CHAPTER 22

USING INTERLISP-D ON THE XEROX 1100

Interlisp-D is an implementation of the Interlisp programming environment for the Xerox 1100 series of Scienti c Information Processors. Interlisp-D uses a byte-coded instruction set, deep binding, `CDR` encoding (in a 32-bit `CONS` cell) and incremental, reference-counted garbage collection. Virtually all of the Interlisp-D system is written in Lisp. A relatively small amount of microcode implements the Interlisp-D instruction set and provides support for a small set of other performance-critical operations.

Interlisp-D is completely upward compatible with Interlisp-10. All Interlisp system software runs under Interlisp-D. The only exceptions are those explicitly indicated in the manual as applicable only to Interlisp-10. In addition, Interlisp-D provides signi cant extensions that allow the user to exploit its personal computing environment.

Interlisp-D has incorporated both low level network access and a collection of various higher level protocols used to communicate over an Ethernet with printing and le servers, as well as with other Interlisp-D processors. In particular, all le operations are independent of whether the le resides on the local disk or on a remote le server. See page 21.1 for further information about Interlisp-D facilities for communicating over an Ethernet.

Interlisp-D includes an extensive collection of user graphics facilities. The most important of these is the window system, which allows the user to divide the screen into multiple autonomous overlapping regions.

The window package provides both interactive and program accessable constructs for creating, moving, reshaping, overlapping, and destroying windows in such a way that a program can be embedded in a window in a completely transparent fashion. This allows existing Interlisp-10 programs to be used without change, while providing a base for experimentation with more complex window operations in new applications. The basic graphics facilities are described in page 19.2.

Using the window system, Interlisp-D extends the existing Interlisp programming tools to use the display and also provides some completely new tools. The display break package provides pushbutton access to the stack (page 20.10). A data inspection program uses windows to symbolically examine and change arbitrary system data structures (page 20.12). A display-oriented structure editor allows convenient modi cation of Interlisp-D programs (page 20.1).

22.1 REPORTING PROBLEMS

The Interlisp-D support team is committed to providing full support services to the entire Interlisp-D user community. Users are encouraged to report problems with the system, errors in the documentation, and their general observations. Messages may be sent to us in several ways:

At most places using Interlisp-D there is one person designated as the Interlisp-D Site Liason, typically an experienced user. If the Site Liason is not able to help you with your problem, he or she should be able to contact the Interlisp-D support team.

Xerox 1100 Hardware

Users with access to the ARPANET are encouraged to send mail to the net address:

1100Support.pasa@PARC- MAXC.

Users without ARPANET sponsorship should call (213) 351-2351 ext. 2222 collect or write to:

1100 Support
Xerox SIS
250 North Halstead St.
Pasadena, CA 91107
ATTN: 1100Support

22.2 XEROX 1100 HARDWARE

Interlisp- D is currently available on three machines, the Xerox 1100 Scienti c Information Processor, the Xerox 1108, and the Xerox 1132. This section is only applicable to the Xerox 1100.

The Xerox 1100 Scienti c Information Processor consists of a cabinet approximately 19 inches wide by 25 inches high by 28 inches deep, a separate `CRT` display, keyboard, and pointing device. The hardware housed in the cabinet has no special environmental requirements and is suitable for installation in an o ce. It draws 10 amperes of power from a 120 volt, 50-60 Hz circuit.

22.2.1 Keyboard

The keyboard resembles a standard typewriter keyboard with several special function keys. `SHIFT` and `LOCK` operate similarly to those on a standard typewriter, except that the `LOCK` key causes only the alphabet keys to be uppercase, while the `SHIFT` key causes all keys on the keyboard to take on their uppercase representation.

A special button used for booting the system at the beginning of a session is located on the back side of the keyboard, immediately to the right of the heavy black wire that connects the keyboard to the rest of the system.

22.2.2 Display

The video display is an 875 line, horizontally- mounted television monitor with a 13-inch by 11-inch, 1024 by 808 black and white pixel screen. The display is mounted on a swivel base to allow easy viewing- angle adjustment. A brightness knob is located on the rear of the base along with a power switch for the display.

22.2.3 Mouse

Direct interaction with the display is carried out with a device called a mouse, a hand- sized object with three keys on top. The mouse is moved about on the working surface next to the keyboard and may be placed on whichever side of the keyboard is most comfortable for the user. As the mouse is moved, a cursor moves around on the display screen.

The three keys on top of the mouse are used for various functions depending on the software being run and the cursor's location on the screen.

22.2.4 Processor Cabinet

The processor cabinet houses the electronic circuitry and magnetic disk. The only operator controls on the processor cabinet are two pushbuttons that are used to turn the power on and o . The pushbuttons are located on the front panel. Also located on the front panel is a numeric display that is used primarily for maintenance purposes.

The Xerox 1100 is microprogrammed with a control store of 4K, 36-bit words with a microinstruction cycle time of about 200 nsec. The data paths within the machine are 16 bits wide and the processor-memory data path is 64 bits wide. The standard real memory is 1.15 MBytes. The memory is expandable to 1.5 MBytes.

The processor cabinet contains a Shugart SA 4008 disk, RS232 interface, and a printer port. The disk is xed (non-removable) with a formatted capacity of approximately 23 MBytes. Optional additions are a 3MHz Experimental Ethernet controller, a 10MHz Ethernet controller, and a color monitor controller board.

22.3 TURNING THE SYSTEM ON

To prepare the Xerox 1100 system for operation, proceed as follows:

a. Reach behind the base of the display and ensure that the toggle switch is up. This turns on power to the display.

b. Determine if power is applied to the processor by observing the small `LED` display panel on the front. The display should be blank except for a small illuminated red dot in the lower right part. If no red dot is visible, check with your local maintenance personnel. (Make sure the processor is plugged in.)

c. With the red dot illuminated, momentarily press the `START/ON` (blue colored) button located on the front panel. A series of numbers should appear in the `LED` display panel (See section X.X.X). After a few seconds, a large white rectangular region should appear in the middle of the display. If this does not happen, check that the cables between the Xerox 1100, the keyboard, and the display are rmly connected, and try pressing the `START/ON` button again.

d. If your system is connected to a (3MBit) Experimental Ethernet it may now automatically load and start a limited version of the Executive obtained from the net.

e. Wait approximately 2 minutes for the disk to reach operating speed and temperature. No light or other indication is provided to inform the user when the disk is ready for operation.

f. When the disk is ready, boot the system by momentarily pressing the button located on the back side of the keyboard just to the right of the black wire that connects the keyboard to the processor. This will start the Alto Executive (described below) and display some status information at the top of the screen. If this does not work, retry starting at step c, or check with your local maintainance personel.

22.4 TURNING THE SYSTEM OFF

When Interlisp-D and the Alto Executive access the local disk, they may momentarily leave the disk directory in an inconsistant state. Therefore, the user should not power down the Xerox 1100 while programs are running. To turn the system o , do the following:

First, exit Interlisp-D with `(LOGOUT)`.

Next, type `Quit`^{cr} to run the Alto `Quit` program.

Once `Quit` is running, it is safe to power o the Xerox 1100, by pressing the `OFF` button on the processor cabinet.

22.5 PARTITIONS

There are two distinct working areas (called ‘‘partitions’’) on the 1100 disk, which enable two di erent working environments to reside on the same machine. If desired, each can have its own unique password. This password allows two di erent individuals (or groups) to share the same machine without fear that a critical le may be accidentally deleted by the other user(s). The partition to be used is selected as follows:

During Startup

To Select Partition 1	Press boot button on rear of keyboard.
To Select Partition 2	Hold down the 0 (zero) key while pressing the boot button.

From the Alto Executive

To Switch to Partition N	Type: `Partition N`^{cr}

22.6 ALTO EXECUTIVE

After booting, the Xerox 1100 will be running the Alto Executive. The Alto Executive has a small number of capabilities, the most important of which is running Interlisp-D. Normally, little time will be spent at the Executive level, as most interaction will occur within the Interlisp-D environment. The Alto Executive is described in section 22.11. The Interlisp-D user may nd the following commands useful:

`Login`	For setting the user name
`Install`	For changing passwords and partition names
`Scavenger`	For recovering from disk problems
`SetTime`	For setting the time

When entering commands to the Executive, typing the following will correct mistakes:

`BS` (backspace) or `CTRL-A`	Delete last typed character
`CTRL-W`	Delete last typed word
`DEL` (delete)	Cancel command

WARNING: It is not recommended that Lisp les be manipulated at the Executive level, as the Executive and other non-Lisp programs do not follow Lisp's version numbering conventions. The recommended way of manipulating les is to use the Interlisp-D functions (such as `SEE`, `COPYFILE`, `RENAMEFILE`, `DIRECTORY`, etc.).

22.7 LOCAL DISK FILES

22.7.1 Naming Conventions

File names on the local disk are limited to character strings of less than 40 alphanumeric characters and the punctuation characters `+ - $`. No blanks are allowed. Upper and lower case letters may both be used, but are not distinguished. Interlisp creates le names with all upper case letters. See section X.X.X for additional details regarding Lisp les.

22.7.2 Critical Files

It is extremely important not to overwrite or delete certain les on the Xerox 1100 local disk. In the Xerox 1100 le system, les cannot be individually protected against reading, writing, or erasing. Because of this, the user should be extremely careful when deleting les, especially when using the ‘‘*’’wildcard. The following les are necessary for the system to function properly and should not be deleted:

AltoD0MC.eb	Lisp.run	Sys.boot
Create le.run *	Lisp.syms	Sysdir
DiskDescriptor	Lisp.sysout *	Sys.errors
DolphinLispMC.eb	Scavenger.run *	Sysfont.al
Executive.run	Swat	User.cm
Fonts.Widths *	Swatee	Lisp.virtualmem
Display Font Files (*.strike) *		

* If these les are available from a local le server, they do not have to be on the disk.

22.8 STARTING LISP

To start Lisp, type one of the following:

`Lisp/i`^{cr}	Brings in a completely fresh version of Interlisp-D from the disk le `LISP.SYSOUT` (like `LISP` in TOPS20).
`Lisp`^{cr}	Returns user to state saved from `(LOGOUT)` (like `CONTINUE` in TOPS20).
`Lisp FILENAME`^{cr}	Returns user to state saved in `(SYSOUT FILENAME )`. For machines connected to an Ethernet with a PUP FTP server, a `SYSOUT` can be retrieved from a le server by including the le server, directory and subdirectories in the eld name, e.g. `{PHYLUM}<LISP>CURRENT>LISP.SYSOUT` (like `SYSIN` in TOPS20).

In addition, any of these commands can be terminated with a semi-colon and a Lisp expression. The code after `MAKESYS`, `SYSOUT` and `LOGOUT` will read this expression and evaluate it if it starts with an open paren or doublequote. If it is a list, it is `EVAL`ed at the next prompt; if it is a string, it is just unread. This is a way of ‘‘typing ahead’’ over `Lisp.run`.

For example, `LISP {PHYLUM}<LISP>CURRENT>LISP.SYSOUT;(FILESLOAD FOO)` will start up Interlisp-D and load in `FOO.DCOM`.

22.8.1 Greeting

When Interlisp-D is started, the currently logged in user (see the Executive `Login` command) will be greeted. There are two phases to the greeting process: First, site-speci c parameters are established by loading the le `INIT.LISP` from the local disk. This le should de ne where various resources (printers, Lispusers les, fonts) are to be found given the machine's location and con guration. It also speci es a search list (the variable `USERGREETFILES`) for the greeting les of individual users. If a user-speci c initialization le is found by searching that list, the second phase of greeting consists of loading that le. User initialization les customize Interlisp-D for a particular user, by specifying values for various parameters such as the default directory variable `LOGINHOST/DIR` (page 18.12). The disk is delivered with an `INIT.LISP` that assumes there are no network facilities and that the user's initialization le is named `INIT.USERNAME` on the local disk. The le `INIT.PARCPLACE` is included on the disk as an example of a site-dependent `INIT.LISP` le for network sites.

22.9 CURSOR STATUS INDICATIONS

When Interlisp-D is running, the cursor will occasionally change to inform the user of certain system states. These are:

Inverted cursor	Garbage collecting (if `GCGAG` is `T`; see page 18.1)
Flashing line on top	Swap read
Flashing line on bottom	Swap write
Flashing lines in middle	Stack operations
"`SAV/ING`" cursor	If Interlisp-D has been idle long enough, the system automatically updates the disk virtual memory le. While this is happening, the cursor is changed to show the word `SAV/ING`. For more information about this feature (including how to turn it o), see page 18.3.

22.10 SYSTEM CRASH

The following table is designed to help you in case of a system crash. For more information on using Raid and Swat, see section 22.12.

System State	Cause	User Action
Raid	Accidentally typed CTRL-C	Type CTRL-N
(You are in Raid if the screen is blank, except for a message of the form ‘‘Raid: xxx’’ at the top. Prompt=‘‘@’’.)	Stack over ow (System message: No Free stack block found)	Type CTRL-D (ushes stack)
	System bug	STOP. Note message and prepare bug report. * Type CTRL-D (Attempts to reset to the top level EVALQT, where you may be able to save state).
	Disk hardware error. Message ‘‘hard disk error’’ appears.	Run Scavenger. See page X.XX.
Swat	Machine malfunction	Reboot
(You are in Swat if the screen displays two black bars, one labeled SWAT. Prompt=‘‘#’’.)	Typed CTRL-N to continue Raid at a bad time.	Type RAID CTRL-C (Puts you in Raid)
No System Response		If cursor follows mouse, type: CTRL-SHIFT-SWAT** (Puts you into Swat from which you can move to Raid, and then, back to Lisp.)
Machine halts or reboots, screen loses sync and/or cursor is no longer tracking	Machine malfunction, overheat condition or power line surge.	Reboot. If this does not seem to work, turn o the power to the display and/or the processor for a moment, and then try rebooting. If a power surge or sag occurs, the machine will automatically reboot.

* The bug report should answer the following questions: What were you doing? Is it reproducible? What les have you loaded? Anything special in your init? Does bug occur in Interlisp-10 (if available)?

** Simultaneously hold down the SHIFT, CTRL, and SWAT keys. SWAT is the unmarked key in the lower right corner of the keyboard.

22.11 ALTO EXECUTIVE

When the Alto Executive is running, it displays two lines of status information near the top of the screen. Included in this information is the amount of space that is left for storing les. This space is measured in disk pages; each page contains 512 bytes. It is prudent to keep at least 150 disk pages available. If your disk has fewer, delete some les, perhaps after sending them to a le server.

22.11.1 Correcting Typing Errors

When typing at the Executive and a mistake is made, there are a few special keys that can be used to correct the mistake. The BS (backspace) key or CTRL-A erases the last character you typed and CTRL-W erases the last word typed. The DEL key cancels the command that was currently being typed. It displays ‘‘XXX’’, and starts a new line with a fresh ‘‘>’’ prompt character.

22.11.2 File Name Patterns in Executive

The Executive provides simple facilities for handling les. It allows a group of les to be named by using le name patterns containing the characters ‘‘*’’ and ‘‘#’’. The ‘‘*’’ character stands for any string of characters. For example, the pattern ‘‘*.ABC*’’ stands for all the les which have ABC as the rst three characters of the extension. The ‘‘#’’ stands for any single character; for instance, ‘‘###.memo’’ stands for all the les which have a three character main name and the extension ‘‘memo.’’ To see what a pattern expands into, type CTRL-X immediately after typing it to get it expanded.

If a le name or pattern is typed to the Executive, and then TAB is pressed, a list of all the les whose names start with that name will be displayed. For example, typing:

```
>*.ABCTAB
```

will get a list of all les which have an extension starting with the characters ABC.

If ESC is pressed after typing the rst few letters of a le, the Executive will add as many characters as it can to complete a le name. If ‘‘?’’ is typed, a list of all les that start with what has already typed will be displayed. The user can then go on and nish the le name.

The characters for getting le names expanded are summarized in the following list:

ESC	Completes the le name if possible; if not, completes as much as it can, and ashes the screen.
TAB	Shows all the le names that match what has been typed since the last blank, and erases what has previously been typed.
?	Like TAB, but does not erase anything.
CTRL-X	Retypes the command line with all le name patterns replaced by the list of le names they expand to.

22.11.3 Executive Commands

The Executive contains a number of facilities that can be invoked from the command line. The commands that invoke these can be identi ed by the extension character ‘‘~’’, which is illegal in a le name. Executive commands include the following:

Executive Commands

`Type.~ FileName`^{cr} Displays the contents of the named le(s) on the screen. After each page, it asks whether you want to see more of the current le. A `DEL` at this point terminates the entire `Type` command. A space (or other charactaer) will display the next page.

`Delete.~ FileName`^{cr}

Removes the named le from the directory and frees the disk space occupied by them. Typing `CTRL-C` will abort the command cleanly between deletions. Note: Be very careful when using this command. Its e ect cannot be reversed.

`Copy.~ DestFileName_ SourceFileName`^{cr}

Copies a le. If there are several `SourceFileName`s, the copy will contain the concatenation of the information in the source les in the order listed. Copying a single le preserves the creation date of the le; concatenating les generates a new creation date.

`Rename.~ OldFileName NewFileName`^{cr}

`Rename.~ NewFileName _ OldFileName`^{cr}

Changes the name of `OldFileName`. `NewFileName` must not already exist unless `OldFileName` and `NewFileName` are the same (di ering only in case).

`Quit.~`^{cr} Starts a diagnostic program which does not use the Operating System. This is done automatically after a machine has been idle in Executive for about 20 minutes. If `DMT.Boot` is not on the disk and the machine is connected to a communication net, `DMT` will be obtained from a remote source.

If the system is not connected to a PUP Ethernet with Boot servers, the screen goes white following a `Quit` command.

`Login.~`^{cr} Saves away the user's name and password for use by programs that interact with access-controlled resources (such as time-sharing or le systems).

`SetTime.~`^{cr} Sets the internal time-of-day clock. The time is obtained from the communications net (if available). Failing that, the user will be asked to supply manually the time (and possibly time zone) in the form `12-OCT-81 14:45`. Use `SetTime/m` to bypass the communications net and set time manually. Use `SetTime/z` to force setting of time zone in manual mode. (When the Executive is started, it examines the time-of-day clock. If the value is not reasonable, the Executive attempts to obtain the time from the communications net before proceeding. If the time cannot be obtained, the time-of-day displayed at the top of the screen will be ''Date and TimeUnknown'' indicating that the user should invoke the `SetTime.~` command manually.)

As a side e ect of obtaining the time from the communications net, the Executive learns the network number of the local communications net (if available) and displays it along with the machine's host address in one of the header lines at the top of the screen. A network number of 0 means ''I don't know.''

`Dump.~ DumpFile SourceFile1 SourceFile2`^{cr}

Writes `DumpFile` as a structured le (in ''Dump'' format) containing the names and data of all the `SourceFile`s. This is a convenient way of packaging up a collection of related les into a single composite le that can later be decomposed into its constituent parts.

`Load.~ DumpFile`^{cr} This reads through a Dump format le and creates individual les corresponding to its constituent parts. The `/V` switch causes `Load` to ask the user about each constituent part, whether to copy it from `DumpFile` to an individual le or not. Acceptable responses are `Y`, `N`, and `C`. The latter indicates that the le is to be copied, but into a le with a di erent name than that indicated. The user is then asked to supply the name of the new le.

`Release.~`^{cr} Tells the release number and date of Executive. The release number is also shown in the rst Executive herald line, just after the slash following ‘‘Xerox Alto Executive.’’

`Install.~`^{cr} Install is used for changing names and passwords. The program will ask the following questions. Provide the answers indicated:

Question	Answer
Do you want the long installation dialog?	`No`
What is your name?	<Enter your name>^{cr}
Please give your disk a name:	<Enter name>^{cr}
Do you wish to give your disk a password?	`Y` or `N`
Enter password (if answer is yes)	<Enter password>^{cr} *(note: password is echoed)*

`Scavenger.~`^{cr} This invokes the Scavenger program (described below) which attempts to recover from a variety of possible disk problems.

`FileStat.~ FileName`^{cr}
This command will tell several things about a le: its length in bytes; size in pages; serial number and disk address; and creation, read, and write dates. If `FileName` is of the form `octal/s` (or `octal1,octal2/s`) the le will be looked up by serial number rather than by name. This is useful if Scavenger or some other program gives a serial number without providing the name. The forms `octal/v` and `octal/r` tell about the les that own the speci ed virtual or real disk address.

`WriteDirectory.~`^{cr}
This command causes the Executive to write a sorted version of the directory back onto SysDir on the disk. Keeping the directory approximately sorted on the disk greatly reduces the time required for the Executive to sort it during initialization. The Executive will periodically perform a WriteDirectory in an attempt to keep the directory reasonably sorted. WriteDirectory also will compact the directory, collecting all the free space at the end and will report several statistics about directory usage.

Some other Executive commands are found on all Xerox 1100 systems but are usable only by those systems whose installation support a complete set of PUP communication net facilities.

22.11.4 Recovering From Disasters: The Scavenger

There are various ways in which a disk can become damaged. If this happens, the procedures described here will usually allow the user to recover the disk, or at worst will allow les to be copied from the sick disk to a healthy one. (Copying les from one disk to another assumes that the system has access to a communications net or that the system con guration allows communication between two Xerox 1100 systems.)

Here are the symptoms of trouble:

You can't boot the disk and get to the Executive.

You are out of disk space, but think you should have plenty; in other words, some disk space has apparently gotten lost.

You get an error message which says something about disk errors or le errors, and perhaps recommends that the Scavenger program be run.

You attempt to run a program that has previously functioned well and it fails.

The rst step is to run a program called Scavenger. If the disk is healthy enough to let you boot and use the Executive, invoke Scavenger by typing:

`Scavenger` ^{cr}

The program will ask the following questions. Provide the answers indicated:

Question	Answer
Do you want to change disks?	`No`
Ready to scavenge dual model 44 (14 sector) le system [con rm]	`Y` or^{cr}
May I alter your disk to correct errors?	`Y`

The Scavenger will now run for about 5 minutes. As it runs, it may ask you whether it is OK to correct ‘‘read errors.’’ If they are ‘‘soft transient’’ errors, answer `Y` (yes). When the Scavenger is done, it will list what it found. If it has succeeded in making the disk healthy, work can then proceed normally.

The Scavenger leaves all the material that it was not able to put into a recognizable le on a le called `Scavenger.Garbage$`, and it leaves a readable record of everything it did on another le called `Scavenger.Log$` (unless it says that you have a beautiful disk). There are two kinds of entries in `Scavenger.Log$`; names of les removed from the directory or otherwise modi ed, and names of le pages that were put into `Scavenger.Garbage$`. Delete the le `Scavenger.Garbage$` and `Scavenger.Log$` that the Scavenger leaves around. It is a good idea to go through this scavenging procedure once a month or so, just to keep disks in good shape.

22.12 RAID

Raid is a low-level debugger that knows about the Interlisp-D stack and certain Interlisp-D objects. It is principally a tool for those who maintain Interlisp-D and most users will not normally need its facilities. Raid is entered when Interlisp-D encounters an error (generally a system bug) that cannot be handled in a normal break. It can be entered explicitly by the `CTRL-C` interrupt.

Raid saves away the display and puts up its own. It prints a message indicating why it was called (e.g. ‘‘`CTRL-C` interrupt’’ or some error message), prints the prompt `@`, and awaits input of one of the single-letter commands described below.

If Raid is called unexpectedly, stop and read the error message. If the call occurred by typing `CTRL-C`, type `CTRL-N` to resume Interlisp-D transparently. Otherwise, a system bug probably was encountered. Do not type `CTRL-N` in this case, as the bug may compound itself. Use the `L` command to look at the Interlisp-D stack to see where you were, and the `F` command to look at individual frames. Once enough information has been learned about the context to report the error, try typing `CTRL-D` to reset the system (this clears the stack and starts over again at the top level). This may allow the system to continue long enough to save your work.

The Raid commands are described below in case the user wishes to use Raid deliberately. However, knowledge of the low-level implementation details may be necessary to make e ective use of Raid. The documentation provided here is not intended to be exhaustive, as Raid and the low-level system implementation are subject to change.

22.12.1 Using RAID

This section describes the Raid commands in more detail, should the user wish to explore the context of an error further. Typing ‘‘`?`’’ to Raid prints out a brief list of the commands.

Raid can be called from system errors and with the `CTRL-C` interrupt. (Note: The `CTRL-C` interrupt can be disabled with `(INTERRUPTCHAR (CHARCODE ^C) NIL)`, or assigned to `CTRL-Z` with `(INTERRUPTCHAR (CHARCODE ^Z) 'RAID)`. See page 9.19.)

Raid can also be entered by calling the following function:

`(RAID MESS1 MESS2 )` [Function]

Calls the Raid debugger, displaying the message `MESS1` (followed by a carriage return and `MESS2` , if it is speci ed).

22.12.1.1 How Information is Displayed

Addresses

Raid knows about the form of lists, atoms, strings, and integers, and prints them in roughly the manner expected. All other Interlisp-D structures, such as arrays and datatypes, print out as the address of the datum. An Interlisp-D address consists of a `{segment, wordo{set}` pair, where `segment` identi es the segment (high eight bits) which often indicates what kind of address it is (e.g. value space, statistics space, stack space, etc.) and `wordo{set` is a 16 bit word o set. These are usually printed by Raid as two unsigned octal numbers enclosed in braces.

Stack frames

Raid displays stack frames as a series of two-word pairs. Each two-word pair appears on a single line with the halves separated by spaces, and some interpretation (e.g., variable names) to the right.

22.12.1.2 Commands

Each command is listed by the letter that invokes it, followed (in { }) by the parameters that the command expects and a brief description of its behavior. Parameters are denoted:

`address`	Two octal numbers separated by space
`atom`	An unquoted character string
`con|rm`	Command requires con rmation (a carriage return)
`dnum`	A decimal number
`onum`	An octal number
`CTRL-B` {`address`,`onum`}	Displays `onum` bytes starting at `address`
`CTRL-D` {`con|rm`}	Returns from Raid, ushing computation to top level. This is similar, but not identical to the Interlisp-D `CTRL-D` interrupt. It clears all stack pointers. This cannot be done when interrupts are o (e.g. if one entered Raid by typing `CTRL-C` and Raid's message includes ‘‘Warning: interrupts o ’’), as the world might be left in an inconsistent state.
`CTRL-F` {`onum`}	Displays the basic stack frame at stack o set `onum`.
`CTRL-K` {`con|rm`}	Kills Interlisp-D and returns to the Exec. The virtual memory image on the disk will probably not be resumable.
`CTRL-N`	Returns from Raid, resuming the Interlisp-D computation with `NIL` as the value of the call on Raid.
`CTRL-S` {`con|rm`}	Calls Swat (the Bcpl debugger).
`CTRL-T`	Returns from Raid, resuming the Interlisp-D computation with `T` as the value of the call on Raid.
`CTRL-U`	Displays the Interlisp-D screen in place of the Raid screen. The display will remain coupled to the Interlisp-D display until any character is typed.
`CTRL-V` {`atom`}	Sets the top-level value of the speci ed atom to `NIL`. This should be used only as a last resort, since it is *NOT* completely equivalent to `(SETTOPVAL 'atom NIL)`, and, of course, it is not undoable.

`CTRL-X` {onum}	Displays the frame extension at stack offset onum.
`+` {onum, onum}	Adds the two octal numbers and displays the result in octal.
`,` {onum, onum}	Displays a 16-bit word whose bytes are the onums.
`_` {address, onum}	Sets the contents of the 16-bit word at address to onum.
`?`	Displays a command summary.
`A` {atom}	Displays the top level value of atom.
`B` {address, onum}	Displays the contents of the cells starting at address and proceeding for onum words.
`C`	Displays a count, by segment, of how many virtual pages are resident, plus some summary information.
`D` {atom}	Displays definition cell of atom.
`E`	Reprints the error message and object on which Raid was called.
`F` {dnum, `A` or `C` if the access and control links differ}	Displays the basic frame and extension for the function activation which is dnum deep along either the stack's access or control links, as specified by `A` or `C`. Thus, `F3 A` displays the basic frame for the third function call along the access chain from the top level. dnum can most easily be ascertained by displaying the stack entries using `L`.
`J` {dnum}	Sets the maximum length of the list Raid will print before abbreviating the tail.
`L` {`A` or `C` if the access and control links differ}	Displays the Interlisp-D stack, one line per function. `LA` follows the access links; `LC` follows the control links. The frames can be examined in more detail using `F`.
`N` {dnum}	Sets the maximum depth to which Raid will recurse while printing list structure before abbreviating.
`P` {atom}	Displays the property list of atom.
`S` {onum1, onum2}	Displays onum2 words from the stack starting at offset onum1.
`U` {dnum}	Sets the radix in which Raid displays numeric Interlisp-D data items to dnum.
`V` {address}	Displays the contents of the virtual address as a Interlisp-D object.

22.13 SWAT

Swat is the debugger for Bcpl, the language in which Raid and some of the Interlisp-D system primitives are written. Swat is entered as a result of an unexpected error condition arising in the Bcpl, and generally indicates that the Lisp world is in very bad shape.

When the system is in Swat, the Swat screen is displayed, with the prompt #. To display the Interlisp-D screen, type `CTRL-U`. That will restore the Interlisp-D screen until the `SWAT` key (the lowest of the three blank keys on the right side of the keyboard) is depressed.

If the system goes into Swat, the Interlisp-D image probably cannot be continued, at least not for long, but a salvage attempt can be tried by typing `RAID<CTRL-C>`. This invokes Raid, from which the stack can be examined, and `CTRL-D` can be tried to reset the system. If the situation is beyond repair, type `CTRL-K` to kill the program and return to the Executive.

Swat can also be entered from Raid via the `CTRL-S` command, in which case type `CTRL-P` to return to Raid.

In some circumstances, the Lisp system may appear dead; `CTRL-C` does not call Raid, but the cursor still tracks the mouse. In this case, the system state can perhaps be saved by deliberately invoking Swat by typing `CTRL-SHIFT-SWAT`. Once inside Swat, type `RAID<CTRL-C>` to get into Raid and proceed as above.

It is possible that Swat will respond to an attempt to call Raid (using `RAID<CTRL-C>`) with the message ‘‘Stack dubious. Do you want to proceed with the call?’’. Respond with `N` (for No!). Next, type `"TOPLEVELFRAME<CTRL-O>"`; this prints a number; then type, `"AC2<CTRL-O>nnnnn<CR>"`, where `nnnnn` is the number printed in the previous step. Finally, try to call Raid again.

The very rst time Swat is entered, load the symbol le. The symbol le name is shown in the lower Swat window. If it is not ‘‘Lisp.syms,’’ type `<CTRL-Y>Lisp<CR>` to load the correct le.

22.14 XEROX 1100 BOOTING AND MAINTENANCE PANEL CODES

The Xerox 1100 processor cabinet contains the electronic circuitry, memory, and magnetic disk of the 1100. The only controls on the front panel of the processor cabinet are two pushbuttons used to turn the power on and o , and a numeric display (the ‘‘Maintenance Panel’’ or ‘‘MP’’) used for maintenance purposes.

This section intends to provide enough information to diagnose malfunctions to the extent that MP codes permit. These codes are mainly of use when booting the machine; usually they mean nothing during an Interlisp-D session. Ordinarily, error conditions in Interlisp-D cause at worst a recoverable (by typing `CTRL-D`) call to Raid. If something is seriously wrong, however, Interlisp-D halts, the screen loses synch and the code in the MP may be signi cant; in these cases Interlisp-D is not continuable.

22.14.1 1100 Booting Sequence

When an 1100 is booted, the following steps occur before you see the Alto Executive:

(1) Hardware loads the Boot microcode from an EPROM;

(2) Boot tests the processor and loads Initial;

(3) Initial tests the map and storage and loads an emulator;

(4) Emulator microcode initializes itself;

(5) Alto Executive or NetExec is booted.

A bootstrap is generally initiated by pushing and releasing the start (power-on) button, by pushing the keyboard boot button, or by executing the Boot function in a microinstruction. In addition to these normal methods of booting, if your machine is malfunctioning, it might spontaneously boot itself when a fault happens while the fault task is running. If the problem is really bad, then it might boot itself over and over.

Some de ciencies in error handling by the Initial microprogram cause certain classes of recoverable hardware failures (disk and ethernet problems) to reboot the machine rather than recovering from the error.

If your machine won't boot or boots very slowly, it is important to go through the following check list:

(1) If you have just powered up, make sure your display is turned on sometimes the emulator won't run if the display is powered o , and you obviously won't be able to see anything.

(2) Sometimes the keyboard microcomputer will power on in a bad state; if this happens, you can have all kinds of trash happen on the backchannel erroneous keystrokes, mouse button clicks, and mouse movement. To x this, push the keyboard boot button. Sometimes, it is necessary to turn the power to the display o and then on again.

(3) Otherwise, watch the MP while the problem is happening. The detailed sequence of numbers may indicate what is going wrong. You will have to get your head down low to observe the numbers on the MP reliably; people have frequently reported numbers with 1's translated into 7's, and some other observation errors are occasionally made.

22.14.2 Hardware Boot

While you depress the start button, the hardware shows 8888 in the MP as a light test. When you release the power-on button, the machine will then run through the boot sequence discussed above. During the hardware boot, you might see either 8808 or 8880 in the MP if RM or IMX parity errors are detected by the hardware during loading.

22.14.3 EPROM Boot Microcode

Boot is a tiny diagnostic and bootstrap loader. Its function is to test the processor thoroughly but quickly, reporting any failures on the MP, and then boot the Initial microcode from any of the I/O devices which might reasonably contain it; these are the SA4000 disk and the 3 Megabit Experimental Ethernet at present; 10 Megabit Ethernet booting will be made available at a future date.

Boot runs a few processor tests to nd out whether or not the processor is healthy enough to continue loading. Many machines malfunction when rst powered up, then work correctly. If the processor tests fail, Boot will show an error MP code (0000 to 0039) for a second or so and then reboot. Otherwise, registers are initialized.

Boot tries to read the rst program on the SA4000 boot record (normally Initial) directly into the microstore. When Boot starts this, it will show 0040 in the MP. If you see this (or 0041 or 0046 which frequently follow immediately), your processor is at least somewhat alive (Initial lives in a special reserved portion of the disk, so you won’t see it among your disk les.)

If the disk won’t work (0041 to 0045, 0047 to 0048) or isn’t ready yet (0046), Boot will try to obtain Initial.Eb from the (3MB) Ethernet; when this decision is made, a one second pause allows you to read the MP; if the reason for the boot is NOT a button push, the delay is extended to one minute to prevent a sick machine from hogging the boot servers. When Ether booting starts, 0060 (trying to Etherboot) will appear in the MP; other 006x MP codes indicate Etherboot problems. While waiting for a boot server to respond and while transmitting Initial microcode from the boot server, 004x slowly alternates with 006x, so that you can see both the reason why the disk boot failed and the current Etherboot indication.

Unfortunately, many machines experience a short period of unreliability after being powered on, but then work correctly. These machines encounter the one minute wait intended to prevent a sick machine from hogging the boot servers, and this can be frustrating, if you are waiting for the machine to become ready. If your machine does this, keep pushing and releasing the start button until you see a healthy 0060.

The standard trick for forcing an Etherboot is to turn power o and then back on. It takes the disk about two minutes to become ready again. If you push the start button before the disk is ready, you should get to the Alto NetExec. Note that Initial is loaded directly into the microstore without using either the map or storage.

22.14.4 Initial Microcode

Initial is primarily responsible for testing and initializing the map and storage, reporting any failures in the MP, then loading and starting an emulator. When Initial receives control, it puts 0700 (’starting map test’) into the MP; if you see 0040 then 0700 (without an intervening 0060), your disk is at least somewhat healthy since Initial was loaded from the disk. Initial rst tests the map; it will hang with the ’bad map’ MP code (0702) displayed, if the map is imperfect.

Then Initial tests storage and uses only ’good’ pages. 0700 is visible barely long enough to see, and 0400 is seen in the MP during storage testing, which lasts less than 4 seconds with eight perfect 96k-word storage boards.

If Initial detects any storage imperfection, it will do additional testing, and 0400 will be shown for 4 to 9 seconds (timing approximately proportional to the amount of storage). If the number of pages with correctable failures exceeds 1/8 of all pages, and if the amount of good storage is less than 128k words, then the entire test will be repeated allowing the pages with only soft failures to be used; otherwise, only perfect pages are used. After all testing is complete, three numbers will be shown in the MP for about 1.3 seconds each: (1) 0400+2^n, where board n+1 is imperfect, (2) the count of ‘‘hard’’ bad pages (uncorrectable failures), and (3) the count of ‘‘soft’’ bad pages (correctable failures only). These numbers are NOT shown when all storage boards are perfect.

Even when some storage is bad, unless the amount of ’good’ storage is reduced by failures to less than 64k words, initialization will continue normally following the bad-page MP codes.

On a disk boot, Initial then puts 0720 into the MP and continues reading the SA4000 boot record. But this time the emulator from the boot record is placed into storage rather than directly into the microstore. On an ether boot, it instead shows 0758 in the MP and reads Alto.Eb from the Ethernet into storage. At the end of le, the microcode image in storage is loaded into the microstore and started.

[Note: Older versions of Initial had di erent code sequences.]

22.14.5 Alto Emulator Microcode

Early in initialization the emulator shows the MP code 'Start device init' (0104) barely long enough to see. Seeing 0104 means that Initial tested and zeroed storage, loaded the microcode image into storage from the disk or Ethernet and then from storage into the microstore, transferred control to it, and executed at least the rst few microinstructions successfully.

The emulator has two main entry points. After 0104, on a disk boot, 0118 (GotBreathOfLife) is shown for 0.3 seconds after the rst page from the disk boot record has been read successfully. On an ethernet boot, 0114 (start booting the NetExec) is shown, then 0118 when the ‘‘breath-of-life’’ program has been received from the Ethernet boot server. Finally, on both disk and ethernet bootstraps, the emulator shows the number of good pages found by Initial in the MP. For the 96k-word storage boards using 16k RAMs, the MP will show 384 times the number of storage boards (i.e., 0768, 1152, 1536, 1920, 2304, or 2688). Some MPs will show this value plus 1 occasionally for unknown reasons. Then the emulator loads the nal microcode overlay and starts the emulator at the breath-of-life program's disk boot or Ether boot address.

When started at its normal entry point, the emulator will boot the OS from SA4000 partition 1 if you have no keys down. If you depressed the ‘‘0’’, or backspace key and then pushed the keyboard boot button when your emulator was correctly running the display task, then the OS will be booted from SA4000 partition 2 (‘‘0’’typed) or the NetExec from the Ethernet (BS typed). If none of these keys was depressed, or if some other keys were also depressed, then a partition 1 disk boot occurs.

Pushing the keyboard boot button does nothing if an emulator isn't running.

22.14.6 Normal Maintenance Panel Code Sequences

If your hardware is working properly, the viewable MP sequence if your disk is not yet up to speed is:

8888, [46, 60], 700, 400, 758, 104, 114, 118, `NPages`,

where ‘‘46, 60’’ may repeat several times before continuing with the rest of the sequence, if your boot server is busy. Other MP codes are not up long enough to see unless something goes wrong; 700 is up barely long enough to see.

If you have just powered up your machine, it might get a 22 or 24. After a second or so, your 1100 will automatically reboot, but this time 46 will stay in the MP for a minute or so. If you get tired of waiting, poke the button again.

The viewable MP sequence for a ‘‘normal’’ disk boot is:

8888, 40, 700, 400, 720, 104, 118, `NPages`.

22.14.7 Maintenance Panel Codes

In the following pages, a # indicates a nal MP code. The machine will hang with this number in the

MP until you boot again. All other MP codes are either errors that will be retried automatically or simple indications of progress.

22.14.7.1 MP Codes from the hardware

8888	Lamp test (shown while the start button is down).
8808#	RM parity error. The MP freezes with this code, indicating (??) a failure during the hardware boot.
8880#	IMX or control store parity error. The MP freezes with this code indicating (??) a failure of the hardware boot.

22.14.7.2 MP codes from Boot microcode

0000	One of the rst instructions in the EPROM clears the MP; seeing this number means that the ALU all-zeroes, all-ones, or GoTo tests failed.
0001	The ALU all-zeroes, all-ones, and GoTo tests passed. You should never see this since some other error should happen or the MP should change to 0040 when disk booting starts.
0002	Midas boot.
0004 to 0015	One of the preliminary ALU tests failed.
0016	Mismatch after write then read of an RM register via the stack.
0017	The contents of an RM register have changed.
0018	Register read and compare error using the Stack.
0020	A Breakpoint microinstruction was executed.
0021	Memory error. Since the EPROM code doesn't touch the map or storage, this is probably an H4 Parity error.
0022	RM register parity error.
0024	Control Store parity error.
0028	Stack error.

0020 to 0035 indicates that a fault happened. The MP contains 20 plus the contents of the Parity register. A fault in the fault handler will reboot the machine, so you may not get to see these codes. The value shown is 20d + (1 if memory error) + (2 if RM parity error) + (4 if control store parity error) + (8 if stack over ow or under ow).

0022 and 0024 are to be expected if you have just powered up your machine. (The bias on the RAM chips hasn't been pumped up yet.) This will invoke a one minute delay to avoid hogging the boot servers when something is broken. Poke the button again if you want faster service.

0040	Starting to load microcode from disk.
0041	Can’t nd disk. (Will now try to EtherBoot)
0042	SA4000 disk read error.
0043	SA4000 seek timed out.
0044	SA4000 disk checksum failure.
0045	SA4000 bad Control Store address attempt to load into EPROM area.
0046	SA4000 disk not ready. (Will now try to EtherBoot)
0047	The label word which should contain a link to the next page of microcode to be loaded has an invalid disk address.
0048	Didn’t load microcode from disk within 1 second. (Will now try to EtherBoot)

Most disk errors (0042, 0043, 0044, 0045, 0047) can be caused by simple transient read problems. The disk task simply retries all of them while the emulator task is counting down a timer. If the timer runs out, you will see 0048.

0060	Trying to load microcode via Ethernet.
0061#	Can’t nd Ethernet board.
0062	Bad Ethernet checksum while reading microcode.
0063	Bad Control Store address attempt to load into EPROM area.
0064	Hardware error (bad status) at end of packet.
0065#	Timeout after 15 tries at about 10 seconds each.

If EtherBooting doesn’t work, the MP will slowly alternate between 006x and 004x so that you can see both what was wrong with the disk and what is wrong with the Ethernet. If the Etherboot eventually times out, you will see 0065 alternating with the bad disk code.

0070 to 0085	An unexpected wakeup happened. The MP contains 70 plus the number of the o ending task (see list of tasks later in this section). If you get one of these, one of the IO controllers is probably broken. For example, its reset logic is not working, or the wakeup logic on the disk or Ethernet board is generating the wrong task number.

22.14.7.3 MP codes from Initial

0700	Start map test.
0702#	Bad map.

0400	Start storage test.

If any bad storage boards detected, the following three numbers are displayed for about 1.3 seconds each:

0401 to 0655	Failures detected on one or more boards. MP shows 0400 + (1 if the memory board in slot 5 is imperfect) + (2 if slot 6) + (4 if slot 7) + (8 if slot 8) + (16 if slot 9) + (32 if slot 10) + (64 if slot 11) +(128 if slot 12). This interpretation is for the 96k-word storage boards with 16k RAMs. Each such board occupies 128k words of real address space but implements only the rst 96k words of the space. Hence, the bits actually represent imperfections in the existing storage of each 128k-word bank of storage.
`NHardBad`	Count of hard bad pages (ones with uncorrectable failures), where pages are 256 16-bit words; shown after 0401-0655.
`NSo$Bad`	Count of soft bad pages (only correctable failures); shown after NHardBad.
0701#	Not enough memory. (Initial requires 64K words.)
0720	Starting to load emulator microcode from disk.
0721#	No disk.
0722#	Disk Read error.
0741#	Can't nd UTVFC (display).

When Initial starts Etherbooting, it puts 740+2* `b|` into the MP. (`b|` is related to the boot le number). If EtherLoad can't load that le within a reasonable length of time it will give up and bump the MP by one.

0758	Trying to load AltoD0.Eb from the Ethernet.
0759#	Timeout trying to load AltoD0.Eb
0812	Label check from SA4000. (Try to EtherBoot.)

22.14.7.4 MP codes from Emulator Microcode

These codes are shown when the (Alto) emulator microcode starts up.

0101#	Not enough memory. (You need 512 pages or 128K words more than the 64k words required by Initial.)
0104	Start device initialization.
0105	Started UTVFC initialization (invisible).
0106	Finished loaded keyboard overlay (invisible only happens on systems).
0107	Finished display initialization (invisible).

0110	Started disk boot (invisible).
0111#	Timeout waiting for disk status.
0112#	Hardware error reading disk (after 10 retries).
0114	Start booting the NetExec.
0118	Breath-of-life read successfully from disk or Ethernet.
`NPages`	The number of 'good' 256-word pages determined by Initial is put in the MP just before the nal microcode overlay overwrites initialization; it normally remains in the MP until you boot, crash, or start Interlisp-D. The emulator starts immediately after loading the nal overlay.

22.14.7.5 Fault MP Codes

[The old Initial has di erent fault MP codes.]

0115	Unexpected Ethernet output task wakeup.
0117	Two MC2 errors; both MC2A and MC2B pipes indicate errors; since LogSE is illegal, this might mean two consecutive references experienced uncorrectable storage failures. Alternatively, this is caused by microcode bug.
0119	Stack over ow or under ow. This is a microcode bug.
0120 to 0135	RM or CS parity error, possibly in combination with other errors. MP code is 120 + (1 if MC1 or MC2 error) + (2 if RM parity error) + (4 if CS parity error) + (8 if stack over ow or under ow).

Many codes show a multiple of 20d + a task number, where the task assignments are given below. The 'pipe task' is the one issuing the reference causing an error this can determined from the error pipe; the 'current task' is the one running when the fault aborted execution.

Task assignments:

0	emulator
5	color display
6	3 Megabit Experimental Ethernet output
7	10 Megabit Ethernet, or 3 Megabit Experimental Ethernet input
8	disk (SA4000 controller)
10	Display
14	timer task
15	fault task

Fault MP Codes

The following codes imply no RM or CS parity error.

0136	MC12 error occurred but none of the reasons for it is indicated; i.e., neither H4PE, MOB, MC1A, MC1B, MC2A, nor MC2B errors are true. Conceivably, this can be caused by an MC1 fault on the reference following a PFetch4, if the PFetch4 experiences error correction. Due to a hardware bug, the fault isn't started soon enough in this case, so an extra non-fault-task instruction is executed. If the extra instruction is a reference it wipes out the MC1ErA and MC1ErB indicators.
140 to 155	Map out of bounds (MOB) indicating virtual address greater than 22d bits. Code shown is 140 + current task. An MOB crash can't happen for Interlisp-D because MOB errors fault to software.
160 to 175	H4 parity error indicating bad parity on the processor bus used by Input, IOStore4, and IOStore16 references. Code shown is 160 + current task. This can never happen at present since these errors are ignored for all tasks.
180 to 195	Some fault when the preceding instruction contains a LoadPage and the fault handler decides to continue execution. This indicates a microcode bug and should be reported.
200 to 215	MC2 crash, indicating correctable storage failure of PFetch1, 2, or 4 with LogSE true in the map entry or an uncorrectable storage failure on any reference. The code shown is 200 plus the pipe task. Since LogSE is presently illegal, this code should indicate an uncorrectable storage failure; some microcode bugs may cause it.
220 to 235	MC1 crash, indicating a page or write protect violation. The code shown is 220 plus the pipe task. This can't happen for Interlisp-D because MC1 faults are passed to software; MC1 faults always crash for Alto.
240 to 255	SetFault or Breakpoint crash. The code shown is 240 plus the current task. This is used by the microcode in a few places when impossible conditions are detected; for unused tasks, it represents an unexpected wakeup.