to)
        ;; When the microcode finishes the samples, cause it to clear
        ;; the flag and stop.
        (audio:push-audio-zero-flag flag-index)
        (audio:push-audio-stop)
        ;; Start it up and wait for it to finish.
        (audio:audio-start start)
        (audio:wait-for-audio-flag flag-index "Play samples")))))
```

14.4.6 Playing Large Pieces Example

Larger pieces (those that are too big to fit in physical memory) can still be played. This program plays data that is stored on the FEP filesystem. Storage must be on the FEP filesystem for several reasons. The digital audio system must produce data at the rate of one sample every 20 microseconds (including all overhead). This is 1.6 megabits per second, which is a small factor away from raw disk speed. After overhead, this is getting close to the limits of the system. The LMFS file system incurs too much overhead. Also, we cannot copy (as LMFS would try to do if we used **:string-in** into an array) and we cannot spend time wiring buffers (as we would need to do with LMFS if we used **:read-input-buffer**).

The FEP filesystem allows us to do disk direct memory access (DMA) directly into a buffer that we can keep wired. We can also setup the audio facility to point to these buffers (using **push-array-of-audio-samples**) once so we do not have to do it often.

The macro **with-multi-disk-buffering** takes care of multibuffering bookkeeping. The user decides how many pages to devote to each buffer and the number of buffers. Disk arrays (the buffers) are allocated and wired on entry and unwired on exit.

```
(defmacro with-multi-disk-buffering
    ((npages nbuffers) (array-of-buffers size-of-each-buffer) &body body)
  "npages and nbuffers are inputs, array-of-buffers and size-of-each-buffer are outputs"
  `(let ((,array-of-buffers (make-array ,nbuffers))
         (,size-of-each-buffer (* ,npages 288.)))
     (unwind-protect
       (progn (loop for .idx. below ,nbuffers
                    as .buffer. = (allocate-resource 'si:disk-array
                                                     (+ ,size-of-each-buffer 288.))
                    do (setf (aref ,array-of-buffers .idx.) .buffer.)
                    (si:wire-structure .buffer.))
              ,@body)
       (loop for .idx. below ,nbuffers
             as .buffer. = (aref ,array-of-buffers .idx.)
             do (when (si:structure-wired-p .buffer.)
                  (si:unwire-structure .buffer.))
             (deallocate-resource 'si:disk-array .buffer.)))))
```

The function **play-disk-file** is the workhorse. There are many "if we are fast enough" clauses in this example. As long as there is not much other activity (especially paging activity) we usually are fast enough.

```
(defun play-disk-file (pathname)
  (setq pathname (fs:merge-pathnames pathname "FEP:>ǂ.mus.newest"))
  ;; get the FEP file opened.
  (with-open-file (file pathname :direction :block
                        :if-exists :overwrite
                        :if-does-not-exist :error)
    ;; These numbers were picked after much experimentation and tuning.
    (let* ((npages 40.) (nbuffers 8))
      (audio:with-audio ()
        (with-multi-disk-buffering (npages nbuffers) (buffers buffer-size)
          ;; allocate a flag for each buffer for synchronization.
          (let* ((flags (audio:reserve-audio-flags nbuffers))
                 (start (audio:audio-index)))
            ;; build the audio program.  Push each buffer as an array of
            ;; samples and then cause the flag associated with the
            ;; buffer to be zeroed.
            (loop for buffer below nbuffers
                  do (audio:push-array-of-audio-samples (aref buffers buffer)
                     0 buffer-size) (audio:push-audio-zero-flag
                     (+ flags buffer)))
```

```
;; Loop back to the beginning.  To play new data (if we are
            ;; fast enough, there /will/ be new data in the buffers).
            (audio:push-audio-jump start)
            ;; n-queued is the number of buffers filled with valid data
            ;; that the microcode can use.  (The microcode will use
            ;; all of them, but if we are fast enough we can keep them full.)
            ;; We fill up all the buffers and then start the audio facility.
            ;; This is done by an interaction with need-to-start and n-queued.
            ;; (There is also provision for small files.)  When all the buffers
            ;; are queued, we need to wait for the microcode to finish
            ;; the next one before we can do disk dma into it.
            (loop with n-queued = 0
                  with need-to-start = t
                  with n-file-blocks = (sys:ceiling (send file :length) 1152.)
                  with current-file-block = 0
                  initially (format t "~&~F seconds~%"
                                    (// (* n-file-blocks 288.) audio:*sample-rate*))
                  as blocks-this-whack =
                     ;; This is the number of blocks to do this time
                     ;; around.  It is at most the number of pages of
                     ;; buffering.  It is also at most the number of
                     ;; blocks remaining in the file.
                     (min npages (- n-file-blocks current-file-block))
                  for buffer-number =
                      ;; This is the current buffer number we are going
                      ;; to try to fill.  It is gets incremented modulo
                      ;; the number of buffers.
                      0 then (\ (1+ buffer-number) nbuffers)
                  as flag-index = (+ flags buffer-number)
                  do ;; If all the buffers are queued, or if the end of
                     ;; the file has been reached, wait for the
                     ;; microcode to finish the buffer and then count it
                     ;; as dequeued.
                     (when (or (= n-queued nbuffers) (zerop blocks-this-whack))
                       (audio:wait-for-audio-flag flag-index "Play disk file")
                       (decf n-queued))
                     ;; If we have some blocks to queue, make sure the
                     ;; flag for this buffer is reset, read in the
                     ;; blocks from the FEP file, increment the block
                     ;; pointer into the file, and count another buffer
                     ;; as queued.
                     (when (not (zerop blocks-this-whack))
                       (audio:reset-audio-flag flag-index)
                       (send file :block-in current-file-block blocks-this-whack
                             (aref buffers buffer-number))
                       (incf current-file-block blocks-this-whack)
                       (incf n-queued))
```

```
;; If the audio facility hasn't been started and
                    ;; all buffers are filled, start the audio facility
                    ;; (and remember we did start it).
                    (when (and need-to-start
                               (or (= n-queued nbuffers)
                                   (≥ current-file-block n-file-blocks)))
                      (audio:audio-start start)
                      (setq need-to-start nil))
                  until
                    ;; We are finished when nothing is queued and we are
                    ;; at the end of the file.
                    (and (zerop n-queued)
                         (≥ current-file-block n-file-blocks)))))))))
```

14.4.7 Polyphony Example

This is a simple muse. It uses roughly the same multibuffering strategy as the disk example, so that portion will not be commented as heavily. (See the section "Playing Large Pieces Example".) The muse muses some number of voices (user specified) between 1 and 6. All voices start at DO (C). Each step (approximately every 1/4 second) causes each voice to wander randomly between 2 diatonic tones below the previous value and 2 diatonic tones above the previous value.

```
;; This is the wave-array for the muse.  It is big enough to ensure that
;; there will be at least 256 (one page) consecutive words.
(defvar *muse-wave-array*
          (make-array (+ sys:page-size sys:page-size -1)
                      :initial-value 0 :area audio:audio-area))

(defun polyphonic-muse (&optional (n-voices 4))
  (check-arg n-voices (and (fixp n-voices)
                           (≤ 1 n-voices audio:*number-of-polyphonic-voices*))
                           "an integer between 1 and 6")
  (audio:with-audio ()
    (si:with-wired-structure *muse-wave-array*
      (let ((offset-to-page
              ;; This is how one gets to the number of Qs to
              ;;  the beginning of a page boundary
              (ldb sys:%%vma-word-offset
                   (- sys:page-size
                      (ldb sys:%%vma-word-offset
                      (%pointer (locf (aref *muse-wave-array* 0))))))))
```

```
;; Setup the muse wave array for a 1/6 (minus a bit) amplitude
        ;; sinewave (sawtooth doesn't seem to sound good here).  1/6
        ;; allows all six voices to proceed without overflow.  The
        ;; "minus a bit" avoids clipping at 1.0.
        (loop for index below sys:page-size
              do (setf
                   (aref *muse-wave-array* (+ index offset-to-page))
                   (audio:float-polyphonic-wave-table-entry
                     (// (sin (// (* 2.0 si:pi index) sys:page-size)) 6.2))))
        ;; Initialize each voice to a reasonable value.  It is essential
        ;; that each voice gets a proper wave-array pointer and
        ;; increment value.  An increment value of 0 will cause the
        ;; pointer never to be incremented.  (This isn't strictly true,
        ;; since the voice number is stored in the low 3 bits, but this
        ;; advances the pointer very slowly.)
        (let ((start (audio:audio-index)))
          (loop for voice below audio:*number-of-polyphonic-voices*
                do
          (audio:push-audio-load-voice voice *muse-wave-array* offset-to-page 0))
          (audio:push-audio-stop)
          (audio:audio-start start)
          ;; Put the audio index back to the start
          (setf (audio:audio-index) start))
        (loop with nbuffers = 4
              with n-queued = 0
              with need-to-start = t
              with flags = (audio:reserve-audio-flags nbuffers)
              with start = (audio:audio-index)
              with chords-per-whack =
                ;; Take the room remaining, divide by the level of
                ;; buffering and then divide by the sum of [2 locations
                ;; per voice for the push-audio-load-voice command, one
                ;; for the push-audio-polyphony command, and one for a
                ;; possible flag or jump].
                (// (audio:audio-room) nbuffers (+ (* n-voices 2) 1 1))
              with half-tone-offsets =
                ;; 0 (and the multiples of 12) are DO.  The other
                ;; numbers are offsets (from 0) to consecutive notes in
                ;; the diatonic scale.
                '(-25. -24. -22. -20. -19. -17. -15. -13.
                       -12. -10. -08. -07. -05. -03. -01.
                       000. +02. +04. +05. +07. +09. +11.
                       +12. +14. +16. +17. +19. +21. +23.
                       +24. +26. +28. +29. +31. +33. +35.)
              with half-tone-offsets-length = (length half-tone-offsets)
              with voice-indices =
                ;; A list, one element for each voice,
                ;;  starting at middle DO.
                (make-list n-voices
```

```
                    :initial-value (find-position-in-list 000. half-tone-offsets))
              for buffer-number = 0 then (\ (1+ buffer-number) nbuffers)
              until (kbd-tyi-no-hang)          ;stop when user hits a key
              do
    (when (≥ n-queued nbuffers)
          ;; This also resets the flag
          (audio:wait-for-audio-flag (+ flags buffer-number) "Muse")
          (decf n-queued))
        ;; if this is buffer zero, make sure we are back to the start.
        (when (zerop buffer-number)
          (setf (audio:audio-index) start))
        ;; Set up the chords for this buffer
        (loop repeat chords-per-whack
            ` do ;; update each voice
                (loop for voice-indices-scan on voice-indices
                      as old-index = (car voice-indices-scan)
                      as new-index =
                           (let ((index (+ old-index (random 5) -2)))
                                ;; clip at the boundaries of the list
                                (cond ((< index 0) 1)
                                      ((≥ index half-tone-offsets-length)
                                          (- half-tone-offsets-length 2))
                                      (T index)))
                      do (setf (car voice-indices-scan) new-index))
                ;; Queue the new values to polyphony facility
                (loop for index in voice-indices
                      for voice-number upfrom 0
                      as half-tone-offset = (nth index half-tone-offsets)
                      as octave-offset = (// half-tone-offset 12.0)
                      as frequency-factor = (expt 2.0 octave-offset)
                      as frequency = (* 256.0 frequency-factor)
                      do
                        (audio:push-audio-load-voice
                          voice-number *muse-wave-array* offset-to-page
                          (audio:frequency-polyphonic-increment frequency)))
                ;; Do polyphony for 1/4 second
                (audio:push-audio-polyphony
                        (sys:round audio:*sample-rate* 4)))
        ;; Synchronize this buffer
        (audio:push-audio-zero-flag (+ flags buffer-number))
        (incf n-queued)
        (when (and (≥ n-queued nbuffers) need-to-start)
          (audio:push-audio-jump start)
          (audio:audio-start start)
          (setq need-to-start nil)))))))
```

Index

A **A** **A**

New Feature: Floating Point Accelerator 9
Time Parser No Longer Accepts Dates in European Format 21
Time Parser Accepts Dates in ISO Standard Format 24
Direct Access Bidirectional File Streams 31
Direct access bidirectional streams 31
Direct Access File Streams 30
Effect of Character Set Translation on Direct Access File Streams 63
Direct Access Output File Streams 31
Active patches 18
Using FEP Clear Paging-files Command Before Add Paging-file 44
New Default for FEP Add Paging-file Command 43
tv:add-system-key renamed to **tv:add-select-key**; **tv:add-escape-key** renamed to **tv:add-function-key** 21
tv:add-system-key renamed to **tv:add-select-key**; **tv:add-escape-key** renamed to **tv:add-function-key** 21
tv:add-system-key renamed to **tv:** **add-select-key**; **tv:add-escape-key** renamed to **tv:add-function-key** 21
tv: **add-system-key** renamed to **tv:add-select-key**; **tv:add-escape-key** renamed to **tv:add-function-key** 21
Internet Domain Addressing 28
Amplitude envelopes 69
Previously Undocumented Feature: Audio Level and Screen Brightness Controls on 3670 and 3640 Consoles 23
Command arrays 73
UNIX ASCII print server 60
Installing an ASCII Print Server on UNIX 60
Installing an ASCII Print Server on VMS 59
Existence of audio 74
Functions, Variables, and Macros for Digital Audio 73
Testing for the Existence of Audio 74
Audio Command Format 67
Audio command lists 71
Building Audio Command Lists 74
Looping Through Audio Command Lists 77
Audio Command Opcodes 67
Examples of Using the Audio Facilities 81
Introduction to the Digital Audio Facilities 65
Lisp Primitives for the Digital Audio Facilities 73
Microcode Support for the Digital Audio Facilities 65
Previously Undocumented Feature: Audio Level and Screen Brightness Controls on 3670 and 3640 Consoles 23
Audio microtask 65
Starting and Stopping the Audio Microtask 79
Digital Audio Parameters 73
New Audio System in Release 5.2 65
The Audio Wrapping Form 74

audio: **audio-exists** variable 74
audio: **audio-index** function 75
audio: **audio-limit** function 75
audio: **audio-loop** macro 78
audio: **audio-push-audio-stop** function 75
audio: **audio-room** function 75
audio: **audio-start** function 79
audio: **audio-stop** function 79
audio:*number-of-polyphonic-voices* variable 74
audio:*sample-rate* variable 73
audio:audio-exists variable 74
audio:audio-index function 75
audio:audio-limit function 75
audio:audio-loop macro 78
audio:audio-push-audio-stop function 75
audio:audio-room function 75
audio:audio-start function 79
audio:audio-stop function 79
audio:computing-immediate-audio-samples macro 77
audio:fix-channel-float function 79
audio:fix-polyphonic-wave-table-entry function 80
audio:fix-sample function 79
audio:float-channel-fix function 79
audio:float-polyphonic-wave-table-entry function 81
audio:float-sample function 80
audio:frequency-polyphonic-increment function 81
audio:modify-audio-command-arg function 76
audio:polyphonic-wave-table-entry-channels function 81
audio:push-array-of-audio-samples function 77
audio:push-audio-jump function 75
audio:push-audio-load-voice function 75
audio:push-audio-polyphony function 76
audio:push-audio-zero-flag function 75
audio:push-immediate-audio-sample macro 77
audio:reserve-audio-flags function 78
audio:sample-add-fix function 80
audio:sample-add-float function 80
audio:sample-add-sample function 80
audio:sample-channels function 80
audio:set-audio-repeat-count macro 78
audio:wait-for-audio-flag function 78
audio:with-audio macro 74

B

B

Start Background Save (m-X) Zmail Command 27
Beep Example 84
Optional Argument to **fed** Can Now be a BFD Object 17
Bidirectional file streams 30
Direct Access Bidirectional File Streams 31
Direct access bidirectional streams 31
Binary files 45
Previously Undocumented Feature: Audio Level and Screen Brightness Controls on 3670 and 3640

B

Consoles 23
Size in bytes of packet buffer 37
Size of Packet Buffers 35
New Hardcopy Option to Kill or Save Buffers (m-X) 15
New Command: Select Some Buffers As Tag Table (m-X) 15
Buffers tape spec parameter 54
Buffs tape spec parameter 54
Bufs tape spec parameter 54
Building Audio Command Lists 74
Size in bytes of packet buffer 37

C

c-m-L is a Synonym for [Select] 27
c-m-L Zmail command 27
c-m-Y 27
New Numeric Arguments for c-m-Y 27
Previously Undocumented Feature: the 3600-family Calendar Clock 23
Carry lister 49
Describe contents of carry tape 49
New Feature: Carry Tape System 45
tape: **carry-dump** function 45
tape: **carry-list** function 49
tape: **carry-load** function 47
char-code function 10
char-flipcase function 11
char-standard function 10
Character files 45
Effect of Character Set Translation on Direct Access File Streams 63
:host-or-local tv: **choose-variable-values** variable type 24
Using FEP Clear Paging-files Command Before Add Paging-file 44
:clear-eol message 29
:clear-rest-of-line message 29
:clear-screen message 29
:clear-window message 29
Previously Undocumented Feature: the 3600-family Calendar Clock 23
Garbage Collection of Compiled Functions 17
c-m-L Zmail command 27
h-c-*upper-left* command 41
New Default for FEP Add Paging-file Command 43
Select Patch (m-X) Zmacs command 18
Show Draft Disposition (m-X) Zmail Command 27
Start Background Save (m-X) Zmail Command 27
Command array 74
Command arrays 73
Using FEP Clear Paging-files Command Before Add Paging-file 44
FEP Halt Command Differences in 3600s and 3670s 41
Audio Command Format 67
FEP Shutdown Command Replaces Halt Command in FEP Version 22 41
Command list 65
Audio command lists 71
Building Audio Command Lists 74
Looping Through Audio Command Lists 77
Audio Command Opcodes 67
Polyphony Command Opcodes 71

FEP Shutdown Command Replaces Halt Command in FEP Version 22 41
New Dired Command, L, Loads a File 15
Previously Undocumented FEP Command: Load FEP 43
New Command: Select Some Buffers As Tag Table (m-X) 15
New FEP Command: Set Display-string 42
New FEP Command: Set Monitor-type 42
Opcodes for audio commands 67
Two New Commands: Select Patch (m-X) and View Patch (m-X) 15
Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) 15
Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) 15
Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) 15
FEP Version, Serial I/O, and Model Number Compatibility in Release 5.2 39
Garbage Collection of Compiled Functions 17
New Compiler Special Form: **compiler:make-message-obsolete** 17
New Compiler Special Form: **compiler:make-message-obsolete** 17
compiler:make-message-obsolete special form 17
Computing Polyphonic Increments 81
audio: **computing-immediate-audio-samples** macro 77
Using the Terminal Program with Hosts Connected to the Serial Line 35
Previously Undocumented Feature: Audio Level and Screen Brightness Controls on 3670 and 3640 Consoles 23
Describe contents of carry tape 49
Previously Undocumented Feature: Audio Level and Screen Brightness Controls on 3670 and 3640 Consoles 23
Conversions Between Sample Formats 79
Conversions for the Polyphony Feature 80
Clarifications and Corrections for Release 5.2 63
Create new logical host 32
Faster Creation of Logical Pathname Hosts 32
Current patch 18
Wavetable cursor 69
Cursors 69
Customizing and Saving the World 3

D D D

Time Parser No Longer Accepts Dates in European Format 21
Time Parser Accepts Dates in ISO Standard Format 24
Clarification of use of **dbg:with-erring-frame** 63
dbg:with-erring-frame macro 63
Debugging in the FEP 43
Previously Undocumented Topic: Debugging in the FEP 43
defconstant 12
defconstant's Query Offers Three Choices 12
New special form: **define-prompt-and-read-type** 22
define-prompt-and-read-type special form 22
Improvements to **:error-disposition** Option of **net:** **define-server** 37
Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) 15
Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest

Definition (m-X) 15
Den tape spec parameter 54
Dens tape spec parameter 54
:density option for tape:carry-dump 45
Density tape spec parameter 54
Describe contents of carry tape 49
Dev tape spec parameter 54
Device tape spec parameter 54
FEP Halt Command Differences in 3600s and 3670s 41
Functions, Variables, and Macros for Digital Audio 73
Introduction to the Digital Audio Facilities 65
Lisp Primitives for the Digital Audio Facilities 73
Microcode Support for the Digital Audio Facilities 65
Digital Audio Parameters 73
Direct Access Bidirectional File Streams 31
Direct access bidirectional streams 31
Direct Access File Streams 30
Effect of Character Set Translation on Direct Access File Streams 63
Direct Access Output File Streams 31
Sending *who-am-i* Requests Directly to Namespace Servers 61
New Dired Command, L, Loads a File 15
Maxtor disk 50
New disk support 50
3600 **disk-save** Improvement 62
Changes to Tape and Disks in Release 5.2 45
Improvements to Tape and Disks in Release 5.2 50
Incompatible Changes to Tape and Disks in Release 5.2 45
New Features in Tape and Disks in Release 5.2 45
Tape Spec Prompting: Changes to Tape and Disks in Release 5.2 53
New FEP Command: Set Display-string 42
Inspector Displays Hash Tables 19
Show Draft Disposition (m-X) Zmail Command 27
Transport and distribute FEP files 49
Distribute transportable systems and libraries 49
Improvements to Distribution Tape Loader 51
Distribution Tape Program Changes 50
Dumper: Distribution Tape Program Changes 50
New Tape Spec Prompting: Distribution Tape Program Changes 51
The Dump Menu: Distribution Tape Program Changes 50
Internet Domain Addressing 28
Show Draft Disposition (m-X) Zmail Command 27
Installing a Tape Drive 59
Incompatible: Must Specify Dumper Tape Drive Identifier 45
New Feature: TD-80 Tape Drive Support 50
Dump individual files 49
The Dump Menu: Distribution Tape Program Changes 50
The Dumper 45
Incompatible: Must Specify Dumper Tape Drive Identifier 45
Dumper: Distribution Tape Program Changes 50

E

Improvements to **tv:** **edit-namespace-object** 37
Effect of Character Set Translation on Direct Access File Streams 63
Amplitude envelopes 69
Improvements to **:error-disposition** Option of **net:define-server** 37
Time Parser No Longer Accepts Dates in European Format 21
Existence of audio 74
Testing for the Existence of Audio 74

F

Changes in FEP Fault Lights on the 3600 42
Changes in FEP Fault Lights on the 3670 39
Previously Undocumented Feature: Audio Level and Screen Brightness Controls on 3670 and 3640 Consoles 23
New Feature: Carry Tape System 45
New Feature: Floating Point Accelerator 9
New Feature: Maxtor XT-1140 Support 50
New Feature: TD-80 Tape Drive Support 50
Previously Undocumented Feature: the 3600-family Calendar Clock 23
Optional Argument to **fed** Can Now be a BFD Object 17
FEP 7
Debugging in the FEP 43
Previously Undocumented FEP Command: Load FEP 43
Previously Undocumented Topic: Debugging in the FEP 43
New Default for FEP Add Paging-file Command 43
Using FEP Clear Paging-files Command Before Add Paging-file 44
Previously Undocumented FEP Command: Load FEP 43
New FEP Command: Set Display-string 42
New FEP Command: Set Monitor-type 42
Changes in FEP Fault Lights on the 3600 42
Changes in FEP Fault Lights on the 3670 39
Transport and distribute FEP files 49
Previously Undocumented Tape Utility for Writing FEP Files to Tape 44
Previously Undocumented Utility: Writing FEP Files to Tape 52
Writing FEP Files to Tape 52
FEP Halt Command Differences in 3600s and 3670s 41
Additional Information About the FEP in Release 5.2 43
Changes to the FEP in Release 5.2 39
FEP Shutdown Command Replaces Halt Command in FEP Version 22 41
FEP Version 18: Improvements in Release 5.2 39
FEP Version 18: New Features in Release 5.2 40
FEP Shutdown Command Replaces Halt Command in FEP Version 22 41
FEP Version 22: Improvements 42
FEP Version 22: Incompatible Changes 41
FEP Version 22: New Features 42
FEP Version 23: Incompatible Changes 43
FEP Version, Serial I/O, and Model Number Compatibility in Release 5.2 39
si:halt has new *fep-commands* argument 40
New Dired Command, L, Loads a File 15
:finish is supported by some output file streams 34

Bidirectional file streams 30
Direct Access File Streams 30
Direct Access Bidirectional File Streams 31
Direct Access Output File Streams 31
Effect of Character Set Translation on Direct Access File Streams 63
File System 7
Changes to the File System in Release 5.2 29
Improvements to the File System in Release 5.2 32
Incompatible Changes to the File System in Release 5.2 29
New Features in the File System in Release 5.2 29
:file-or-files option for tape:carry-dump 45
Binary files 45
Character files 45
Dump individual files 49
Transport and distribute FEP files 49
V*nn*-debug.flod files 43
Transfer source files between hosts 49
Input files from tape 47
Snapshot files on a local LMFS 49
Previously Undocumented Tape Utility for Writing FEP Files to Tape 44
Previously Undocumented Utility: Writing FEP Files to Tape 52
Write files to tape 45
Writing FEP Files to Tape 52
Fill pointer 74
Filters 27
Moving a Message by Filters Now Uses All Filters 27
Moving a Message by Filters Now Uses All Filters 27
:finish 34
:finish is supported by some output file streams 34
audio: **fix-channel-float** function 79
audio: **fix-polyphonic-wave-table-entry** function 80
audio: **fix-sample** function 79
Synchronization Flags 78
New Flavor Functions: **send-if-handles**, lexpr-send-if-handles, and **operation-handled-p** 12
audio: **float-channel-fix** function 79
audio: **float-polyphonic-wave-table-entry** function 81
audio: **float-sample** function 80
New Feature: Floating Point Accelerator 9
compiler:make-message-obsolete special form 17
define-prompt-and-read-type special form 22
The Audio Wrapping Form 74
Audio Command Format 67
Sample Format 66
Time Parser Accepts Dates in ISO Standard Format 24
Time Parser No Longer Accepts Dates in European Format 21
Conversions Between Sample Formats 79
audio: **frequency-polyphonic-increment** function 81
Processor front panel lights 39, 42
fs:make-logical-pathname-host function 32
fs:set-logical-pathname-host function 33
si: **full-gc** function 17
audio:audio-index function 75
audio:audio-limit function 75
audio:audio-push-audio-stop function 75
audio:audio-room function 75
audio:audio-start function 79

audio:audio-stop function 79
audio:fix-channel-float function 79
audio:fix-polyphonic-wave-table-entry function 80
audio:fix-sample function 79
audio:float-channel-fix function 79
audio:float-polyphonic-wave-table-entry function 81
audio:float-sample function 80
audio:frequency-polyphonic-increment function 81
audio:modify-audio-command-arg function 76
audio:polyphonic-wave-table-entry-channels function 81
audio:push-array-of-audio-samples function 77
audio:push-audio-jump function 75
audio:push-audio-load-voice function 75
audio:push-audio-polyphony function 76
audio:push-audio-zero-flag function 75
audio:reserve-audio-flags function 78
audio:sample-add-fix function 80
audio:sample-add-float function 80
audio:sample-add-sample function 80
audio:sample-channels function 80
audio:wait-for-audio-flag function 78
char-code function 10
char-flipcase function 11
char-standard function 10
fs:make-logical-pathname-host function 32
fs:set-logical-pathname-host function 33
lexpr-send-if-handles function 12
operation-handled-p function 12
send-if-handles function 12
si:full-gc function 17
si:halt function 40
string-flipcase function 11
tan function 9
tape:carry-dump function 45
tape:carry-list function 49
tape:carry-load function 47
tape:write-fep-files-to-tape function 53
time:read-calendar-clock function 23
time:set-calendar-clock function 23
Garbage Collection of Compiled Functions 17
Functions, Variables, and Macros for Digital Audio 73

G

G

Garbage Collection of Compiled Functions 17
Symbols Added to the **global** Package in Release 5.2 9
Gran tape spec parameter 54
Granularity tape spec parameter 54
Group 47

G

H

h-c-*upper-left* command 41
h-c-*upper-left* waits for Lisp to stop itself 41
FEP Halt Command Differences in 3600s and 3670s 41
FEP Shutdown Command Replaces Halt Command in FEP Version 22 41
si: **halt** function 40
si: **halt** has new *fep-commands* argument 40
New Hardcopy Option to Kill or Save Buffers (m-X) 15
Inspector Displays Hash Tables 19
Release 5.2: Introduction and Highlights 7
New Variable to Hold Packet Size 37
Create new logical host 32
Specifying a New Logical Host Name 51
Host tape spec parameter 54
:host-or-local option for **prompt-and-read** 24
:host-or-local tv:choose-variable-values variable type 24
Faster Creation of Logical Pathname Hosts 32
Transfer source files between hosts 49
Using the Terminal Program with Hosts Connected to the Serial Line 35

I

FEP Version, Serial I/O, and Model Number Compatibility in Release 5.2 39
Identifier 45
Incompatible: Must Specify Dumper Tape Drive Identifier 45
Inactive patches 18
Incompatible: Must Specify Dumper Tape Drive Identifier 45
Increment 69
Left increment 79
Right increment 79
Computing Polyphonic Increments 81
Polyphonic increments 81
Polyphonic wavetable increments 80
Initial patch 18
Input files from tape 47
Inspector Displays Hash Tables 19
Introduction to Installation Instructions 1
Installation Instructions (3600 and 3670 Only) 1
Installation Instructions (3640 Only) 3
Installation Instructions for Release 5.2 1
Installing a Tape Drive 59
Installing an ASCII Print Server on UNIX 60
Installing an ASCII Print Server on VMS 59
Changes to the User Interface in Release 5.2 21
Improvements to the User Interface in Release 5.2 24
Incompatible Changes to the User Interface in Release 5.2 21
New Features in the User Interface in Release 5.2 22
Internet Domain Addressing 28
Time Parser Accepts Dates in ISO Standard Format 24

K

Keyboard keys 7
Keyboard keys 7
New type keyword for **prompt-and-read** and tv:choose-variable-values: **:host-or-local** 24
New keyword options to **open** 32
New Hardcopy Option to Kill or Save Buffers (m-X) 15

L

New Dired Command, L, Loads a File 15
Changes to the Lisp Language in Release 5.2 9
Left increment 79
Len tape spec parameter 54
Length tape spec parameter 54
Previously Undocumented Feature: Audio Level and Screen Brightness Controls on 3670 and 3640 Consoles 23
lexpr-send-if-handles function 12
New Flavor Functions: **send-if-handles**, **lexpr-send-if-handles**, and **operation-handled-p** 12
Distribute transportable systems and libraries 49
Processor front panel lights 39, 42
Changes in FEP Fault Lights on the 3600 42
Changes in FEP Fault Lights on the 3670 39
Using the Terminal Program with Hosts Connected to the Serial Line 35
Improvements to Lisp in Release 5.2 12
Incompatible Changes to Lisp in Release 5.2 9
New Features in Lisp in Release 5.2 9
Lisp language 7
Changes to the Lisp Language in Release 5.2 9
Lisp Primitives for the Digital Audio Facilities 73
Lisp Primitives for Wiring Memory 72
h-c-*upper-left* waits for Lisp to stop itself 41
Command list 65
Carry lister 49
The Lister 49
Audio command lists 71
Building Audio Command Lists 74
Looping Through Audio Command Lists 77
Snapshot files on a local LMFS 49
LMFS Tape Spec Prompting 29
Previously Undocumented FEP Command: Load FEP 43
:nowarn option to **load-patches** 63
Clarification of Use of **:nowarn** Option to **load-patches** 63
Improvements to Distribution Tape Loader 51
The Loader 47
Responses to loader queries 47
New Dired Command, L, Loads a File 15
Snapshot files on a local LMFS 49
Create new logical host 32
Specifying a New Logical Host Name 51
Faster Creation of Logical Pathname Hosts 32
Time Parser No Longer Accepts Dates in European Format 21
Looping Through Audio Command Lists 77

M M M

New Command: Select Some Buffers As Tag Table (m-X) 15
New Hardcopy Option to Kill or Save Buffers (m-X) 15
Two New Commands: Select Patch (m-X) and View Patch (m-X) 15
Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) 15
Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) 15
Two New Commands: Select Patch (m-X) and View Patch (m-X) 15
Select Patch (m-X) Zmacs command 18
Show Draft Disposition (m-X) Zmail Command 27
Start Background Save (m-X) Zmail Command 27
Machine tape spec parameter 54
audio:audio-loop macro 78
audio:computing-immediate-audio-samples macro 77
audio:push-immediate-audio-sample macro 77
audio:set-audio-repeat-count macro 78
audio:with-audio macro 74
dbg:with-erring-frame macro 63
Functions, Variables, and Macros for Digital Audio 73
fs: **make-logical-pathname-host** function 32
New Compiler Special Form: **compiler:** **make-message-obsolete** 17
compiler: **make-message-obsolete** special form 17
:no-increment-patch option to **make-system** 63
Clarification of Use of **:no-increment-patch** Option to **make-system** 63
make-system 'fep-tape 52
Release 5.2: Operations and Site Management 59
Maxtor disk 50
Lisp Primitives for Wiring Memory 72
Unwired memory 71
Wired memory 71
The Dump Menu: Distribution Tape Program Changes 50
Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) 15
How Tape Specs Are Merged 55
Tape Spec Merging Examples 58
:clear-rest-of-line message 29
:clear-window message 29
:read-bytes message 30
Moving a Message by Filters Now Uses All Filters 27
Stream Messages 30
New Microcode in Release 5.2: 296 on 3600 8
Microcode Support for the Digital Audio Facilities 65
Audio microtask 65
Starting and Stopping the Audio Microtask 79
Minimum tape spec parameter 54
Minimum-length tape spec parameter 54
Minimum-record tape spec parameter 54
Minimum-record-length tape spec parameter 54
Minrec tape spec parameter 54
Minreclen tape spec parameter 54
FEP Version, Serial I/O, and Model Number Compatibility in Release 5.2 39
audio: **modify-audio-command-arg** function 76
New FEP Command: Set Monitor-type 42

Moving a Message by Filters Now Uses All Filters 27
Patch Facility Supports Multiple Patches In-progress 18
Music systems 69

N

N-buffers tape spec parameter 54
N-buffs tape spec parameter 54
N-bufs tape spec parameter 54
Specifying a New Logical Host Name 51
:clear-screen and **:clear-eol** Obsolete Names 29
Sending *who-am-i* Requests Directly to Namespace Servers 61
Improvements to **:error-disposition** Option of **net:define-server** 37
Neti:raw-packet-buffer-size 37
neti:raw-packet-buffer-size variable 37
Networks 7
Changes to Networks in Release 5.2 35
Improvements to Networks in Release 5.2 37
Incompatible Changes to Networks in Release 5.2 35
New Features in Networks in Release 5.2 35
Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) 15
Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) 15
:no-increment-patch option to **make-system** 63
Clarification of Use of **:no-increment-patch** Option to **make-system** 63
Non-real-time synthesis 85, 86
Non-real-time Synthesis Example 85
:nowarn option to **load-patches** 63
Clarification of Use of **:nowarn** Option to **load-patches** 63
FEP Version, Serial I/O, and Model Number Compatibility in Release 5.2 39
Number-of-buffers tape spec parameter 54
audio: ***number-of-polyphonic-voices*** variable 74
New Numeric Arguments for c-m-Y 27

O

Optional Argument to **fed** Can Now be a BFD Object 17
Audio Command Opcodes 67
Polyphony Command Opcodes 71
Opcodes for audio commands 67
New keyword options to **open** 32
Operation of Polyphony 69
New Flavor Functions: **send-if-handles**, **lexpr-send-if-handles**, and **operation-handled-p** 12
operation-handled-p function 12
Release 5.2: Operations and Site Management 59
Notes on Operations in Release 5.2 59
:finish is supported by some output file streams 34
Direct Access Output File Streams 31

P

Symbols Added to the **global**
Size in bytes of
Size of
New Variable to Hold
Using FEP Clear Paging-files Command Before Add
New Default for FEP Add
Using FEP Clear
Processor front
Digital Audio
Tape Spec
Time
Time

Current
In-progress
Initial
Two New Commands: Select Patch (m-X) and View
Two New Commands: Select
Select

Active
Inactive
Patch Facility Supports Multiple
Faster Creation of Logical

New Feature: Floating
Fill

Computing

audio:

Operation of

Conversions for the
The
Lisp
Lisp
UNIX ASCII
Installing an ASCII
Installing an ASCII

Distribution Tape
Dumper: Distribution Tape
New Tape Spec Prompting: Distribution Tape
The Dump Menu: Distribution Tape
Using the Terminal

:host-or-local option for
New type keyword for

LMFS Tape Spec
Tape Spec

P

Package in Release 5.2 9
packet buffer 37
Packet Buffers 35
Packet Size 37
Paging-file 44
Paging-file Command 43
Paging-files Command Before Add Paging-file 44
panel lights 39, 42
Parameters 73
Parameters 54
Parser Accepts Dates in ISO Standard Format 24
Parser No Longer Accepts Dates in European Format 21
patch 18
patch 18
patch 18
Patch (m-X) 15
Patch (m-X) and View Patch (m-X) 15
Patch (m-X) Zmacs command 18
Patch Facility Supports Multiple Patches In-progress 18
Patch states 18
patches 18
patches 18
Patches In-progress 18
Pathname Hosts 32
Playing Large Pieces Example 86
Point Accelerator 9
pointer 74
Polyphonic increments 81
Polyphonic Increments 81
Polyphonic wavetable increments 80
polyphonic-wave-table-entry-channels function 81
Polyphony 69
Polyphony 69
Polyphony Command Opcodes 71
Polyphony Example 89
Polyphony feature 65
Polyphony Feature 80
Polyphony Feature 69
Primitives for the Digital Audio Facilities 73
Primitives for Wiring Memory 72
print server 60
Print Server on UNIX 60
Print Server on VMS 59
Processor front panel lights 39, 42
Program Changes 50
Program Changes 50
Program Changes 51
Program Changes 50
Program with Hosts Connected to the Serial Line 35
prompt-and-read 24
prompt-and-read and **tv:choose-variable-values: :host-or-local** 24
Prompting 29
Prompting: Changes to Tape and Disks in Release

5.2 53
New Tape Spec Prompting: Distribution Tape Program Changes 51
zwei: ***prune-headers-after-yanking**** 27
audio: **push-array-of-audio-samples** function 77
audio: **push-audio-jump** function 75
audio: **push-audio-load-voice** function 75
audio: **push-audio-polyphony** function 76
audio: **push-audio-zero-flag** function 75
audio: **push-immediate-audio-sample** macro 77

Q

Q Q

Responses to loader queries 47
defconstant's Query Offers Three Choices 12

R

R R

Sample rate 73
Neti: raw-packet-buffer-size 37
neti: **raw-packet-buffer-size** variable 37
:read-bytes message 30
time: **read-calendar-clock** function 23
Reclen tape spec parameter 54
Record-length tape spec parameter 54
Recsize tape spec parameter 54
:reel option for tape:carry-dump 45
Reel tape spec parameter 54
report option for tape:carry-dump 45
Sending *who-am-i* Requests Directly to Namespace Servers 61
audio: **reserve-audio-flags** function 78
Responses to loader queries 47
Right increment 79

S

S S

Sample Format 66
Conversions Between Sample Formats 79
Sample rate 73
audio: **sample-add-fix** function 80
audio: **sample-add-float** function 80
audio: **sample-add-sample** function 80
audio: **sample-channels** function 80
audio: ***sample-rate**** variable 73
Storing Samples 76
Start Background Save (m-X) Zmail Command 27
New Hardcopy Option to Kill or Save Buffers (m-X) 15
Saving Subsequent Worlds 5
Customizing and Saving the World 3
Sawtooth Wave Example 83
Previously Undocumented Feature: Audio Level and Screen Brightness Controls on 3670 and 3640 Consoles 23
Two New Commands: Select Patch (m-X) and View Patch (m-X) 15
Select Patch (m-X) Zmacs command 18
New Command: Select Some Buffers As Tag Table (m-X) 15
[Select] 27
c-m-L is a Synonym for [Select] 27
send-if-handles function 12

New Flavor Functions: **send-if-handles**, **lexpr-send-if-handles**, and **operation-handled-p** 12
zwei: ***send-mail-about-patch**** 18
Sending *who-am-i* Requests Directly to Namespace Servers 61
FEP Version, Serial I/O, and Model Number Compatibility in Release 5.2 39
Using the Terminal Program with Hosts Connected to the Serial Line 35
UNIX ASCII print server 60
Installing an ASCII Print Server on UNIX 60
Installing an ASCII Print Server on VMS 59
Sending *who-am-i* Requests Directly to Namespace Servers 61
New FEP Command: Set Display-string 42
New FEP Command: Set Monitor-type 42
Character set translation 63
Effect of Character Set Translation on Direct Access File Streams 63
audio: **set-audio-repeat-count** macro 78
time: **set-calendar-clock** function 23
fs: **set-logical-pathname-host** function 33
:set-pointer 30
Show Draft Disposition (m-X) Zmail Command 27
FEP Shutdown Command Replaces Halt Command in FEP Version 22 41
si:full-gc function 17
si:halt function 40
si:halt has new *fep-commands* argument 40
Sine Wave Example 81
Release 5.2: Operations and Site Management 59
New Variable to Hold Packet Size 37
Size in bytes of packet buffer 37
Size of Packet Buffers 35
Snapshot files on a local LMFS 49
Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) 15
Two New Commands: Source Compare Newest Definition (m-X) and Source Compare Merge Newest Definition (m-X) 15
Transfer source files between hosts 49
compiler:make-message-obsolete special form 17
define-prompt-and-read-type special form 22
New Compiler Special Form: **compiler:make-message-obsolete** 17
New special form: **define-prompt-and-read-type** 22
Tape spec parameter specification 56
Incompatible: Must Specify Dumper Tape Drive Identifier 45
Specifying a New Logical Host Name 51
Square Wave Example 83
Start Background Save (m-X) Zmail Command 27
Starting and Stopping the Audio Microtask 79
Patch states 18
h-c-*upper-left* waits for Lisp to stop itself 41
Starting and Stopping the Audio Microtask 79
Storing Samples 76
Stream Messages 30
:finish is supported by some output file streams 34
Bidirectional file streams 30
Direct access bidirectional streams 31

Direct Access Bidirectional File Streams 31
Direct Access File Streams 30
Direct Access Output File Streams 31
Effect of Character Set Translation on Direct Access File Streams 63
New functions: **char-flipcase** and **string-flipcase** 11
string-flipcase function 11
Notes on Wired Structures 71
Saving Subsequent Worlds 5
New disk support 50
New Feature: Maxtor XT-1140 Support 50
New Feature: TD-80 Tape Drive Support 50
Microcode Support for the Digital Audio Facilities 65
:finish is supported by some output file streams 34
Patch Facility Supports Multiple Patches In-progress 18
Symbols Added to the **global** Package in Release 5.2 9
Synchronization Flags 78
Non-real-time synthesis 85, 86
Non-real-time Synthesis Example 85
File System 7
New Feature: Carry Tape System 45
Changes to the File System in Release 5.2 29
Improvements to the File System in Release 5.2 32
Incompatible Changes to the File System in Release 5.2 29
New Audio System in Release 5.2 65
New Features in the File System in Release 5.2 29
Music systems 69
Distribute transportable systems and libraries 49

T T T

New Command: Select Some Buffers As Tag Table (m-X) 15
Inspector Displays Hash Tables 19
New Command: Select Some Buffers As Tag Table (m-X) 15
tan 9
New Function: **tan** 9
tan function 9
Tangent 9
:tape-host option for tape:carry-dump 45
:density option for tape:carry-dump 45
:file-or-files option for tape:carry-dump 45
:reel option for tape:carry-dump 45
:tape-host option for tape:carry-dump 45
report option for tape:carry-dump 45
tape:carry-dump function 45
tape:carry-list function 49
tape:carry-load function 47
tape:write-fep-files-to-tape function 53
New Feature: TD-80 Tape Drive Support 50
Using the Terminal Program with Hosts Connected to the Serial Line 35
Testing for the Existence of Audio 74
Changes in FEP Fault Lights on the 3600 42
New Microcode in Release 5.2: 296 on 3600 8
Installation Instructions (3600 and 3670 Only) 1
3600 **disk-save** Improvement 62
Previously Undocumented Feature: the 3600-family Calendar Clock 23

FEP Halt Command Differences in 3600s and 3670s 41
Previously Undocumented Feature: Audio Level and Screen Brightness Controls on 3670 and 3640 Consoles 23
Installation Instructions (3640 Only) 3
Changes in FEP Fault Lights on the 3670 39
Previously Undocumented Feature: Audio Level and Screen Brightness Controls on 3670 and 3640 Consoles 23
Installation Instructions (3600 and 3670 Only) 1
FEP Halt Command Differences in 3600s and 3670s 41
Time Parser Accepts Dates in ISO Standard Format 24
Time Parser No Longer Accepts Dates in European Format 21
time:read-calendar-clock function 23
time:set-calendar-clock function 23
Transfer source files between hosts 49
Character set translation 63
Effect of Character Set Translation on Direct Access File Streams 63
Transport and distribute FEP files 49
Distribute transportable systems and libraries 49
tv:add-system-key renamed to **tv:add-select-key**; **tv:add-escape-key** renamed to **tv:add-function-key** 21
tv:add-system-key renamed to **tv:add-select-key**; **tv:add-escape-key** renamed to **tv:add-function-key** 21
tv:add-system-key renamed to **tv:add-select-key**; **tv:add-escape-key** renamed to **tv:add-function-key** 21
tv:add-system-key renamed to **tv:add-select-key**; **tv:add-escape-key** renamed to **tv:add-function-key** 21
:host-or-local **tv:choose-variable-values** variable type 24
New type keyword for **prompt-and-read** and **tv:choose-variable-values**: **:host-or-local** 24
Improvements to **tv:edit-namespace-object** 37
:host-or-local **tv:choose-variable-values** variable type 24
New type keyword for **prompt-and-read** and tv:choose-variable-values: **:host-or-local** 24

U U U

Unit tape spec parameter 54
Installing an ASCII Print Server on UNIX 60
UNIX ASCII print server 60
Unwired memory 71
Changes to Utilities in Release 5.2 17
Improvements to Utilities in Release 5.2 17
New Features in Utilities in Release 5.2 17
Previously Undocumented Tape Utility for Writing FEP Files to Tape 44
Previously Undocumented Utility: Writing FEP Files to Tape 52

V V V

audio:*number-of-polyphonic-voices* variable 74
audio:*sample-rate* variable 73
audio:audio-exists variable 74
neti:raw-packet-buffer-size variable 37
New Variable to Hold Packet Size 37
:host-or-local **tv:choose-variable-values** variable type 24
Functions, Variables, and Macros for Digital Audio 73

FEP Version, Serial I/O, and Model Number Compatibility in Release 5.2 39
Two New Commands: Select Patch (m-X) and View Patch (m-X) 15
Installing an ASCII Print Server on VMS 59
V*nn*-debug.flod files 43
Voices 69
Vol tape spec parameter 54
Volume tape spec parameter 54

W

audio: **wait-for-audio-flag** function 78
Sawtooth Wave Example 83
Sine Wave Example 81
Square Wave Example 83
Wavetable 69
Wavetable cursor 69
Polyphonic wavetable increments 80
Sending *who-am-i* Requests Directly to Namespace Servers 61
Wired memory 71
Notes on Wired Structures 71
Lisp Primitives for Wiring Memory 72
audio: **with-audio** macro 74
Clarification of use of **dbg:** **with-erring-frame** 63
dbg: **with-erring-frame** macro 63
Customizing and Saving the World 3
Saving Subsequent Worlds 5
The Audio Wrapping Form 74
Write files to tape 45
tape: **write-fep-files-to-tape** function 53
Writing FEP Files to Tape 52
Previously Undocumented Tape Utility for Writing FEP Files to Tape 44
Previously Undocumented Utility: Writing FEP Files to Tape 52

Z

Zmacs 7
Select Patch (m-X) Zmacs command 18
Changes to Zmacs in Release 5.2 15
New Features in Zmacs in Release 5.2 15
Zmail 7
c-m-L Zmail command 27
Show Draft Disposition (m-X) Zmail Command 27
Start Background Save (m-X) Zmail Command 27
Changes to Zmail in Release 5.2 27
zwei:*prune-headers-after-yanking* 27
zwei:*send-mail-about-patch* 18