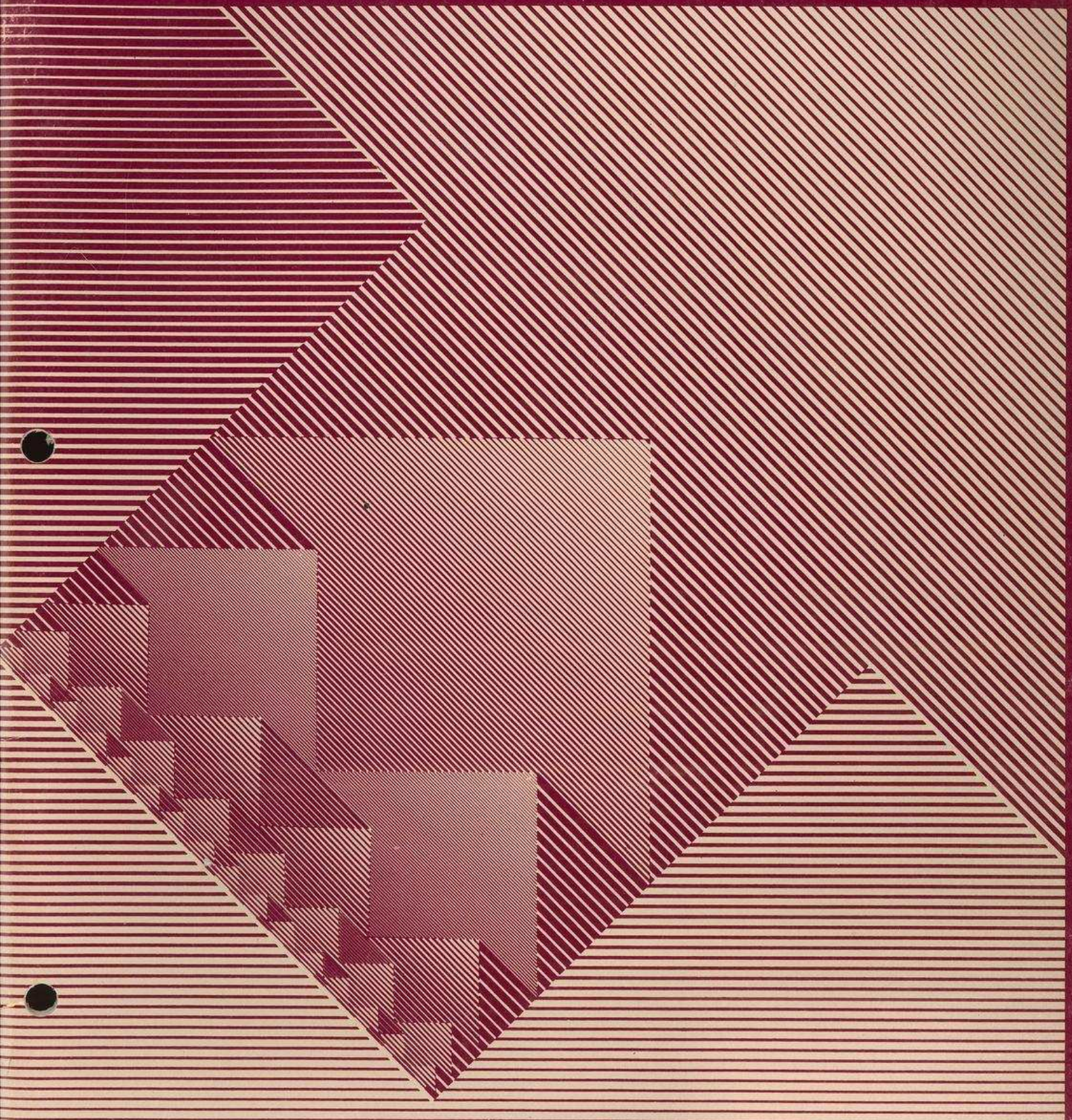
symbolics™

Lisp Machine Summary

3600 Edition



symbolics™

Lisp Machine Summary

3600 Edition August 1983

Jeff Katz

Cambridge, Massachusetts

Lisp Machine Summary 3600 Edition

 # 990075

August 1983

This document corresponds to Release 4.4.

This document was prepared by the Documentation and Education Services Department of Symbolics, Inc.

The information in this document is subject to change without notice and should not be construed as a commitment by Symbolics, Inc. Symbolics, Inc. assumes no responsibility for any errors that might appear in this document.

Symbolics, Inc. makes no representation that the interconnection of its products in the manner described herein will not infringe on existing or future patent rights, nor do the descriptions contained herein imply the granting of a license to make, use, or sell equipment constructed in accordance with its description.

The terms and conditions governing the sale of Symbolics hardware products and the licensing of Symbolics software consist solely of those set forth in the written contracts between Symbolics and its customers. No representation or other affirmation of fact contained in this document, including but not limited to statements regarding capacity, response-time performance, suitability for use or performance of products described herein, shall be deemed to be a warranty by Symbolics for any purpose, or give rise to any liability of Symbolics whatsoever.

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.

Symbolics, Inc. assumes no responsibility for the use or reliability of its software on equipment that is not supplied or maintained by Symbolics, Inc.

Table of Contents

Introduction

Scope and Audience

The *Lisp Machine Summary 3600 Edition* provides basic information about the Symbolics 3600 Lisp Machine. It is designed to assist both new and experienced users in effectively operating the 3600.

New users will find helpful information about getting started on the 3600 and about such topics as windows, errors and stuck states, pathnames, Zmail, the editor, and the Lisp environment. Experienced users will find the document useful as a quick reference guide.

Detailed technical information about the 3600 can be found in the Symbolics documentation referred to throughout this document.

References

The margin label *See...* marks references to other Symbolics documents. They contain more detailed information about the current topic.

Notation Conventions

Keyboard

Modifier keys are designed to be held down while pressing other keys. They do not themselves transmit characters. A combined keystroke like META-X is pronounced "meta x" and written as m-X. This notation means press the META key and, while holding it down, press the X key.

Modifier keys are abbreviated as follows:

Key	*Abbreviation*
CTRL	c-
META	m-
SUPER	s-
HYPER	h-
SHIFT	sh-

The keys with white lettering (like X or SELECT) all transmit characters. Combinations of these keys are meant to be pressed in sequence. This sequence is written as, for example, SELECT L. This notation means press the SELECT key, release it, and then press the L key.

Text

This document uses the following notation conventions:

Appearance in document	*Representing*
send, chaos:host-up	Printed representation of Lisp objects in running text.
RETURN, ABORT, c-F	Keyboard keys.
SPACE	Space bar.
`login`	Literal type-in.
`(make-symbol "foo")`	Lisp code examples.
(**function-name** *arg1* [*arg2*])	Syntax description of the invocation of **function-name**.
arg1	Argument to the function **function-name**, usually expressed as a word that reflects the type of argument (e.g., *string*).
[*arg2*]	Optional argument; you can leave it out.
Undo, Tree Edit Any	Command names in Zmacs and Zmail appear with initial letter of each word capitalized.
Insert File (m-X)	Extended command names in Zmacs and Zmail. Use m-X to invoke one.
[Map Over]	Menu items.
(L), (R2)	Mouse clicks: L=left, L2=double click left, M=middle, M2=double click middle, R=right, R2=double click right.

Notation Conventions, *cont'd.*

Mouse

Mouse commands use notations for menu items and mouse clicks in the following ways:

1. Square brackets delimit a mouse command.
2. Slashes (/) separate the members of a compound mouse command.
3. The standard clicking pattern is as follows:
   - For a single menu item, always click left. For example, the following two commands are exactly the same:

     [Previous]
     [Previous (L)]

   - For a compound command, always click right on each menu item except the last, where you click left. For example, the following two compound commands are exactly the same:

     [Map Over / Move / Hardcopy]
     [Map Over (R) / Move (R) / Hardcopy (L)]

4. When the notation does not follow the standard, it shows explicitly which button to click. For example:

   [Map Over / Move (M)]
   [Previous (R)]

Getting Started and Logging Out

Introduction

This section provides information about the front-end processor (the FEP), and how to start, cold boot, log in to, and log out of the 3600.

The FEP

The 3600 includes a front-end processor, known as the FEP. The FEP is a small computer, inside the 3600 processor cabinet, based on a Motorola 68000 microcomputer chip.

The FEP plays several roles in the operation of the 3600 system. The FEP's most visible function is that of booting (starting the software of) the Lisp system. The FEP allows you to communicate with the 3600 via a variety of commands.

See...

- page 7, for information about FEP command completion.
- *Front-End Processor* for more information about the FEP and a listing of available commands.

Getting Started

To power up and start using the 3600, use the following procedure:

1. Turn on the 3600.
   a. Plug in the 3600. The front panel lights on the processor cabinet display "3600" when the machine is plugged in.
   b. Turn the key on the front panel to the vertical position, marked LOCAL.
   c. After the front panel lights display "Power up?", push the spring-loaded switch marked YES. The front panel lights then display "3600 on".

   After you have turned the machine on, the FEP has control of the console.
2. Cold boot the 3600.

   Cold booting is a complete reset of Lisp. It loads in a new world. Cold booting erases the contents of the Lisp environment, including the contents of editor and mail buffers. Never cold boot a 3600 that is being used by someone else.

   You can cold boot the 3600 when all of the following conditions hold:
   - The screen is white.
   - The FEP prompt (`Fep>`) appears in the upper left-hand corner.
   - A blinking cursor appears.

Getting Started and Logging Out, *cont'd.*

However, you cannot cold boot if any of the following conditions are true:

- The screen is black. This might indicate that the console is not turned on.
- The screen is white, but no characters appear. This might indicate that the video cable is disconnected. If the video cable is connected this condition might indicate a malfunction of the FEP and you should call your Symbolics Field Service Representative.

Cold boot the 3600 by typing the FEP command `boot` at the FEP prompt (`Fep>`) and pressing RETURN.

As it cold boots the 3600, the FEP executes the commands in the >configuration.fep file. Booting takes about two minutes. When the FEP has successfully cold booted the 3600, the *herald* (a multiline message beginning `Symbolics System`) appears.

3. Log in.

After cold booting, you are in a window named Lisp Listener 1. The following examples illustrate several options for logging in, where `whit` is your login name. Type all punctuation and parentheses as shown.

- to log into the default host machine, using your init file, type
```
(login "whit")
```
- to log into the default host machine, without your init file, type
```
(login "whit" t)
```
- to log into another host machine "sc3", using your init file, type
```
(login "whit" 'sc3)
```

If the host machine you log into is a timesharing computer system, you must have a directory and account on that host machine.

See...

- *Front-End Processor*, page 2, for more information about cold booting.
- *Lisp Machine Manual*, page 506, for more information about logging in.
- *Release 4.0 Release Notes*, page 63, for information about **login-forms**, a special form for wrapping around a set of forms in your init file.

Getting Started and Logging Out, *cont'd.*

Logging Out

1. Use any Lisp Listener by pressing SELECT L.
2. Log out by typing `(logout)`.

   Wait until the Lisp Listener returns T before you go to the next step.
3. Cold boot the 3600.

   This step is optional. It is not necessary to cold boot if the 3600 has been used only a short while and if no major changes to the machine state have been made. If the 3600 has been used for several hours and many files have been loaded or read into it, we recommend that the machine be cold-booted.

   Cold booting frees up virtual memory and puts the 3600 in a fresh state. In this way you do not contaminate the next user's environment with your customizations.

Note: You need not turn the machine off each night; however, it does not hurt the machine to do so.

Getting Help

Reference Material

The following is a list of the fundamental manuals available for the 3600.

Operating the Lisp Machine
Lisp Machine Manual
Front-End Processor
Zmacs Manual
File System
Font Editor
Program Development Help Facilities
Program Development Tools and Techniques
Zmail Tutorial and Reference Manual
Zmail Concepts and Techniques
Introduction to Using the Window System
Choice Facilities
Scroll Windows

HELP Key

The key labelled HELP looks up context-dependent documentation.

HELP	Shows documentation available for the current activity. In some programs, c-HELP, m-HELP, and so on, provide additional documentation.
FUNCTION HELP	Shows a list of useful functions that you can invoke using the FUNCTION key.
SELECT HELP	Shows programs and utilities that you can select using the SELECT key.

FEP Command Completion

While the keyboard is connected to the FEP, the following forms of completion are available:

- Pressing the HELP key at the FEP prompt (Fep>) or after typing part of the first word of a command shows the commands understood by the FEP command processor.
- Pressing the HELP key after typing the first word of a command shows a list of commands that begin with that word. Example: set SPACE HELP gives a list of commands that begin with the word set.

See...

- Page 22 and page 28 for more information about help facilities in editing and mail handling, respectively.
- *Front-End Processor* for more information about the FEP and a listing of available commands.

Recovering from Errors and Stuck States

Introduction

Sometimes it is hard to know whether or not you are in trouble, because some operations, particularly those involving other network machines, can take a long time. Periodically check the process state and the run bars on the status line (see page 10). Some process states mean trouble if they persist, say, for a minute or more.

Look at the clock in the status line. If the clock is ticking, processes are being scheduled. If the clock is not ticking, the 3600 is halted. As long as the FEP is working, it prints a message near the top of the screen when the 3600 has halted and then gives its `Fep>` prompt. When the 3600 resumes its previous state, it updates the clock with the correct time.

Recovery Procedures

If the status line displays one of the following process states, recover by using the appropriate procedure:

State	*Recovery procedure*
`Wait Forever`	Select a different window, then reselect the one you were in.
`Output Hold`	Press `FUNCTION TRIANGLE` (the `TRIANGLE` key is in the top row, center, and has a triangle symbol on it); if that puts you in the Debugger, use `ABORT`.
`Arrest`	Press `FUNCTION - A` (that is, a three-key sequence).
`Lock`	Try `FUNCTION 0 S` to see if any windows want to type out. If that does not help, press `c-ABORT`.
`Selected`	Press `FUNCTION 0 S`.
`(no window)`	Use the mouse or `SELECT` key to select the window you want.

You can press `c-m-SUSPEND` to force the current process into the Debugger. You can press `FUNCTION SQUARE` in cold-load-stream (not dependent upon windows) to get to a Lisp read-eval-print loop; (the `SQUARE` key is in the top row, center, and has a square symbol on it).

The Debugger

Errors that are not caught and handled by the program that triggered them invoke the Debugger. See page 26.

Resetting the FEP

Resetting the FEP restarts the FEP program. You must reset the FEP if the FEP does not respond to any keyboard characters, especially `c-LOCAL`. You must reset the FEP after the 3600 console has been disconnected and then reconnected.

To reset the FEP, use the following procedure:

1. Push the red button labelled `RESET` on the front panel of the 3600 processor cabinet.

Recovering from Errors and Stuck States, *cont'd.*

2. Push the spring-loaded switch marked YES when the front panel lights display "Reset FEP?".

Warm Booting

Warm booting causes either a **:flush** or **:reset** message to be sent to all processes in the 3600, depending on what type each process is.

To warm boot the 3600, use the following procedure:

1. Type either `(si:%halt)` to a Lisp Listener or press c-LOCAL. You are now connected to the FEP.
2. Type `start` at the FEP prompt (Fep>) and press RETURN.

Sometimes, the 3600 prints `lisp stopped itself` and returns control to the FEP. When this happens, at the FEP prompt (Fep>) you should type `show status`, check the information it provides, and then type `start`.

See...

See the *Lisp Machine Manual*, page 438, for more information about **:flush** and **:reset**.

Halting

Halting the 3600 leaves all Lisp states intact. To halt the 3600 in order to connect to the FEP, type `(si:%halt)` to a Lisp Listener or use c-LOCAL. You are now connected to the FEP. To return to Lisp, type `continue` at the FEP prompt (Fep>) and press RETURN.

The 3600 can halt itself under exceptional conditions. In this case, try typing `continue`. If `continue` does not work, use `start`.

The Screen

Overview

The screen always contains one or more windows (see page 15). Regardless of which windows are displayed, the screen always contains some information displays, including a *mouse documentation line* and a *status line*. These information displays are crucial in determining whether a 3600 is operating normally or needs intervention.

Mouse Documentation Line

The mouse documentation line is normally displayed in reverse video and contains documentation on the current meaning of mouse clicks. Pressing `FUNCTION 1 C` complements the video state of the mouse documentation line. (See page 12 for more information about the mouse.)

Status Line

The status line is the line of text at the bottom of the screen. It contains the following information:

- Date and time
- Login name
- Current package
- Process state
- Run bars
- Other context-dependent information, such as
  - Console idle time
  - Network service indicators

Process State

The process state refers to the processes associated with the selected window. See page 15. The following list shows some common states:

State	*Meaning*
`Mouse Out`	Waiting for the mouse process to notice a change of windows.
`Net In`	Waiting for data from another machine on the network.
`Net Out`	Waiting to send data to another machine on the network.
`Open`	Waiting to open a file on another machine on the network.
`Run`	Process is running.
`Tyi`	Waiting for input from keyboard or mouse.

The Screen, *cont'd.*

Run Bars

The run bars are thin horizontal lines near the process state in the status line. A description of each one follows:

GC bar (under the package name)
: Left half is visible when the scavenger is looking for references to objects that are candidates to become garbage. Right half is visible when the transporter is copying an object.

Disk bar
: Visible when the processor is waiting for the disk, typically because of paging. Nonpaging disk I/O usually waits via **process-wait**, in which case this bar does not appear.

Run bar (under the run/wait state)
: Visible when a process is running and not waiting for the disk. Not visible when the scheduler is looking for something to do.

Disk-save bar
: Visible when **disk-save** is reading from the disk; **disk-save** alternatively reads and writes large batches of pages. The alternating state of this bar tells you that **disk-save** is working while you wait for it.

The Mouse and Menus

The Mouse

The *mouse* is the small object with three buttons on top and a rolling ball on the bottom. Rolling the mouse along a flat surface moves a *mouse cursor* on the screen correspondingly. The mouse cursor is usually an arrow but can change shape to tell you that something is different about the window.

Using the Mouse

Hold the mouse with the wire away from you. The three buttons are referred to by position as left, middle, and right. Pushing a button down momentarily is called *clicking*. Clicking almost always causes some action to occur.

The mouse documentation line (see page 10) shows what action would occur if you pressed one of the mouse buttons. In the mouse documentation line, the following codes indicate which button to click for the desired action to occur:

Code	*Indication*
L	Click the left button.
L2	Click the left button twice in quick succession.
M	Click the middle button.
M2	Click the middle button twice in quick succession.
R	Click the right button.
R2	Click the right button twice in quick succession.

You can also get L2, M2, or R2 by holding down SHIFT and clicking respectively left, middle, or right once.

Mouse-sensitivity

Parts of the screen can be *mouse-sensitive;* that is, clicking one of the mouse buttons on these parts causes some action to occur. When the mouse cursor moves over a portion of the screen that is mouse-sensitive, an outline box appears around the item. Clicking on the boxed item in the manner specified in the mouse documentation line causes the desired action to occur.

Scrolling

Many windows in the system respond to scrolling commands. Bump the mouse against the left side of the pane until a scroll bar and double-headed pointer appear.

The scroll bar, by its size and placement on the left side of the pane, indicates the percentage of the pane or buffer contents that is currently visible on the screen. For example, a very short scroll bar at the bottom of the pane indicates that you are seeing the last part and only a small percentage of the contents of that pane or buffer. A very long scroll bar in the center of the pane indicates that you are seeing a large proportion and are approximately half-way through the pane's entire contents.

The Mouse and Menus, *cont'd.*

To scroll using the scroll bar and the double-headed pointer, use one of the following mouse buttons:

R	Moves the line currently at the top of the screen to the position indicated by the pointer.
M	Displays the percentage of the pane contents that approximately corresponds to the position indicated by the pointer.
L	Moves the line indicated by the pointer to the top of the screen.
L2	Moves the line indicated by the pointer to the bottom of the screen.

Menus

One common application of a mouse button is to call up a *menu* of options, containing mouse-sensitive choices. Menus are lists of mouse-sensitive choices, surrounded by a border. They normally appear in the part of the screen where the mouse cursor was positioned when you clicked the button.

The 3600 has several styles of menus, including the following common ones:

- Momentary menu

  Each item is a possible choice. Positioning the mouse cursor over an item and then clicking the appropriate button makes the choice. The System Menu is a momentary menu.

- Choose-variable-values menu

  Each line presents one or more possible values of a particular variable. The Window Attributes Menu is a choose-variable-values menu.

  Each variable has a type that controls what values it can take on. The way in which the possible values are presented and the way in which you choose a value depend upon the type. Variables can have one of two types.

  - A type with a small number of legal values. Each line in the menu presents the possible legal values of a particular parameter. The current value appears in bold face. Each of the values is mouse-sensitive. Clicking on a value selects it.
  - A type with a large or infinite number of legal values. Each line in the menu presents only the current value of a particular parameter. A numerical value is of this type. To change a value, select the current value by clicking on it, type in a new value, and press RETURN. Rubbing out more characters than have been typed in restores the original value instead of changing it.

You exit menus in a variety of ways. For some menus, like the System Menu, making the choice causes the menu to disappear. Moving the mouse cursor off this kind of menu also causes the menu to disappear. Other menus have explicit commands, such as [Do It], [Exit], or [Abort], which you

The Mouse and Menus, *cont'd.*

must click on to make the menu disappear. Other menus are displayed in the frame permanently, such as the Zmail Command Menu.

System Menu

The System Menu is a momentary menu that lists several choices for acting upon windows and calling programs (for example, Lisp Listener, Zmacs, or the Inspector). You can always call the system menu by clicking R2 (the right mouse button twice). Use the System Menu to do many things, among them:

- Create new windows.
- Select old windows.
- Change the size and placement of windows on the screen.
- Hardcopy a file.

See...

Introduction to Using the Window System and *Choice Facilities* for more information about the mouse and menus.

Selecting and Creating Windows

Introduction

All user interaction with the 3600, except with the FEP, occurs in *windows.* The screen always contains one or more windows. The window that you are interacting with is called the *selected window.* You select a window via the mouse, a menu, or a keyboard key. If the window is not already exposed, it appears on the screen. See *Introduction to Using the Window System* for more information about windows.

Default Windows

The 3600 has a default set of windows available, some of which are available via a System Menu and some via the SELECT key as well.

Use [Select] in the Windows column of the system menu to see a menu of currently available windows. Some default windows are

```
Main Zmail Window
Lisp Listener 1
Edit: pathname
```

Moving

On the 3600, you do not "leave" a window with an explicit terminating command; instead, you select a different window. You can return to the most recently used window by pressing FUNCTION S.

SELECT Key

A set of windows is always available by pressing the SELECT key and then one of the following keys:

Key	*Program*
C	Converse, for messages to other users
E	Editor, the Zmacs text and program editor
F	File system editor for access to files and directories
I	Inspector, for inspecting and modifying data structures
L	Lisp
M	Mail reading and sending system
P	Peek, a system status display
S	Supdup, a display-oriented virtual terminal utility for logging in to other hosts
T	Telnet, a virtual terminal utility for logging in to other hosts

Multiple Instances of a Window

Use SELECT c-*x* to create an additional instance of a window of the specified type, where *x* is a Key from the table above. The 3600 beeps if the system does not support multiple instances of windows of that type. Pressing SELECT *x* multiple times in a row cycles through all existing windows of type *x*.

Pathnames, Files, and Directories

Introduction

You can store files on the 3600 you are using, its file-server machine, on some other Lisp Machine, or on a host on a network. To allow the 3600 to deal with files in a host-independent way, *pathnames* are represented inside the 3600 in a format that is independent of any operating system. These are also called *physical pathnames.* By designating a file by its pathname, you indicate to the 3600 where to find it.

See...

Symbolics File System for more information about files.

Pathname Components

A full pathname consists of six components.

host	The machine the file resides on.
device	The device on the specified host.
directory	The directory on the device — used to group files.
name	The name of the file.
type	Usually the type of file, for example, lisp, text, bin.
version	A type of version number; usually increases in chronological order of creation of the file.

Pathname Syntax

Each site has a table of hosts that it recognizes in pathnames, including the default host that it uses in pathname defaults and the special host names **sys** and **local**. The part of a pathname following the host name has to be in the format expected by that host.

The components of a pathname are optional; the file system supplies default values for missing components. For example, the default value for the version is usually the most recent one.

Pathnames for various kinds of hosts are

Host type	*Pathname format*
3600 or LM-2	`host:>directory>name.type.version`
UNIX	`host:/directory/name.type`
VAX/VMS	`host:device:[directory]name.type.version`
TOPS-20	`host:device:<directory>name.type.version`
TENEX	`host:device:<directory>name.type;version`
Multics	`host:>directory>name.type`
ITS	`host:device:directory;name type`

Logical Pathnames

Another kind of pathname that does not correspond to any particular file server is called a *logical pathname* and its host is called a *logical host.* Every logical pathname can be translated into a corresponding physical pathname: A mapping from logical hosts into physical hosts performs this

Pathnames, Files, and Directories, *cont'd.*

translation. Since the logical-to-physical mapping can be changed easily, logical pathnames simplify keeping bodies of software on more than one file system. So, for instance, when software is moved from one site to another, the same pathnames can be used at all sites regardless of where the files are really stored.

An example of a logical pathname is `sys:site;version lisp`

See...

Lisp Machine Manual, page 388, for more information about logical pathnames.

File Handling

The 3600 does not currently have the same kind of commands for file handling as most timesharing hosts. It has two utilities for examining and manipulating files and directories. The first is the directory editor (Dired), selected in Zmacs using `c-X D` or `m-X Dired`. `?` shows the commands that are available in Dired. The second is the file system editor (FSEdit), selected using the key sequence `SELECT F`.

Several extended commands in Zmacs allow you to manipulate individual files. While in Zmacs, use `HELP A file` to see a list of these commands.

Introduction to the Lisp Environment

Introduction

The 3600 runs a dialect of Lisp called *Zetalisp*, which is a descendant of Maclisp. A window running the read-eval-print loop of this Lisp is called a Lisp Listener.

Use SELECT L, or [Lisp] in the System Menu, to select a Lisp Listener. Use SELECT c-L to create and then select an additional Lisp Listener.

The environment for Lisp program development includes the following:

- Zmacs, the editor, used to create Lisp source code and to compile functions and files.
- Lisp Listener, used to run the code.
- The Debugger, used to examine the environment.
- The Inspector, used to inspect and modify Lisp data structures.

See...

Lisp Machine Manual and *Program Development Tools and Techniques* for more information about the Lisp environment.

Packages

Since the Lisp Machine is a large-scale virtual-memory, single-user computer, many programs — the editor, the compiler, and so on — coexist in the same environment (address space). Once Lisp functions are loaded into the Lisp Machine environment, they remain there until the machine is cold-booted, that is, until a fresh version of the Lisp Machine software is loaded, wiping out the current Lisp world.

Since you might have two large programs that both define a function named **load**, the Lisp Machine must distinguish between functions with the same name that belong to different programs. The **package-declare** function provides a mechanism for separating the like-named functions in different programs by assigning each its own distinct context, or *name space*. The name space is called the package. The package name prefixes two identically named functions.

Example: **chaos:get-packet** and **arpa:get-packet** define two different functions named **get-packet**, one in package **chaos**, the other in package **arpa**.

In the Lisp world, packages are organized hierarchically, with subordinate packages having access to their own Lisp symbols as well as *inheriting* those of superior, or parent, packages. The package at the root of the tree of packages is called **global** and contains the common symbols of the entire Lisp language.

The standard system contains a number of packages, including the following:

Introduction to the Lisp Environment, *cont'd.*

Package name	*Package contents*
chaos	Chaosnet
fs	System-independent file access
lmfs	Lisp Machine file system
si	Internals for many system functions (subpackage of **sys**)
sys	Internal symbols for Lisp Machine "operating system"
time	Timekeeping software
tv	Window system
user	Default package for typing to Lisp
zwei	Editor
compiler	Lisp compiler

The default package is **user**. To invoke functions from a package other than the current package, you must either supply the package prefix or change the current package. For example:

Procedure	*Type-in*
Supply the package prefix	`(time:print-universal-time)`
Change the current package	`(pkg-goto 'time)` `(print-universal-time)`

The status line always displays the current package.

See...

Lisp Machine Manual, page 392, for more information about packages.

Functions and Variables at Lisp Top Level

`*` Represents the value printed the last time through the read-eval-print loop. (`**` and `***` are from the second-last and third-last time.)

```
(describe *)
```

`+` Last type-in (`++` and `+++` are second-last and third-last type-in).

(apropos *string package*)
Displays all symbols in the package whose print names contain the string. If you do not specify the *package*, **apropos** displays all symbols in all packages whose print-names contain the string.

```
(apropos "time")
(apropos "profile" "zwei")
```

`(arglist` *function*`)`
Displays the argument list of the function.

```
(arglist 'fs:merge-pathnames)
```

`c-C` Inserts the last form read, with everything but the closing parenthesis.

`m-C` After `c-C`, replaces last form read with previous form read with everything but the closing parenthesis; used to cycle through kill ring.

`(describe` *object*`)` Displays information (appropriate to the type of the object) about a particular object.

```
(describe terminal-io)
```

`(inspect` *object*`)` Displays information (appropriate to the type of the object) about a particular object. Uses a window display. Allows you to scroll up or down through a large object and modify some components of the object.

```
(inspect terminal-io)
```

`(ed)` Calls the editor. You can supply arguments to control what you want to edit.

`(ed t)`	Provides a new buffer
`(ed` *pathname*`)`	Edits the named file
`(ed` *function*`)`	Edits the file containing the definition for a particular function.

`(load` *pathname*`)` Loads the file (either source or compiled) into the Lisp environment

```
(load "local:>pdq>kmfile")
```

`(login` *user host init-flag*`)`
Performs several tasks, among them, setting the file name defaults, telling the 3600 where the init files are, usually loading them, setting the mail return address, and marking the 3600 in use.

`(logout)` Evaluates any logout forms (reversing effects of an init file), clears the user name, and performs file and network cleanup.

`(plist` *symbol*`)` Returns the list representing the property list of the symbol.

```
(plist 'zwei:*word-syntax-table*)
```

Functions and Variables at Lisp Top Level, *cont'd.*

`(print-disk-label)`
Lists the FEP:>* directory that contains all the files required to boot and diagnose the 3600 processor and that acts as the first level of disk organization.

`(print-herald)` Displays software versions, world load file name, and the names of the site, of this 3600, and of the associated host.

`(print-notifications)`
Displays all notifications received since cold boot.

`(print-sends)` Displays all sends received from users on other machines since cold boot.

`(make-system)` Does the actual work of compiling and loading. See the *Lisp Machine Manual*, page 411, for more information.

`(print-system-modifications)`
Displays description of patches loaded in the current system.

`(compiler:compile-file` *pathname*`)`
Compiles the file.

`(compiler:compile-file-load` *pathname*`)`
Compiles the file and then loads the resulting compiled code file.

`(trace` *function1* ...`)`
Specifies one or more functions to be traced. The functions are not evaluated here. See the *Lisp Machine Manual*, page 457, for further information about `trace`.

```
(trace zwei:find-file zwei:find-file-internal)
```

`(untrace` *function1* ...`)`
Disables tracing of one or more functions. See the *Lisp Machine Manual*, page 459, for further information about `untrace`.

`(viewf` *pathname*`)` Displays the specified file on the screen.

```
(viewf "f:>sys>lmfs>patch.directory")
```

`(who-calls` *symbol package*`)`
Displays one line of information about each function that calls a function, or uses the symbol as a variable or as a constant. By default it looks in all packages unless you specify a package.

```
(who-calls 'time:get-universal-time 'tv)
```

Editing

Introduction

The editor on the 3600 is called Zmacs. Use SELECT E to select it. Its commands are very similar to those of the EMACS editor (from ITS, UNIX, and TOPS-20 timesharing systems). This section includes lists of some of the commands available in the editor, summarizing them by category.

Some editor commands accept a preceding argument in the form c-*n*, m-*n*, s-*n*, h-*n*. That is, hold down the CTRL, META, SUPER, or HYPER key and press the digits you need for the argument.

See...

Zmacs Manual and *Program Development Help Facilities* for more information about the editor.

General Help Facilities

c-ABORT	Aborts the function currently executing.
c-G	Aborts a command while it is being entered, unselects the region, or unmerges a kill; that is, resets "state".
HELP A *string*	Shows every command containing *string* (try HELP A Paragr or HELP A Buffer).
HELP C *x*	Explains the action of any command (try HELP C c-K as an example).
HELP D *string*	Describes a command (try HELP D Query Rep).
HELP L	Displays the last 60 keys pressed.
SUSPEND	Starts a Lisp Listener (return from it with RESUME).

Zmacs Help Facilities

Undo (m-X)	Reverts to buffer before last kill, unkill, fill, sort, or similar complex command.
c-Y	Yanks back the last thing killed.
m-Y	After a c-Y, yanks back things previously killed; used after a c-Y to cycle through the kill ring.

Extended Commands

Extended commands (the m-X commands) put you in a small area of the screen with full editing capabilities (a *minibuffer*) for entering names and arguments. Several kinds of help are available in a minibuffer.

COMPLETE	Completes as much of the current command as possible.
HELP	Gives information about special characters and possible completions.
c-?	Shows possible completions for the command currently being entered.
END or RETURN	Complete the command, and then execute it.
c-/	Does an apropos on what has been typed so far.

Editing, *cont'd.*

Writing Files

c-X c-S	Writes the current buffer into a new version of the current file name.
c-X c-W	Writes the current buffer into a file with a different name.
Save All Files (m-X)	Offers to save each file whose buffer has been modified.

Buffer Operations

c-X c-F	Gets a file into a buffer for editing.
c-X B	Selects a different buffer (prompts; default is the last one).
c-X c-B	Displays a menu of available buffers; lines are mouse-sensitive.
c-X K	Kills a buffer (prompts for which one; default is current one).
m-<	Moves to the beginning of the current buffer.
m->	Moves to the end of the current buffer.

Character Operations

c-B	Moves left (back) a character.
c-F	Moves right (forward) a character.
c-P	Moves up (previous) a character.
c-N	Moves down (next) a character.
RUBOUT	Deletes a character left.
c-D	Deletes a character right.
c-T	Transposes the two characters around point; if at the end of a line, transposes the two characters before point, ht -> th.

Word Operations

m-B	Moves left (back) a word.
m-F	Moves right (forward) a word.
m-RUBOUT	Kills a word left (c-Y yanks it back at point).
m-D	Kills a word right (c-Y yanks it back at point).
m-T	Transposes the two words around point (if only -> only if).
m-C	Capitalizes the word following point.
m-L	Lower-cases the word following point.
m-U	Upper-cases the word following point.

Line Operations

c-A	Moves to the beginning of the line.
c-E	Moves to the end of the line.
c-O	Opens up a line for typing.
c-X c-O	Closes up any blank lines around point.
CLEAR-INPUT	Kills from the beginning of the line to point (c-Y yanks it back at point).
c-K	Kills from point to the end of the line (c-Y yanks it back at point).

Editing, *cont'd.*

Sentence Operations

m-A	Moves to the beginning of the sentence.
m-E	Moves to the end of the sentence.
c-X RUBOUT	Kills from the beginning of the sentence to point (c-Y yanks it back at point).
m-K	Kills from point to the end of the sentence (c-Y yanks it back at point).

Paragraph Operations

m-[	Moves to the beginning of the paragraph.
m-]	Moves to the end of the paragraph.
m-Q	Fills the current paragraph (see HELP A Auto fill).
n c-X F	Sets the fill column to *n* (example: c-6 c-5 c-X F).

Lisp Form Operations

c-m-@	Marks a form.
c-m-A	Moves backward, to the beginning of a definition.
c-m-E	Moves forward, to the end of a definition.
c-m-F	Moves forward, to the end of a Lisp object.
c-m-B	Moves backward, to the beginning of a Lisp object.
c-m-U	Moves up a level of list structure.
c-m-D	Moves down a level of list structure.
c-m-K	Kills a Lisp object right.
c-m-RUBOUT	Kills a Lisp object left.
c-m-T	Interchanges the Lisp expressions before and after point.

Screen Operations

c-V	Shows next screen.
m-V	Shows previous screen.
c-0 c-L	Moves the line where point is to line 0 (top) of the screen.
c-m-R	Repositions the window to display all of the current definition, if possible.
c-m-L	Selects the most recently selected buffer in this window.

Search and Replace

c-S *string*	"Incremental" search; searches while you are entering the string; terminate search with END.
c-R *string*	"Incremental" backward search; terminate search with END.
c-% *string1* RETURN *string2* RETURN	Replaces *string1* with *string2* throughout.
m-% *string1* RETURN *string2* RETURN	Replaces *string1* with *string2* throughout, querying for each occurrence of *string1*; press SPACE meaning "do it", RUBOUT meaning "skip", or HELP to see all options; (see HELP C m-%).

Editing, *cont'd.*

Region Operations

c-SPACE	Sets the mark, a delimiter of a region. Move the cursor from mark to create a region; (the editor underlines to show the region). Use with region commands c-W, m-W, and c-Y.
c-W	Kills region (c-Y yanks it back at point).
m-W	"Saves" region (c-Y yanks it back at point).
c-Y	Yanks back the last thing killed.

Window Operations

c-X 2	Splits the screen in two windows (same buffer shown in each).
c-X 1	Resumes single window, using the current window.
c-X O	Moves cursor to other window.
c-m-V	Shows next screen of the buffer in the other window; with a numeric argument scrolls that number of lines — positive for the usual direction, negative for the reverse direction.
c-X 4	Splits the screen into two windows and asks what should be shown in the other window.

Lisp Code Operations

m-.	Selects the definition of a function for editing, finding the file if necessary.
c-sh-A	Displays the argument list of the function called by the form in which the cursor is located.
c-sh-C	Compiles the current definition.
c-sh-D	Displays the short documentation for the function called by the form in which the cursor is located.
c-sh-E	Evaluates the current definition.
c-sh-M	Expands the macro to the right of the cursor.
c-sh-V	Displays information about the variable before the cursor.
Arglist (m-X)	Displays the argument list of a function (prompts for a function).
Compile Buffer (m-X)	Compiles the current buffer.
Find Unbalanced Parentheses (m-X)	Finds any parenthesis balancing error in current buffer.
Source Compare (m-X)	Compares two files or buffers.

The Debugger

Introduction

Program errors place you in the Debugger. Enter the Debugger explicitly by pressing m-SUSPEND or c-m-SUSPEND. The Debugger prompt is a small right-facing arrow. You can type Lisp forms to the Debugger and it will evaluate them.

The error reports show the Lisp function call that caused the error. You can usually press ABORT or RESUME to terminate the erring command and return to top level in the process. (Pressing RESUME is appropriate only after explicit entry or after certain kinds of errors from which you can proceed.) Otherwise you can use the Debugger to investigate the error.

Press the HELP key to display a short help message; press c-HELP to display all of the Debugger's single key commands.

Additionally, different error conditions define different commands that allow you to proceed in ways specific to the current error. These commands appear after the error message and are usually assigned to the SUPER key.

In the Debugger, pressing c-M sends mail containing a backtrace and the error message. You type in a description of how to make the error happen before pressing END to send the mail. Where the report goes depends on the particular site.

The Debugger, *cont'd.*

Debugger Commands

c-A	Shows the arglist, that is, the arguments expected, for the current stack frame.
c-E	Calls the editor to edit the function from the current stack frame.
REFRESH	Clears the screen and redisplays the error message.
c-L	Clears the screen and redisplays the error message.
c-M	Sends mail containing a backtrace and the error message. You type in a description of how to make the error happen before pressing END to send the message.
c-N or LINE	Goes down the stack by one frame to caller.
c-P or RETURN	Goes up the stack by one frame to callee.
c-R	Returns a value, values, or no value from the current frame.
c-S *string*	Searches down the stack for a frame calling a function with *string* in the name.
c-X	Toggles the trap-on-exit flag for the current frame.
c-n c-m-A	Displays value of the *n*th argument in the current frame.
c-m-R	Reinvokes a function (for example, after using c-E and c-sh-C).
c-m-W	Invokes the Display Debugger (the window-display version of the Debugger).
m-B	Shows a backtrace of function names with argument names and values.
m-L	Shows local variables and disassembled code for the current stack frame.
ABORT	Aborts to the previous Debugger level or to top level.
RESUME	Proceeds when possible.
HELP	Displays basic information about the Debugger.
c-HELP	Displays all of the Debugger's single-key commands.

Messages and Mail

Introduction

The 3600 message facility is called Converse. Use SELECT C to select it. It manages conversations with other network host users.

The 3600 mail system is called Zmail. Use SELECT M to select it. It offers most of the functions that are available in other mail systems. For most operations, you can use either menus or single-letter commands. When you are composing mail in Zmail, you are using the Zwei editor.

Zmail arranges the screen with a summary window in the top half, the currently selected message in the bottom half, and its main menu across the middle. You can use the mouse on the menu or on the summary to operate on messages. Watch the mouse line documentation for information.

See...

Zmail Concepts and Techniques and *Zmail Tutorial and Reference Manual* for more information about Zmail.

Zmail Help Facilities

HELP *	Shows a list of all things for which help is available.
HELP *item*	Press the HELP key, then click on a menu item. Zmail explains the action of that menu item.

Summary of Zmail Actions

Action	*How to do it*
Read your mail	Use [Get Inbox] from menu, or G from keyboard.
Select a different message	Use [Next] or [Previous] from menu, N or P from keyboard, or click left on the summary line for a particular message.
Compose a message	Use [Mail] from menu, or M from keyboard.
Reply to a message	Use [Reply] from menu, or R from keyboard. Mouse buttons select different styles of creating the header fields.
Stop composing a message or reply	Press ABORT. The message draft is saved in case you later need to resume (use [Continue] from the menu or C from the keyboard).
Delete a message	Use [Delete] from menu (mouse buttons indicate which message to select after deleting), D from keyboard, or click middle on the message-summary line.

Messages and Mail, *cont'd.*

Forward a message	Use [Mail/Forward] from menu.
Redistribute a message	Use [Mail/Redistribute] from menu.
See more of a long message	Use the screen operation command from Zmacs. (Space is equivalent to c-V while you are reading a message.)
Move around in the summary window	Several styles of moving are available. In one, use [Next (R)] or [Previous (R)] in main menu and select a destination from the resulting menu. In another, bump the mouse against the left edge of the window to get a double-arrow cursor for percentage scrolling.
Print a message on paper	Use [Move/Hardcopy] (if you have a printer).
File a message in another file	Use [Move/Read/Create file]. It prompts for a message file pathname, loads that file, and adds the current message to it. Use [Save] from the main menu to save the change.
Inspect another file of messages	Use [Select/Read/Create file]. It prompts for a message file pathname.
Examine messages that you have composed and sent or aborted	Use [Continue (R)].
Customize Zmail	Use [Profile] from menu. After making changes, use [Save] to save the init file. Use [Exit] to finish.
Expunge and saving the mail file	Use [Save] from main menu, or S from keyboard.
Complete a session	Use [Quit] or type Q to save all message sequences and exit Zmail; or select some other window.

Index

Lisp Machine Summary 3600 Edition
#990075

Design: Schafer/La Casse
Typesetting: Cover – Litho Composition Co.
Text – Symbolics LGP–1 Laser Graphics Printer
Printing: Henry Sawyer Co.