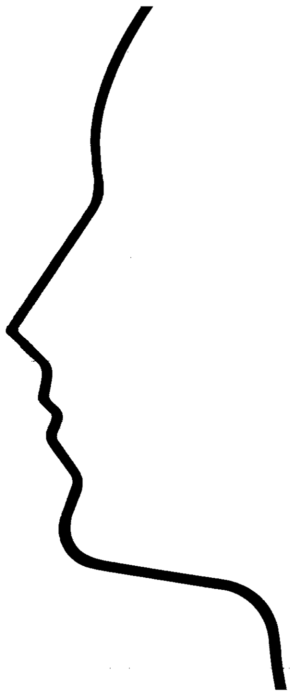
TEXAS INSTRUMENTS

microExplorer™

Development Software User's Guide



microExplorer™ DEVELOPMENT SOFTWARE USER'S GUIDE

MANUAL REVISION HISTORY

microExplorer™ Development Software User's Guide (2552702-0001)

Original Issue	April 1988
Revision A	December 1988
Revision B	September 1989
Revision C	October 1990

Produced by the Publishing Center
Texas Instruments Incorporated
Information Technology Group
Austin, Texas

THE EXPLORER™ SYSTEM SOFTWARE MANUALS

Little/No Interest	Medium Interest	Required
░	▥	▓

	First Day of Explorer Use	Casual or New Developer	Experienced Developer	Applications Programmer	Systems Manager
Introduction to the Explorer System	▓	▓	░	░	▥
Zmacs Editor Tutorial	▓	▥	░	░	░
Lisp Reference	▥	▓	▓	▓	▥
Input/Output Reference	▥	▓	▓	▓	▥
Tools and Utilities	▥	▓	▥	▓	▓
Zmacs Editor Reference	░	▓	▥	▥	░
Window System Reference	░	▥	▓	▓	░
Programming Concepts	▥	▓	▥	▥	▥
Networking Reference	▥	▥	▥	▥	▓
microExplorer User's Guide	▓	▥	░	░	░
microExplorer Development Software User's Guide	▥	▓	▓	▓	▥
System Software Installation	▥	░	░	░	▓
System Software Design Notes	░	░	▥	▥	░

Little/No Interest	Medium Interest	Required
░	▥	▓

	First Day of Explorer Use	Casual or New Developer	Experienced Developer	Applications Programmer	Systems Manager
Release Notes	▥	▥	▥	▥	▓
SLE X Window System Programmer's Reference	░	▥	▓	▓	░
SLE Common Lisp Object System	░	▥	▓	▓	░

THE EXPLORER™ SYSTEM SOFTWARE MANUALS

Mastering the Explorer Environment	Introduction to the Explorer System	2243190-0001
	Explorer Zmacs Editor Tutorial	2243191-0001
	Explorer Networking Reference	2243206-0001
	Explorer Diagnostics	2533554-0001
	Explorer System Software Installation Guide	2243205-0001
	microExplorer™ User's Guide	2552701-0001
Programming With the Explorer	Explorer Programming Concepts	2549830-0001
	Explorer Lisp Reference	2243201-0001
	Explorer Input/Output Reference	2549281-0001
	Explorer Zmacs Editor Reference	2243192-0001
	Explorer Tools and Utilities	2549831-0001
	Explorer Window System Reference	2243200-0001
	microExplorer Development Software User's Guide	2552702-0001
	Release Information, Explorer System Software	2549844-0001
	Explorer Software Release Information and Installation Guide for microExplorer Delivery Software	2555022-0001
	SLE X Window System™ Programmer's Reference	2553272-0001
	SLE Common Lisp Object System	2559293-0001
Explorer Options	Explorer Natural Language Menu System User's Guide	2243202-0001
	Explorer Relational Table Management System User's Guide	2243203-0001
	Explorer Grasper User's Guide	2243135-0001
	Explorer TI Prolog User's Guide	2537248-0001
	Programming in Prolog, by Clocksin and Mellish	2249985-0001
	Explorer TCP/IP User's Guide	2537150-0001
	Explorer LX™ User's Guide	2537225-0001
	Explorer LX System Installation	2537227-0001
	Explorer NFS™ User's Guide	2546890-0001
	Explorer Communications Interface to DECnet™ User's Guide	2537223-0001
	Explorer Software Release Information and Installation Guide for microExplorer Source Option	2555023-0001
	Explorer Software Release Information and Installation Guide for microExplorer Development Software	2555024-0001
	Explorer SNA II	2537281-0001
	Personal Consultant™ Plus Explorer	2537259-0001
System Software Internals	Explorer System Software Design Notes	2243208-0001

THE EXPLORER SYSTEM HARDWARE MANUALS

System Level Publications	Explorer 7-Slot System Installation	2243140-0001
	Explorer System Field Maintenance	2243141-0001
	Explorer System Field Maintenance Documentation Kit	2243222-0001
	Explorer System Field Maintenance Supplement	2537183-0001
	Explorer System Field Maintenance Supplement Documentation Kit	2549278-0001
	NuBus™ Systems Explorer/Explorer LX Field Maintenance Handbook	2553315-0001
	Explorer NuBus System Architecture General Description	2537171-0001
System Enclosure Equipment Publications	Explorer 7-Slot System Enclosure General Description	2243143-0001
	Explorer Memory General Description (8-megabytes)	2533592-0001
	Explorer 32-Megabyte Memory General Description	2537185-0001
	Explorer Processor General Description	2243144-0001
	68020-Based Processor General Description	2537240-0001
	Explorer II™ Processor and Auxiliary Processor Options General Description	2537187-0001
	Explorer II Plus™ Processor General Description	2553312-0001
	Explorer System Interface General Description	2243145-0001
	Explorer Color System Interface Board General Description	2537189-0001
	Explorer NuBus Peripheral Interface General Description (NUPI board)	2243146-0001
	Computer Enclosure Installation and Operation	2557942-0001
Display Terminal Publications	Explorer Display Unit General Description	2243151-0001
	CRT Data Display Service Manual, Panasonic code number FTD85055057C	2537139-0001
	Explorer Color Console General Description	2537195-0001
	TRINITRON™ Graphic Display Monitor GDM-1603 Service Manual, Sony™ part number 0-558-986-01	2551107-0001
	Model 924 Video Display Terminal User's Guide	2544365-0001
143-Megabyte Disk/Tape Enclosure Publications	Explorer Mass Storage Enclosure General Description	2243148-0001
	Explorer Winchester Disk Formatter (Adaptec) Supplement to Explorer Mass Storage Enclosure General Description	2243149-0001
	Explorer Winchester Disk Drive (Maxtor) Supplement to Explorer Mass Storage Enclosure General Description	2243150-0001
	Explorer Cartridge Tape Drive (Cipher) Supplement to Explorer Mass Storage Enclosure General Description	2243166-0001
	Explorer Cable Interconnect Board (2236120-0001) Supplement to Explorer Mass Storage Enclosure General Description	2243177-0001

Explorer II, Explorer II Plus, and NuBus are trademarks of Texas Instruments Incorporated.

TRINITRON and Sony are trademarks of Sony Corporation.

143-Megabyte Disk Drive Vendor Publications	XT-1000 Service Manual, 5 1/4-inch Fixed Disk Drive, Maxtor Corporation, part number 20005 (5 1/4-inch Winchester disk drive, 112 megabytes)	2249999-0001
	ACB-5500 Winchester Disk Controller User's Manual, Adaptec, Inc., (formatter for the 5 1/4-inch Winchester disk drive)	2249933-0001
1/4-Inch Tape Drive Vendor Publications	Series 540 Cartridge Tape Drive Product Description, Cipher Data Products, Inc., Bulletin Number 01-311-0284-1K (1/4-inch tape drive)	2249997-0001
	MT01 Tape Controller Technical Manual, Emulex Corporation, part number MT0151001 (formatter for the 1/4-inch tape drive)	2243182-0001
	Viper™ Half-High Intelligent 4 1/4-Inch Streaming Cartridge Tape Drive SCSI Models 2060S and 2125S, Archive Corporation, part number 21136-001	2551106-0001
182-Megabyte Disk/Tape Enclosure MSU II Publications	Mass Storage Unit (MSU II) General Description	2537197-0001
182-Megabyte Disk Drive Vendor Publications	Control Data™ WREN™ III Disk Drive OEM Manual, part number 77738216, Magnetic Peripherals, Inc., a Control Data Company	2546867-0001
515-Megabyte Mass Storage Subsystem Publications	SMD/515-Megabyte Mass Storage Subsystem General Description (includes SMD/SCSI controller and 515-megabyte disk drive enclosure)	2537244-0001
515-Megabyte Disk Drive Vendor Publications	515-Megabyte Disk Drive Documentation Master Kit (Volumes 1, 2, and 3), Control Data Corporation	2246129-0002
	Volume 1, General Description, Operation, Installation and Checkout, and Part Data	2246125-0004
	Volume 2, Theory, General Maintenance, Trouble Analysis, Electrical Checks, and Repair Information	2246125-0005
	Volume 3, Diagrams	2246125-0006
1/2-Inch Tape Drive Publications	MT3201 1/2-Inch Tape Drive General Description	2537246-0001
380-Megabyte Disk/Tape Enclosure MSU IIA Publications	Mass Storage Unit (MSU IIA) Installation and Operation Kit	2557935-0001

Viper is a trademark of Archive Corporation.

Control Data and WREN are trademarks of Control Data Corporation.

380-Megabyte Disk Drive Vendor Publications	XT-4000S Product Specification and OEM Manual, Maxtor Corporation, part number 1014995 (DB380 Disk Drive)	2554653-0001
1/2-Inch Tape Drive Vendor Publications	Cipher CacheTape™ Documentation Manual Kit (Volumes 1 and 2 With SCSI Addendum and, Logic Diagram), Cipher Data products	2246130-0001
	1/2-Inch Tape Drive Operation and Maintenance (Volume 1), Cipher Data Products	2246126-0001
	1/2-Inch Tape Drive Theory of Operation (Volume 2), Cipher Data Products	2246126-0002
	SCSI Addendum With Logic Diagram, Cipher Data Products	2246126-0003
Printer Publications	Model 810 Printer Installation and Operation Manual	2311356-9701
	Omni 800™ Electronic Data Terminals Maintenance Manual for Model 810 Printers	0994386-9701
	Model 850 RO Printer User's Manual	2219890-0001
	Model 850 RO Printer Maintenance Manual	2219896-0001
	Model 850 XL Printer User's Manual	2243250-0001
	Model 850 XL Printer Quick Reference Guide	2243249-0001
	Model 855 Printer Operator's Manual	2225911-0001
	Model 855 Printer Technical Reference Manual	2232822-0001
	Model 855 Printer Maintenance Manual	2225914-0001
	Model 860 XL Printer User's Manual	2239401-0001
	Model 860 XL Printer Maintenance Manual	2239427-0001
	Model 860 Xl Printer Quick Reference Guide	2239402-0001
	Model 860/859 Printer Technical Reference Manual	2239407-0001
	Model 865 Printer Operator's Manual	2239405-0001
	Model 865 Printer Maintenance Manual	2239428-0001
	Model 880 Printer User's Manual	2222627-0001
	Model 880 Printer Maintenance Manual	2222628-0001
	OmniLaser™ 2015 Page Printer Operator's Manual	2539178-0001
	OmniLaser 2015 Page Printer Technical Reference	2539179-0001
	OmniLaser 2015 Page Printer Maintenance Manual	2539180-0001
	OmniLaser 2108 Page Printer Operator's Manual	2546348-0001
	OmniLaser 2108 Page Printer Technical Reference	2546349-0001
	OmniLaser 2108 Page Printer Maintenance Manual	2546350-0001
	OmniLaser 2115 Page Printer Operator's Manual	2546344-0001
	OmniLaser 2115 Page Printer Technical Reference	2546345-0001
	OmniLaser 2115 Page Printer Maintenance Manual	2546346-0001
Communications Publications	990 Family Communications Systems Field Reference	2276579-9701
	EI990 Ethernet™Interface Installation and Operation	2234392-9701
	Explorer NuBus Ethernet Controller General Description	2243161-0001
	Communications Carrier Board and Options General Description	2537242-0001

Cache Tape is a trademark of Cipher Data Products, Inc.

Omni 800 and OmniLaser are trademarks of Texas Instruments Incorporated.

Ethernet is a trademark of Xerox Corporation.

CONTENTS

ABOUT THIS MANUAL

Introduction

The *microExplorer™ Development Software User's Guide* describes the Development System Software of the microExplorer.

Three microExplorer software packages are available from Texas Instruments:

- The Delivery configuration (also called the System Software) resembles a Lisp kernel that allows you to run Common Lisp-based applications on an Apple™ Macintosh™ II. For more information about the System Software, see paragraph 1.2, System Software, and the *microExplorer User's Guide*.
- The Development System Software contains all of the System Software, as well as additional software that allows you to *develop* your own applications for execution on the microExplorer or on a Texas Instruments Explorer™ system.
- The Network System Software incorporates additional network functions into the development system. Note that this manual applies to both the Development System Software and the Network System Software.

NOTE: The installation procedures for the microExplorer hardware are described in the *microExplorer User's Guide*.

Audience

The *microExplorer Development Software User's Guide* is intended for people who will be writing programs that run on the microExplorer. The manual assumes that a developer has some experience writing application programs in a high-level language. Familiarity with the Lisp language is considered to be a benefit, but is not absolutely a requirement for using the Development System Software. The documentation provided with this software describes Lisp in detail.

The developer is assumed to be generally familiar with the Macintosh Finder operating environment.

NOTE: Software installation procedures can be performed by all levels of users. However, if you have purchased the networking option with your microExplorer, have your system administrator handle all the tasks associated with configuring the network.

Contents of This Manual

This manual includes the following sections and appendixes:

Section 1: Development System Software Overview — This section tells what the Development System Software provides, lists its components, and describes the minimum requirements needed to use the software. This section also briefly describes the System Software. The section tells which of the Explorer system's tools and facilities are not applicable in the microExplorer environment. Lastly, the section describes the hardware/software options available for the microExplorer.

Section 2: Launching — This section tells how to activate the microExplorer Development System Software after installation of the hardware and software is complete.

Section 3: About the Software — The first portion of this section briefly describes each part of the microExplorer software and tells where to find more information about each item. The last portion of the section gives a more detailed accounting of some of the utilities useful for exploring the Lisp environment (such as Peek, the Inspector, dribble files, and so on).

Section 4: Memory and Disk Requirements — This section describes the memory and disk requirements for the microExplorer.

Section 5: User Interface — This section provides information about various microExplorer help features and explains how to use them. Specifically, this section describes the following features:

- Status line and mouse documentation window
- Help key
- Histories
- Universal Command Loop (UCL)
- Online documentation

Section 6: Fonts — Because the Macintosh does all actual drawing for the microExplorer, fonts are represented and handled differently in the microExplorer environment than they are in the Explorer system. This section describes font handling on the microExplorer, font paradigm differences between the Explorer and the microExplorer, and how to add and modify fonts in the microExplorer environment.

Section 7: Customizing Your Environment — When you first log in to your microExplorer, its responses are set to certain default values. As you use the microExplorer, you may wish to modify its responses to suit your preferences. This section describes how to customize the microExplorer environment.

Section 8: Sharing File System Resources — The microExplorer environment has no file system of its own. Instead, a microExplorer application acts as any application running on the Macintosh, reading from and writing to the Macintosh file system. This file access capability extends both to hard disks and microfloppy diskettes.

Section 9: Partition-Files — This section describes partition-files: their relationship to Explorer system partitions, and how you can create and use them.

Section 10: Maintaining Your System Configuration — This section provides an overview of methods for managing updates to your microExplorer configuration.

Section 11: Printing — This section describes how you can print to a printer attached to the Macintosh by using the Explorer **print-file** routines. You can print either locally or remotely within constraints.

Section 12: Inter-Environment Communication — This section describes microExplorer-specific information that is required for an application to provide services between the Macintosh and the microExplorer environments. This information deals with the remote procedure call (RPC) protocol. Until you activate RPC in your application, the application cannot receive client requests for services.

Section 13: HyperLisp — This section describes HyperLisp, the microExplorer HyperCard™ interface. HyperLisp allows HyperCard applications to pass Lisp forms to the Explorer environment for evaluation.

Appendix A: Troubleshooting — This appendix gives some information about diagnosing and correcting problems encountered on the microExplorer.

Appendix B: Character Codes and Displays — This appendix gives information about keyboard mapping and character display differences between the Macintosh and the Explorer environments.

Appendix C: The Explorer System Manuals — This appendix identifies areas where information described in the companion Explorer manuals performs differently in the microExplorer environment.

Appendix D: Porting Applications — This appendix tells about the requirements for porting an application program from another type of Lisp machine to the microExplorer.

Appendix E: Network Option — This appendix describes the hardware and software requirements to use the network option. This appendix also provides information you need to know before using the DECnet™ communications interface software or the Network File System (NFS™) protocol software on the microExplorer.

microExplorer Manuals

Depending on which system you purchased, the following manuals are included to provide additional information about the microExplorer:

Title	TI Part Number
microExplorer User's Guide	2552701-0001
Explorer Software Release Information and Installation Guide for microExplorer Delivery Software	2555022-0001
Explorer Software Release Information and Installation Guide for microExplorer Source Option	2555023-0001
Explorer Software Release Information and Installation Guide for microExplorer Development Software	2555024-0001

Explorer Manuals Included With the microExplorer

Because the microExplorer environment strongly resembles the environment of its Explorer predecessor, and because that environment has many programming capabilities, several Explorer manuals have been included to help you with your development efforts. The following manuals are included:

Title	TI Part Number
Explorer Lisp Reference	2243201-0001
Explorer Zmacs Editor Reference	2243192-0001
Explorer Input/Output Reference	2549281-0001
Explorer Window System Reference	2243200-0001
Explorer Tools and Utilities	2549831-0001
Explorer Networking Reference	2243206-0001
Explorer NFS™ User's Guide	2546890-0001
Introduction to the Explorer System	2243190-0001
Explorer Zmacs Editor Tutorial	2243191-0001
Explorer Programming Concepts	2549830-0001
SLE X Window System™ Programmer's Reference	2553272-0001
SLE Common Lisp Object System	2559593-0001

Each configuration that contains the Development System with the networking option also includes the following manuals:

Title	TI Part Number
Explorer TCP/IP User's Guide	2537150-0001
Explorer DECnet™ User's Guide	2537223-0001

Related Manuals

The following manual, which is provided with your hardware, provides additional information about the Macintosh:

- *Macintosh II* owner's guide

The following manuals provide additional information about the Macintosh and can be obtained from Apple Computer, Inc.:

- *Inside Macintosh* manual series
- *Macintosh Programmer's Workshop Reference*
- *Macintosh MultiFinder™ User's Guide*
- *Macintosh HyperCard User's Guide*

The following manual provides additional information about Common Lisp:

- *Common Lisp — The Language* by Guy L. Steele Jr., 1984 (Digital Press, 30 North Avenue, Burlington, MA 01803)

Keystroke Sequences

Many of the commands used with the microExplorer are executed by a combination or sequence of keystrokes that are native to the TI Explorer system. You should become familiar with the conventions from that environment so that you can move easily between the documentation sets provided for each.

Perhaps the most used keystroke convention in the microExplorer environment is *chording*. Chording is the simultaneous pressing of two or more keys. The microExplorer documentation attempts to be consistent with the Explorer documentation that accompanies it; therefore, chording sequences refer to the Explorer key combination rather than the Apple key combination. The keyboard overlay that comes with your microExplorer shows the mapping for the modifier keys used so frequently in chording; however, here are the more common mappings:

Documented microExplorer Key	**Apple Key Equivalent**
CTRL	Control
META	Option
SUPER	Apple
SYMBOL	Control *and* Apple (simultaneously)
HYPER	Apple *and* Option (simultaneously)
SYSTEM	F5
TERM	F8

In this manual, hyphens connect the names of keys that you should chord. Spaces separate the names of keys that you should press sequentially. The following table illustrates this manual's conventions for describing keystroke sequences.

Documented microExplorer Mouse Keystroke Sequence	**Action on the Apple Keyboard**
META-CTRL-D	Hold the Option and Control keys while pressing the D key.
SYSTEM HELP	Press and release the F5 key; then press and release the help key.
CTRL-X CTRL-F	Hold the Control key and press the X key, release the X key, and then press the F key. Alternatively, press CTRL-X, release both keys, and press CTRL-F.
META-X Find File	Hold the Option key while pressing the X key, release the keys, type the letters `Find File`, and then press the RETURN key.
TERM SUPER-HELP	Press the F8 key and release it; then press and hold the Apple key while pressing the HELP key.

If the **sys:*kbd-zmacs-mode*** variable is non-**nil**, then the CTRL and SUPER keys are interchanged. To find the value of this variable, enter the following form in the Lisp Listener:

```
sys:*kbd-zmacs-mode*
```

If the microExplorer returns any value except **nil**, the two keys are interchanged.

Although none of the following key combinations are currently used on the microExplorer system, it is not possible to chord HYPER with META and/or CTRL. Nor is it possible to chord SYMBOL with META.

Mouse Usage

In order to maintain consistency with Explorer documentation, the microExplorer mouse documentation retains the concept of a three-button mouse. However, the Apple mouse has only one button, therefore several differences exist in mouse usage.

The most obvious difference between the Macintosh and microExplorer mouse environments concerns the user interface. For example, how does a user "click right" when the Macintosh mouse has only one button? The microExplorer Development System Software has mapped the various Explorer three-button mouse-clicks to equivalent mouse-click/keystroke combinations. These mappings, which are shown on the keyboard overlay that comes with the microExplorer package, are as follows:

Documented microExplorer Mouse Click	**Action on the Apple Mouse**
L (click left)	Click the button (press the button once and release it).
M (click middle)	Click while holding down the Option key.
R (click right)	Click while holding down the Apple key.
L2 (double click left)	Click the button twice quickly. (Press the button, release it, and press it again quickly.)
M2 (double click middle)	Click the button twice quickly while holding down the Option key.
R2 (double click right)	Click the button twice quickly while holding down the Apple key.
LHOLD MHOLD RHOLD	Press the specified button/key combination and hold it down.

Note:

If you double click too fast, the system sees only one click, If you double click too slowly, the system sees two single clicks. Clicking time for the mouse can be modified in the Macintosh Control Panel.

Lisp Code

Three fonts are used in this manual to denote Lisp code:

- System-defined words and symbols are in **boldface**. A system-defined word or symbol is any word or symbol appearing in the system source code, including names of functions, variables, macros, flavors, methods, and so on.
- Examples of programs and output are in a special `monowidth` font. System-defined words in an example are also in this font.
- Sample names are in *italics*. Names in italics can be replaced by any value you choose to substitute. (Italics are also used for emphasis and to introduce new terms.)

For example, this sentence contains the word **setf** in boldface because **setf** is defined by the system.

Some function and method names are very long—for example, **get-ucode-version-of-band**. Within the text, long function names may be split over two lines because of typographical constraints. When you code the function name **get-ucode-version-of-band**, however, you should not split it or include any spaces within it.

Within manual text, each example of actual Lisp code is shown in the monowidth font. For instance:

```
(setf x 1 y 2) => 2
(+ x y) => 3
```

The form `(setf x 1 y 2)` sets the variables `x` and `y` to integer values; then the form `(+ x y)` adds them together.

In this example of Lisp code with its explanation, `setf` appears in the monowidth font because it is part of a specific example.

For more information about Lisp syntax descriptions, see the *Explorer Lisp Reference* manual.

Words for which you can substitute another value are shown in italics, as in the following example:

> The variables *vars* contained in the lambda list of some function *foo* are bound to the argument values of the function invocation.

Occasionally, in examples where you could substitute a specific value, the boldface and italics fonts are used together.

DEVELOPMENT SYSTEM SOFTWARE OVERVIEW

1

Introduction

1.1 The microExplorer provides an embedded Explorer subsystem, complete with Explorer system windows, on an Apple Macintosh II. It is an extremely powerful system, yet inexpensive for the versatility that it provides.

Three microExplorer software environments are available: the Delivery configuration (also called the System Software), the Development System Software, and the Network System Software. The System Software allows you to run Common Lisp-based applications on your Macintosh II. See paragraph 1.2, System Software, for an overview of the System Software. For details about the System Software, see the *microExplorer User's Guide*.

The Development System Software provides the same capabilities as the System Software, as well as additional features to allow you to write and run your own Lisp-based applications on the microExplorer or on an Explorer system. The Network System Software incorporates additional network functions into the development system. Note that this manual applies to both the Development System Software and the Network System Software.

The Development System Software provides most of the same tools used by the full Explorer system:

- Lisp Listener
- Zmacs editor and Dired
- Compiler
- Interactive debugger (error handler)
- Trace and step facilities
- Inspector utility
- Peek utility
- Profile utility
- Universal Command Loop (UCL)
- Band down-sizing tools
- Metering
- Remote procedure call (RPC)
- Printing
- Extended address space (EAS)

However, the following features provided by the Explorer system are currently not available in the microExplorer environment:

- Shared NuBus™ memory
- Color support
- Suggestions utility
- Glossary utility
- NLMenu
- RTMS utility
- Font editor (This utility is provided by the Macintosh software.)
- Backup and restore utilities (However, you can use the Macintosh software to back up and restore by using Apple-formatted tapes, and with the networking option, you have access to backup/restore facilities on remote Explorers with cartridge tape.)

The microExplorer environment has no file system of its own. Instead, the microExplorer reads from and writes to the Macintosh file system. This file access capability extends both to hard disks and microfloppy diskettes. See Section 8 for details about microExplorer file interaction.

System Software

1.2 Developers should be aware of the capabilities and limitations of the System Software (used in the Delivery configuration) when using the Development System Software to prepare application software to execute in that environment.

The System Software is a base for running application programs. It provides basic Common Lisp support, access to disk files on the host computer, and a loader that can be used to add additional functionality as required. The System Software can be used as-is if the microExplorer is being used as a Lisp server that runs only Common Lisp programs for an application program running on the host Macintosh, where the user interface is entirely controlled from the host application. More commonly, though, the System Software is used as a starting point for building a customized band, where the application developer will load the application program (with its run-time support systems), possibly delete some items that are not needed by the application, perform garbage collection, and **disk-save**.

The System Software supports the full language specified by the book *Common Lisp — The Language* except for some of the functions in Chapter 25. The following functions do not exist: **compile**, **compile-file**, **trace**, **step**, **inspect**, **ed**, and **apropos**. Run-time support for the following Explorer extensions to Common Lisp is included: flavors, stack groups, processes, dynamic closures, resources, initializations, areas, pathname and stream message handling, logical hosts, signaling and handling error conditions, and the extended **loop** macro. The System Software also includes a very simple Lisp Listener based on the cold load stream, the basic debugger, the disassembler, garbage collector, RPC, loader, **defsystem**, **make-system**, **load-patches**, and **disk-save**.

NOTE: Applications written to run in the Delivery configuration should use Common Lisp functions only. Zetalisp functions with arguments that differ from their Common Lisp counterparts are not present in the Delivery configuration although they are supported in the Development System Software.

Minimum Requirements

1.3 To build a system with the microExplorer Development System Software, you must have the following items as part of your Macintosh system:

- Macintosh II chassis
- Macintosh II ROMs version 1.2 or later
- 640x480 (or greater) monochrome or color display
- Apple Extended Keyboard
- Apple single-button mouse
- 2M bytes of RAM (minimum)
- Diskette or tape drive for software distribution
  - 800K-byte microfloppy diskette drive (3.5-inch)
  - 40M-byte tape drive
- 80M-byte (minimum) hard disk, with 70M bytes of unused disk space for installation of the microExplorer Development System Software and for virtual memory swap space
- Macintosh operating system with MultiFinder

In addition to the preceding Macintosh-related items, you must have these microExplorer items:

- microExplorer board with 4M bytes (or more) of RAM
- microExplorer software

Options

1.4 The following optional items are available for use with a microExplorer:

- 4M-byte memory expansion board — Attaches piggyback to the microExplorer board to increase onboard memory to a total of 8M bytes.

- 8M-byte memory expansion board — Attaches piggyback to the microExplorer board to increase onboard memory to a total of 12M bytes.

- Networking (via Ethernet™/TCP/IP) — Provides networking services (when coupled with Apple's EtherTalk™ card and access to an Ethernet local area network). If you purchase the networking option, you then can use several other utilities from the Explorer environment such as Mail, Converse, Remote Tape, and the Bug Reporting System.

  The Mail system allows communication between network users. It is a Zmacs-based utility whose overall design is based on the directory editor (Dired) and can be accessed and used in a similar way.

  The Converse utility allows interactive communication between users in a microExplorer network; that is, you can send messages to or receive messages from any other network user. The Converse message editor displays all messages that you have sent or received. Unlike the mail system, no message files are maintained for Converse. For more information about Mail and Converse, see the *Explorer Tools and Utilities* manual.

  The Remote Tape facility allows remote access from the microExplorer to cartridge tape resources present on Explorers in your network environment. The Backup/Restore utility allows tape backup and restore of individual files, groups of files, directories, and partitions.

  The Bug Reporting System provides a template that allows the user to report a specific microExplorer software problem. Customers who have access to ARPANET or any other appropriate network can use this template to submit bug reports by network mail. After you fill in the template and press the END key, the system creates a mail message and sends the report.

- Development System Software source code — Permits *borrowing* of system code and allows modification of the Lisp operating system. If you purchase the source code, you also get the Visual Interactive Documentation (Visidoc) Online Manual Viewer, which allows you to view documentation from online reference manuals. Visidoc retrieves online reference material and places it in a buffer where the text and graphics information can be viewed. You can also perform operations such as evaluating examples and copying text. See the *Explorer Tools and Utilities* manual.

  The online manuals included as part of Visidoc are the following:

  - *Explorer Lisp Reference*
  - *Explorer Zmacs Editor Reference*
  - *Explorer Tools and Utilities*
  - *Explorer Input/Output Reference*
  - *Explorer Networking Reference*
  - *Explorer Window System Reference*
  - *SLE X Window System Programmer's Reference*

LAUNCHING 2

Introduction

2.1 This section tells you how to launch (or start) the microExplorer, how to log in, and how to end a session. As you use the microExplorer, notice that a portion of the bottom line of the screen displays valuable information about the current status of the microExplorer environment. If you are not the first person to use the microExplorer since its application was launched on the Macintosh II, you may want to relaunch the microExplorer application to reinitialize it.

Investigate the microExplorer on your own; it is almost impossible to harm. The worst that can happen is that you might overwrite or delete a file.

INSTALLATION NOTE: If you have just installed the microExplorer software, you can skip to paragraph 2.3, Enable the Multifinder. If, however, you have a factory-installed microExplorer, power up the Macintosh before proceeding further.

Power Up the microExplorer

2.2 The microExplorer processor board resides in the Macintosh II chassis; therefore, it receives its power from the Macintosh. When you power up the Macintosh, you power up the microExplorer. If the Macintosh is connected properly and is plugged in, you can power it up simply by pressing the Power On key in the upper right-hand corner of the Apple Extended Keyboard.

NOTE: For information about resetting the microExplorer, see paragraph 2.8, Resetting the microExplorer.

Enable the MultiFinder

2.3 The microExplorer relies on coprocessing among several Macintosh applications that constitute the microExplorer software. This coprocessing requires that the Macintosh MultiFinder be enabled before attempting to launch the microExplorer. To enable the MultiFinder, perform the following steps:

1. Select your startup disk volume by clicking on its icon on the Macintosh desktop.

2. Select the Set Startup item from the Special menu in the menu bar. A Set Startup dialog will be displayed indicating whether the system on this disk will use the Finder or the MultiFinder.

3. If Finder is selected, click on the button next to MultiFinder to select it instead.

4. Click on the OK button to end the dialog.

5. You must now reboot the Macintosh in order to start the MultiFinder. Select the Restart item from the Special menu.

You may want to repeat this process for all system disk volumes in your configuration so that if you start up using a different volume, its system will use the MultiFinder.

CAUTION: If the MultiFinder is not enabled, the microExplorer may appear to hang shortly after you launch it, requiring a restart of the Macintosh.

Current versions of the MultiFinder do not implement preemptive process scheduling. If the microExplorer is running as a background task because you have selected a different application as the foreground task, the microExplorer will wait much longer for I/O services. Performance of file input/output, paging, and so forth, will degrade considerably when the microExplorer is not the selected application. Furthermore, some system facilities and ill-behaved applications will not give background tasks *any* time slices while they are active. Be sure not to use these system facilities or applications if you want your microExplorer to run at full speed.

Verify the Startup File

2.4 At launch time, the microExplorer searches a file called Startup (in the microExp folder) for system parameters. A couple of these parameters, the load and microcode volume names, require the name of your Macintosh hard disk. The Startup file initially supplies a default name of HD (a common hard disk name).

NOTE: Do not use non-ASCII special characters in your hard disk names. Use only letters, numbers, and special characters in the ASCII character set. The system may freeze during boot otherwise.

Change the Startup file so that the load and microcode volumes reflect the name of your Macintosh hard disk that contains the microExp folder. Also change the load and microcode partition-file names to reflect the names of the partition-files from which you are booting the microExplorer application. You may need to change other fields also. For more information on the Startup file, see paragraph 2.9, The Startup File.

Because Startup resides in the Macintosh file system, you can use any editor available on the Macintosh to modify the file, including TeachText or MacWrite®.

CAUTION: When you edit Startup, be very careful not to modify the spacing in the file.

Launch the microExplorer

2.5 The microExplorer resides as an application on the Macintosh hard disk in the microExp folder. To launch the microExplorer, perform the following steps:

1. If it is not already open, open your hard disk directory icon window by double clicking the mouse on it. Inside the window, you will see the microExp folder.
2. Open the microExp folder by double clicking the mouse on it. Inside the folder is the Texas-shaped microExplorer application icon.
3. Double click the mouse on the microExplorer icon.

CAUTION: Only one microExplorer application can be active in your system at any given time. Attempting to launch a second microExplorer application will cause the Macintosh system to crash.

The standard Macintosh II has no hardware memory protection. Ill-behaved applications can corrupt other applications' memory without detection. If the microExplorer is behaving erratically, reboot and run without the other application(s) loaded to see if the problem disappears.

If you encounter any difficulties during the launch procedure, see Appendix A. This appendix describes many common sources of errors during the microExplorer launch.

Launch Status

2.5.1 When you launch the microExplorer application, numerous messages appear in a window titled the Cold Load Stream. These messages identify the status of the launch. Figure 2-1 shows the typical contents of the Cold Load Stream at completion of the launch sequence:

Figure 2-1 **Typical Launch Sequence Messages**

```
Initializing comm.
Initializing allocation queues.
>>DebugBuffer Initialization. Size= 8192 bytes.
Reading Startup File hd:microExp:startup
Resetting
Booting mcr hd:microExp:MX96.MCR
Load Band   hd:microExp:NA17.load
```

When you launch the microExplorer application, the microExplorer processor board performs the following series of operations:

1. The processor board executes a self-test, tests the optional memory board (if present), and identifies any faults detected.

2. If the microExplorer board passes its tests, the microExplorer then continues the booting process. If the microExplorer board fails its tests, the microExplorer displays a dialog box (usually referred to as a pop-up window in Explorer documentation) describing the failure.

3. Depending on which *microcode* partition-file is the default, the microExplorer loads the microcode contained in that partition-file into the processor board's control memory. (See paragraph 2.9, The Startup File, for information about default partition-files.)

4. Depending on which *load* partition-file is the default, the microExplorer loads the contents of that partition-file into the processor board's memory.

5. The microExplorer performs other tasks that are defined on the initialization lists.

As soon as these operations complete, the Cold Load Stream disappears, and the Lisp Listener window appears.

Lisp Listener

2.5.2 Each time you launch the microExplorer software, the Lisp Listener utility is automatically invoked. Towards the lower-left corner of the screen, the words `Lisp Listener 1` appear. This label indicates that the micro-Explorer is now active in the Lisp Listener, the interactive Lisp interpreter.

Initial Screen

2.5.2.1 At the completion of the microExplorer launch sequence, the Lisp Listener displays the *initial screen,* a virtual screen produced by the microExplorer application. (For more information about virtual screens, see paragraph 5.2.1, Virtual Screens.) The initial screen lists the name of your microExplorer, the memory available in your microExplorer, and the names and version numbers of the software available on your microExplorer.

At the top of the initial screen is the screen's name. A typical virtual screen name would be `microExplorer - 001`. When you are in another application on the Macintosh, you can reselect the microExplorer application by dragging down the Apple menu until the microExplorer name is highlighted, and then releasing the mouse button. You can also reselect the microExplorer application while in another application by clicking on any visible portion of a microExplorer window.

When you are already in the microExplorer application, you can select a different virtual screen by dragging down the Screens menu until the title of the screen you want is highlighted and then releasing the mouse button. You can also select a different screen while in another by clicking on any visible portion of the desired screen.

Default Screen Size

2.5.2.2 Normally, the size of the microExplorer's first virtual screen depends on the size of the monitor you are using. That is, on larger monitors, the initial screen will automatically be sized to occupy most of the available display space. If you do not want this automatic resizing, you can disable it by holding down the Option key while double clicking on the microExplorer icon to launch it.

On some of the previous releases, the screen saved in the load band is the size of a small monitor. Holding the option key down on a large monitor would prevent the screen from growing larger. In release 6.1, the screen saved in the load band is the size of a large monitor. Holding the option key down on a small monitor will prevent the screen from shrinking to fit the small monitor, in which case much of the screen will be obscured and you will need to use the scroll bars to view the entire screen.

Because larger screens require more Macintosh memory, the default memory size of the microExplorer application may need to be increased for monitors with large displays. See Section 4, Memory and Disk Requirements, for more information about microExplorer memory size requirements.

Prompt Symbol

2.5.2.3 After the initial screen is shown, the microExplorer displays a blank line and the prompt symbol (>) at the left margin. This prompt symbol, with the status message of `Keyboard`, indicates that the Lisp Listener is now ready to evaluate the functions you type. The blinking keyboard cursor (▮) indicates where the typed characters are represented on the video display.

Lisp Top Level

2.5.2.4 Usually on the microExplorer, a user interacts with the Lisp language through a top-level loop called the read-eval-print loop. This loop continuously monitors the Lisp Listener for input of Lisp expressions. When input is detected, the loop reads and evaluates the expression, then prints the result on the display.

Logging In

2.6 You can perform many operations on a microExplorer without logging in. However, to set up the microExplorer environment with your preferences and to gain access to files, you must log in; that is, you must execute the **login** function.

The **login** function is a Lisp function that identifies you and provides access to the microExplorer.

To execute the **login** function, type the function in the Lisp Listener window that appears when you boot the microExplorer. If you have never logged in to the microExplorer before, you should use the following specific form:

(login '*user-name*)

You can specify any name as *user-name*. The apostrophe, or single quote ('), indicates that the characters following it should be treated as a literal value and should not be evaluated.

The following definition of the **login** function and its optional arguments explains further.

login *user-name* &optional *host inhibit-init-file-p* — Function

The **login** function provides access to the file system and to the Mail utility.

Arguments:

user-name — Provides a logical address for electronic mail or messages and specifies your default directory. The *user-name* argument can be either a symbol, such as `'name` or a string such as `"name"`. If you use a symbol, the letters are changed to all uppercase letters. If you use a string, the case of the letters remains the same.

host — Specifies which host's file system to use. The specified *host* becomes your default file server; all references to files that do not specify another host use files in that system. The default value is the local machine (lm). In particular, this is where your login-initialization file (login-init file) comes from. If the host is specified as T, then **login** assumes that host equals lm and *inhibit-init-file-p* equals true.

inhibit-init-file-p — Specifies whether to use a login-init file. If *inhibit-init-file-p* is **nil**, the microExplorer attempts to find a file called LOGIN-INIT (extension .XLD first, then extension .LISP) or PROFILE-LISPM.INIT saved under a directory of the same name as *user-name* on the host. If the microExplorer cannot find a login-init file, it simply returns the Lisp Listener prompt. The default value of *inhibit-init-file-p* is **nil**, which means to use a login-init file if one exists. Initially, always accept the default value of **nil** for this argument.

As you type the **login** function, the microExplorer reacts as follows:

1. The opening parenthesis may initiate microExplorer activity.
2. If you type rapidly, the video display may not immediately repeat what you type. The characters are stored in a buffer until the machine displays them.
3. When you type the closing parenthesis, the microExplorer executes the function. (You need not press the RETURN key.) The run bars show activity, and the status message changes to Run.
4. The microExplorer displays the specified *user-name* on the status line between the who-state and the package name.
5. When execution is complete, the microExplorer displays a T (the returned value of the function) and moves the cursor and the microExplorer prompt to the beginning of a new line.

Ending the Session

2.7 If your microExplorer is shared among several users, you may need to end a session by logging out when you are finished with your work. However, if you are the only user of your microExplorer, you will probably not need to log out except when you reboot or power down your machine.

Logging Out

2.7.1 Logging out cancels your association with the machine. It does not change the environment, destroy the contents of buffers, or kill or bury windows. To log out, follow these steps:

1. Select a Lisp Listener window. If you are not in a Lisp Listener window, select one by pressing SYSTEM L or by selecting the Lisp Listener item from the System menu.
2. Type the Lisp function `(logout)`. The microExplorer evaluates the function, thereby logging you out. In some cases, it resets the values of variables that were set from an initialization list. Then, the microExplorer returns the value **t**.

Powering Down

2.7.2 Because the microExplorer is often part of a network, powering down the Macintosh containing a microExplorer requires more than simply turning off the Macintosh's components. If your microExplorer station is part of a network and provides a file system or a printer for other stations on the network, you should leave your station powered up while other stations require it.

Typically, the Macintosh (and therefore the microExplorer) remains powered up at all times. A situation may arise, however, that forces you to power down the Macintosh. To ensure that all operations within the microExplorer are completed before you power down the Macintosh, follow these steps:

1. Wait until the run bars disappear and the status line does not report any file activity, such as the name of a pathname being transferred to another file system or to a printer.

2. Type the Lisp function `(sys:shutdown)`. A message appears asking if you really want to shut down the system.

3. Press the letter `Y`. The system completes any current activity and shuts down the processor. The keyboard cursor in the selected window stops blinking.

4. Terminate the microExplorer interface by moving the mouse cursor to the `File` item in the Macintosh menu bar for the microExplorer application. Drag down to the `Quit` option; then, release the mouse button.

5. If you want to power down the Macintosh, move the mouse cursor to the `Special` item in the Macintosh menu bar. Drag down to the `Shutdown` option; then, release the mouse button.

Resetting the microExplorer

2.8 The microExplorer environment can be reset in several ways.

The first way is to quit the microExplorer application and then relaunch it. This action resets the entire microExplorer environment; new microcode and load partition-file information are read into your environment, and you begin fresh. To quit the microExplorer application, move the mouse cursor to the File title in the menu bar. Drag the mouse down to the Quit command and release the mouse button.

NOTE: Unlike the Explorer system, the microExplorer ignores all META-CTRL-META-CTRL key-chording sequences. This means that you *cannot* launch or reset the microExplorer by pressing that key sequence.

If your microExplorer has crashed, or if no activity is present when you think there should be, you can also perform a partial reset of the microExplorer by performing a warm boot. A warm boot reinitializes the Lisp environment in the same virtual address space; however, a warm boot does not reload the load or the microcode partition-files. (If the Macintosh itself has crashed, a warm boot of the microExplorer is not possible.) To perform a warm boot on your microExplorer, move the mouse cursor to the Special title in the menu bar. Drag the mouse down to the Warm Boot command, and release the mouse button. This may not work for severe crashes.

Finally, you can force a microExplorer crash. This action first writes information about the current microExplorer session into the :microExp:Crash.log file. You can then relaunch the microExplorer application as described in paragraph 2.5, Launch the microExplorer. To force a microExplorer crash, move the mouse cursor to the Special title in the menu bar. Drag the mouse down to the Force Crash command, and release the mouse button.

The Startup File

2.9 At launch time, the microExplorer searches for certain information required by the microExplorer application as a prerequisite to launching. Part of this information includes the identity of the default load, microcode, and page partition-files which define the microExplorer environment. The microExplorer automatically searches for this information in the :microExp:Startup file.

If you have more than one disk volume with microExp folders, the microExplorer will use the first Startup file it finds, scanning the disk volumes in mount order. Mount order is the order in which the volumes appear on the right-hand side of the desktop from top to bottom.

NOTE: Partition-files serve a purpose similar to partitions in the Explorer environment. If you are not familiar with partitions or partition-files, see Section 9, Partition-Files.

Figure 2-2 shows the typical contents of a Startup file:

Figure 2-2

Typical Contents of the Startup File

```
loadvolume     = HD
loadname       = NA17.load
mcrvolume      = HD
mcrname        = MX96.mcr
hostname       = mx04
defaultdevice  = HD
forslot        = 12
```

By changing the values of the assignment lines in this file, you can specify which load and microcode files get loaded when the microExplorer application is launched. If you have the room, you can keep multiple configurations on your hard disk, such as a Delivery and a Development configuration. You can launch whichever configuration you want by changing the values in the Startup file and relaunching.

Startup resides in the Macintosh file system; however, you can use any available editor to modify the file (MacWrite, TeachText, Zmacs, and so on).

NOTE: If you edit the Startup file, you need to ensure that assignments (x = y) are each on a separate line. The order in which the assignments appear is not critical. Also, you must ensure that the equal sign in each line has at least one space on either side of it.

The loadvolume line identifies the disk on which your load partition-file resides (HD in this example). microExplorer requires that you keep your load partition-file in the microExp folder of your Macintosh system's hard disk.

The loadname line identifies the default load partition-file (NA17.load in this example). This partition-file is loaded at launch time, and contains the microExplorer environment under which you will be operating.

The mcrvolume line identifies the disk on which your microcode partition-file resides (HD in this example). As with the load partition-file, your microcode partition-file must reside in the microExp folder of your Macintosh system's hard disk.

The mcrname line identifies the default microcode partition-file (MX96.mcr in this example). This partition-file is loaded at launch time and contains the microcode for the microExplorer processor.

The hostname line uniquely identifies your Macintosh microExplorer host. In a standalone environment (no networking), the hostname can be an arbitrary name. In a network environment, the name you supply for hostname must be a microExplorer host object properly defined in your namespace. For more information on setting up a microExplorer host, see your Network Software Release Information.

The defaultdevice line identifies the default disk volume for the Macintosh. The default volume is where the microExplorer looks for login files and other default files. This volume name serves as the **:default-device** component of pathnames in your local Macintosh file system. (For more information on the microExplorer file system interface, see Section 8, Sharing File System Resources.) If your microExplorer is a standalone unit (no networking), the defaultdevice also serves as your system host volume. Typically, defaultdevice should be your Macintosh's hard disk.

The forslot line is only required if the version number of your Macintosh II ROMs is earlier than version 1.2 (which date to January, 1988). If your Macintosh has these ROMs, you must identify the slot number where the microExplorer processor board resides (a value from 9 to 14) in the forslot line of the Startup file. The default value of the forslot line is 12. As you are facing the front of the Macintosh, slot 9 is the first slot on your left. If the ROMs in your Macintosh are a later version, you need not change the default value of the forslot line. In this situation, the microExplorer will launch without problems even if your microExplorer processor board is in another slot.

If your system has ROMs earlier than version 1.2 and you plan to use TBSERVER, the microExplorer processor board must be in slot 13 (hexidecimal D). TBSERVER will only launch with the board in slot 13.

NOTE: The slot numbers in the Macintosh chassis are numbered in hexadecimal; however, when entering a value for forslot, you must enter the decimal equivalent of the hexadecimal slot number.

Setting the Date and Time

2.10 The date and time functions are controlled entirely by the Macintosh II. The Lisp functions that deal with date and time access their counterparts on the Macintosh II, so the microExplorer user sees no change in capabilities. You can use Lisp functions such as **time:set-local-time** to set the Macintosh clock. Alternatively, you can use the Macintosh Control Panel facility to set the date and time. For more information on the Control Panel, refer to the Control Panel section of the *Macintosh II* owner's manual.

ABOUT THE SOFTWARE 3

Introduction

3.1 The microExplorer features a highly productive programming environment that allows you to develop very complex programs in incremental steps. You can use any of the system software, as well as optional software packages, as building blocks to create your software. The environment provides many tools to help you modify and enhance that software until you have created a finished product that satisfies you.

The first part of this section briefly describes each part of the microExplorer software and tells where to find more information about each item.

The last portion of the section provides a brief overview of Lisp functions and explains how to interpret and use them. It introduces some functions that help you investigate the microExplorer's Lisp environment. Finally, it introduces the following microExplorer features:

- Dribble files — Contain records of all keyboard input and all system output for a given session

- Print-notification function — Allows you to obtain a complete list of all notifications received during a given session

Development System Software

3.2 The programs described in the following paragraphs are available as part of the Development System Software. Some of these programs are not accessible from the System menu because they are not interactive programs. Each paragraph refers to other parts of this manual and/or other manuals that provide detailed information about the software.

Compiler

3.2.1 The Lisp compiler converts Lisp functions into macrocode so that they run faster and require less memory. See the *Explorer Lisp Reference* manual.

Debug Tools

3.2.2 The interactive debugger enables you to examine the environment in which an error (or some other condition) is signaled and to take the necessary corrective action. Other tools include break, trace, step, and advise facilities. See the *Explorer Tools and Utilities* manual.

Inspector

3.2.3 The Inspector enables you to view and modify the components of Lisp objects. See the *Explorer Tools and Utilities* manual.

Lisp Listener

3.2.4 Using the Lisp Listener, you can communicate directly with the Lisp interpreter. The Lisp Listener reads a Lisp form, evaluates it, and prints the results. When you launch the microExplorer application, the Lisp Listener is the default window. See paragraph 2.5.2, Lisp Listener, and the *Explorer Tools and Utilities* manual.

Peek Utility

3.2.5 The Peek utility displays different types of system activities and their current states. These activities include processes, statistic counters, areas, windows, file system status, and network status. See the *Explorer Tools and Utilities* manual.

Profile Utility

3.2.6 The Profile utility allows you to customize your system environment. See the *Explorer Tools and Utilities* manual.

Universal Command Loop (UCL)

3.2.7 UCL accepts various forms of user input (such as menu selection, mouse clicks, keystrokes, or typed command names) and interprets the input as requests for command execution. See the *Explorer Tools and Utilities* manual.

Zmacs Editor and Dired

3.2.8 Zmacs is a real-time editor that provides extensive support for writing Lisp programs as well as other types of text. Dired, a directory editor embedded within Zmacs, enables you to access files and directories. See the *Explorer Zmacs Editor Reference* manual.

Macintosh Toolbox Interface

3.2.9 The Macintosh Toolbox Interface facility provides direct access from Lisp to most of the Macintosh Toolbox routines described in Apple's *Inside Macintosh* manual series.

HyperLisp

3.2.10 HyperLisp is the microExplorer HyperCard interface. This interface allows you to write application programs that use HyperCard in the Macintosh environment as the user interface to Lisp applications running in the Explorer environment. See Section 13, HyperLisp.

Extended Address Space (EAS)

3.2.11 EAS provides the capability to extend the virtual address space of the Explorer hardware beyond 128M bytes. See the *Explorer Lisp Reference*.

Metering

3.2.12 The metering system allows you to perform a detailed analysis of your program's execution. See the *Explorer Tools and Utilities* manual.

Remote Procedure Call (RPC)

3.2.13 The RPC protocol allows program-to-program communication with other Macintosh applications that are linked with your RPC routines. See Section 12, Inter-Environment Communication.

Printing

3.2.14 Printing capabilities are provided for a locally-connected printer such as LaserWriter™ or ImageWriter™. See Section 11, Printing.

Lisp Functions

3.3 All Lisp programs consist of one or more *Lisp functions* that specify procedures to be performed. Some procedures are not complete within themselves; that is, they must be given one or more pieces of information to work with. In the Lisp environment, these pieces of information are called *arguments*. For example, consider the following elementary Lisp function:

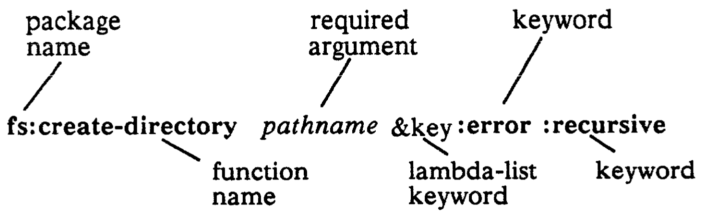
```
(+ 5 2)
```

This expression consists of the procedure (+) and its arguments (5 and 2). The procedure specifies what is to be done to the arguments. That is, the procedure specifies that 5 and 2 are to be added together.

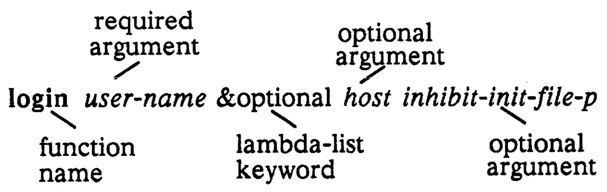
Every Lisp function is enclosed by a set of parentheses. The opening parenthesis identifies the expression as a Lisp function; the closing parenthesis signals the Lisp machine to perform the specified operation and to return the value specified by the expression. This process is called *evaluation*.

Lisp syntax governs the legal order of the elements of any Lisp expression, including those of Lisp functions. Some Lisp functions require several types of elements; some require only the procedure name.

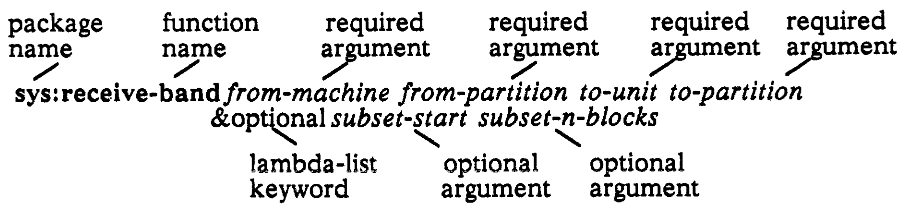
The following syntax examples show three typical Lisp functions. These examples illustrate the legal order of the elements of typical Lisp functions. The six element types are described after the three examples.



Function



Function



Function

Element Descriptions:

- Package name — The name of the package in which the function resides. When a package name is specified, it is considered part of the function name.

  If the package name is not specified, the function resides in the current package, or the LISP or TICL (TI Common Lisp) packages. (The last two packages are usually inherited.) If you do not specify a package name and the function is not in one of the previously mentioned packages, the microExplorer attempts to find the function in other packages. If the system finds a function of the same name in another package, it asks you whether you want to use that function.

- Function name — The name of the function. The function name usually indicates the type of operation specified by the function. For example, the **fs:create-directory** function creates a directory as specified by the pathname argument.

- Required argument — Required arguments are the arguments directly following the function name; they are not preceded by a qualifying character such as &. When you type a Lisp function to be evaluated, you must supply a value for each of the required arguments. The required arguments in the previous syntax examples are *pathname*, *user-name*, *from-machine*, *from-partition*, *to-unit*, and *to-partition*.

- Lambda-list keyword — Lambda-list keywords are words that describe the arguments that follow. That is, these words are argument separators rather than arguments. All lambda-list keywords are preceded by an ampersand (&). The lambda-list keywords in the preceding syntax examples are as follows:

  - &optional — Indicates that the arguments that follow are optional

  - &key — Indicates that the arguments that follow are keywords

- Optional argument — Optional arguments follow the lambda-list keyword &optional. These arguments do not have to be specified when you use the function with which they are associated. However, when you do specify these options, you must specify all the options in the order in which they are listed. Thus, if you want to specify the third optional argument, you must also specify the first and second optional arguments. The optional arguments in the syntax examples are *host*, *inhibit-init-file-p*, *subset-start*, and *subset-n-blocks*.

- Keyword — Keywords describe an optional argument. When you use a keyword to call a function, you must specify both the keyword (preceded by a colon) and its value. However, you need to specify only those keywords and values that you want to include; therefore, if you want to specify the third keyword, you do not specify the first or second keywords. Also, the order in which you specify keywords is unimportant. The keywords in the syntax examples are **:error** and **:recursive**. Note that keywords are preceded by a colon (:) when they are called, not when you define them.

Useful Functions

3.4 The following functions provide the means to explore relationships between various Lisp forms and the workings of the microExplorer:

- **describe**
- **apropos** and **sub-apropos**
- **who-calls**
- **what-files-call**
- **where-is**
- **symbol-plist**

Most of these functions can be used to great advantage in conjunction with either the Inspector or Zmacs.

describe Function

3.4.1 The **describe** function provides information about objects such as arrays, closures, function entry frames (FEFs), symbols, lists, locatives, stack groups, packages, floating-point numbers, fixnums, bignums, complex numbers, select methods, named structures, and **defstructs**.

describe *x* — Function

The **describe** function provides information about a Lisp object *x*. It prints the attributes of the specified object and can describe an object located within another; such recursive descriptions are indented appropriately.

For example, when you apply the **describe** function to a symbol, it shows the symbol's value, its function definition, and each of its properties. Calling the **describe** function on a floating-point number shows the number's internal representation in a way that is useful for tracking down errors, such as round-off errors. Executing **describe** on an array shows the array's type, dimension, and length, but the contents of each array element are not shown. (The Inspector shows array contents.)

Other functions that are useful for obtaining information about objects are the **describe-area**, **describe-flavor**, and **describe-package** functions. The *Explorer Tools and Utilities* manual provides information about these functions. Try the following functions to see what they return:

```
(describe-area fs:pathname-area)

(describe-flavor 'sys:cold-load-stream)

(describe-package 'sys)
```

Apropos Functions

3.4.2 Apropos is a facility that searches the system for all symbols whose print names contain a specified string and prints information about the symbols found. Following are *abbreviated* definitions of two of the more common **apropos** functions. For more detail, see the *Explorer Tools and Utilities* manual.

apropos *string* &optional *package* Function

The **apropos** function tries to find all symbols in the entire system whose print names contain *string* as a string. Whenever it finds a symbol, this function prints the symbol's name; if the symbol is defined as a function and/or is bound to a value, this function tells you so and prints the names of the arguments (if any) to the function. Specifying a package name restricts the search for *string* to *package* and any other packages to which *package* is subordinate.

sub-apropos *string starting-list* &key **:predicate :dont-print** Function

The **sub-apropos** function finds all symbols in *starting-list* whose names contain *string* and which satisfy **:predicate**. If **:predicate** is **nil**, the string is the only condition. The symbols are printed if **:dont-print** is **nil**. In any case, a list of the symbols found is returned.

To see some examples of these functions, try the following:

```
(apropos 'edit)

(sub-apropos 'edit *)
```

who-calls Function

3.4.3 The **who-calls** function searches for all the functions that call a specified object and prints one line of information for each caller it finds.

who-calls *x* &optional *package* Function

With the **who-calls** function, *x* must be a symbol or a list of symbols. This function tries to find all the Lisp functions that call *x* as a function or macro or that use *x* as a variable or constant. (It does not find forms that use constants containing *x*, such as a list whose elements include *x*; it only finds the form if *x* itself is used as a constant.) It tries to find all the functions by searching all function calls of all symbols in *package* and *package*'s descendants. The *package* argument defaults to **nil**, meaning that all packages are checked.

The editor command META-X List Callers is similar to **who-calls** and defaults to the USER package. Type the following to see what the function returns:

```
(who-calls 'format 'time nil nil)
```

what-files-call Function

3.4.4 The **what-files-call** function is similar to the **who-calls** function except that it finds all the files containing functions that call a particular object.

what-files-call *x* &optional *package* Function

This function returns a list of the pathnames of all files containing functions that **who-calls** would have printed. This function is useful if you need to recompile and/or edit all of these files. Type the following to see what the function returns:

```
(what-files-call 'format 'time nil nil)
```

where-is Function

3.4.5 The **where-is** function prints the names of all packages that contain a specified symbol.

where-is *print-name* &optional *package* Function

The *print-name* argument specifies the object for which the specified packages are searched. If *print-name* is a string, it is converted to uppercase. Unless *package* is specified, **where-is** assumes you want to search all packages. If you specify a package for *package*, the function searches only for symbols from the specified package. The **where-is** function returns a list of all the symbols it finds. Type the following to see what the function returns:

```
(where-is 'member)
```

symbol-plist Function

3.4.6 The **symbol-plist** function returns the property list of symbols for a specified object. The *Explorer Lisp Reference* manual provides further information about the **symbol-plist** function.

The Inspector

3.5 The Inspector is a window-based utility for viewing and modifying Lisp objects. Inspecting a particular object displays the components of this object. The type of object determines exactly what is displayed. For example, the components of a list are its elements; the components of a symbol are its value binding, function definition, property list, package, and print name.

The components of displayed objects are mouse-sensitive; positioning the mouse cursor over an object causes a box to appear around that object. Once the object is boxed, you can press the proper mouse button to inspect the boxed object, modify it, or give the object as an argument to a command.

Two different types of Inspectors exist: one type has its own process and the other does not. For this reason, you can invoke the Inspector in these three different ways:

- Press SYSTEM I.
- Use the mouse to select Inspect from the System menu.
- Call the **inspect** or **inspect*** function from the Lisp Listener.

inspect &optional *x* — Function

The **inspect** function enters the Inspector, displaying the contents of *x* (if specified). This function invokes the Inspector in its own process instead of in the calling process, just as if you pressed SYSTEM I.

An Inspector invoked with **inspect** or SYSTEM I does not have access to locally bound variables.

inspect* &optional *x* — Function

The **inspect*** function enters the Inspector, displaying the contents of *x* (if specified). This function runs in the stack group of the calling process, rather than in a separate stack (as when you press SYSTEM I or use **inspect**).

This function maintains the environment. It has access to locally bound variables.

The inspection window contains the following panes:

- An interaction pane called Inspect
- Three inspection panes (which are configurable to 1, 2, or 3)
- A history pane
- A command menu pane

The *Explorer Tools and Utilities* manual provides more information about the **inspect** function and the Inspector.

Peek Utility

3.6 Peek is a window-oriented utility that displays a continuously updated system status of the following mouse-selectable items:

- Windows mode — Lists the hierarchy of windows currently displayed on the microExplorer
- Areas mode — Displays for each area the area name, number of regions, and allocation
- Processes mode — Displays the process name, status, priority, quantum, and idle time of each process
- File Status mode — Lists all the hosts that have remote file connections to or from this host and shows any connections (host units) associated with those hosts
- Servers mode — Displays a list of all the servers running on this host that are serving other hosts, including the server's contact name, host, process, and connection
- Histogram — Displays a continuously updated statistical sampling of which functions use the most system resources
- Network mode — Allows you to find useful information about all network connections (assuming you have the networking option)
- Counters mode — Displays the statistics of all the microcode meters
- Documentation (Help) command — Displays concise, general information about the Peek utility
- Set Timeout command — Allows you to set the time delay between Peek utility updates
- Host Status command — Displays the status of each host contacted since booting
- Exit command — Returns to the process from which Peek was called

You can invoke the Peek utility by selecting it from the System menu, pressing the keystroke sequence SYSTEM P, or entering the **peek** function in the Lisp Listener. For detailed descriptions of the various Peek utility modes and commands, refer to the *Explorer Tools and Utilities* manual. Try some of the options to see what type of data is shown.

Other Utilities and Functions

3.7 In addition to the utilities discussed in the preceding paragraphs, the Development System Software features several other useful utilities and functions, which are discussed in the following paragraphs.

Dribble Files

3.7.1 A microExplorer dribble file records terminal interaction performed at a session; that is, it contains a record of all keyboard input and most system output. The following example assumes that you are running on a machine called mx9. To open a dribble file, you enter a form, such as the following, in a Lisp Listener:

```
(dribble "lm:hd:misc:dribble.text")
Entering Dribble Read-Eval-Print Loop.
    Type (DRIBBLE-END) to exit.
```

The following example shows how the dribble file might appear after the **copy-directory** and the **dribble-end** functions have been executed:

```
> (copy-directory "mx9: hd: applications:*" "mx4: hdl: applications:")
mx4: hdl: applications: windows.xld → [Different file exists at target]
mx4: hdl: applications: panes.xld → [Already Exists]
Copied mx9: hd: applications: cull.xld to mx4: hdl: applications: cull.xld
Byte size 16, Characters NIL
Copied mx9: hd: applications: grow.xld to mx4: hdl: applications: grow.xld
Byte size 16, Characters NIL
Copied mx9: hd: applications: jump.xld to mx4: hdl: applications: jump.xld
Byte size 16, Characters NIL
Copied mx9: hd: applications: noop.xld to mx4: hdl: applications: noop.xld
Byte size 16, Characters NIL
Copied mx9: hd: applications: trim.xld to mx4: hdl: applications: trim.xld
Byte size 16, Characters NIL
(#⊂FS:MAC-PATHNAME "mx9: hd: applications: cull.xld"⊃
#⊂FS:MAC-PATHNAME "mx9: hd: applications: grow.xld"⊃
#⊂FS:MAC-PATHNAME "mx9: hd: applications: jump.xld"⊃
#⊂FS:MAC-PATHNAME "mx9: hd: applications: noop.xld"⊃
#⊂FS:MAC-PATHNAME "mx9: hd: applications: trim.xld"⊃)

> (dribble-end)
```

When you enter the `dribble-end` function to close the dribble file, a message such as the following appears:

```
Closing dribble file, mx9:hd:misc:dribble.text.
NIL
```

The microExplorer provides four Lisp functions for dribble file operations: **dribble, dribble-start, dribble-all,** and **dribble-end.** The *Explorer Tools and Utilities* manual provides detailed information about the use of these functions.

print-notifications Function

3.7.2 At times you may need a list of all notifications received during the current session. For example, you might have missed notifications or inadvertently deleted them before reading them. The **print-notifications** function allows you to reprint all notifications.

MEMORY AND DISK REQUIREMENTS

4

Introduction

4.1 The microExplorer uses different memory and disk resources for various functions. The Lisp system uses local memory on the microExplorer board, while the Macintosh microExplorer application runs in Macintosh memory. The Macintosh disk system is used both for Lisp paging storage (via page partition-files) and for general file storage by Lisp, Macintosh system software, and other Macintosh applications. While optional memory and disks are available, certain minimum requirements exist for proper microExplorer functioning. This section explains how these minimum values are reached and how to calculate the amount of additional memory or disk space you might require.

NOTE: All actual and minimum figures listed here are subject to change with software updates and new releases. The figures provided should be used only as guidelines for calculating the appropriate requirements.

Macintosh System Memory Requirements

4.2 When estimating how much Macintosh system memory will be needed, you should consider the following issues:

- The minimum memory required to run the microExplorer application
- Additional memory required for the microExplorer because of a large monitor and/or additional virtual screens
- Memory needed by your Macintosh system software
- Memory required for other applications

Each of these issues will be addressed separately.

Macintosh Application Memory Usage

4.2.1 All Macintosh applications have a suggested memory size. This size is the amount of free Macintosh memory that these applications should require. To obtain an application's suggested memory size, you select the application by clicking once on it and then select the Get Info command from the File menu. This method works as long as the application has not been launched. Once the application is running, you can view its memory usage by selecting the Finder (by clicking on a disk icon, for example) then selecting the About the Finder item from the Apple menu. A window appears that shows the memory allocation and usage of each running application.

Some Macintosh applications are written so that they can use additional Macintosh memory if it is available. Such applications have an editable Application Memory Size field in addition to the Suggested Memory Size field of the Get Info display. The microExplorer is such an application. Its minimum memory size is always given in the Suggested Memory Size field. Currently this value is 550K. However, if the microExplorer runs on a large monitor, more memory is needed for the larger screen's bitmaps. The minimum large-monitor size is given in the Application Memory Size field of the Get Info display. Currently, this value is 1000K.

NOTE: In this section, the notation K refers to K bytes.

---

microExplorer Application Memory Requirements

4.2.2 The microExplorer requires the support of three concurrent Macintosh applications: microExplorer, NfsDaemon, and PortMap. When you boot the microExplorer by launching the Macintosh application named microExplorer, the NfsDaemon and PortMap applications are launched automatically by the microExplorer. The total memory size required for the microExplorer is the sum of the memory sizes for the three applications:

550K for microExplorer (minimum)
122K for NfsDaemon
122K for PortMap

794K for microExplorer (minimum)

This figure represents the minimum memory requirement needed for running a microExplorer on a standard-sized Macintosh black and white or color monitor. With the microExplorer application sized at this 550K minimum, only a few virtual screens can be created, and some window-selection operations may be slow. A method for calculating an appropriate size for the microExplorer application in other environments is discussed later.

Macintosh System Software Memory Requirements

4.2.3 Another memory consideration is the amount of memory that will be needed by your Macintosh system software. The System application does not have a Suggested Memory Size; you can only determine how much it is using by consulting the About the Finder memory display.

The basic Macintosh II System application provided by Macintosh requires about 350K. This figure is current for the Macintosh operating system release 6.0.2 running MultiFinder. To this figure Texas Instruments adds several Desk Accessories and Fonts, bringing the size to about 351K. In addition, Finder requires its own memory separate from the System application. Finally, a certain amount of memory should be reserved for running other Macintosh utilities. Examples of other utilities and their suggested memory sizes include the following:

104K for PrintMonitor (background printing)
384K for Apple HD SC Setup
384K for HDBackup
384K for Tape Backup 40SC

A minimum memory requirement for Macintosh system software would then be the following:

351K for the Macintosh System application (minimum)
160K for the Finder
425K average for another Macintosh utility

936K for Macintosh system software (minimum)

The minimum required microExplorer memory configuration can then be computed as follows:

936K for Macintosh system software (minimum)
794K for microExplorer (minimum)

1730K for a complete microExplorer system (minimum)

As this figure indicates, a microExplorer system should not be configured with less than 2M bytes of physical memory.

Note that installed Desk Accessories, Control Panel applications, device drivers, and system debuggers can increase the amount of memory needed by the basic Macintosh System application. Here are some examples:

140K for CloseView™ Control Panel application
6K for Responder Control Panel application
1K for the Apple EtherTalk driver
130K for the MacsBug debugger (runs as part of the system)

Other Applications

4.2.4 If you plan to run other Macintosh applications concurrently with microExplorer, you may want even more Macintosh memory. Following are typical Macintosh applications and their suggested memory sizes.

480K for MacDraw™
224K for MacWrite
384K for MacPaint™
1000K for HyperCard
1024K for MPW shell

Note that because of the large memory requirement of HyperCard, running HyperLisp (which requires both the microExplorer and HyperCard to be running) adds a significant amount to the required memory for a HyperLisp microExplorer system. Such a configuration should not have less than 3M bytes of memory.

microExplorer Application Memory for Bitmaps

4.2.5 In addition to acting as the primary interface agent for Lisp on the Macintosh side, the Macintosh microExplorer application also caches bitmaps for Lisp window displays. If there is not enough bitmap cache space on the Macintosh side, then the Lisp code uses the NuBus to swap bitmaps between its virtual memory and the cache. Because the backup storage for Lisp's virtual memory is the Macintosh disk, swapping between the Lisp virtual memory and the Macintosh cache may require disk transactions as well as NuBus activity. As a result, microExplorer bitmap cache flushes can be time-consuming operations.

The microExplorer application is shipped ready to use. However, for improved performance you may want to allocate more Macintosh memory to this microExplorer application for bitmap caching space if either one of the following occurs:

- You add a larger monitor (that is, larger bitmaps).
- Your Lisp application frequently uses windows (that is, more bitmaps).

A symptom of insufficient bitmap cache size is a user application that starts out with acceptable speed and then slows down as more and more windows are used. The user application may appear to be spending its time doing paging (because of bitmap swapping) even though the application is not supposed to be paging intensive.

Changing microExplorer Application Memory Size

4.2.6 You can change the memory allocation of a Macintosh application that is not currently running by following these steps:

1. From the Macintosh Finder, click on the application icon once to highlight it.
2. Select the Get Info item from the File menu. The Application Memory Size (K) entry at the bottom of the Info dialog box indicates the current allocation.
3. Click on the box adjacent to Application Memory Size, and enter the new value.
4. Click on the box in the upper left corner of the Info dialog box to close it.

The next time you launch the application, it will run with its new memory allocation.

microExplorer Memory Allocation Guidelines

4.2.7 If you plan to use your Macintosh only to run the microExplorer, you can simply allocate all remaining Macintosh memory to the microExplorer application. However, if you intend to use other Macintosh applications while you are running the microExplorer or if you are contemplating buying more Macintosh memory, then calculate the microExplorer application memory size in bytes as the sum of the following quantities:

400K for the basic application code (required)

+ Total area for all microExplorer screens listed in the Screens menu of the microExplorer menu bar (required, minimum one)

+ Total area for three extra microExplorer screens equal in size to the largest screen included in the item above (required)

+ Total area for all microExplorer windows (for example, Lisp Listener, Zmacs, Peek, and so on) and all application windows and menus (performance option)

where the area for a screen in bytes is calculated as the screen's pixel height times pixel width divided by 8. The default size of a standard-sized screen is 608x424 pixels, but that can be changed by using the Set Screen Size dialog box from the microExplorer Options menu. An existing screen's size can be changed with the screen resizing operation (described in Section 5, User Interface).

The first three items are required for general operation. The last item is required for optimum performance.

Memory Allocation Example

4.2.8 The microExplorer allows you to define microExplorer screens of different sizes. Normally, however, all microExplorer screens are the same size, which is slightly smaller than the Macintosh monitor (so that part of the desktop is still visible).

Assume that you want to determine the microExplorer application size for the large Macintosh 1152x870 monitor. In this case, the default microExplorer screen is 1120x814, resulting in a microExplorer screen size for the equation of (1120 * 814)/8, or 113,960 bytes (111K bytes).

Furthermore, assume that you will have a Lisp Listener and a Zmacs editor window on each of those two microExplorer screens. (These particular windows are full-screen size. Other windows and especially menus may be less than full-screen size.) The calculation is given below. Note that only the last item is optional for the configuration described.

microExplorer application code size		400K
Total area for two microExplorer screens		222K
Total area for three extra screens		222K
Total area for four full-size windows	+	444K
		1288K bytes

Therefore, if you set your microExplorer application size to 1288K, then it can handle two virtual screens, each with a Lisp Listener and an editor, without swapping the bitmap cache. Extra windows or pop-up menus would cause swapping.

Lisp Memory and Disk Requirements

4.3 The microExplorer, being a genuine Lisp machine, uses a virtual memory architecture.

NOTE: If you are unfamiliar with the concepts associated with virtual memory architecture, see the section entitled Storage Management in the *Explorer Lisp Reference* manual.

For maximum performance, the most frequently used sections of code and data must reside in physical memory. The more often code or data must be paged in and out, the slower your application will run. To execute within acceptable time constraints, a system must balance its physical and virtual memory requirements. To achieve this balance, you can use the tools provided with the Development System Software to indicate the memory and disk requirements for your application. The procedures for using these tools are as follows:

1. Load your applications on the microExplorer development system.

2. Use the **gc-status** function and the **sys:time-stats** function to view statistics on the system's actual memory requirements.

3. Use these statistics along with the application's actual execution experience to determine the paging storage adequacy.

Lisp Physical Memory Considerations

4.3.1 If the physical memory on your microExplorer is smaller than the set of instructions and data actively in use, you will spend an excessive amount of time waiting on the disk. An adequate amount of physical memory for the microExplorer's development configuration is 8M bytes of physical memory.

The Lisp function **sys:time-stats** provides total real time, total CPU time and use percentage, and total disk wait time and percentage for your system since it was last booted. Disk wait percentage is the percentage of processing time spent waiting on the disk. If the percentage exceeds 10 to 20 percent, the system's responsiveness is limited by physical memory, and additional physical memory will help performance.

You may also want to compare the output of the **sys:time-stats** function before starting to execute your application, then after running it for awhile. Such a comparison might be useful for separating the disk time for your application from that of the microExplorer system in general.

Adaptive Training and Physical Memory Usage

4.3.2 The use of the Adaptive Training facility can substantially reduce the amount of physical memory needed by the working set (dynamically active portion) of a given application. Adaptive Training is enabled by using the **training-on** function or by logging in with a **nil** value (the default) for the *inhibit-init-file-p* argument of the **login** function.

The typical Development load partition-file contains 25M bytes of system code and data structures. Only a small amount (approximately 4.5M bytes) is in use even by the most interactive application. The **gc-status** function (invoked by pressing TERM G) presents statistics for how much of the system is dynamically active. These figures are based on information accrued by Temporal Garbage Collection (TGC) and its Adaptive Training facility.

The **gc-status** display presents the amount of active data (in K bytes) for each generation and the total for all generations under its Active column. If you inspect this total after your application has been executing for awhile, you can obtain the dynamically active size of your application environment. Adding physical memory beyond this active size is generally unnecessary.

Load Band Training

4.3.3 The Garbage Collector's Load Band Training facility improves the locality of reference as the program executes and the garbage collector copies objects from old space to new space. The most recently accessed objects are grouped together on the same page in memory and are far more likely to reside in physical memory, or at least to be paged from the disk as a unit.

You can apply the Load Band Training before you deliver your application. Then, you can disk-save the load partition-file to be delivered. When this is done, your customer launches a partition-file that has already been trained to the optimum locality of reference. Typically, this reduces the application's paging time by a factor of four on the first execution. Additional information on training load partition-files is available in the *Explorer Lisp Reference* manual.

Paging Storage Considerations

4.3.4 An adequate amount of paging storage for a microExplorer configuration is generally an amount between one and one-half and two and one-half times the size of the load partition-file running. If the system runs with automatic GC on (the default) and Adaptive Training disabled, the amount needed is closer to the size of the load band. For sessions with Adaptive Training on (which increases paging storage usage) or if a **full-gc** or **disk-save** is planned, the amount of paging storage should be two or more times the size of the load band. Here are some examples:

Example 1:

20M-byte development band
    TGC on, Adaptive Training off

Recommended paging storage: 30M bytes minimum

Example 2:

20M-byte development band
5M-byte application code
TGC and Adaptive Training on

Recommended paging storage: 40–50M bytes minimum

Example 3:

Boot a 30M-byte network band
TGC on, Adaptive Training off
Load 5M bytes of application code

Recommended paging storage to complete a **gc-and-disk-save**: 80M bytes

How much paging storage is actually necessary depends on the rate new objects are created in your microExplorer environment, how often the microExplorer performs garbage collection, and what changes you make to the basic load partition-file. Inadequate paging storage has two possible consequences:

- You will be unable to perform a full garbage collection and **disk-save** your environment.
- The paging system will fill up too rapidly with newly created or changed objects. Then, garbage collections will occur more frequently in order to reclaim reusable memory, ultimately resulting in a premature need to reboot in order to regain a sufficient amount of free virtual memory.

In either case, if additional disk storage is available, more paging storage can be added dynamically to the application you are running. Use the **gc-status** function to view the garbage collection statistics for the collection frequency and the amount of free paging storage available. You can add a new page partition-file by using the **sys:add-page-band** function.

Disk Space Considerations

4.3.5 In considering total disk storage requirements for the microExplorer system, the following components must be taken into account. Each item is discussed separately in later paragraphs.

- microExplorer system files:
  - Load and page partition-files
  - Other files required for microExplorer operation
  - Optional microExplorer system files
- Macintosh system files
- File space required by your application
- Space for other Macintosh applications and their data files

microExplorer System Files

4.3.5.1 Space for load and page partition-files has already been addressed in paragraph 4.3.4, Paging Storage Considerations. As discussed, a reasonable minimum for a Development System Software configuration is between 30 and 80M bytes (depending on the use of Adaptive Training, the need to **disk-save**, size of application code, and so forth). Additional partition-file space is required if you need to have more than one load band available in your configuration.

After you complete the system installation process, all microExplorer-related system files can be found under the top-level microExp folder. (Normal microExplorer operation creates several top-level folders on your disk, but these folders are generally empty and consume little space.) The contents of the microExp folder (exclusive of the page and load partition-files) can be divided into three parts: the MacSys folder, the ExpSys folder, and the other items in the microExp folder. The current sizes for each of these in the standard Development or Network System Software is as follows:

```
1.4M bytes for MacSys
4.2M bytes for ExpSys
 .9M bytes for other microExp items
          (excluding .LOAD and .PAGE files)
---------------------------------------------
6.5M bytes total
```

The MacSys folder contains Macintosh application source code needed to build custom applications that communicate with the microExplorer via the remote procedure call (RPC) protocol. Refer to Section 12, Inter-Environment Communication, for information on constructing such applications. If you do not intend to create such Macintosh applications, you can discard the MacSys folder contents.

The ExpSys folder contains files for use in the microExplorer environment. Some subdirectories, such as Public and GWIN, contain optional software that you may or may not want to maintain on your local disk. Some ExpSys folders that generally should not be deleted are PATCH, SITE, and UBIN.

Note that if the microExplorer source option is installed, it requires a significant amount of file space in the ExpSys folder. The current sizes for the source option are 30M bytes for the basic source files and 10M bytes for Visidoc online documentation files.

Macintosh System Files

4.3.5.2 The basic Macintosh system software is contained in the System and Utilities folders on your hard disk. The minimum contents of these folders for a standard microExplorer with printing capability is currently about 2M bytes. Acquiring additional utilities or desk accessories or installing additional fonts will require more space.

Application File Space

4.3.5.3 The amount of Macintosh file system storage required by microExplorer applications varies widely. In a network environment with ample file storage available on remote file servers, very little local file system storage might be required. Conversely, an application may need a large amount of Macintosh file storage for its data files. If your application has previously run on an Explorer, the amount of the Explorer local file system space it uses will be a good indicator of how much space to allocate on the microExplorer. As a very rough figure, most applications require between 10 and 40M bytes of local file system disk space.

Other Macintosh Applications

4.3.5.4 The amount of disk space required for other Macintosh applications and their data files can also vary widely. The HyperCard application with its standard stacks requires about 2.2M bytes of disk space, not including any user-defined data. A word processing program might not take much space, but you might need a substantial amount of disk space to store files created with it. As a rough figure, you might want to allocate between 5 and 10M bytes of disk space for Macintosh application usage.

5 USER INTERFACE

Introduction

5.1 This section provides information about various microExplorer help features and explains how to use them. Specifically, this section describes the following features:

- Window system interface
- Keyboard
- Histories
- Universal Command Loop (UCL)
- Online documentation

Window System Interface

5.2 The microExplorer uses the Explorer window system; however, all microExplorer graphics (text, lines, and so on) are displayed using Apple's QuickDraw toolkit. This approach gives the microExplorer hardware device independence; that is, any display available for the Macintosh works for the microExplorer. More importantly, by using QuickDraw, the microExplorer achieves a high degree of parallel processing. When the microExplorer needs to output to the screen, it simply sends a message to the Macintosh (such as **:draw-string**) and immediately resumes execution of the Lisp program. In the meantime, the Macintosh receives the message, and performs the output operation at the same time the Lisp program is executing.

Virtual Screens

5.2.1 The microExplorer uses Explorer screen objects called *virtual screens*. Each virtual screen acts like a separate microExplorer; that is, each virtual screen can have its own Lisp Listeners, Zmacs buffers, Peek screens, and so on.

Each virtual screen maps directly to a Macintosh window. Therefore, a virtual screen acts like any other Macintosh window in that it can be moved by dragging its title bar, sized by dragging its size box, scrolled by using its scroll bars, and so on. You can even have multiple virtual screens existing at the same time on your desktop. To create a new virtual screen, use the following keystroke:

TERM SYSTEM *x*

where *x* represents a standard key reference for a microExplorer window (such as L for Listener, E for Zmacs editor, P for Peek, and so on). Afterwards, if you examine the Screens menu, you will see that a new item has been added representing the virtual screen you just created.

CAUTION: Creating new virtual screens or increasing the size of existing ones requires additional Macintosh memory for proper operation and maximum performance. Refer to Section 4, Memory and Disk Requirements, for more information on memory size requirements for additional virtual screens.

Because the microExplorer uses the QuickDraw toolkit, virtual screens behave just like other Macintosh application windows. Therefore, virtual screens can co-exist on your desktop with other Macintosh application windows, allowing you to move freely from one to another using the same user-interface tools.

In the Explorer window system, multiple overlapping windows are possible. However, output cannot go to a window that is partially obscured by another. With the microExplorer, you can avoid this situation by placing the background Explorer window (such as a Peek display) on a separate virtual screen. Because the Macintosh window system has no restriction on output to partially obscured windows, output can then go to an obscured virtual screen.

Viewing and Scrolling

5.2.2 Macintosh windows are considered viewports into a document. When you grow or shrink a Macintosh window by using the size box in the lower right corner of the window, you are changing the size of the viewport into the currently displayed document. In the case of a microExplorer window, you change the size of your viewport into the microExplorer virtual screen when you resize the window using its size box. Similarly, when you scroll the Macintosh window, you are changing the position of your viewport into the microExplorer virtual screen. If you shrink your windows by using the size box, you can have all your microExplorer windows visible simultaneously on your Macintosh desktop.

This implementation results in the ability to scroll a virtual screen within a Macintosh window *that is also scrollable*. Two sets of scroll bars exist. First, you have the familiar Macintosh vertical and horizontal scroll bars that change the position of the viewport into the Explorer screen. Secondly, you have the Explorer scroll bar that performs the same functions that Explorer users are accustomed to, such as scrolling a Zmacs buffer or an Inspector pane. Generally, the Macintosh scroll bars are located on the extreme right and bottom of the Macintosh window, while the Explorer scroll bar is located to the left on the window. For the microExplorer, the look and feel of the Explorer scroll bar has been changed to be compatible with the operation of the Macintosh scroll bar. This change prevents users from having to learn how to use two different scrolling mechanisms. However, the microExplorer scroll bar still retains the features implemented for the middle and right mouse buttons that give Explorer its additional scrolling features.

Resizing Virtual Screens

5.2.3 The way text is laid out on a window depends on the size of the window's virtual screen. Line wrapping, for example, depends on the width of the underlying screen. While adjusting the size of a microExplorer window by dragging the size box only changes the viewport into the virtual screen, pressing and holding the Option key while dragging the size box performs the *virtual screen resizing* operation. The screen resizing operation takes place when the user releases the mouse button after dragging the size box to the desired location on the screen. Resizing a virtual screen changes its display characteristics. Text that is subsequently displayed on the screen's windows may wrap differently.

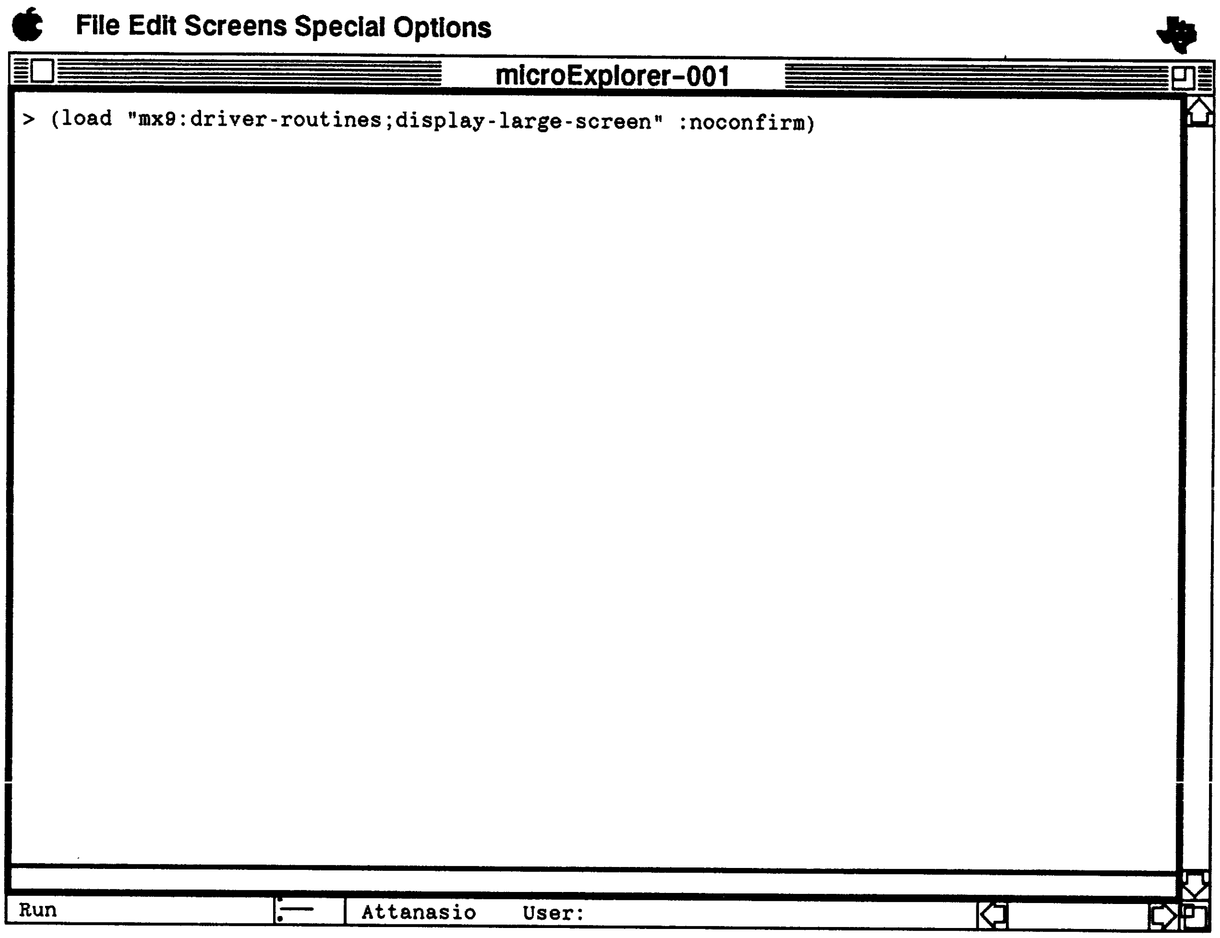
CAUTION: Increasing the size of a virtual screen by resizing requires additional Macintosh memory for proper operation and maximum performance. Refer to Section 4, Memory and Disk Requirements, for more information.

If you want to specify the default screen size of any additional virtual screens that you create, you can use the Set Screen Size command in the Macintosh Options menu. Several possible screen sizes appear on the Set Screen Size dialog window. The current default size for virtual screens is indicated by the presence of a Macintosh default button (a partially filled in circle). You can set the screen size either by clicking on the button next to one of the standard sizes displayed or by providing your own values in the width and height boxes. Then click on the OK button, and invoke a new version of the virtual screen by pressing TERM SYSTEM *x*, where *x* represents a standard key reference for a microExplorer window (such as L for Listener, E for Zmacs editor, P for Peek, and so on). The new screen will have the size you specified.

Status Line and Mouse Documentation Window

5.2.4 The status line and the mouse documentation window, both located at the bottom of the display, present information you will find useful when working on the microExplorer. Figure 5-1 shows a screen with a status line.

Figure 5-1 Screen Showing Status Line



```
> (load "mx9:driver-routines;display-large-screen" :noconfirm)
```

```
Run                    Attanasio    User:
```

At the lower left of the screen is the state information for the microExplorer. In Figure 5-1, the microExplorer is in the Run state.

To the right of the state is a set of run bars. The top run bar indicates the amount of current CPU activity. The middle run bar indicates current disk usage, and the bottom run bar indicates garbage collection activity.

To the right of the run bars is the current user's login name and the current package. When the file system is transferring a file, the current user's login name and the current package are replaced by the name of the file, the number of bytes transferred, and the percentage of the transfer completed.

The mouse documentation window (not shown in Figure 5-1) displays the operations currently available with the mouse, and information about mouse-selectable items.

Multiple Displays

5.2.5 As with other applications running on the Macintosh, the microExplorer is able to support multiple Macintosh physical displays. The Macintosh paradigm for multiple displays is as follows: there is a single very large display surface; individual displays are positioned (at boot time) on this display surface. Thus windows can be moved between displays by dragging them on the display surface in the normal way. By using a single display, the Macintosh allows surface windows to be enlarged or to be positioned such that they actually span multiple physical display devices—even if those devices are of completely different types (color, monochrome, third-party, and so on). A single Macintosh II can drive up to six displays.

The Keyboard

5.3 The microExplorer maps the functionality of an Explorer system keyboard onto the Apple Extended Keyboard. The microExplorer comes with its own keyboard overlay, showing Explorer keystroke designations over the equivalent keys on the Apple Extended Keyboard.

Many of the keystrokes mapped on the keyboard overlay perform the same or similar functions as Explorer system keystrokes:

Clr Input — Erase the current line of input

Clr Screen — Erase the screen and position the cursor at the top

Abort — Cancel the current operation and return to top level

On the Apple Extended Keyboard, two useful keys include the Delete key and the Help key.

The Delete key erases the last character you typed. This key is the equivalent of the Explorer's RUBOUT key.

By pressing the Help key or a keystroke sequence that includes the Help key, you can obtain helpful information about the various programs and utilities on the microExplorer.

The Help key is context-sensitive; that is, the type of help provided by the Help key can vary depending on the type of operation being performed at the time the key is pressed. Pressing the key in some utilities displays a menu of mouse-selectable help options; in other utilities, it displays a list of help options that you can choose by typing a specified letter.

For example, if you press the Help key after you have typed the opening parenthesis of a Lisp form in the Lisp Listener, the system displays general system help that suggests various alternatives. However, if you press the Help key while at the top level of most utilities (before typing input), the system displays a menu of the following items:

- Explorer Overview — Describes the Explorer environment (which the microExplorer closely resembles), how to move around the system, how to get started, and other available help features.

- System Menu — Displays the System menu and allows you to select an item.

- System Applications — Displays a list of applications available on the microExplorer.

- Term Key Help — Displays a help screen with Term key sequences and their functions.
- Application Help Option — Prints a message describing the application. The name of this help option is specific to the current UCL application (for example, Lisp Listener Help).
- Command Name Search — Allows you to search for commands that contain a specified substring.
- Command Display — Displays a list of the currently active universal and input editor commands.
- Command Type-in Help — Provides information about typing command names and Lisp forms.
- Command History — Displays a list of recently entered commands.
- Keystroke Search — Allows you to search for commands that contain a specified keystroke or keystroke sequence.
- Customization Menu — Provides a menu of items that allow you to customize the command interface.

Some other common uses of the Help key are as follows:

- CTRL-HELP — Produces a list of commands available in the input editor.
- HYPER-CTRL-HELP — In a utility that uses the UCL, produces a list of commands available in the utility.
- TERM HELP — Produces a list of keys you can use with TERM.
- SYSTEM HELP — Produces a list of keys you can use with the SYSTEM key to invoke a utility.

Histories

5.4 *Histories* are buffers that store information about previously used commands. Entries in histories can be used to reduce typing, to store intermediate results, and sometimes to move text from one utility to another. The three types of histories are the input history, the output history, and the kill history. All three types are accessible from the Lisp Listener.

Input History

5.4.1 The input history stores any Lisp form that you enter in the Lisp Listener. You can save keyboard entry time by retrieving previous forms and executing them again, or by making minor modifications and executing them again. Try each of the following keystrokes with your own input history. The following keystroke sequences allow you to access input histories:

- CTRL-C — Copies the most recent input history entry into the Lisp Listener input line.
- CTRL-*n* CTRL-C — Copies the input history entry specified by the number *n*, with 1 denoting the most recent entry, 2 the previous entry, and so on.

- META-C — When used immediately after CTRL-C, replaces what CTRL-C just copied with the second most recent input history entry into the Lisp Listener input line. When used immediately after CTRL-*n* CTRL-C, copies the input history entry prior to that specified by *n*. In either case, if you press META-C again without moving the keyboard cursor, the next most recent input history entry replaces the one currently displayed.

- STATUS — Invokes a menu containing the most recent entries from the input history. You can select one of the entries in the menu to copy it into the Lisp Listener input line. For example, if you have logged in and then logged out since the last cold boot operation, you can use the following sequence to log in again:

  1. Press the STATUS key to invoke the menu of the most recent commands. This menu contains your last login entry.

  2. Use the mouse to select your login command from the menu. This selection copies the command into the Lisp Listener input line.

  3. Press RETURN.

- META-STATUS — Displays a numbered list of the entries from the input history. You can use the number of the entry as an argument with CTRL-*n* CTRL-C to copy a specific entry from the input history into the input line.

Figure 5-2 shows a typical input history.

Figure 5-2 **Typical Input History**

```
Contents of input history:

 =>  1. (apropos 'reset :net)
     2. (print-disk-label)
     3. *package*
     4. (setq *package* user)
     5. (make-system 'indexor :noconfirm)
     6. (report-last-shutdown)
     7. (login 'clark 'lm t)
```

Output History **5.4.2** The HYPER-STATUS keystroke allows you to access output histories that store what the Lisp Listener returns. HYPER-STATUS invokes a menu of output history entries from which you can select an entry to copy into the Lisp Listener input line.

Kill History

5.4.3 The kill history stores any segment of text larger than a character that you mark for deletion while in Zmacs or that you delete while in the Lisp Listener. The following keystroke sequences allow you to access kill histories:

- CTRL-Y — Copies the most recent kill history entry into the Zmacs buffer input line. This action is commonly called *yanking*.
- CTRL-*n* CTRL-Y — Copies the kill history entry specified by the number *n*, with 1 denoting the most recent entry, 2 the previous entry, and so on.
- META-Y — When used immediately after CTRL-Y, replaces the entry copied by CTRL-Y with the second most-recent kill history entry into the Zmacs buffer input line. When used immediately after CTRL-*n* CTRL-Y, copies the kill history entry prior to that specified by *n*. In either case, if you press META-Y again without moving the keyboard cursor, the next most-recent kill history entry replaces the one currently displayed.
- META-CTRL-STATUS — In the Lisp Listener, displays a numbered list of the entries from the kill history. You can use the number of the entry as an argument with CTRL-*n* CTRL-Y to copy a specific entry from the kill history into the input line. This keystroke is not defined in Zmacs.

Universal Command Loop (UCL)

5.5 Most of the system utilities were built using UCL to provide a common interface to command-input, help, and mouse handling. UCL provides menus of commands from which you can select commands instead of typing command names or pressing keystroke sequences. The UCL also gathers and presents online documentation of commands. In addition, the UCL enables you to do the following:

- Have the system complete a partially typed item, such as a command, function, or flavor name
- Search for a specific command name or keystroke sequence
- Use input editor commands, which enable you to edit any time you type

The *Explorer Tools and Utilities* manual provides more information on how to use the UCL.

Searches

5.5.1 The UCL menus provide a quick way to find command names or keystroke sequences. If you press the HELP key in any utility built with the UCL, the system displays a basic help menu from which you can choose several mouse-selectable items that can help you find commands or keystroke sequences. You can use the Peek utility window to explore the following items:

- Command display — Displays available commands. You can select a command and execute it, receive more information about it, or display the keystroke sequence associated with it. You can also use the HYPER-CTRL-HELP keystroke sequence to invoke the command display.

- Command history — Lists the command name, keystroke sequence, and a short description of each command you have used. You can select a command and execute it, receive more information about it, or see the keystroke sequence associated with it. You can use this option to learn the keystroke sequence for a command executed from a menu. You can also use the HYPER-CTRL-P keystroke sequence to invoke the command history.

- Command name search — Searches for any command containing a word you specify. You can also use the HYPER-CTRL-N keystroke sequence to invoke the command name search. When you type a substring, the system searches for all commands containing that substring. For example, if you type `ed`, the command name search will list all commands with `ed` in them.

- Keystroke search — Searches for the command executed by the keystroke sequence you specify. You can also use the HYPER-CTRL-K keystroke to invoke the keystroke search. When you enter a keystroke sequence, a description of that sequence is displayed.

Completions

5.5.2 In the Lisp Listener, you can use several completion options. For items such as commands, flavors, and symbols, you can press a keystroke sequence to execute any of the following types of completion:

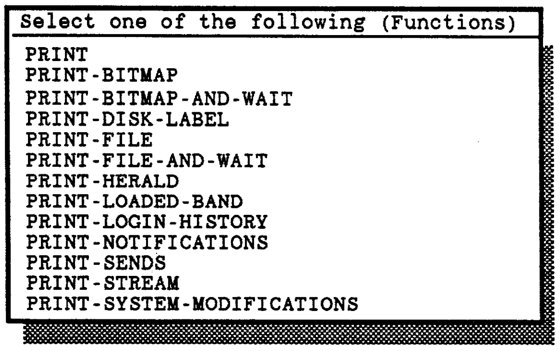
- Recognition — Tries to complete a word exactly as typed. Pressing ESCAPE completes the word if only one match is found, or displays the possible matches in the mouse documentation window if several matches are found. Pressing CTRL-/ displays a menu containing the possible matches. Pressing the space bar completes the word with the first match found.

- Apropos — Searches for a symbol containing, in the specified order, the letters you type. The symbol need not begin with an alphabetic character. Pressing SUPER-ESCAPE displays the possible matches in the mouse documentation window. Pressing SUPER-/ displays a menu containing the possible matches.

- Spelling — Searches for a symbol containing letters in the specified order you type or in a similar order. Pressing HYPER-ESCAPE displays the possible matches in the mouse documentation line. Pressing HYPER-/ displays a menu containing the possible matches. When running a completion on a word defined as a function, you must enter the word as a function; that is, the word must be prefaced with an opening parenthesis.

  For example, if you type the misspelled word `(primt` and press HYPER-ESCAPE (the command for spelling completion), the system changes the letters to PRINT. If you then press HYPER-/, the system displays a menu similar to that shown in Figure 5-3. As you box each item, a brief description of that function appears in the mouse documentation window. You can select one of the items and copy it into the input line.

Figure 5-3 **Functions Menu**



```
Select one of the following (Functions)
PRINT
PRINT-BITMAP
PRINT-BITMAP-AND-WAIT
PRINT-DISK-LABEL
PRINT-FILE
PRINT-FILE-AND-WAIT
PRINT-HERALD
PRINT-LOADED-BAND
PRINT-LOGIN-HISTORY
PRINT-NOTIFICATIONS
PRINT-SENDS
PRINT-STREAM
PRINT-SYSTEM-MODIFICATIONS
```

Miscellaneous Features

5.5.3 The microExplorer offers two miscellaneous features for use with its commands: the Redo command and numeric arguments.

Redo Command

5.5.3.1 *The Redo command, executed by pressing HYPER-CTRL-R,* repeats the last significant command. (Significant commands typically perform editing operations.) For example, if you have just used the Save command from the Customization menu in the Lisp Listener, you can immediately reselect it by pressing HYPER-CTRL-R.

Numeric Arguments

5.5.3.2 *With some editing commands, you can use numeric arguments to* repeat the commands a specific number of times. To use numeric arguments with most commands, you press CTRL-*n*, where *n* is a positive number, and then press the keystroke sequence for the command. For example, pressing CTRL-3 META-D deletes three words to the right of the cursor.

Other commands use the numeric argument to indicate whether the command is to be modified in a particular way. For example, in the Zmacs Copy File command, each of the numeric arguments 2 through 6 cause a different operation to be performed.

Input Editor

5.5.4 The input editor offers many keystroke sequences to help you enter input with a minimum of effort. Pressing CTRL-HELP invokes an online list of these keystroke sequences.

The input editor commands are valid in any Lisp Listener as well as in various windows, such as Peek and the Inspector. Also, Zmacs has its own input editor that is very similar to this one. Table 5-1 shows the input editor commands and where they are valid. The arrow keys shown in the table are the keys located to the left of the numeric keypad.

NOTE: For information about keystroke and character mapping between the Macintosh and the microExplorer environments, see Appendix B.

Table 5-1 Common Keystroke Sequences

Keystroke Sequence	Command Name	Description
Used in Input and Zmacs Editors:		
ESCAPE	Complete	When a match is found, performs an Auto Complete (see Space Bar)
CTRL-A or SUPER-←	Beginning of Line	Moves the cursor to the left margin of the current line
CTRL-B or ←	Backward Character	Moves the cursor one position to the left
META-B or META-←	Backward Word	Moves the cursor to the first character of the current word unless the cursor is already on the first character of a word or between words, in which case this command moves the cursor to the first character of the previous word
META-CTRL-B or META-CTRL-←	Backward Parenthesis	With the cursor immediately to the right of the closing parenthesis of a given set of parentheses, moves the cursor to the opening parenthesis (in other words, finds the matching parenthesis)
CTRL-D	Delete Character	Deletes the character highlighted by the cursor
META-D	Delete Word	Deletes the word (or fragment) from the position of the cursor onward to the right
CTRL-E or SUPER-→	End of Line	Moves the cursor to the right end of the line
CTRL-F or →	Forward Character	Moves the cursor one position to the right
META-F or META-→	Forward Word	Moves the cursor one word to the right (to the space before the next word)
META-CTRL-F or META-CTRL-→	Forward Parenthesis	With the cursor on the left parenthesis of a given set of parentheses, moves the cursor to the right of the closing parenthesis (in other words, finds the matching parenthesis)

Table 5-1 Common Keystroke Sequences (Continued)

Keystroke Sequence	Command Name	Description
CTRL-K	Kill Line	Kills input from the keyboard cursor to the end of the current line
CTRL-N or ↓	Next Line	Moves the cursor to the next line
CTRL-P or ↑	Previous Line	Moves the cursor to the previous line
CTRL-Q*	Quote Character	Enables you to insert the name of a command key rather than performing the command (For example, pressing CTRL-Q CLEAR INPUT inserts ⟨CLEAR INPUT⟩ into the input line rather than actually clearing the line.)
CTRL-T	Transpose Character	Exchanges the character highlighted by the cursor with the one to the left
META-T	Transpose Word	Transposes words on either side of the keyboard cursor
RUBOUT (Delete on the Apple Extended Keyboard)	Rubout Character	Deletes the character to the left of the cursor
META-RUBOUT	Rubout Word	Deletes the word to the left of the cursor
META-CTRL-RUBOUT	Rubout Parenthesis	Deletes input from the left of the cursor to the matching left parenthesis
META-SHIFT-< or HYPER-↑	Beginning of Buffer	Moves the cursor to the beginning of the buffer or file
META-SHIFT-> or HYPER-↓	End of Buffer	In Zmacs, moves the cursor to the end of the buffer; in the Input Editor, moves the cursor to the end of Zmacs and the beginning of the Lisp Expression
CTRL-O	Open Line	Inserts a carriage return to the left of the cursor

Note:

* CTRL-Q works in the Zmacs text buffer, but not in the minibuffer.

Table 5-1 Common Keystroke Sequences (Continued)

Keystroke Sequence	Command Name	Description
Used Only in the Input Editor:		
SUPER-ESCAPE	Apropos Complete	Searches for a symbol containing, in the specified order, the letters you type (the symbol need not begin with an alphabetic character); displays the possible matches in the mouse documentation window
HYPER-ESCAPE	Spelling Complete	Searches for a symbol containing letters in the specified order or a similar order; displays the possible matches in the mouse documentation window
META-CTRL-K	Kill to Next Parenthesis	Deletes the Lisp form to the right of the cursor
CTRL-/	List Completions	Displays a menu containing the possible matches for recognition completion
Space Bar (used after CTRL-/)	Auto Complete	Completes the word with the first match found for recognition completion (based on the commands listed on the Command Display of the HELP screen for utilities that use UCL)
SUPER-/	List Apropos Completions	Displays a menu containing the possible matches for apropos completion
HYPER-/	List Spelling Completions	Displays a menu containing the possible matches for spelling completion
CTRL-L	Refresh Screen	Clears the selected window and reprints the current input
HYPER-CTRL-N	Command Name Search	Brings up the menu to specify the commands to find
HYPER-CTRL-P	Command History	Displays previous significant commands in the scroll window
META-CTRL-HELP	Display Internal State	Displays the internal state of the input editor
CTRL-S	Position at Error	Yanks the previous input that contained a typing error and positions the cursor where the typing error occurred

Table 5-1 Common Keystroke Sequences (Continued)

Keystroke Sequence	Command Name	Description
Used Only in the Zmacs Editor:		
CTRL-V	Scroll Down	Scrolls the window backward
META-V	Scroll Up	Scrolls the window forward
CTRL-space bar	Set Mark	Sets the start of a region beginning with the cursor
CTRL-SHIFT-<	Mark Beginning	Marks a kill region from the position of the cursor to the beginning of the buffer
CTRL-SHIFT->	Mark End	Marks a kill region from the position of the cursor to the end of the buffer
CTRL-W	Kill Region	Kills a region of input marked by the user
CTRL-SHIFT-U	Undo	Undoes small changes from a recent edit

Online Documentation

5.6 In addition to the various help menus that you can invoke while in certain programs and utilities, the microExplorer provides several kinds of online documentation. This online documentation is designed to provide immediate help while you perform various operations.

Mouse Documentation

5.6.1 The mouse documentation window provides definitions of various mouse operations and lists the functions of the mouse buttons for any given operation at any given time.

The actual message in the documentation window at any given time depends upon which microExplorer facility is in use at the time as well as the specific location of the mouse cursor. For example, when the mouse cursor is located in the main screen of the Lisp Listener, the mouse documentation window displays the following message:

```
R:Bring up the System Menu
```

The letter `R` represents a single click of the right mouse button (which means to hold down the Apple key on your Apple Extended Keyboard and then click the mouse). In this case, the message indicates that a single click of the right mouse button invokes the System menu.

If the mouse cursor is located within the System menu, the mouse documentation window displays a definition of the currently boxed menu item and lists the mouse button that invokes the item.

Function Documentation

5.6.2 The Lisp language provides thousands of valid Lisp functions, each of which has a unique definition and a unique set of required and/or optional arguments. The microExplorer provides the following online documentation features that allow you to obtain immediate help when entering a Lisp function:

- Mouse documentation window — When you type a function name in the Lisp Listener and press the space bar, the mouse documentation window immediately displays the function name and a list of valid arguments for the function. For example, if you type `(print` and press the space bar, the mouse documentation line displays the following message:

  ```
  PRINT: (OBJECT &OPTIONAL STREAM)
  ```

- CTRL-SHIFT-A — The keystroke sequence CTRL-SHIFT-A provides an alternate means of obtaining the arguments of a given function. After you type a function in the Lisp Listener or editor, pressing this keystroke sequence displays a message identical to the one displayed in the mouse documentation window. For example, if you type `(print` and press CTRL-SHIFT-A, the following message appears on the main screen or in the minibuffer of the editor:

  ```
  > PRINT: (OBJECT &OPTIONAL STREAM)
  ```

- CTRL-SHIFT-D — The keystroke sequence CTRL-SHIFT-D displays the function name and its arguments followed by the function definition. For example, if you type `(print` and press CTRL-SHIFT-D, the following message appears on the main screen or in the minibuffer of the editor:

  ```
  > PRINT: (OBJECT &OPTIONAL STREAM)

  Print OBJECT on STREAM with quoting if needed, with a Return before
  and a Space after.
  ```

  If you try to enter an invalid or undefined Lisp function, the system returns a message indicating that the function is undefined. For example, if you type `(sink-the-bismarck` and press CTRL-SHIFT-D, the following message appears on the main screen or in the minibuffer of the editor:

  ```
  > Can't find a definition for SINK-THE-BISMARCK.
  ```

Variable Documentation

5.6.3 Online documentation of variables is similar to function documentation except that it is valid only in the Zmacs editor. The keystroke sequence CTRL-SHIFT-V provides this documentation. The following are typical examples of the use of the variable documentation feature:

- Variables for which online definitions are provided — When you type one of these variables in a Zmacs buffer and press CTRL-SHIFT-V, a typeout window provides a message that gives the current state of the variable and its definition. For example, if you type `*package*` in a Zmacs buffer and press CTRL-SHIFT-V, a message similar to the following appears in a typeout window:

```
*package* has a value and is declared special
The current package, the default for most package operations
including INTERN.
```

- Variables for which online definitions are not provided — When you type one of these variables in a Zmacs buffer and press CTRL-SHIFT-V, a message giving the current state of the variable appears in the Zmacs minibuffer. For example, if you type `tv:selected-window` and press CTRL-SHIFT-V, a message similar to the following appears in the Zmacs minibuffer:

```
TV:SELECTED-WINDOW has a value and is declared special by file
SYS:WINDOW;TVDEFS.#

The currently selected window, or NIL.
```

- Undeclared variables — When you type an undeclared variable in a Zmacs buffer and press CTRL-SHIFT-V, the minibuffer momentarily displays a message indicating that the term has not been declared a variable. For example, if you type `still-water` and press CTRL-SHIFT-V, a message similar to the following appears in the minibuffer:

```
STILL-WATER is not a declared variable.
```

FONTS 6

Introduction

6.1 Because the Macintosh does all actual drawing for the microExplorer, fonts are represented and handled differently in the microExplorer environment than they are in the Explorer system. These differences may affect applications in complex ways, especially applications that manipulate fonts directly or ones that need access to custom Macintosh fonts. An understanding of the issues and their implications can be important for successful microExplorer application development. This section describes font handling on the microExplorer, font paradigm differences between Explorer and microExplorer, and how to add and modify fonts in the microExplorer environment.

microExplorer Fonts and Drawing

6.2 A window system must maintain two types of information about its fonts:

- Sizes — Height, width, and so on
- Glyphs — Small bitmaps used to draw each character

In the Explorer system, fonts are both created and used in the Explorer window system environment. Hence, the Explorer window system stores both the glyph and the size information in the Explorer font object.

The microExplorer, however, uses Macintosh fonts. When the microExplorer Lisp software writes to a window, the Macintosh QuickDraw window manager does the actual drawing using Macintosh fonts. The font information originates and is used for drawing in the Macintosh environment. The microExplorer's window system only keeps a copy of some of the font sizing information needed to instruct the Macintosh on how to lay out the text. As a result, the font object in the microExplorer environment is only a copy of the Macintosh font's sizing information.

For example, the cptfont font is the default font for both the Explorer and the microExplorer. On the Explorer system, the formal name of this font is **fonts:cptfont**. The value of this symbol is a font object describing both the sizes and glyphs of all characters in the font. On the microExplorer system, software still references the **fonts:cptfont** symbol, but the font object is different: it only contains position and sizing information such as character width, kerning, and font baseline. The actual glyphs for the cptfont characters are stored on the Macintosh side in the microExplorer application's resources.

Font Mapping

6.3 When drawing using a particular font on the microExplorer, the microExplorer window system must specify which Macintosh font should be used for the drawing operation. The microExplorer looks up a Lisp-to-Macintosh font mapping, which is defined by the **mac:*mac-font-translation-table*** variable. To understand the format of the font mapping specification, an explanation of Macintosh font families is required.

On the Explorer system, each size and style of font has a separate font name (always a symbol in the FONTS package) representing a distinct font object. For example, the Explorer has four sizes in the Times Roman typeface, each represented by a different font name (tr8, tr10, tr12 and tr18). There are additional font names for various styles in this font, such as bold, italic, and combination bold italic (tr12b, tr12i, tr12bi). If you want a style or point size not provided (an italic version of tr18 for example), you must create a new font by using the Font Editor. This font is stored in a new font symbol (**fonts:tr18i** in this example).

In contrast, the Macintosh classifies a font by three characteristics: family (such as Times, Helvetica, Chicago, and so on), point size, and style (bold, italic, underline, and so forth). Each font family represents all different sizes and styles in a given font typeface and has one or more collections of actual glyphs or representative fonts. A feature of Macintosh font handling is that it can algorithmically scale point sizes and change styles. If an application requests a point size in a font family for which there is no representative font, the Macintosh will scale the existing representative as best it can. The same is true for different styles of a font, such as boldface and italic. If the requested style is not available in a font family, the Macintosh will apply a styling algorithm to an existing representative, such as thickening a font to create a bold version or slanting it to create a pseudo-italic version.

Algorithmic scaling provides great flexibility but has certain consequences. The main disadvantage is speed: the scaling is done in real time so drawing may be slower. Also, some scaling or styling operations may distort a font's appearance. Scaling a large font to half size and scaling a small font to double size both work fairly well. However, scaling a 10-point font to 8 points is sometimes a problem. Also, a pixel round-off error can cause one character to change appearance depending upon its position in the line. For example, in a word such as bookkeeping, the adjacent character pairs may look different because of dynamic scaling.

However, the designer of a Macintosh font always has the option to provide representatives for the most commonly used sizes and styles in a font family (for efficiency) or to provide representative fonts for some of the difficult scaling/styling operations that might be performed (for legibility). In this way, the Macintosh can represent a large range of font sizes and styles by using only a few sets of representative glyphs while providing the application developer a great deal of flexibility.

The Macintosh's font search and substitution algorithm is designed so that some valid font will always be found for a font reference. If the font family requested cannot be found, the default application or system font will be used. This guarantees that some characters will always be displayed on the screen but can lead to subtle display problems if the fonts in the Macintosh system are not properly installed.

When mapping an Explorer font, the main question is which Macintosh font family to use. The size and style must also be specified; however, if no representative font exists in the Macintosh font family for that point size and style, one will be created dynamically by the Macintosh Font Manager. The **mac:*mac-font-translation-table*** mapping table contains an entry for each Explorer font symbol specifying the Macintosh font family, point size, and style that will be used for the Explorer font.

mac:*mac-font-translation-table* Variable

The value of this symbol is a list of mapping specifications for Explorer fonts. Each specification is a list containing the name of the Explorer font, the family name of the Macintosh font to use to represent the Explorer font, the point size to use, and any style modifiers such as boldface, italic, and so forth.

Examples: The following example maps `fonts:tr8` to the Macintosh font family Times in 9 point:

```
(fonts:tr8 mac:timesfont 9)
```

The next example maps `fonts:tr10i` to the Macintosh font family Times in 10-point italic:

```
(fonts:tr10i mac:timesfont 10 (symbol-value mac:italicbit))
```

The following example maps `fonts:tr12bi` to the Macintosh font family Times in 12-point bold italic:

```
(fonts:tr12bi mac:timesfont 12 (symbol-value mac:boldbit)
    (symbol-value mac:italicbit))
```

Font Mapping Consistency

6.3.1 On the Explorer system, a standard set of fonts is distributed in every load band. Therefore, if your applications use only standard Explorer fonts, they will always look the same regardless of which Explorer they run on.

A feature of the Macintosh, however, is that fonts can easily be added to and removed from the environment. Third-party vendors offer a wide selection of alternative fonts. Furthermore, if the font family requested by an application is not available, the Macintosh Font Manager will use one of its established defaults (and the defaults themselves can be altered by applications). This prevents an application from getting an error for requesting a nonexistent font but also may lead to text being displayed in unexpected ways.

As you move your application from machine to machine, you should be aware that different fonts may be available in the different environments. Any custom fonts your application uses will have to be installed on all machines running your application or else made part of the microExplorer application.

In addition, the Explorer-to-Macintosh font mappings can be altered programmatically. That is, it is possible to add and modify entries in the **mac:*mac-font-translation-table*** mapping table. As a consequence of providing this flexibility, application developers must take care to ensure that their desired font mappings do not conflict with those of other applications. In general, this means that an application should not alter any of the standard mappings provided in the **mac:*mac-font-translation-table*** alist.

If text is not displayed as you think it should be on your microExplorer, first check to see that all the Macintosh fonts required by the microExplorer are installed in your Macintosh System resource file (which is the System file under the System folder of your startup disk). A list of these fonts appears in paragraph 6.3.2, Macintosh Fonts Required by the microExplorer. If all the correct fonts are installed, it is possible that the microExplorer font mappings have been changed. You can use the **mac:undo-mac-font-translations** function to restore the microExplorer's original font mappings.

mac:undo-mac-font-translations &optional (*remap-fonts* **t**) (*fixup-data-structures* **nil**) — Function

Restores the original microExplorer Explorer-to-Macintosh font translations. Repeated calls will undo the undo. When *remap-fonts* is true (the default), this function verifies the current font object information with the Macintosh.

When *fixup-data-structures* is true, this function also tries to fix up font references in existing windows and symbols. This process takes awhile, so the default is **nil**.

This function returns two values:

- The translation table (after the undo)
- A flag indicating whether it is the original version or a user-modified version (one of the keywords **:original** or **:user-modified**)

Macintosh Fonts Required by the microExplorer

6.3.2 Macintosh supplies a large number of predefined fonts in the Fonts file under the Font/DA Mover folder. Only a few of these fonts are installed as part of the standard System resource file Macintosh includes with its system releases. In addition to these standard fonts, several fonts are required for the default Explorer-to-Macintosh font mappings performed by the microExplorer window system. Texas Instruments installs these fonts in Macintosh systems delivered on hard disks. For Macintosh systems that are not supplied by Texas Instruments, the user must install these fonts.

The fonts necessary for microExplorer font translations are listed under the Required Fonts column in the listing that follows. An asterisk indicates that the font must be added to the standard fonts installed by Macintosh. The Optional Fonts column lists other predefined Macintosh-supplied fonts that are available for installation by users but are not usually part of a Macintosh system.

This list is not a comprehensive list of Macintosh fonts available to users since the Macintosh system, like the Explorer system, allows the addition of user-defined fonts.

Also note that the standard set of Macintosh fonts may change in the future. The list may be updated by information in your microExplorer Software Release Information document.

Required Fonts	**Optional Fonts**
Chicago 12	Athens 18
Courier 10	Cairo 18
Courier 12	Courier 9
Geneva 9	Courier 14
Geneva 10	Courier 18
Geneva 12	Courier 24
Geneva 14	Helvetica 14
Geneva 18	Helvetica 18
Geneva 20	Helvetica 24
Geneva 24	London 18
Helvetica 9 *	Los Angeles 12
Helvetica 10	Los Angeles 24
Helvetica 12	Mobile 18
Monaco 9	New York 9
Monaco 12	New York 10
Symbol 12 *	New York 12
Times 9 *	New York 14
Times 10	New York 18
Times 12	New York 20
Times 18 *	New York 24
Venice 14 *	San Francisco 18
	Symbol 9
	Symbol 10
	Symbol 14
	Symbol 18
	Symbol 24
	Times 14
	Times 24

Note:
*** The font was added by Texas Instruments to the standard fonts installed by Macintosh.**

Missing Font Mappings

6.3.3 The **mac:*mac-font-translation-table*** maps most, but not all, of the fonts normally distributed on an Explorer system. Some of the large display fonts that have no obvious counterpart on the Macintosh are missing. Furthermore, custom fonts you may have created for your Explorer application are not in this map either. You can examine the **mac:*mac-font-translation-table*** alist to determine which font mappings are currently in effect in your environment.

If an application refers to a font that is not in the mapping table, the user will be given an opportunity to supply a mapping for the font. Typically, the font is in an application window's font map; when the window is exposed, a **:parse-font-descriptor** message is sent to the window's screen with a font name that is not in the **mac:*mac-font-translation-table*** alist. The user is then presented with a Choose Variable Values (CVV) window and asked to choose a mapping. The user can simply click on the DEFAULT margin box to map the unknown Explorer font to the microExplorer default font (usually cptfont), or the user can choose a specific Macintosh family name, point size, and style to which to map the Explorer font. Clicking left on the font slot of the CVV window presents a menu of available Macintosh fonts (those font families recognized by the microExplorer).

You can suppress the presentation of this CVV menu by setting the **fonts:*use-default-mac-font*** variable to true. Then, the Macintosh font specified by the **mac:*default-mac-font*** variable is used. This is equivalent to automatically clicking on the DEFAULT margin box of the CVV menu.

Font Character Mapping Differences

6.3.4 The Explorer, the microExplorer, and the Macintosh systems all support the full set of 96 ASCII printing characters. However, there is some divergence among the other 160 characters available in an 8-bit byte. The original Lisp machines supported 128 printing characters by substituting printable characters for the 32 ASCII control characters that are normally non-printing. These 128 characters constitute the so-called Lisp machine character set.

Both the Explorer and the Macintosh systems use the character codes above 127 to represent an international symbol set. These two symbol set extensions are similar but not identical. The choice of symbols is slightly different, and their assigned character codes are different.

On the microExplorer system, only the cptfont fonts (cptfont, cptfonti, cptfontb, and cptfontbi) and bigfnt are defined to have the traditional Lisp machine character set. The Macintosh fonts used to represent the cptfonts and bigfont are not among the predefined Macintosh fonts. Rather, they are custom Macintosh fonts containing the standard Lisp machine character set, and they are stored in the microExplorer application's resources. Currently, all other Macintosh fonts used to map Explorer fonts are among the standard Macintosh system fonts and hence use the Macintosh character set definition. When these fonts are used on the microExplorer, Lisp machine extension character codes are mapped to the closest character in the full Macintosh standard character set. However, some fonts do not provide the full Macintosh character set, and even when they do in many cases there is no exact match for the glyph. As a result, some special characters in fonts other than the cptfonts and bigfont will not be displayed as they would be on an Explorer system.

These character mapping differences have the following consequences for the application programmer:

- If your code only uses the 96 ASCII printing characters plus newlines, tabs, and formfeeds, then you will not have any problems.

- If you must print or display something that needs some of the traditional Lisp machine extension characters (such as the horseshoes used in printing pathname objects), then you must use one of the cptfonts, bigfont, or a custom font that maps to the microExplorer application's cptfont or bigfont.

- If you want to use a character from the Lisp machine character set extension but in some font other than a cptfont or bigfont, or if you want to use a character from the Explorer's international character set extension, a glyph that matches the corresponding Explorer glyph may not be available in the Macintosh font. You should first run a test to see if the glyphs you need are available (a good way to do this is by using META-X List Fonts in Zmacs).

- If the glyphs you must display are not available in a particular font, your best solution is to create a new Macintosh font that has the glyphs you want and then add a new font to the microExplorer environment that maps to your font. The new Macintosh font can be created by copying and modifying an existing font using a Macintosh font-editing application or the ResEdit facility. The procedures for adding and modifying existing fonts are outlined in later paragraphs.

CAUTION: Do not modify the glyphs of predefined Macintosh fonts. The results of doing so will be seen by all applications using that font, not just the microExplorer.

Static Versus Dynamic Font Paradigms

6.3.5 In the Explorer window system, every font is represented by a symbol in the FONTS package that is bound to a unique font object (a structure of type **w:font**). When an application decides to add font *foo* to its window's font map, the font object value of the symbol **fonts:***foo*, not the symbol itself, is placed into the window's font map. On the Explorer system, this loss of a level of indirection insulates the application's window from any future changes in the design of the characters of the font *foo*. Any calculations the application might make about the sizes of panes based on character sizes are unaffected by the creation of a new font object to describe a changed font *foo*. The window will continue to point to the old structure, which is a self-contained description of the *foo* font.

In the microExplorer system, all drawing is done by the QuickDraw toolkit on the Macintosh. Consequently, the font description that is actually being used to put pixels on the screen is controlled by the Font Manager on the Macintosh. Furthermore, fonts can be highly dynamic in the Macintosh environment. Font descriptions can be freely added, modified, and deleted. The microExplorer Lisp software still has font symbols bound to font objects, but these font objects only need to contain the font sizing information required by the microExplorer window system to calculate character positions on screens. Moreover, this sizing information is only a local copy of the corresponding information kept by the Macintosh Font Manager. The microExplorer system builds the font object from information provided by the Macintosh when a **:parse-font-descriptor** message is sent to a microExplorer screen. If the font description on the Macintosh changes, another **:parse-font-descriptor** must be performed if the changes are to be seen by the microExplorer window system.

For example, if a microExplorer application programmer wants to use microExplorer font *foo* for which a Macintosh font mapping exists, then the programmer uses the following form:

```
(send w:default-screen :parse-font-descriptor 'fonts:foo)
=> #<FONT MAC-foo ...>
```

This form returns the **font** object to be used in the remainder of the application. If you examine this **font** object, you will see that it is named **mac-***foo*. The **mac-** prefix identifies a converted font in which the **font** object contains only the character size information and not the glyphs.

In particular, this converted **font** object contains **w:font-char-height**, **w:font-char-width-table**, and **w:font-chars-exist-table** slots. Its **w:font-char-width** value is the width of the font's lowercase m. This size information is sufficient for the microExplorer's window system software to format text on a window, but only the Macintosh needs the glyphs for drawing characters.

CAUTION: When designing a microExplorer application, you should take care to work solely in terms of symbolic font names rather than font objects. In particular, you should not store font objects in data structures that will be saved in the load band. Instead, store the font name symbols in the data structures and then use :parse-font-descriptor at run time to obtain a working copy of their font objects. This programming convention will help ensure that your application is not adversely affected if the Macintosh font is modified.

When a Macintosh Font Changes

6.3.6 Because the microExplorer font objects are only copies of Macintosh font information, the microExplorer window system can become confused if a Macintosh font is modified. For example, if a microExplorer window is using **fonts:medfnt** as its default font, the window's font map contains an object such as `#<FONT MAC-MEDFNT 37474017>`. This font is mapped to the Macintosh Monaco font in 12 point. If the 12-point Monaco font is modified (for example, with a Macintosh font-editing application) after the microExplorer application has captured the font information via a call to **:parse-font-descriptor**, then the application's concept of the location of the characters it writes in the window will be wrong. Many times the microExplorer window system must identify the precise pixel where a character is located (rubout handling, for example), and in this situation the window system will be unable to do so.

The Macintosh fonts named in the **mac:*mac-font-translation-table*** alist are rarely changed. Hence, the font objects in the microExplorer environment usually match the Macintosh descriptions exactly. However, an application designer may wish to introduce a refined design for a particular size and style of some Macintosh font to improve its visual effect. When this is done, special steps must be taken to propagate the new font description to all its users on the microExplorer.

When you know a Macintosh font has changed and you wish to propagate its new description to the microExplorer, you can use the **mac:remap-all-fonts** function to cause **:parse-font-descriptor** to forget that it has already obtained font descriptions and get them again the next time it is called. Then, any windows created after invoking this function will get the new font descriptions. The **:parse-font-descriptor** method checks the new font description just obtained from the Macintosh against the corresponding existing microExplorer font description; if they are identical, the existing microExplorer font description is left in place. Only a font that has changed on the Macintosh since its description was last brought over to the microExplorer is converted to a new font object.

This new font structure returned by **:parse-font-descriptor** is initially pointed to by only the font's symbol. All windows that were using the font still point to the font's old font structure. The **mac:fixup-font-refs-in-screens-and-windows** function must be called to propagate the font change into the font maps, label descriptors, and item lists of all screens and active windows. Existing deactivated windows are cleaned up automatically when they are activated. Simple font references in special symbols in all packages are updated by the **mac:clean-up-existing-font-refs** function. These operations are time-consuming so the application should decide when they would best be done.

Note that no other functions have been supplied to find and fix font references buried in arbitrary data structures such as lists or other window instance variables. When such direct references to font objects exist, the application must provide routines such as the ones described to catch the references and make sure they are updated to use any new font descriptions.

Adding a New Macintosh Font to the microExplorer

6.4 The following paragraphs describe how to add a new Macintosh font to the microExplorer environment.

Installing the Font on the Macintosh

6.4.1 The first step is to identify the Macintosh font you want to use. This font can be one of the following:

- One of the Macintosh fonts in the Fonts file under the Font/DA Mover folder that is not yet installed in the System resource file
- A new font created by using ResEdit or other font-editing Macintosh applications

Install the new font in the System resource file located in the System folder of your boot device. You should not install fonts in the microExplorer application file if the font will be used for printing. Doing so creates the possibility of crashing the Macintosh when printing a file or bitmap.

Installing the font can be done in the following ways:

- The Font/DA Mover can be used to install a font in the Macintosh System resource file provided that the font is stored in a font file such as the one in the Font/DA Mover folder.

  Note that the Font/DA Mover will not open any arbitrary file to look for fonts, only special font files. However, most font-editing applications will store fonts in files of this special form.

- Some font-editing applications will allow you to install your font directly in the System resource file.
- ResEdit can be used to copy the resources associated with your fonts to the System resource file. The following paragraphs discuss these resources and how to copy resources with ResEdit.

Once you have installed the font in the microExplorer application or in the System resource file, the correct mapping to this font must be established in the Lisp window system environment. This mapping process is described in paragraph 6.4.4, Mapping the Font in Lisp.

Font Resources and ResEdit

6.4.2 A Macintosh font consists of two related resources: one resource of type FOND and one or more resources of type FONT. The FOND resource represents a group or family of fonts, usually of different sizes and styles. The FOND resource contains the font family name, the font family ID, and a list of one or more fonts belonging to the family. This list of fonts is actually a list of FONT resources representing individual fonts in the family. Each FONT resource contains the actual glyphs for the font's characters. For example, the Macintosh Courier fonts have a FOND ID of 22 that represents six members of the Courier family (of point sizes 9, 10, 12, 18, 20, and 24). Each of the different point sizes has glyphs stored in a FONT resource.

To copy a font from one application to the System resource file by using ResEdit, you must copy both the FOND resource for the font family and all the FONT resources for the fonts in the family. If you are unfamiliar with using the ResEdit application, paragraph 6.4.3, Using ResEdit, describes the procedure step-by-step. A copy of the ResEdit application is shipped with the microExplorer in the microExp:HyperLisp: folder.

Note that the font family ID for the font you are installing must not conflict with any of the microExplorer's own font family IDs and in general should not conflict with any of the font family IDs in your Macintosh System resource file (unless you intend for your font to shadow the system font with the same ID). To ensure that your font family ID is unique, use ResEdit to view the already installed FOND IDs in the System resource file. If your font's FOND ID is already in use, you can use ResEdit to change it before copying the font's resources.

Using ResEdit

6.4.3 The ResEdit application can be found in the HyperLisp folder under the microExp folder. Once launched, ResEdit displays a stack of scrollable windows, one per disk volume, showing the files and top-level folders on each volume. To move a font into the System resource file, perform the following steps:

NOTE: Do not edit the booted System file. Boot the Macintosh using another System file or edit a copy of the booted System file.

1. Click on the volume window for the volume where your new font resides.

   a. Double click on folders until you find the file containing your font.

   b. Double click on this file to display a scrollable menu of resource names attached to the file.

   c. Scroll the menu, if necessary, until you see the file's FOND and FONT resource names.

   d. Move this ResEdit window to an unobstructed portion of your screen by dragging on its title bar.

2. Double click on the FOND resource name. This displays a menu of all FONDs in this file showing their names and IDs.

   a. Identify the font you want to move by its name, and note its family ID number.

   b. Drag this window also to an unobstructed portion of your screen.

3. Click on the volume window where the System folder resides. Then double click on the window to display the files within the folder.

4. Double click on the System file to display the names of all its resources.

   a. Scroll the menu to view the FOND resource name.

   b. Drag this window to an unobstructed portion of your screen.

5. Open the System file's resource name to display all the FOND resources. This displays a menu of all font family names and IDs in the Macintosh system. See if your font family ID is unique by checking it against this display.

6. If your font family ID is unique, proceed to step 7. Otherwise, return to your font file's menu of FONDs by clicking on that window.

   a. Select the appropriate FOND name/ID line by clicking on it once.

   b. Now use the Duplicate operation under ResEdit's Edit menu to make a copy of the FOND with a different family ID.

   c. Repeat step 4 to ensure that your font family ID is unique in the System FONDs. If not, repeat this step until a FOND with an appropriately unique ID is generated.

7. Return to your font file's menu of FONDs by clicking on that window.

   a. Select the appropriate FOND name/ID line by clicking on it once.

   b. Now use the Copy operation under ResEdit's Edit menu to store a copy of the FOND into a hidden buffer.

8. Return to the System file menu of FONDs by clicking on that window. Now retrieve the copy of your FOND by using the Paste operation from the Edit menu. The new FOND now appears on the list of FONDs.

9. Now each of the FONT resources associated with your FOND must be copied individually to the System file.

   a. Return to your font file's menu of FONDs by clicking on that window.

   b. Double click on the appropriate FOND name/ID line to display a menu of the FOND's information.

   c. Scroll down until you see a line indicating the # of font entries. (Note that this number is zero-based.) Following this line is a group of fields for each of the fonts in this family.

   d. Write down the Res ID value for each of the fonts. These values are the FONT resource IDs you need to copy.

10. Return to your font file's menu of resource names by clicking on that window.

    a. Double click on the FONT item to display a menu of all FONT resources.

    b. Select one of the FONT resources you need to move by clicking once on the line containing the appropriate ID.

    c. Now use the Copy operation under the Edit menu to store a copy of the FONT into a hidden buffer.

11. Return to the System file menu of resource names by clicking on that window.

    a. Double click on the FONT item to display a list of all System file FONTs.

    b. Now retrieve your FONT by using the Paste operation from the Edit menu. The FONT now appears on the list of the System file FONTs.

12. Repeat steps 10 and 11 until all the FONTs in your FOND have been copied.

13. Select the Quit operation on ResEdit's File menu to exit the resource editor.

    a. Answer YES to the question `Save "System" before closing?`.

    b. If you had to generate a FOND with a unique ID, you will also be asked the question `Save` *<name of your font's file>* `before closing?`. Answer NO to this question.

Mapping the Font in Lisp

6.4.4 Once the new font has been installed in the microExplorer application or in the System resource file, its presence must be made known to the microExplorer window system. Start by writing down the family name and family ID of your font. (Most Macintosh font-editing applications will provide a mechanism for viewing the font family ID, or you can use ResEdit to find this information.)

Assume the Macintosh font name is MyMacFont and its ID is 350. A mapping between a font name in the microExplorer environment and this font must now be established. The font name used in Lisp does not have to be the same as the Macintosh font name. The only restriction is that the font name should be a symbol in the FONTS package. Assume the font name you want to use in Lisp is **fonts:mylispfont**.

The following code adds your font to the list of known Macintosh fonts:

```
;;; Define Macintosh font family name and ID number as a Lisp constant.
(defconstant mac:mymacfont 350.)

;;; Add the new constant to the list of known Macintosh font families.
(push-end "MyMacFont" mac:*mac-font-list*)
(push-end 'mac:mymacfont
          (get 'mac:*mac-font-list* :mac-font-names))
```

Next, a mapping must be established between the Lisp symbol **fonts:mylispfont** and the Macintosh font. You can do this by using code similar to the following:

```
(push `(fonts:mylispfont mac:mymacfont point-size
      ,optional-modifier ... ,optional-modifier)
      mac:*mac-font-translation-table*)
```

In this example, *point-size* is a number that represents a point size (normal sized text is 12 point), and the *optional-modifiers* are zero or more of the following symbols:

mac:boldbit	**mac:shadowbit**
mac:italicbit	**mac:condensedbit**
mac:underlinebit	**mac:extendedbit**
mac:outlinebit	

Note that if your Macintosh font family contains more than one font (of different point sizes or styles), you will normally want to establish an explicit mapping from a Lisp font name to each size or style. For example, if your Macintosh font family had explicit fonts for sizes 10 point and 12 point and also for 10-point bold and 10-point bold italic, you might use the following code fragments:

```
(push `(fonts:myfont-10 mac:mymacfont 10.)
      mac:*mac-font-translation-table*)
(push `(fonts:myfont-12 mac:mymacfont 12.)
      mac:*mac-font-translation-table*)
(push `(fonts:myfont-10-bold mac:mymacfont 10. ,mac:boldbit)
      mac:*mac-font-translation-table*)
(push `(fonts:myfont-10-bold-ital mac:mymacfont 10.
      ,mac:boldbit ,mac:italicbit) mac:*mac-font-translation-table*)
```

Now examine the list associated with the **mac:*mac-font-translation-table*** symbol to ensure your mapping has been added. Explorer fonts in this translation table are normally converted to Macintosh fonts only at boot time so that later changes (such as the one just made) will not be seen. Once you perform a **disk-save**, your change will be effective when the new load band is rebooted. To check your change before disk-saving, however, you can enter the form `(mac:remap-all-fonts)`. If you then do META-X List Fonts in Zmacs, your new font name is listed. Clicking on the listed font name displays the characters of your font.

After the previous steps have been performed, the **fonts:mymacfont** symbol contains a new font object that describes your Macintosh font. This object will not contain bitmaps for the actual character glyphs but will have all necessary information concerning character widths and kerning, font baseline, and so forth.

Modifying an Existing Font Mapping

6.4.5 If you want to change an existing font mapping rather than adding a new font, you would first use a form such as the following to modify the mapping specification for the font:

```
(setf mac:*mac-font-translation-table*
      (remove (assoc 'fonts:mylispfont-bold
                     mac:*mac-font-translation-table*)
              mac:*mac-font-translation-table*))

(push '(fonts:mylispfont-bold mac:mymacfont 14 ,mac:boldbit)
      mac:*mac-font-translation-table*)
```

This form changes the **mac:*mac-font-translation-table*** alist. Next, you should either call **mac:remap-all-fonts** or **disk-save** and reboot in order for new font references to see the changes. Either way will put a new font object on your Lisp font symbol. In addition, you must take special care to ensure that any existing references to the old font object are updated.

When you modify the translation table, font references may already have been made to the now outdated font object and this object may have been saved in existing window objects. Likewise, the old font object may be stored in global variables other than the font symbol and in other miscellaneous data structures used by an application. This can occur if an application manipulates the font object directly rather than the font symbol as is recommended on the microExplorer. In this case, every time you expose one of these windows or use the application, you will still see the old font.

Some of the cached font object references can be rectified by using the Lisp functions **mac:fixup-font-refs-in-screens-and-windows** (to correct existing windows and their associated structures) and **mac:clean-up-existing-font-refs** (to clean up font references in special symbols in all packages). This second function examines all symbols in all packages so it takes awhile. Even these routines cannot find font references buried arbitrarily in various application data structures. Cleaning up such references is the responsibility of the application programmer.

Remember that this fix-up problem can be avoided in application code by saving font symbol names in application data structures rather than the font object directly. Then at run time, the application can use the following form to initialize the font symbol to the current Macintosh font information before the font is first used:

```
(send w:default-screen :parse-font-descriptor fonts:mylispfont)
```

To summarize, when modifying an existing font mapping, all three of the following forms should be executed after the **mac:*mac-font-translation-table*** mapping entry has been altered:

```
(mac:remap-all-fonts)
(mac:fixup-font-refs-in-screens-and-windows)
(mac:clean-up-existing-font-refs)
```

7 CUSTOMIZING YOUR ENVIRONMENT

Introduction

7.1 When you first log in to your microExplorer, its responses are set to certain default values. For example, the maximum number of entries saved in the kill history is set to 16, and the default prompt is set to a greater-than symbol (>). As you use the microExplorer, you may wish to modify its responses to suit your preferences. This section describes how to customize the microExplorer environment.

The Profile Utility

7.2 The Profile utility provides a simple way to change the microExplorer environment. This utility provides the following capabilities:

- Allows you to change your environment for the current session only
- Allows you to create a login-initialization (login-init) file that sets up the same customized environment each time you log in and returns the environment to its default condition when you log out

To use the Profile utility, you must be logged in; to create a login-init file, you must have an existing login directory in the Macintosh file system. The login directory is the directory with the same name as your login name.

NOTE: Because profile login-init files can exist in either the microExplorer or the Explorer environments, you may see numerous variables in the profile utility window that are not applicable in the microExplorer environment (such as **profile:keyclick-state** and others). These variables have been retained to allow you to maintain a single profile for use on multiple systems. If you are creating a profile for use only on a microExplorer, ignore these options.

Accessing the Profile Utility

7.2.1 To access the Profile utility, select the Profile item on the System menu. The Profile utility window that appears consists of a choose-variable-values menu labeled `Variables currently displayed`, an Actions menu, and a menu of types of variables.

When you first invoke the Profile utility, the menu displayed at the top is the Important Variables menu. The microExplorer tells you which menu is displayed by highlighting the name of the menu in reverse video in the choose-variable-values menu. The Profile utility allows you to change the values of variables and/or execute commands.

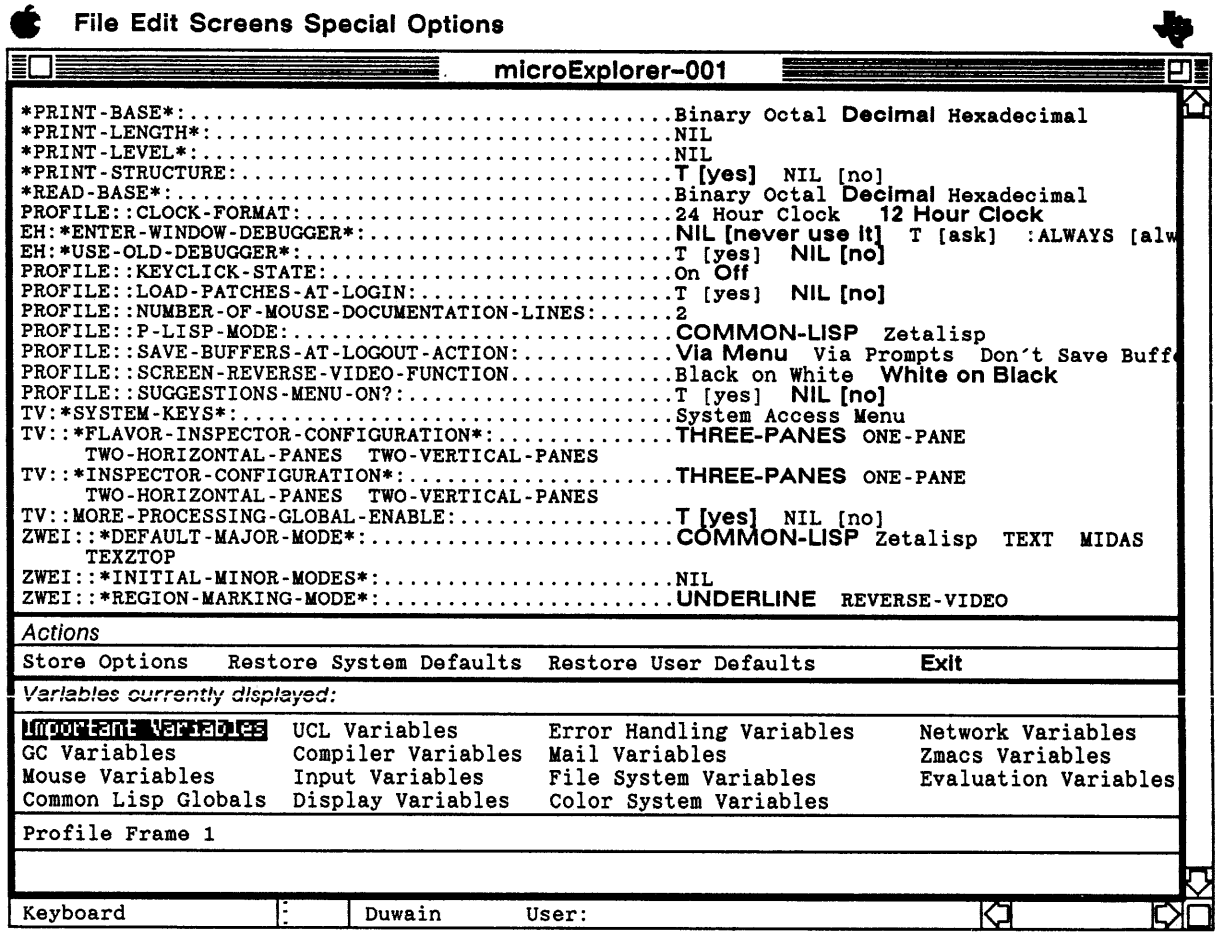
Accessing Variables

7.2.2 Variables in the choose-variable-values menu are grouped according to the areas they affect (Input, Display, Compiler, and so on). The Important Variables group includes frequently changed variables from several other groups.

To edit a group of variables, select the name of the group in the bottom menu. The selected group is then displayed in reverse video, and the top pane of the window displays the variables that you can change.

You can edit the values of variables listed in the display just as you would edit the values in a choose-variable-values menu. An explanation of the selected variable appears in the mouse-documentation window.

Figure 7-1 Typical Profile Utility Window



```
*PRINT-BASE*:........................................Binary Octal Decimal Hexadecimal
*PRINT-LENGTH*:......................................NIL
*PRINT-LEVEL*:.......................................NIL
*PRINT-STRUCTURE:....................................T [yes]  NIL [no]
*READ-BASE*:.........................................Binary Octal Decimal Hexadecimal
PROFILE::CLOCK-FORMAT:...............................24 Hour Clock   12 Hour Clock
EH:*ENTER-WINDOW-DEBUGGER*:..........................NIL [never use it]  T [ask]  :ALWAYS [alw
EH:*USE-OLD-DEBUGGER*:...............................T [yes]  NIL [no]
PROFILE::KEYCLICK-STATE:.............................On Off
PROFILE::LOAD-PATCHES-AT-LOGIN:......................T [yes]  NIL [no]
PROFILE::NUMBER-OF-MOUSE-DOCUMENTATION-LINES:........2
PROFILE::P-LISP-MODE:................................COMMON-LISP  Zetalisp
PROFILE::SAVE-BUFFERS-AT-LOGOUT-ACTION:..............Via Menu  Via Prompts  Don't Save Buff
PROFILE::SCREEN-REVERSE-VIDEO-FUNCTION...............Black on White  White on Black
PROFILE::SUGGESTIONS-MENU-ON?:.......................T [yes]  NIL [no]
TV:*SYSTEM-KEYS*:....................................System Access Menu
TV::*FLAVOR-INSPECTOR-CONFIGURATION*:................THREE-PANES ONE-PANE
     TWO-HORIZONTAL-PANES  TWO-VERTICAL-PANES
TV::*INSPECTOR-CONFIGURATION*:.......................THREE-PANES ONE-PANE
     TWO-HORIZONTAL-PANES  TWO-VERTICAL-PANES
TV::MORE-PROCESSING-GLOBAL-ENABLE:...................T [yes]  NIL [no]
ZWEI::*DEFAULT-MAJOR-MODE*:..........................COMMON-LISP  Zetalisp  TEXT  MIDAS
     TEXZTOP
ZWEI::*INITIAL-MINOR-MODES*:.........................NIL
ZWEI::*REGION-MARKING-MODE*:.........................UNDERLINE  REVERSE-VIDEO
Actions
Store Options   Restore System Defaults   Restore User Defaults        Exit
Variables currently displayed:
Important Variables  UCL Variables        Error Handling Variables    Network Variables
GC Variables         Compiler Variables   Mail Variables              Zmacs Variables
Mouse Variables      Input Variables      File System Variables       Evaluation Variables
Common Lisp Globals  Display Variables    Color System Variables
Profile Frame 1

Keyboard          :    Duwain        User:
```

The Profile utility allows you to change variables from the Input Variables menu, as well as many others from other groups. For instance, you can change the default value for the ***read-base*** variable, which specifies the base or radix of numbers that you type. By default, the microExplorer assumes you are entering decimal numbers.

Selecting Actions

7.2.3 In addition to changing the values of variables, you can select one of the following actions from the Actions menu:

- Store Options — Creates or updates a special login-init file in your login directory. This login-init file specifies your login environment to be the same as the current environment. When you log out, the microExplorer returns most variables to their default values.
- Restore System Defaults — Restores microExplorer default values to all variables.
- Restore User Defaults — Restores default values listed in the login-init file.
- Exit — Terminates the Profile utility and returns you to the window from which it was called.

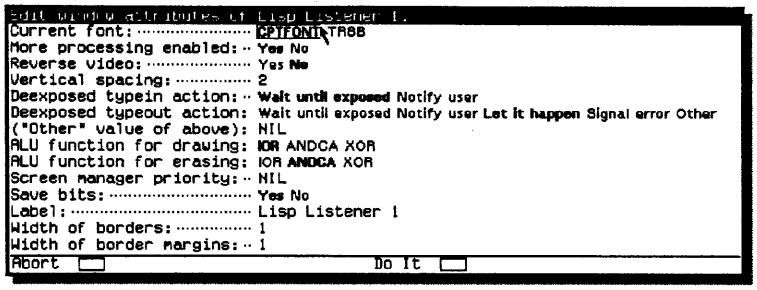
Changing Window Attributes

7.3 You can change your microExplorer window attributes by modifying the values in the Window Attributes menu. You can invoke this menu by selecting the Edit Attributes item from the System menu. Any values you change remain in effect for the current session.

Figure 7-2 shows a typical Window Attributes menu for a Lisp Listener window. Some of the variables in this menu are as follows:

- Current font — Sets the font or font list used in this window; allows you to select one of the listed fonts or specify a new font
- Reverse video — Selects regular or reverse video for the specific window only
- Vertical spacing — Specifies the amount of space between lines of print within the window
- Label — Specifies the window label
- Width of borders — Specifies the width of the window's borders
- ALU for drawing — Specifies the ALU for drawing output on the window
- ALU for erasing — Specifies the ALU for erasing objects from the window

Figure 7-2 **Window Attributes Menu**



```
Edit window attributes of Lisp Listener 1.
Current font: ························ CPTFONT TR8B
More processing enabled: ·· Yes No
Reverse video: ····················· Yes No
Vertical spacing: ················ 2
Deexposed typein action: ·· Wait until exposed Notify user
Deexposed typeout action: Wait until exposed Notify user Let it happen Signal error Other
("Other" value of above): NIL
ALU function for drawing: IOR ANDCA XOR
ALU function for erasing: IOR ANDCA XOR
Screen manager priority: ·· NIL
Save bits: ···························· Yes No
Label: ································· Lisp Listener 1
Width of borders: ················ 1
Width of border margins: ·· 1
Abort                               Do It
```

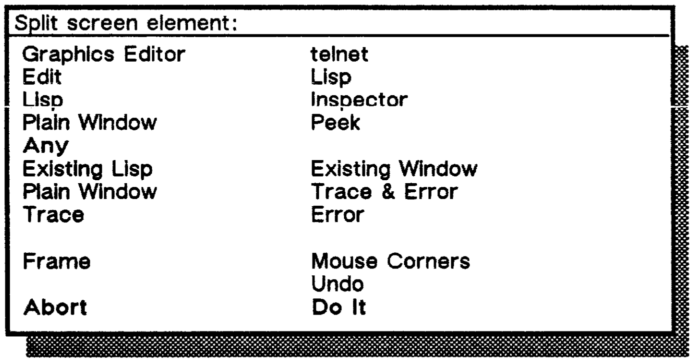
Viewing Several Windows

7.4 The microExplorer allows you to display several windows at once in the main screen, either in a standard layout or in a layout you create yourself. You can also change the size and position of windows on the video display and of panes within a frame. In addition, you can save the layout and assign a keystroke sequence to invoke it during a session.

Using a Standard Layout

7.4.1 To use a standard layout, choose the Split Screen option on the System menu. When you select this option, the microExplorer displays a menu of windows you can select, as well as a few functions. Figure 7-3 shows a typical Split Screen Element menu.

Figure 7-3 **Typical Split Screen Element Menu**



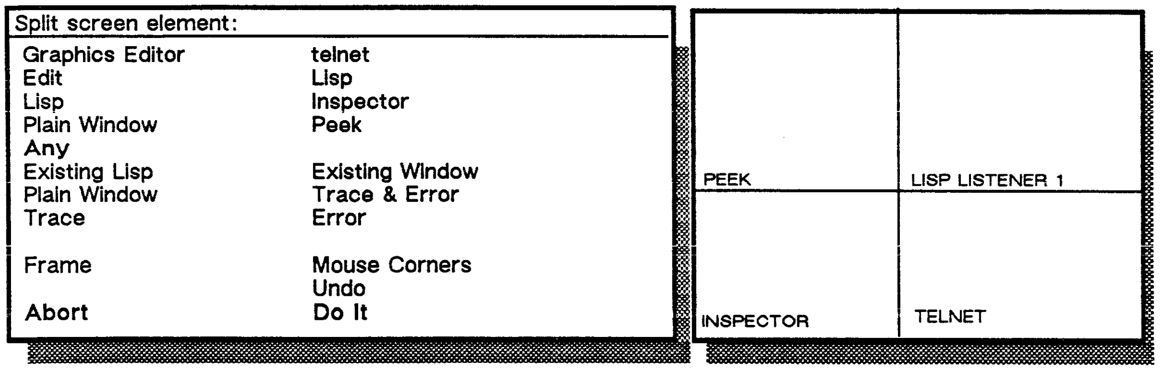
```
Split screen element:
Graphics Editor        telnet
Edit                   Lisp
Lisp                   Inspector
Plain Window           Peek
Any
Existing Lisp          Existing Window
Plain Window           Trace & Error
Trace                  Error

Frame                  Mouse Corners
                       Undo
Abort                  Do It
```

Split Screen Choices

7.4.1.1 When you select an item from the Split Screen Element menu, you create a small screen that shows the position of the window relative to the video display and to other windows in the split screen (Figure 7-4). From the Split Screen Element menu, you can make one of the following choices:

- You can select a specifically named window, such as Peek or Inspector.
- You can select the Existing Window item to obtain a menu of active windows. You can then select one of these windows to include in the split screen.
- You can select the Any item to obtain a small window that prompts you for the flavor of window to include.

Figure 7-4 Standard Layouts Created by the Split Screen Option



Split Screen Commands

7.4.1.2 You can select any of the following commands in the Split Screen Element menu:

- Abort — Cancels the layout and returns you to the previous window.
- Do It — Creates the specified layout.
- Frame — Invokes a menu that enables you to save a layout as a frame and to assign a keystroke sequence to invoke the layout. The name you assign to this frame appears in the Select menu invoked by the System menu.
- Mouse Corners — Specifies the area in which to place the split screen layout.
- Undo — Cancels the last window included as part of the layout.

Creating a Layout

7.4.2 Instead of using a layout created by the Split Screen Element menu, you can create your own layout by using the Create option of the System menu.

Because all windows are rectangular, the Create option enables you to specify the size and position of a window by setting two corners. To create a window, perform these steps:

1. Invoke the System menu and select the Create option. The microExplorer displays a menu of utilities.
2. Select the type of window you want to create. The menu disappears, and the mouse cursor becomes an upper left corner (┏).
3. Move the corner to the position where you want the upper left corner of the window to be; then, click the mouse as instructed in the mouse documentation window. The upper left corner becomes stationary, and the mouse cursor becomes a lower right corner (┛).
4. Move the cursor to the position where you want the lower right corner of the window to be; then, click the mouse as instructed in the mouse documentation window. The window appears within the rectangle defined by the two corners.

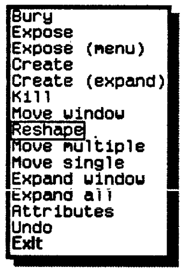
Modifying a Layout

7.4.3 The microExplorer also allows you to modify an existing layout. To change the layout of a window or pane, use the Edit Screen menu shown in Figure 7-5. When you use this menu, the microExplorer returns you to the menu after each command is completed. To leave the menu, you must select Exit. When you are modifying a layout, the contents of some windows may not appear until after you exit the Edit Screen menu.

The following commands can be used to modify a layout:

- Move Window — Changes the position of a window without changing its size. After you select a window, a shadow replaces its borders. As you move the mouse cursor, the outline of the window moves with it.
- Reshape — Changes the shape of a visible window. You specify the upper left and lower right corners of the new window.
- Move Single — Moves and reshapes a single window. You specify a single border or corner of a window and move that border or corner to a new position. Other corners and borders remain fixed.
- Move Multiple — Moves and reshapes several contiguous windows by specifying and moving one or more borders or corners of the block of windows.

Figure 7-5 **Edit Screen Menu**



Naming a Layout

7.4.4 For any layout you use, you can assign a name and keystroke sequence to the layout. Assigning a keystroke sequence enables you to invoke the layout by using a method similar to one for invoking a window for a particular utility. How you assign the name and keystroke sequence depends on whether you used the Split Screen Element menu or the Create option to create the layout.

Some of the steps outlined in the following paragraphs require a keystroke sequence consisting of the SYSTEM key and another character. In these cases, you should not use a keystroke (such as SYSTEM L) that already calls a utility. To list the keystroke sequences already used with the SYSTEM key, press SYSTEM HELP. (Note that uppercase and lowercase letters invoke the same item.)

Using the Split Screen Element Menu

7.4.4.1 When you use the Split Screen Element menu to create a screen layout, you can also use the Frame command in the menu to assign a name and keystroke sequence to the layout. If you have used the Split Screen Element menu without the Frame command to create the layout, use the Change Layouts option described in paragraph 7.4.4.2, Using the Change Layouts Option.

To name a layout you are creating with the Split Screen Element menu, follow these steps:

1. After you choose the windows in the layout, select the Frame option in the Split Screen Element menu. The microExplorer displays a menu of two prompts.

2. Type a name for the layout and press the keystroke sequence you want to use to invoke the layout. (This keystroke sequence must begin with the SYSTEM key.) If you decide not to name the layout, you can select the Cancel item. The microExplorer retains the layout you chose but discards information about the frame.

3. Select the Do It item. The microExplorer invokes the layout you created.

For the remainder of the current session, you can invoke the layout by pressing the specified keystroke sequence or by choosing its name from the Select menu invoked from the System menu.

Using the Change Layouts Option

7.4.4.2 If you use the Create command to create a layout or if you have completed a layout with the Split Screen command, you can use the Change Layouts option of the System menu to name the layout. Follow these steps:

1. Invoke the System menu and select the Change Layouts option. The microExplorer displays a menu that includes one or more items.

2. Select the Save This item. The microExplorer prompts you for the name of the layout.

3. Type the name for the layout. The microExplorer prompts you for the keystroke sequence you want to use to invoke the layout.

4. Press the desired keystroke sequence beginning with the SYSTEM key. The microExplorer returns to the layout.

For the remainder of the current session, you can invoke the layout by pressing the specified keystroke sequence or by choosing the name from the Change Layouts menu invoked from the System menu.

Applications and Windows

7.5 If you are programming an application for use on the microExplorer, or if you are porting an existing application from a Lisp machine other than the microExplorer, you need to be aware of potential differences in the size of Macintosh screens. The Macintosh screen is available in several sizes. For details about programming considerations, see Section 4, Memory and Disk Requirements, and Appendix D.

Adding Custom Macros

7.6 By using the Universal Command Loop (UCL) within the Lisp Listener, you can create and implement macros that incorporate frequently used combinations of commands or keystrokes. A *command macro* consists of a series of commands, such as Delete Character, List Completions, and so on; a *keystroke macro* consists of a series of keystrokes, such as that contained in the name of a Lisp function. In addition to the macro itself, you can specify its name, keystroke sequence, and description to be displayed with other commands when you press HYPER-CTRL-HELP. You should not use a keystroke sequence that is already being used in the microExplorer.

The procedures described in the paragraphs that follow create custom macros for the Lisp Listener. To create a keyboard macro in the Zmacs editor, see the *Explorer Zmacs Editor Reference* manual.

Searching for a Keystroke Sequence

7.6.1 To discover if a keystroke sequence is currently used in the microExplorer, follow these steps from the Lisp Listener:

1. Press HYPER-CTRL-K to search for possible current uses of the keystroke sequence. The microExplorer displays a menu that prompts you for the keystrokes.

2. Press the keystrokes for which you want to search; then select the Do It item. The microExplorer displays either the command that the keystroke executes or a message stating that the keystroke was not found.

Adding a Command Macro

7.6.2 To create a command macro from the Lisp Listener, follow these steps:

1. Press HYPER-CTRL-C to invoke the Build Command Macro command. Alternately, you can invoke this command by pressing HELP, selecting the Customization item from the Help menu, and then selecting the Build Command Macro item from the Customization menu. The microExplorer displays a menu that prompts you for three values.

2. Type the name of the command and a short description of what it does. (After you complete the command macro, this input is displayed when you press HYPER-CTRL-HELP.) Be sure to press RETURN when you complete each entry to signal to the microExplorer that you are finished.

3. Select the field that prompts you for the keystroke sequence to execute this macro. The microExplorer displays another menu that prompts you for the keystroke sequence.

4. Press the keystroke sequence that you want to execute this command macro; then select the Do It item when the keystroke is correct. As you use this menu, you can delete unwanted keystrokes by selecting the Rubout item. You can return to the previous menu by selecting the Abort item. The microExplorer displays a menu of available commands.

5. Follow the instructions in the mouse documentation window to select the commands for your macro. Each command you select is displayed in reverse video in the menu. In addition, the microExplorer lists each command you choose so that you can see the contents of the command macro.

6. After you select all the commands, select the Do It item. The video display again shows the window you selected before you began creating a macro.

The command Save Commands saves all your command modifications and the keystroke and command macros that are normally found in your directory. This command saves not only the modifications to the current application, but also the modifications to any UCL application; that is, it saves your entire command environment in one file that can be loaded at another time.

Adding a Keystroke Macro

7.6.3 To create a keystroke macro from the Lisp Listener, follow these steps:

1. Press HYPER-CTRL-M to invoke the Build Keystroke Macro command. Alternately, you can invoke this command by pressing HELP, selecting the Customization item from the Help menu, and then selecting the Build Keystroke Macro item from the Customization menu. The microExplorer displays a menu that prompts you for three values.

2. Type the name of the command and a short description of what it does. (After you complete the keystroke macro, this input is displayed when you press HYPER-CTRL-HELP.) Be sure to press RETURN when you complete each entry to signal the microExplorer that you are finished.

3. Select the field that prompts you for the keystroke sequence to execute this macro. The microExplorer displays another menu that prompts you for the keystroke sequence.

4. Press the keystroke sequence that you want to execute this keystroke macro; then select the Do It item when the keystroke is correct. As you use this menu, you can delete unwanted keystrokes by selecting the Rubout item. You can return to the previous menu by selecting the Abort item. The microExplorer displays a window with a request for your input.

5. Follow the instructions in the window to remove the window, and then type the keystrokes that you want to insert into the input line when you press the keystroke sequence. If you include several complete Lisp functions, press RETURN between each function.

6. When you are satisfied with the keystrokes, press HYPER-CTRL-M to complete the macro. The video display again shows the window you selected before you began creating a macro.

The command Save Commands saves all your command modifications and keystroke and command macros normally in your directory. This command saves not only the modifications to the current application, but also the modifications to any UCL application; that is, it saves your entire command environment in one file that can be loaded at another time.

SHARING FILE SYSTEM RESOURCES

8

Introduction

8.1 The microExplorer environment has no file system of its own. Instead, a microExplorer application acts as any other application running on the Macintosh: it reads from and writes to the Macintosh file system. This file access capability extends both to hard disks and microfloppy diskettes (although you must be careful not to access a file on a diskette that you have previously dismounted).

By using the Macintosh file system, the microExplorer spares you the effort of maintaining multiple copies of each file (one for the microExplorer and one for the Macintosh). In addition, future enhancements to the Macintosh file system will not alter compatibility between microExplorer files and standard Macintosh files.

To access the Macintosh file system, the microExplorer uses a suite of protocols designed for communicating between systems with different types of processors. At the top of this protocol hierarchy is the Network File System (NFS) protocol developed by Sun™ Microsystems. For specific information about this implementation, you can see Section 12, Inter-Environment Communication. In particular, see the software hierarchy shown in the block diagram in Figure 12-1.

Accessing Files and Directories

8.2 While operating in the microExplorer environment, you can use any of the standard functions and utilities available in the Lisp environment to access files and directories residing on the Macintosh file system.

NOTE: When the Macintosh crashes, changes being made to a file may be lost. The file may also be damaged. Therefore, before you force a crash or reset the Macintosh, be sure to save all your files.

Functions

8.2.1 A few of the more common file-oriented Lisp functions include the following:

- **view-file**
- **copy-file**
- **rename-file**
- **delete-file**
- **load**
- **copy-directory**
- **delete-directory**

Note that when renaming a file, you can only rename it to the directory that it presently resides in.

To use microExplorer file system commands, see either the *Explorer Input/ Output Reference* manual or the *Explorer Zmacs Editor Reference* manual.

Functions (such as **with-open-file**) allow programmatic access of files on the Macintosh file system. See paragraph 8.7, Programmatic File Access, for more information about programmatic file access requirements in the microExplorer environment.

Utilities

8.2.2 microExplorer utilities frequently used with files or directories include Zmacs (particularly Dired) and Peek. Some differences exist between microExplorer and Macintosh directory listings.

Pathname Completion

8.2.3 Pathname completion is supported in the microExplorer environment. Pathname completion allows you to partially specify a filename and have a microExplorer utility merge the partial filename with a set of defaults to produce a full pathname object. One common use of pathname merging is to enter a partial pathname to a prompt in the Zmacs minibuffer. You then press ESCAPE to trigger pathname completion. (For information about merging and the differences between filenames and pathname objects, see the *Explorer Input/Output Reference* manual.)

Pathname Syntax

8.3 Because the microExplorer uses the Macintosh file system, its local pathnames must adhere to the pathname syntax established for Macintosh pathnames.

Macintosh Syntax

8.3.1 A Macintosh pathname has several components, each of which is separated by a colon (:) character. The syntax for a Macintosh pathname appears as follows:

```
VOLUME:DIRECTORY:SUBDIRECTORY:NAME
```

A typical Macintosh pathname would appear as follows:

```
HD:MPW:APPLICATIONS:MY-APPLICATION
```

The first component in a Macintosh pathname is the *volume name* (the HD in the preceding example). This component identifies the device that contains the Macintosh file system; normally, the hard disk holds the Macintosh file system.

If a Macintosh pathname has any directories or subdirectories, they are represented by the components immediately following the volume name. If multiple directories are present, the pathname lists them in descending order. In the example, MPW is a directory name and APPLICATIONS is a subdirectory of the MPW directory.

The last component in a Macintosh pathname represents the actual name of the file. In the example, MY-APPLICATION is the name of the file being referenced.

Unlike Explorer and microExplorer pathnames, Macintosh pathnames have no true type component. See paragraph 8.7, Programmatic File Access, for information about how this affects programmatic file access.

No version number components are supported in pathnames for the microExplorer files.

Although the asterisk is a valid character in Macintosh pathnames, the microExplorer does not recognize it as valid. Attempts to handle such files will meet with various "invalid wildcard" errors. For example, suppose the following error message occurs:

```
>>Error: NFS pathname .... contains wildcards which are not allowed for
operation "PROPERTIES"
```

This message indicates that there is a file within a directory you are attempting to access that contains one or more #\?or #* characters in its name. The #\\ character in a pathname causes even more obscure errors. You should rename the files you want to access from the microExplorer by using the Finder to remove any asterisks, question marks, and slashes.

microExplorer Syntax

8.3.2 The microExplorer adds one extra component to the Macintosh pathname scheme; that component is the host name. The microExplorer pathname syntax appears as follows:

HOST:VOLUME:DIRECTORY:DIRECTORY:NAME.TYPE

A host name is required for reference to nonlocal files. Any pathname that contains a device name must include a host name as well.

A typical microExplorer pathname would appear as follows:

MAC:HD:MPW:APPLICATIONS:MY-APPLICATION.LISP

Logical Pathnames

8.3.3 The microExplorer also allows you to define logical pathnames. Logical pathnames enable the microExplorer to use host-independent file pathnames. Logical pathnames are discussed in detail in the *Explorer Input/Output Reference* manual. However, here is an example where logical pathnames might be useful in the microExplorer environment. What if you wanted to reference Macintosh files using Explorer pathname syntax? You could do so using code similar to the following:

Example:

```
(fs:add-logical-pathname-host
  "me" "lm"
  '(("mine" "hd" ("mine"))
    ("temp" "hd" ("temp"))
    ("test" "hd" ("temp" "test"))
    ))
```

In this example, the `add-logical-pathname-host` function creates a new logical host named `me`. The new logical host has a corresponding *physical* host (that is, the host to which it forwards most operations) called `lm` (the microExplorer's local host name).

The next forms are the translation arguments, that is, a list of translation specifications. First is a string naming the directory on the logical host (`mine`). The rest of the translation specifies the physical directory, including the volume name. The volume name is the second element of the translation specification. The third element is a list of directory names. All together the first translation specification will translate the logical directory `me:mine;` to the physical directory `hd:mine:`.

The example concludes by adding translation specifications for the `"hd:temp:"` and `"hd:temp:test:"` directories on the physical host.

With the logical pathname now defined, you can access the FOO.BAR file in the TEMP:TEST: directory by using the following pathname:

ME:TEST;FOO.BAR

Again, see the *Explorer Input/Output Reference* manual for more information about logical hosts.

Permissible Characters and Case Sensitivity

8.3.4 In the Macintosh file system, the only character that you cannot use in a pathname component is the colon. It is required for separating pathname components. However, in practice certain applications that run in the Macintosh environment make use of filename *extensions*. The applications embed a period (.) and other alphabetic characters to the end of a filename to identify which utility the file can be used with. For example, the Macintosh Programmer's Workshop (MPW) application adds a period-a (.a) to assembly language source files, a period-p (.p) to Pascal source files, and so on. The microExplorer adds a period-MCR (.MCR) to files that contain the microcode required to boot a microExplorer, or a period-LOAD (.LOAD) to those files that contain the system code required to boot a microExplorer.

Another character that you should avoid using is the forward slash character (/). This character is used to separate pathname components in many operating systems derived from the UNIX Operating System, including Apple's A/UX™ system.

The Macintosh is case-insensitive. That is, pathnames that are identical except for upper/lower case character differences refer to the same file on disk. When you enter the name of a file upon its creation, the Macintosh retains the pathname's case composition *as you enter it*. However, this case sensitivity is not preserved. Files that are created on the Macintosh keep the case exactly as specified. Files that are created from the microExplorer in the Macintosh file system always have uppercase pathnames. However, files created either way can be accessed using any combination of uppercase and/or lowercase characters.

File Attributes

8.4 File attributes are not components of a pathname, but they are used internally by the Macintosh file system. Two of the more important file attributes are the *type* attribute and the *signature* attribute.

Type Attributes

8.4.1 File type attributes (or types) are used by the Macintosh file system so that applications can verify that files are the correct type before opening them. For example, when the assembler starts to open a file, it sees the ".a" extension on the file's pathname and immediately checks the type for that file. If the type is not TEXT, the assembler signals an error.

Existing file type attributes include:

- OBJ — Object file
- APPL — Application file
- TEXT — Text (or ASCII) file
- FLDR — Directory (Folder) file

Signature Attributes

8.4.2 File signature attributes (or signatures) are also used by the Macintosh file system. In this case, the signature identifies which application originally created the file, which in many instances is the only application that can successfully modify the file. For example, the TeachText, MacWrite, and MacPaint all have individual signatures for their files. As a result, whenever you initiate an edit session from the Finder, the Macintosh file system automatically invokes the correct editor.

Files created from the microExplorer environment will have a signature of MACA, which is the signature identifying a generic Macintosh application.

You can see signatures by listing a directory from the MPW application. The file's signature is listed under the Creator column.

File Properties

8.5 Certain properties of microExplorer files (most notably the group ID and user ID) are not used by the Macintosh file handler. These two properties are used by NFS for internal purposes only.

The Working Directory

8.6 The Macintosh file system employs several concepts that differ somewhat from the standard Explorer file system. One of these concepts is the *working directory*. In file or directory operations, either the user or an application can set the working directory so that any operation that follows can specify a partial pathname, leaving the Macintosh file system to fill in the working directory portion of the pathname. The following paragraphs describe use of the partial pathname more fully.

In the Macintosh environment, the ability to use a partial pathname makes use of one of two pathname concepts: *absolute pathnames* and *relative pathnames*. An absolute pathname is one that contains all components for identifying a particular file (volume, directory, subdirectories, filenames, and file extension). For example, the following pathname is an absolute pathname:

hd:mpw:rpcsrc:nfs:protocol.obj

A relative pathname, on the other hand, lets you use a shorthand notation for referring to a pathname. Assuming your working directory is the hd:mpw:rpcsrc:nfs directory from the preceding example, you could perform some action on the protocol.obj file by referring to the following pathname:

:protocol.obj

The colon indicates that the pathname following it is relative to the working directory. The Macintosh file system also lets you refer to a directory level above the current working directory. To do so, precede a pathname with two colons. For example, assuming you are still in the hd:mpw:rpcsrc:nfs directory, the following pathname would access a file called spec.txt in the rpcsrc directory:

::spec.txt

The microExplorer has no working directory concept. To handle Macintosh pathnames in the microExplorer environment, see the following paragraph.

Programmatic File Access

8.7 For the microExplorer environment, a new pathname flavor has been created: **fs:mac-pathname**.

Handling the Working Directory

8.7.1 To handle the working directory concept, an application on the microExplorer can produce an absolute pathname by merging a partial pathname with a pathname object that represents a working directory.

To see how this works, see the examples that follow:

```
(setf working-pathname (pathname "mac:hd:microExp:nupi:"))
```

Once this value has been assigned, the coded examples yield pathname objects as follows:

```
(fs:merge-pathnames "::src:nupi.p" working-pathname) ⇒
                         MAC:HD:MICROEXP:SRC:NUPI.P

(fs:merge-pathnames ":src:nupi.p" working-pathname) ⇒
                         MAC:HD:MICROEXP:NUPI:SRC:NUPI.P
```

However, the following Lisp form receives an error because it cannot find the host `src:`

```
(fs:merge-pathnames "src:nupi.p" working-pathname)
```

In this situation, the Macintosh's natural directory delimiter conflicts with the Explorer host delimiter, producing an ambiguity. The Common Lisp function, **pathname**, cannot resolve the dilemma; however, you can use the **fs:parse-pathname** function, which accepts a host object to parse with respect to the value of its argument, *with-respect-to*.

For example if `MAC-HOST` is set to `(sys:parse-host "mac")`, then you can obtain an absolute pathname by evaluating the following code:

```
(fs:parse-pathname "SRC:nupi.p" mac-host)
```

The absolute pathname would be the pathname object MAC:SRC:NUPI.P.

To handle user input of Macintosh pathnames, your application could have code similar to the following:

```
(let* ((working-pathname "mac:hd:microExp:nupi:")
       (final-pathname
         (fs:merge-pathnames
           (fs:parse-pathname user-string mac-host)
           working-pathname))))
```

Type Component

8.7.2 Merging of types into pathnames is not always intuitive. In general, it is not safe to give just a file extension and merge it into a pathname. You should always include at least the filename when specifying a type.

Macintosh pathnames, which have no type component, should have their type component set to **:unspecific** to prevent pathname merging from filling in an incorrect type based on the available default values.

You can control how the type component is set in a pathname by modifying the value of the global variable **fs:*merge-mac-types*** at the time the pathname is parsed. For example, if **fs:*merge-mac-types*** is set to **nil**, the following code would return a pathname with the type component set to **:unspecific**.

```
(pathname "MAC:Fleming:microExp:Nupi:MakeFile")
```

Had **fs:*merge-mac-types*** been non-**nil**, the code would have returned a pathname with the type component set to **nil**. This pattern is somewhat analogous to the way the microExplorer handles the name and type of a UNIX-style pathname.

Device Component

8.7.3 Many existing Explorer applications do not handle the **:device** component of pathnames. Application writers need to use the **:merge-device** method when creating new pathname objects.

Expunge Operations

8.7.4 The expunge operation is not supported by the Macintosh file system. Calls to functions or methods for expunge will cause errors.

9 PARTITION-FILES

Introduction

9.1 This section describes *partition-files*: microExplorer files that reside in the Macintosh file system, but which are specially designed for use by the microExplorer system. The following topics are discussed:

- Partition-file names
- Manipulating partition-files
- Logical volumes
- Obtaining information about partition-files
- Disk saving partition-files
- Page partition-files
- Partition-file functions
- Disk I/O compatibility considerations
- The MakePFiles application
- Copying a partition-file across the network

If you are familiar with the Explorer environment, you will undoubtedly recognize the similarity to Explorer partitions. In fact, partitions have been implemented in the microExplorer environment as partition-files. Rather than physically partitioning sections of the disk, the microExplorer stores each partition as a single file in the Macintosh file system.

Each Macintosh disk volume can have a top-level microExp folder that contains all the partition-files for that volume. All files in the microExp folder that meet the naming conventions described later in this section are treated as microExplorer partition-files. Partition-files are one to four character strings followed by a standard suffix such as .load, .mcr, or .page.

The following types of partition-files are supported in the microExplorer environment:

- Load — A load partition-file (load band) contains the initial environment that is loaded when the microExplorer is launched. Each load partition-file requires a specific microload partition-file.
- Microload — A microload partition-file (mcr band) contains the microcode that is loaded into microExplorer processor memory at launch time.

- Page — A page partition-file is used as the address space for virtual memory. You can use more than one page partition-file on a single disk or on multiple disks. Taken together, they are called the page band.
- Meter — A meter partition-file contains information generated by the Metering utility. This partition-file is usually created before performing metering work and is deleted afterward.

Partition-File Names

9.2 In practice, partition-file name syntax is determined by the Macintosh file system; however, the microExplorer retains certain Explorer naming conventions while applying certain other restrictions.

For example, load partition-files usually have a name resembling N*xxx*.load, such as N908.load. In this example, *xxx* represents the creation date of the partition-file (September 8). The first part of the filename (N*xxx*) is a naming convention. However, partition-file names are restricted to having four or fewer characters in the first portion of the name, followed by an identifier suffix such as period-load (.load). This suffix is *required*. All load partition-file names must have the period-load suffix so that internal microExplorer functions can identify the file as a load partition-file. (Partition-files can have longer names than four characters in the Macintosh file system, but the microExplorer only uses the first four characters.)

NOTE: Do not add any blanks, tabs, or other invisible characters to the beginning or end of a partition-file name.

Similarly, microload partition-files usually have a name resembling M*xxx*.mcr, such as M298.mcr. In this example, *xxx* represents the microcode version number (version 298). Again, the first part of the filename (M*xxx*) follows the naming convention from the previous paragraph. The same restrictions apply for microcode partition-file names; that is, four or fewer characters in the first portion of the name, followed by an identifier suffix such as period-mcr (.mcr). This suffix is *required*.

Page partition-files usually have a name resembling P*xxx*.page, such as P25M.page. In this example, the *xxx* represents the size of this particular partition-file (25M bytes). Again, the first part of the filename (P*xxx*) is a naming convention (see the previous paragraphs). For page partition-file names, the four-character suffix period-page (.page) is *required* in all page partition-file names.

Names of meter partition-files are required to have the four-character suffix period-metr (.metr). A good partition name for a meter partition is METR; be sure to use the period-metr (.metr) suffix (so an example meter partition-file name would be METR.METR).

On a system that includes more than one disk, partition-file names on one disk can be duplicated on another disk.

Manipulating Partition-Files

9.3 The microExplorer has a logical disk label much like an Explorer disk label. However, you do not explicitly manipulate partition-files on a logical disk label with **sys:edit-disk-label** or META-X Edit Disk Partitions. You manipulate partition-files in the following ways:

- When the microExplorer is *not* active, partition-files can be created or resized by using the MakePFiles utility. They can also be copied by using the normal Macintosh file utilities in the Finder.
- After booting the microExplorer, partition-files can be created or modified dynamically from the microExplorer environment by using Lisp functions such as **disk-save, sys:add-page-band,** and others described later in this section.

NOTE: When the microExplorer is launched, it takes a snapshot of information about the partition-files in the microExp folders of all online volumes. After the microExplorer is booted, changes made through MakePFiles or the Finder will not be known by the microExplorer. Remember the following two rules when manipulating partition-files:

- **Use the MakePFiles or Finder utilities only if the microExplorer is not running.**
- **If the microExplorer is running, use only Lisp functions to add, modify, or otherwise manipulate partition-files.**

Failure to follow these rules can lead to strange system behavior and system crashes.

Before attempting to add a partition-file or change the size of an existing partition-file, be aware of the following items:

- Be sure that you have enough disk space available to accommodate the new file or the increased size of an existing one.
- Be sure that the file system of the Macintosh is not fragmented on your hard disk. If it is, clean up the file system first. Fragmentation can substantially lengthen loading time for load partition-files and can slow the microExplorer by affecting its paging performance. Generally, if the MakePFiles utility reports that it required more than three *extents* (groups of contiguous units on the disk) to create a partition-file, the microExplorer may suffer degradation in its paging performance. Third-party applications are available for checking disk fragmentation and for disk compression.

CAUTION: Do not attempt to shorten the length of a load partition-file to less than its data length. If you do, the resulting load band will be unusable and you will be forced to reinstall the entire load partition-file from backup sources. See the sys:describe-partition function in paragraph 9.5 for information about how to read the data length of a load partition-file.

Logical Volumes

9.4 Each Macintosh disk volume that is visible on the Finder desktop can be considered a *logical volume* by the microExplorer software. Logical volumes on the microExplorer are much like logical units on the Explorer system. One logical volume (or logical unit) exists for each Macintosh volume that is mounted (online) when the microExplorer is launched.

The numbering system for logical volumes is zero-based. Logical unit 0 is the volume that appears in the top right-hand corner of the Finder desktop; logical unit 1 is the volume immediately below it; and so forth.

Each logical volume has a logical disk label much like an Explorer disk label. Since microExplorer partitions are implemented as Macintosh files, the files considered as part of the microExplorer's disk label are only a subset of all the files on the Macintosh volume: those partition-files found under the top-level microExp folder of each logical volume.

Note that while it is possible to dismount a Macintosh volume from the Finder by dragging its icon into the trash, this should not be done to a volume that has active partition-files on it if the microExplorer is running. To do so would cause the microExplorer to crash later when it tries to access partition-file data on the unmounted volume.

Obtaining Information About Partition-Files

9.5 Several Lisp functions provide information about partition-files. Two frequently used functions, **print-disk-label** and **sys:describe-partition**, are described here. Other functions are described in the *Explorer Input/Output Reference* manual.

print-disk-label &optional *unit stream* Function

Prints the contents of the disk label of the specified unit.

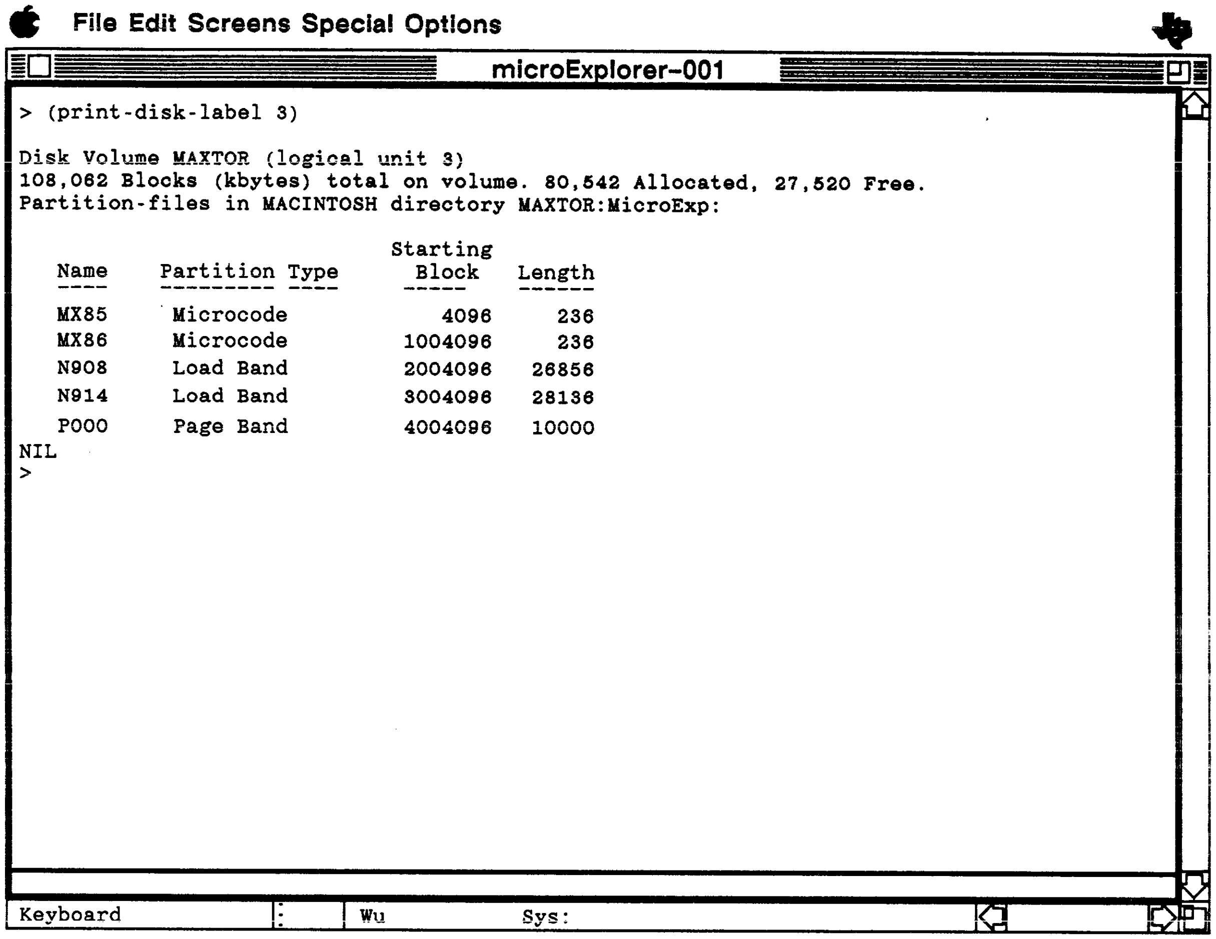
Arguments:

unit — Specifies the logical *unit* number (volume number) whose label is displayed. If a host name is specified without a unit number, the default unit for that host is used. The default value of *unit* is the value of **sys:*default-disk-unit***. *unit* can also be a volume name string such as "HD".

stream — Specifies where to print the disk label. The default value of stream is the value of ***standard-output***, typically the video display.

Example: If you execute the Lisp function **print-disk-label**, the display is similar to that shown in Figure 9-1.

Figure 9-1 Result of the print-disk-label Function



```
> (print-disk-label 3)

Disk Volume MAXTOR (logical unit 3)
108,062 Blocks (kbytes) total on volume. 80,542 Allocated, 27,520 Free.
Partition-files in MACINTOSH directory MAXTOR:MicroExp:

                              Starting
   Name    Partition Type      Block   Length
   ----    --------- ----      -----   ------
   MX85     Microcode           4096      236
   MX86     Microcode        1004096      236
   N908     Load Band        2004096    26856
   N914     Load Band        3004096    28136
   P000     Page Band        4004096    10000
NIL
>
```

sys:describe-partition *partition-name* &optional *unit* Function

Displays information about the partition-file named *partition-name* on the disk specified in *unit*.

Arguments: *partition-name* — Either a symbol or a string that specifies the name of the partition to describe.

If *partition-name* is a load partition-file, this function displays (in addition to other information) the microcode version required by the partition-file, the size of the data in the partition-file, and the highest virtual address used. The size informs you of the amount of space required to copy this partition-file; the highest virtual address indicates the size of the page partition-file needed to run *partition-name*.

unit — Specifies the logical *unit* number (volume number) that contains the partition. If a host name is specified without a unit number, the default unit for that host is used. The default value of *unit* is the value of **sys:*default-disk-unit***. *unit* can also be a volume name string such as "HD".

Example: If you execute the Lisp function `(sys:describe-partition "L121")`, the system displays a message similar to the following:

```
Partition L121 starts at 37053 and is 35000 blocks long.
It is a compressed world-load.
Data-length is 26496 blocks, highest virtual page number is 65984.
Goes with microcode version 33.
It is a (Load Band) of CPU type (Explorer)
NIL
```

The data-length of a load partition-file tells how much of that partition-file contains meaningful information. The value of `blocks` is expressed in blocks of 1024 bytes each.

CAUTION: When modifying the size of a partition either by using MakePFiles or the sys:modify-partition function, do not make the length of an existing load partition-file less than its data length. Otherwise, you will encounter seemingly unexplained crashes when attempting to use that load partition-file.

Disk Saving Partition-Files

9.6 The **disk-save** and **gc-and-disk-save** functions are discussed in detail in the *Explorer Input/Output Reference* manual; however, some of the arguments to these functions are handled differently in the microExplorer environment.

On the Explorer, the partition you use for the **disk-save** must exist and be the right size before **disk-save** begins. On the microExplorer, if the partition provided as the *partition* argument to **disk-save** does not exist, **disk-save** attempts to create it for you after prompting you to confirm. If the partition already exists but is too small, **disk-save** expands the existing partition, again after prompting you to confirm. If **:no-query** is true, **disk-save** creates or expands the partition without prompting.

If you do not have enough disk space on the volume you have specified for a partition of the size **disk-save** needs to create, you will get an error before the save starts. You then must free up space on that volume (or try another volume) before attempting the **disk-save** again. Remember, if you delete other partition-files to free up space, be sure to use **sys:delete-partition** if the microExplorer is running.

Note that **disk-save** estimates the size of the resulting partition by using the **sys:estimate-dump-size** function and adding a little more. The data length of the saved band will generally be somewhat less that the partition size **disk-save** allocates. After the saved band is booted, it can be sized to fit its partition better by using the function **sys:resize-load-band**.

The **:partition-comment** keyword argument to **disk-save** is used in the system information displayed by the **print-herald** function just as is the case on the Explorer. However, the string is not saved with the partition-files as a comment; that is, it will not show up in a **print-disk-label** listing.

gc-and-disk-save behaves similarly to **disk-save** in its handling of the *partition* and **:partition-comment** arguments. The only difference is that **gc-and-disk-save** cannot accurately estimate the size of the save partition because garbage collection (GC) may collect a large amount of garbage before the actual save begins. Therefore, **gc-and-disk-save** uses the value of its **:partition-size** argument to create the save partition initially (the default is 30,000 blocks). If this value is too small after the GC, the partition will be resized automatically if possible.

Page Partition-Files

9.7 At launch time, the microExplorer identifies all files in microExp folders with the period-page (.page) suffix for use as virtual memory paging storage in the current session. The microExplorer also uses page partition-files residing on different hard disks if your system is so configured. You can have multiple page partition-files up to a maximum of 64.

Page partition-files can be created by using the MakePFiles utility before the microExplorer is launched or by using the **sys:add-page-band** function after the microExplorer is booted.

sys:add-page-band &key :partition-name (:unit sys:*default-disk-unit*) (:size 5000) Function

Creates a new page partition-file and dynamically adds it to the system's virtual memory.

:partition-name — If non-**nil**, the value of the **:partition-name** keyword should be a string or a symbol specifying the partition name. Only the first four characters will be used (and they must be unique for this disk). When **:partition-name** is **nil** or unsupplied, the name defaults to a unique partition name for this volume.

:unit — A logical unit number (volume number) of an online disk (not a floppy). **:unit** can also be a volume name string such as "HD".

:size — The partition size in blocks. The default is 5000 (5M bytes).

Note that using the **sys:add-page-band** function on the microExplorer is equivalent to adding a page band by using **sys:edit-disk-label** (or META-X Edit Disk Partitions) on the Explorer followed by a call to **sys:change-swap-space-allocation** to make the new band known to the virtual memory system.

As part of its role as the address space for virtual memory, the page partition-file affects the microExplorer's performance. If the file system on your hard disk is fragmented, more disk seeks are required when information is paged into the microExplorer's memory. Consequently, be sure that your file system is clean and free from fragmentation.

Partition-File Functions

9.8 Most of the partition-file manipulation you do will be implicit through the use of routines that modify partitions automatically, such as **disk-save** and **sys:add-page-band**. However, other routines exist that can be called to manipulate partition-files explicitly. The following paragraphs describe these routines. See paragraph 9.9, Disk I/O Compatibility Considerations, for a discussion of issues involved in porting partition-manipulating applications to the microExplorer.

sys:add-partition *partition-name unit size* &key (**:partition-type :load**) Function

Creates a new partition named *partition-name* on *unit*; the new partition is *size* number of blocks, and its type is specified by the **:partition-type** keyword value.

partition-name — Either a symbol or a string. Up to 31 characters are used in the new partition-file's name, but the first four characters must be unique on this volume.

unit — The logical *unit* number (volume number) of an online unit (not a floppy disk). *unit* can also be a volume name string such as "HD".

size — A positive integer. An error will be signaled if there is not enough free disk space to create a partition that large.

:partition-type — A valid partition type keyword listed in the **sys:*mx-partition-types*** alist.

sys:modify-partition *partition-name unit new-size* Function
&key (**:partition-type :load**) **:new-partition-type**
:new-partition-name (**:query t**)

Modifies the existing partition named *partition-name* on *unit*. The most common use is to change a partition's size to *new-size* number of blocks. The partition's name may optionally be changed to the value of the **:new-partition-name** keyword or its type may be changed to the type specified by the **:new-partition-type** keyword.

partition-name — Either a symbol or a string. Up to 31 characters are used in the new partition-file's name, but the first four characters must be unique on this volume.

unit — The logical *unit* number (volume number) of an online unit (not a floppy disk). *unit* can also be a volume name string such as "HD".

new-size — A positive integer. An error will be signaled if there is not enough free disk space for a partition that large.

:partition-type — A valid partition type keyword as in **sys:*mx-partition-types***.

:new-partition-type — A valid partition type keyword as in **sys:*mx-partition-types***. The value of the **:new-partition-type** keyword can also be **nil**, signifying no type change.

:new-partition-name — Either a symbol or a string with the same constraints as *partition-name*. The value of the **:new-partition-name** keyword can also be **nil**, signifying no name change.

sys:modify-partition will warn you about modifying active partitions such as the current load band or any page band unless **:query** is **nil**. Note that this function allows you to modify an active page band on a microExplorer just as on the Explorer a page band can be modified or deleted by using the disk label editor. However, this operation is dangerous on either system. If the microExplorer system later tries to access a page in the modified partition, the microExplorer will crash.

sys:delete-partition *partition-name unit* &key (**:partition-type :load**) (**:query t**) — Function

Deletes the partition named *partition-name* on *unit*.

partition-name — Either a symbol or a string. Up to 31 characters are used in the new partition-file's name, but the first four characters must be unique on this volume.

unit — The logical *unit* number (volume number) of an online unit (not a floppy disk). *unit* can also be a volume name string such as "HD".

:partition-type — A valid partition type keyword as in **sys:*mx-partition-types***.

sys:delete-partition will warn you about modifying active partitions such as the current load band or any page band unless **:query** is **nil**. Note that this function allows you to delete an active page band on a microExplorer just as on the Explorer a page band can be modified or deleted by using the disk label editor. However, this operation is dangerous on either system. If the microExplorer system later tries to access a page in the modified partition, the microExplorer will crash.

sys:*mx-partition-types* — Variable

An alist of valid microExplorer partition type keyword names: **:load**, **:log**, **:mcr**, **:page**, **:file**, and **:metr**.

sys:add-or-modify-partition *partition-name unit size partition-type* &optional (*query* **t**) — Function

Checks to see if the partition *partition-name* on *unit* (type *partition-type*) exists and is at least *size* blocks long. This function does nothing if the partition exists. If the partition does not exist or is too small, this function either creates or expands the partition automatically (when *query* is **nil**) or asks you first (when *query* is **t**).

If you are writing code that manipulates disk partitions, it is convenient to use **sys:add-or-modify-partition** on the microExplorer before calling a partition-search routine such as **sys:find-disk-partition**.

sys:resize-load-band &optional (*partition-name* **sys:*loaded-band***) (*unit* **sys:*default-disk-unit***) — Function

Resizes the load band named *partition-name* on *unit* so that it does not consume unneeded disk space.

partition-name — A string or a symbol specifying a partition name. Only the first four characters are used (and must be unique for this disk).

unit — A logical *unit* number (volume number) of an online disk (not a floppy). *unit* can also be a volume name string such as "HD".

Disk I/O Compatibility Considerations

9.9 If you have an Explorer application that uses a disk partition for storing data, you can probably port that application to the microExplorer with a few modifications.

Partition-files on the microExplorer are handled internally very much as partitions are handled on the Explorer. In particular, low-level disk-I/O routines such as **sys:disk-read** and **sys:disk-write** can be used to access and manipulate data inside partition-files. The range of valid disk blocks in a partition can be obtained from a **print-disk-label** listing or from the **sys:find-disk-partition** function. Note that there are a large number of invalid blocks on a logical volume; in particular, disk block 0 (which is the block containing the disk label on an Explorer) is not a valid microExplorer disk block number on any volume.

There are a few additional considerations on the microExplorer. First, because data written to files on the Macintosh is not necessarily written to disk immediately, data you write to partitions by using **sys:disk-write** may not be written immediately to disk. However, you can call the microExplorer-specific function **sys:flush-volume**, specifying the logical volume, to ensure that such data is written to disk. In addition, if you used special user-defined partition types in your Explorer application, you should instead use the partition type **:file** on the microExplorer.

The MakePFiles Application

9.10 Because partitions are implemented as Macintosh files on the micro-Explorer, there is not actually a disk label editor. The MakePFiles utility allows you to change the size of a partition-file when the microExplorer is not running. This utility is frequently used to increase the size of a page partition-file (to add more virtual memory), to decrease the size of a page partition-file (for making room for other files in the file system), or to increase the size of a load partition-file (for disk saving over itself).

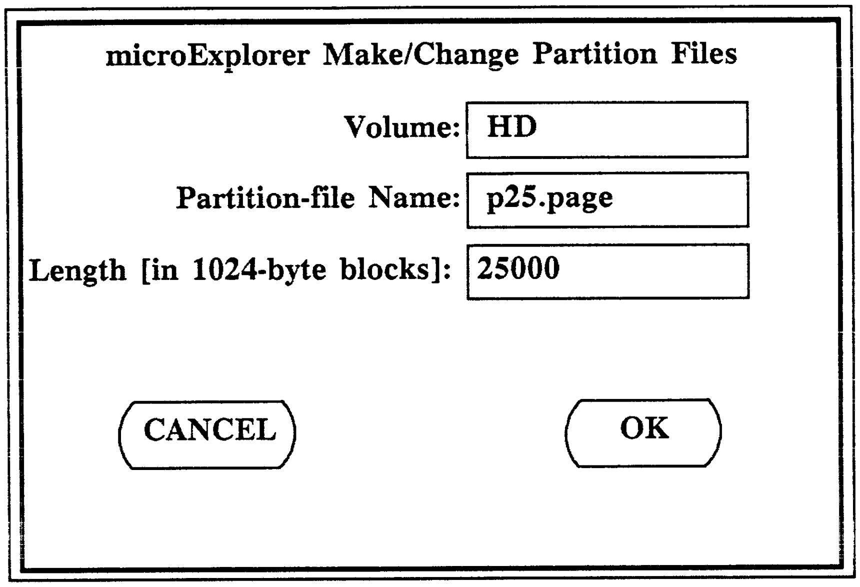
CAUTION: Do not use MakePFiles to create (or change) a partition or the Finder to delete a partition when the microExplorer is running.

The microExplorer only recognizes partitions that exist at launch time and those created from the Lisp environment after boot.

Note that once the microExplorer is booted, you can use routines such as **sys:add-page-band, sys:add-partition,** and **sys:modify-partition** to perform the same functions as MakePFiles. One advantage of MakePFiles, however, is that it tells you how many extents (groups of contiguous units on the disk) the partition occupies. This information is helpful in determining whether your paging performance can be further optimized.

The MakePFiles application resides in the microExp folder. When you double click on the MakePFiles icon, the MakePFiles window appears, and the Make title appears in the menu bar. Drag down the Make title in the menu bar and select the Make command. The Make/Change Partition-Files dialog box appears, as shown in Figure 9-2:

Figure 9-2 **Make/Change Partition-Files Dialog Box**



microExplorer Make/Change Partition Files

Volume: HD

Partition-file Name: p25.page

Length [in 1024-byte blocks]: 25000

The `Volume` prompt requests the volume name of your hard disk. The default value is `HD`, a common volume name for hard disks. If different from `HD`, enter the name of your hard disk by clicking on the box surrounding the default value, typing the correct name, and pressing RETURN.

The `Partition-file Name` prompt requests the name of the partition-file that you are adding or for which you are changing the size. If the value you supply for this prompt identifies a previously existing partition-file, MakePFiles resizes the partition-file; otherwise, MakePFiles creates a new partition-file by that name. Note that whatever name you supply for the name of the affected partition-file, you must adhere to the required naming conventions in paragraph 9.2, Partition-File Names. For example, the name of a page partition-file must end with the period-page (.page) suffix, a load partition-file must always have the period-load (.load) suffix, and so on.

The `Length` prompt requests the length (in 1024-byte blocks) of your partition-file. If you identified an existing partition-file in the `Partition-file Name` prompt, you can increase or decrease the size of that partition-file by modifying its `Length`; otherwise, the value you supply for the `Length` prompt will be the length of the new partition-file to be created by MakePFiles.

NOTE: If you specify a length value that exceeds the room you have on your disk, you will receive a –34 error, indicating that no more room is available. However, MakePFiles will still create a new partition-file or resize an existing partition-file, allocating as much space as it can find for that file.

After you have made sure that the requested information is correct, click on the OK button to activate the MakePFiles utility, or click on the CANCEL button if you have changed your mind. If all the blocks are successfully allocated for the partition-file, MakePFiles informs you of the number of extents allocated.

After the MakePFiles utility completes, exit MakePFiles. To do so, select the Quit command from the File menu.

Copying a Partition-File Across the Network

9.11 Assuming that you have the microExplorer networking option, you can copy partition-files across a network between any combination of Explorers and microExplorers.

When transferring partition-files between two microExplorers, you can use a file transfer function, such as **copy-file** or its equivalent, via a utility such as Dired.

Before you copy a partition-file from one machine to another, print the disk labels of both machines or use Dired on the microExp directories to verify the names of the partitions you want to transfer. Also, compare the size of the partition-file to be copied with the amount of free space left on your disk.

You can also use the Lisp functions **sys:receive-band** and **sys:transmit-band** to transfer partitions between microExplorers or between a microExplorer and an Explorer. Be aware, however, that Explorer partitions cannot be booted as partition-files on the microExplorer; nor can microExplorer partition-files be booted as partitions on an Explorer system.

The **sys:compare-band** and **sys:compare-disk-partition** functions compare partitions on different machines.

For information about transfer and compare partition functions, (also called band-transfer functions) see the *Explorer Input/Output Reference* manual.

MAINTAINING YOUR SYSTEM CONFIGURATION

10

Introduction

10.1 The microExplorer system software consists of a number of component systems. All of these component systems, as well as systems defined by the user, can be updated by using the microExplorer patching facility. This section provides an overview of methods for managing updates to your configuration. The following topics are discussed:

- Obtaining system version information about your configuration
- The **load-patches** function
- Updates distributed on diskettes
- Using the microExplorer as the SYS host

Obtaining Version Information

10.2 The **print-herald** function provides system version information that you can use to determine what patches have been loaded. This function displays information about the configuration you are running. The function is particularly useful on systems that have multiple bands for different applications.

print-herald &optional *stream verbose-p* Function

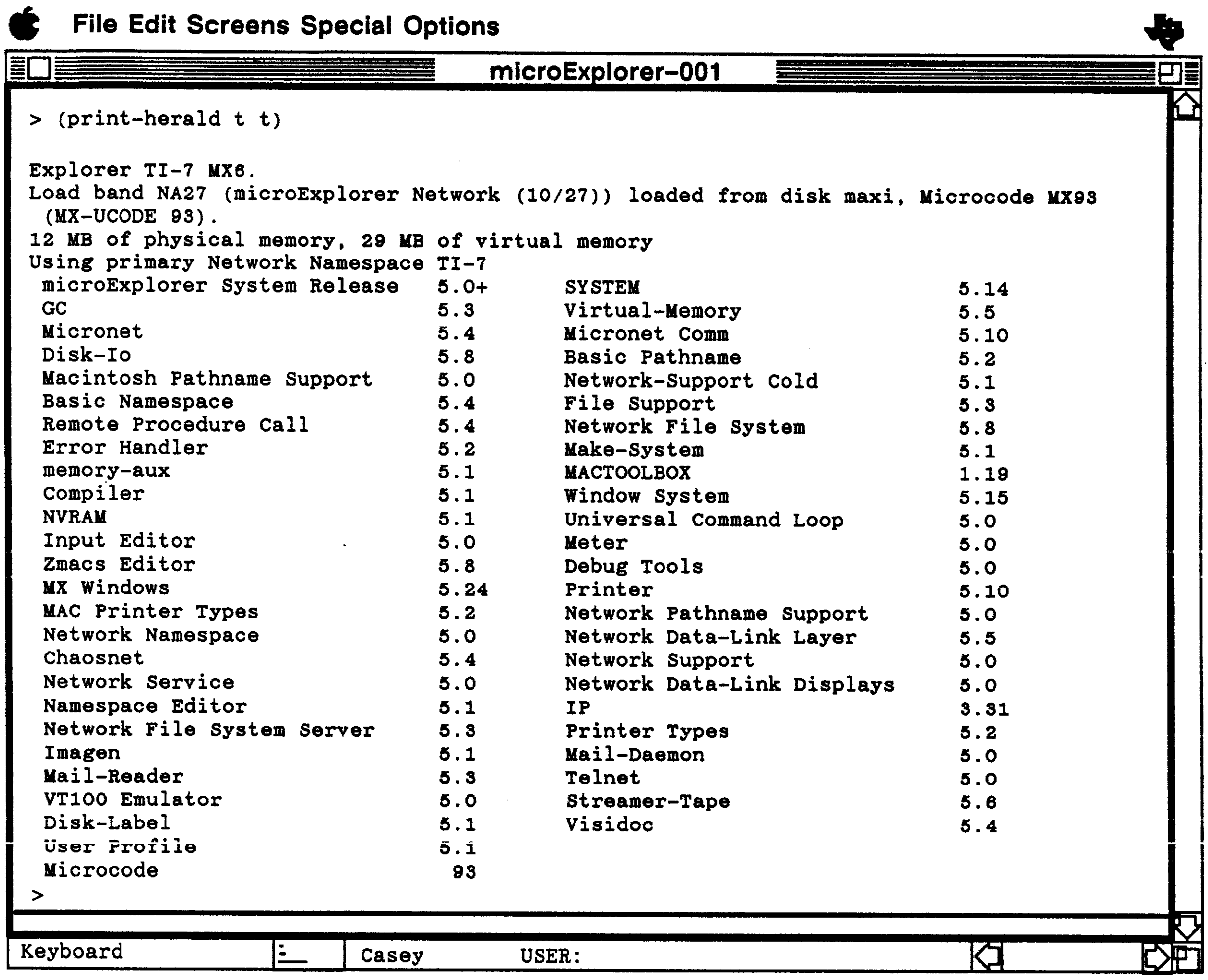
The **print-herald** function lists the versions (patch levels) of the component systems of the load band currently executing. It also displays other useful information about the current environment, such as the local host name, the load band name and description, the volume where the load band resides, and the amounts of physical and virtual memory available.

If *stream* is specified, this function prints the list to *stream* rather than to the default, ***standard-output***.

If *verbose-p* is non-**nil**, then all loaded software is listed. If it is **nil**, an abbreviated listing of the software is displayed. The default is **nil**.

Example: If you execute the Lisp function `(print-herald t t)`, the display is similar to that shown in Figure 10-1.

Figure 10-1 Result of the print-herald Function



```
> (print-herald t t)

Explorer TI-7 MX6.
Load band NA27 (microExplorer Network (10/27)) loaded from disk maxi, Microcode MX93
 (MX-UCODE 93).
12 MB of physical memory, 29 MB of virtual memory
Using primary Network Namespace TI-7
 microExplorer System Release   5.0+     SYSTEM                        5.14
 GC                             5.3      Virtual-Memory                5.5
 Micronet                       5.4      Micronet Comm                 5.10
 Disk-Io                        5.8      Basic Pathname                5.2
 Macintosh Pathname Support     5.0      Network-Support Cold          5.1
 Basic Namespace                5.4      File Support                  5.3
 Remote Procedure Call          5.4      Network File System           5.8
 Error Handler                  5.2      Make-System                   5.1
 memory-aux                     5.1      MACTOOLBOX                    1.19
 Compiler                       5.1      Window System                 5.15
 NVRAM                          5.1      Universal Command Loop        5.0
 Input Editor                   5.0      Meter                         5.0
 Zmacs Editor                   5.8      Debug Tools                   5.0
 MX Windows                     5.24     Printer                       5.10
 MAC Printer Types              5.2      Network Pathname Support      5.0
 Network Namespace              5.0      Network Data-Link Layer       5.5
 Chaosnet                       5.4      Network Support               5.0
 Network Service                5.0      Network Data-Link Displays    5.0
 Namespace Editor               5.1      IP                            3.31
 Network File System Server     5.3      Printer Types                 5.2
 Imagen                         5.1      Mail-Daemon                   5.0
 Mail-Reader                    5.3      Telnet                        5.0
 VT100 Emulator                 5.0      Streamer-Tape                 5.6
 Disk-Label                     5.1      Visidoc                       5.4
 User Profile                   5.1
 Microcode                       93
>
```

Loading Patches

10.3 A *patch file* is a file that contains upgrades and fixes for problems in a software system. Patches for the microExplorer are distributed on micro-diskette or on 40M-byte tape cartridge, depending on the complexity of the patches involved.

The microExplorer uses numbers to indicate the version of a software system or utility. For example, font editor version 2.3 indicates that this font editor is the second *major version* (the 2 of the 2.3) and the third *minor version* (the 3 of the 2.3). Thus, three patches have been made since version 2 of the font editor was installed. You can see the current version of each software system when you execute the **print-herald** function. If you have set Verbose mode on, it lists each system or utility name (along with its version number) installed on your microExplorer.

To make use of the latest version of the software, you must load the patch files for each system you want to update. The **load-patches** function enables you to load patches easily. For example, to load all patches to the system software, type the following Lisp function:

```
(load-patches)
```

For each subsystem with defined patches, you are then queried whether to load patches for that subsystem. To load only some of the patches, reply `no` to the queries about the patches you do not want loaded. If you want to load all patches without being queried for each subsystem, you can use the following form:

```
(load-patches :noselective)
```

If you want to load the patches to only a single subsystem and you know the name of the subsystem, execute the function and specify the name of the subsystem. For example, you know that the name of the Zmacs editor subsystem is `zmacs`. To load patches to the Zmacs editor, type the following Lisp function:

```
(load-patches 'zmacs)
```

If you want to use the patches each time you boot the microExplorer, you can save the patched load band by using the **disk-save** function. See Section 9, Partition-Files, of this manual and the *Explorer Input/Output Reference* manual for more information.

Note that for standard microExplorer component systems, the patches are loaded from the logical host SYS. (Refer to paragraph 10.5, microExplorer as SYS Host, if you plan to use the microExplorer as the SYS host.)

For information about making patches, refer to the *Explorer Lisp Reference* manual.

Updates on Diskettes

10.4 Patches for the microExplorer are usually distributed on diskettes. The following functions can be used to load patches directly from diskette and, in a network environment, to install the patches on your SYS host where other users can easily access them.

sys:**load-patches-from-diskette** &key (**:microexplorer-host** "LM") (**:update-diskette-name** **sys:*patch-diskette-name***) **:options** — Function

Loads patches from the diskette installed on the host specified by the **:microexplorer-host** keyword. This function continues prompting for further diskettes to support multi-volume patch-diskette sets. The **:options** keyword value should be a list of options suitable for **load-patches** (see the *Explorer Lisp Reference* manual). These options are simply passed along. For example:

```
(load-patches-from-diskette :options '(:noselective))
```

sys:**install-update-from-diskette** &key (**:microexplorer-host** "LM") (**:to-host** "SYS") (**:expsys t**) (**:macsys t**) (**:copy-lisp t**)(**:print-only nil**) — Function

Copies patch files and other updated materials from diskettes installed on the microExplorer host specified by the **:microexplorer-host** keyword.

:to-host — Specifies the host to copy to. This host is usually the SYS host, but it can be any Explorer or microExplorer host.

:expsys — If **:expsys** is true, updated files from the microExp:ExpSys: folder will be installed, including patch files.

:copy-lisp — If **:copy-lisp** is false, .LISP files will not be copied for most patches.

:macsys — If **:macsys** is true, all updated files, including any changed microExplorer application source files from the microExp:MacSys: folder, will be installed.

:print-only — If the value of the **:print-only** keyword is true, the copy operations to be done are only displayed but no actual copy operations are performed. This option is useful for verifying that the update materials will be copied to the intended place before actually starting the operation. The default for **:print-only** is **nil**, which will perform the copy operations.

microExplorer as SYS Host

10.5 When an Explorer is a SYS host, there are generally no directory translations for the host since by default the standard system directories for system source, public code, and so forth are located at the top level in the Explorer file system. On a microExplorer, however, there are two complications:

- Any Macintosh pathname must include a device component; therefore, a default device must be specified for any microExplorer host.
- All standard microExplorer system directories are located in the ExpSys subfolder of the microExp folder. Thus, a set of directory translations must be defined so that pathnames such as SYS:PUBLIC; will translate to the right subdirectory pathname (in this case *host*:*device*:microExp:ExpSys:Public:) when SYS is a microExplorer host.

If a microExplorer is to be used as SYS host for a number of machines on a network, you must set up the namespace so that the SYS host initialization can create the proper directory translations. Follow these steps:

1. Using the Namespace editor, add the following properties to the host definition of your microExplorer in the network namespace:

   :site-device — The name of a disk volume that contains the standard microExplorer system folder microExp with its ExpSys and MacSys subfolders. This volume also generally contains the system source files. This name will be used as the device component of the SYS directory translations.

   :site-directory — "microExp:ExpSys:site"

   These two properties tell the system where to find the file SYS.TRANSLATIONS. The SYS.TRANSLATIONS file, which is provided with the development software, contains directory translations for all standard microExplorer system directories.

2. In a Lisp Listener, enter the form `(net:set-sys-host` ***host-name***`)`. This form loads the file SYS.TRANSLATIONS, which defines the system host and the needed directory translations.

If you simply wish to make your local microExplorer your SYS host, you can use the following form instead, where ***device-name*** is a string naming the volume to which the SYS translations will refer.

```
(net:add-logical-pathname-host
   "SYS" "LM" (name:make-microexplorer-sys-translations)
   device-name)
```

In general, the function **name:make-microexplorer-sys-translations** is useful for creating a standard set of system directory translations to be used on a logical host that translates to a microExplorer host.

11 PRINTING

Introduction

11.1 You can print to a printer attached to the Macintosh by using the Explorer **print-file** routines. You can print either locally or remotely within constraints. The following steps are necessary to enable local printing. Each step is discussed in detail in this section.

1. The printer must be properly configured on the Macintosh.

2. The printer must be defined in the Lisp environment by editing the namespace or by issuing the sys:**add-printer-device** function.

3. The Printer Setup item under the microExplorer's File menu must be used to specify page layout parameters such as landscape printing, reduction, enlargement, and so forth.

4. **print-file** commands can then be issued, specifying the name of the printer.

Configuring a Macintosh Printer

11.2 The printers on the Macintosh can be configured in many different ways. The following paragraphs describe two possible configurations: one for an Apple ImageWriter and one for a Texas Instruments OmniLaser™ printer. For further information on printer configurations, consult your Macintosh documentation.

Configuring an ImageWriter Printer

11.2.1 The following steps describe how to set up an Apple ImageWriter using the Macintosh serial interface:

1. Connect the ImageWriter's serial cable to the port labeled with the printer icon on the back of the Macintosh chassis.

2. Turn on the ImageWriter printer, and ensure that it is online.

3. Ensure that the necessary ImageWriter printer driver is installed in your System folder. The ImageWriter printer driver and the other standard Macintosh printer drivers are provided on the Printing Tools diskette in Macintosh operating system releases.

NOTE: For proper microExplorer printing, you must use the Macintosh printing software supplied with operating system version 6.0 or later.

4. Configure the ImageWriter as the system printer by using the Chooser utility as follows:

   a. Select the Chooser utility from the Finder's Apple menu.

   b. If the ImageWriter driver is correctly installed in your System folder, an ImageWriter icon will appear in the left-hand window of the Chooser display. Click on the ImageWriter icon to select it (not on the AppleTalk® ImageWriter icon).

   c. The top right-hand Chooser window displays two port icons and requests that you select a printer port. Click on the printer icon, and answer Yes or Continue to any dialogs displayed.

   d. Exit the Chooser by selecting the Quit item from its File menu.

5. Select the Page Setup operation from the Finder's File menu to bring up an ImageWriter Style dialog box. Confirm the dialog by clicking on the OK button. The Style dialog establishes a set of system printer defaults for other Macintosh applications to use (including the microExplorer).

6. Ensure that the printer is functioning properly by printing your Startup file from the Finder as follows:

   a. Select the Startup file under the microExp folder by clicking on it once.

   b. Select the Print operation from the Finder's File menu. Since the Startup file is marked as a TeachText file, TeachText is launched to drive the printing.

   c. Click on the OK button in the ImageWriter Job dialog box that is displayed.

If this printing operation does not finish successfully, return to the first step of this procedure and carefully follow each step again. If the file is printed successfully from the Finder, you are ready to launch the microExplorer and configure its environment.

Configuring a Texas Instruments OmniLaser Printer

11.2.2 The following steps describe how to configure any of the Texas Instruments OmniLaser printers using the AppleTalk interface. The OmniLaser 2100 series printers support PostScript® operations and the AppleTalk protocol. Note that a similar procedure can be used to set up an Apple LaserWriter printer.

1. Obtain an Apple LocalTalk™ cable set. Connect the AppleTalk connector to the AppleTalk port on the back of the OmniLaser. Connect the circular end of this cable to the port labeled with the printer icon on the back of the Macintosh chassis.

2. Turn on the OmniLaser, and wait for its display to indicate `Online Idle`. (If this does not occur, the printer may not have passed power-up self-tests. Refer to your printer documentation.)

3. Configure the OmniLaser to use the AppleTalk interface and PostScript operations by using the OmniLaser's front panel as follows:

   a. Press the Setup/Test button. The display indicates `Setup/Test, Setup`. Press the Select button to confirm.

   b. Press the Next button until the display indicates `Setup Software Intf`. Press the Select button to confirm.

   c. Press the Next button until the display indicates `Software Intf., POSTSCRIPT BATCH`. Press the Select button to confirm. The display now returns to `Online Idle`.

   d. Press the Setup/Test button. The display indicates `Setup/Test, Setup`. Press the Select button to confirm.

   e. Press the Next button until the display indicates `Setup Hardware Intf`. Press the Select button to confirm.

   f. Press the Next button until the display indicates `Hardware Intf., APPLETALK`. Press the Select button to confirm. The display now returns to `Online Idle`.

4. Ensure that the necessary LaserWriter printer drivers are installed in your System folder. The LaserWriter printer driver and the other standard Macintosh printer drivers are provided on the Printing Tools diskette in Macintosh operating system releases.

NOTE: For proper microExplorer printing, you must use the Macintosh printing software supplied with operating system version 6.0 or later.

---

5. Use the Control Panel to select the Built-in network as follows:

   a. Enter the Control Panel utility from the Finder's Apple menu.

   b. Scroll the left-hand window until the Network icon is visible.

   c. Click on this icon once. The right-hand window displays one or more network icons, including one labeled Built-in and possibly one labeled EtherTalk if you have installed the Apple EtherTalk software.

   d. Make sure that the Built-in network is selected by clicking on it once. The Built-in network must be selected in order to configure the OmniLaser successfully in the Chooser.

6. Configure the OmniLaser as the system printer by using the Chooser utility as follows:

   a. Select the Chooser utility from the Finder's Apple menu.

   b. If the LaserWriter driver is correctly installed in your System folder, a LaserWriter icon that looks like a printer hooked up to an AppleTalk network appears in the left-hand window of the Chooser display. Click on this icon to select it (not on the LaserWriter IISC icon).

c. The top right-hand Chooser window should display the name `OmniLaser` with a request to select a LaserWriter. If so, ensure that the OmniLaser name is highlighted and that AppleTalk is listed as `Active`.

   If the name `OmniLaser` is not displayed in the right-hand window, communications with the printer cannot be established. Return to step 1 of this procedure, and carefully follow each step again.

d. Optionally, you can also turn on background printing by clicking on the appropriate On button.

NOTE: It is recommended that you enable background printing because this allows the microExplorer software to perform other operations while the Macintosh software drives the printing operation. However, background printing requires additional Macintosh memory for the PrintMonitor task to run. Consult Section 4, Memory and Disk Requirements, for more information on Macintosh physical memory requirements.

---

e. Exit the Chooser by selecting the Quit item from its File menu.

7. Select the Page Setup operation from the Finder's File menu to bring up a LaserWriter Style dialog box. Confirm the dialog by clicking on the OK button. The Style dialog establishes a set of system printer defaults for other Macintosh applications to use (including the microExplorer).

8. Ensure that the printer is functioning properly by printing your Startup file from the Finder as follows:

   a. Select the Startup file under the microExp folder by clicking on it once.

   b. Select the Print operation from the Finder's File menu. Since the Startup file is marked as a TeachText file, TeachText is launched to drive the printing.

   c. Click on the OK button in the LaserWriter Job dialog box that is displayed.

If this printing operation does not finish successfully, return to the first step of this procedure and carefully follow each step again. If the file is printed successfully from the Finder, you are ready to launch the microExplorer and configure its environment.

Defining a Printer in the Lisp Environment

11.3 The local Macintosh printer must now be made known to the Lisp environment by editing the namespace or by using the **sys:add-printer-device** function.

The following use of **sys:add-printer-device** defines a Macintosh printer named `My-Mac-Omnilaser` that is attached to the local host specified by *my-host-name*:

```
(sys:add-printer-device "My-Mac-Omnilaser" :mac-printer my-host-name
    :stream :mac-printer :global nil)
```

Because this form does not update the global namespace, it is suitable for use with the Development System Software or if you do not wish your printer to be accessible from other hosts in a network environment. To make the printer known globally in a network environment, specify **t** as the value of the **:global** keyword.

You can also use the namespace editor to define a printer device if you have the Network System Software. Edit the **:printer** class in the namespace to add the printer globally. The following three attributes are required:

- **:type** — The value of **:type** must be **:mac-printer**.
- **:host** — The value of **:host** must be a valid host name.
- **:stream** — The value of **:stream** must be **:mac-printer**.

Default Printer

11.3.1 If you have the microExplorer Network System Software and your microExplorer is on a network with a print server, the network namespace initially defines the default printer for files, as well as the default printer for screen images. If you do not have the Network System Software, your local namespace may contain information about a locally-connected Macintosh printer. In either case, you can specify a different default printer for your microExplorer.

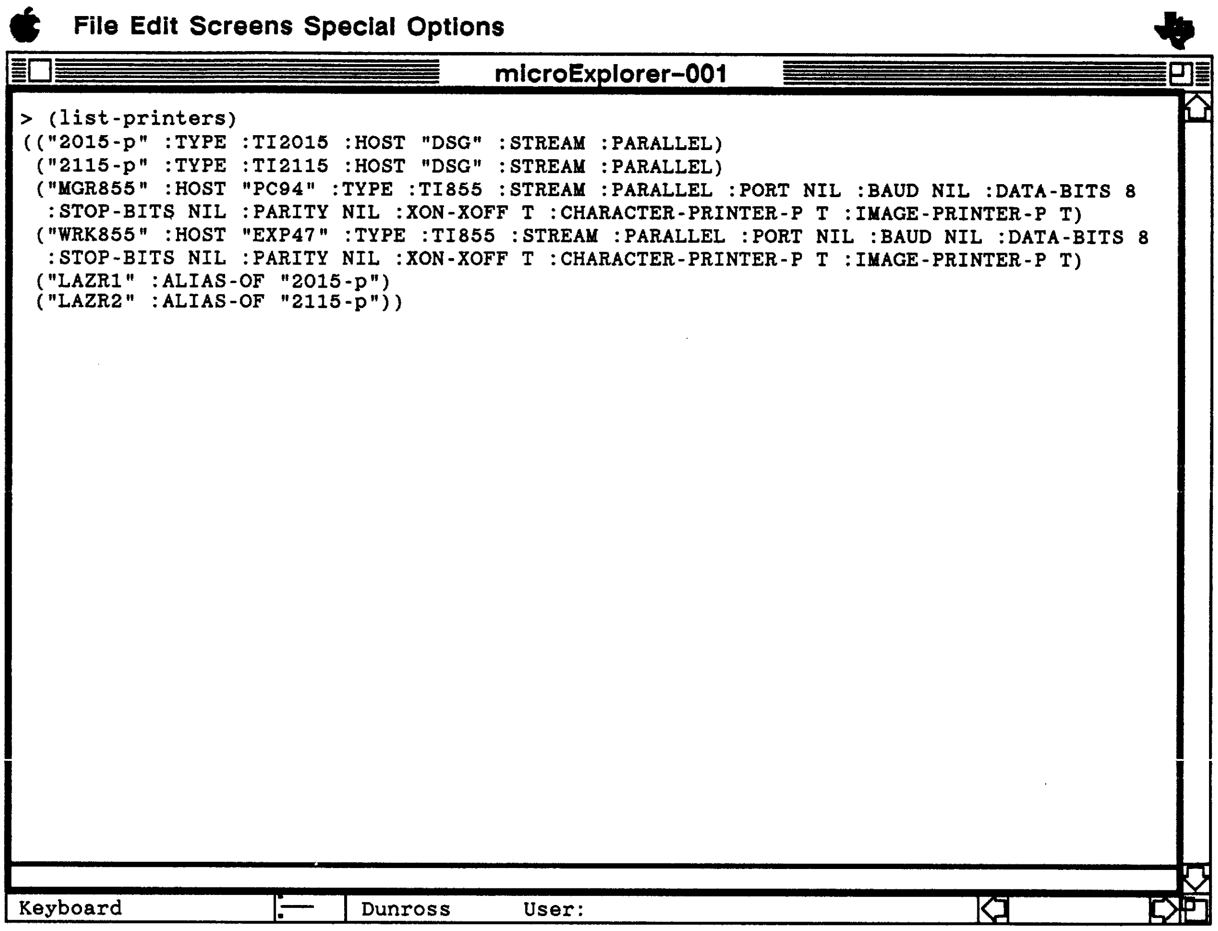
Listing the Available Printers

11.3.2 Before you can specify a default printer, you must learn which printers are available. To display a list of printers available on your network, enter the following function in the Lisp Listener:

```
(list-printers)
```

The microExplorer displays a screen similar to that shown in Figure 11-1 in which the characters enclosed in quotes (strings) are the names of the printers available on your network. If you are running in an environment without networking, only printers of type **:mac-printer** are displayed. The listing also includes the printer type and the host of the printer.

Figure 11-1 Typical List of Available Printers



```
> (list-printers)
(("2015-p" :TYPE :TI2015 :HOST "DSG" :STREAM :PARALLEL)
 ("2115-p" :TYPE :TI2115 :HOST "DSG" :STREAM :PARALLEL)
 ("MGR855" :HOST "PC94" :TYPE :TI855 :STREAM :PARALLEL :PORT NIL :BAUD NIL :DATA-BITS 8
  :STOP-BITS NIL :PARITY NIL :XON-XOFF T :CHARACTER-PRINTER-P T :IMAGE-PRINTER-P T)
 ("WRK855" :HOST "EXP47" :TYPE :TI855 :STREAM :PARALLEL :PORT NIL :BAUD NIL :DATA-BITS 8
  :STOP-BITS NIL :PARITY NIL :XON-XOFF T :CHARACTER-PRINTER-P T :IMAGE-PRINTER-P T)
 ("LAZR1" :ALIAS-OF "2015-p")
 ("LAZR2" :ALIAS-OF "2115-p"))
```

Changing the Default Printer

11.3.3 Information concerning default printers and printer availability is stored in the namespace. The following functions can be used to access this information:

- **set-default-printer** — Establishes the default text printer
- **set-default-image-printer** — Establishes the default image printer
- **get-default-printer** — Returns the name of the default text printer
- **get-default-image-printer** — Returns the name of the default image printer
- **list-printers** — Returns a list of all printers in the current namespace

For further details about these functions, refer to the *Explorer Input/Output Reference* manual.

Printer Setup Operation

11.4 The microExplorer Printer Setup operation should be performed before issuing any print commands from the microExplorer environment. This operation establishes a set of default print parameters for use by the microExplorer printing software on the Macintosh side. These default parameters allow the microExplorer to perform printing operations without prompting the user with dialogs on each operation. Follow these steps to execute the Printer Setup operation:

1. Select the Printer Setup item from the microExplorer's File menu.

   A Style dialog box is displayed. The exact contents of the Style dialog box depend on the kind of printer you have; generally, you can specify parameters such as landscape versus portrait printing.

2. Supply the values you want for the parameters, and then confirm the dialog by clicking on the OK button.

3. A Job dialog box is then displayed. Repeat the same procedure for this dialog, then confirm it by clicking on the OK button.

Any time you want to change one of the printer parameters mentioned in these dialogs, you must use the Printer Setup command. The print routines will not display these dialogs as part of their operation.

Print Operations

11.5 Once the Macintosh printer has been properly configured in both the Macintosh and the microExplorer Lisp systems, you can use any of the standard Explorer print operations such as the following:

- **print-file** and **print-file-and-wait**
- **print-bitmap** and **print-bitmap-and-wait**
- TERM Q screen dumps

If you have the Network System Software, these operations can be performed to a remote microExplorer's Macintosh printer as well as to a local one on your machine.

Macintosh Memory Considerations

11.6 When the Macintosh prints a document, it first allocates a bitmap sized by using the dimensions of the printer page and the reduction or enlargement factor (if any) specified in the Style dialog. Because using a reduction factor increases the apparent dimensions of the printer page, specifying a large reduction factor can require a large amount of Macintosh memory. The microExplorer application is sized so that reduction factors down to 50 percent will generally work. However, if you intend to print documents with a reduction size smaller than 50 percent, you need to increase the size of the microExplorer application. Refer to Section 4, Memory and Disk Requirements, for information on how to do this.

Macintosh Screen Dumping Facilities

11.7 You can create a MacPaint snapshot of microExplorer screens. The current method requires that the Lisp window be visible on the Macintosh desktop, but the Lisp window must not be the currently selected Macintosh application. After selecting another Macintosh application such as the Finder, you can use the Apple-Shift-3 key sequence to dump the current screen image to a file. The file will have a name such as Screen 1 and will be created at the top level on your boot volume. You can print the screen dump file by using the MacPaint application.

In the microExplorer Lisp environment, the keys to produce a screen snapshot are actually mapped to a Zmacs command, which is why the microExplorer application cannot be selected when the Apple-Shift-3 key sequence is used.

Note that the Apple-Shift-3 screen dumping from the Finder may not work on large monitors and is only supported on standard monitors in two-color or two-gray mode.

INTER-ENVIRONMENT COMMUNICATION 12

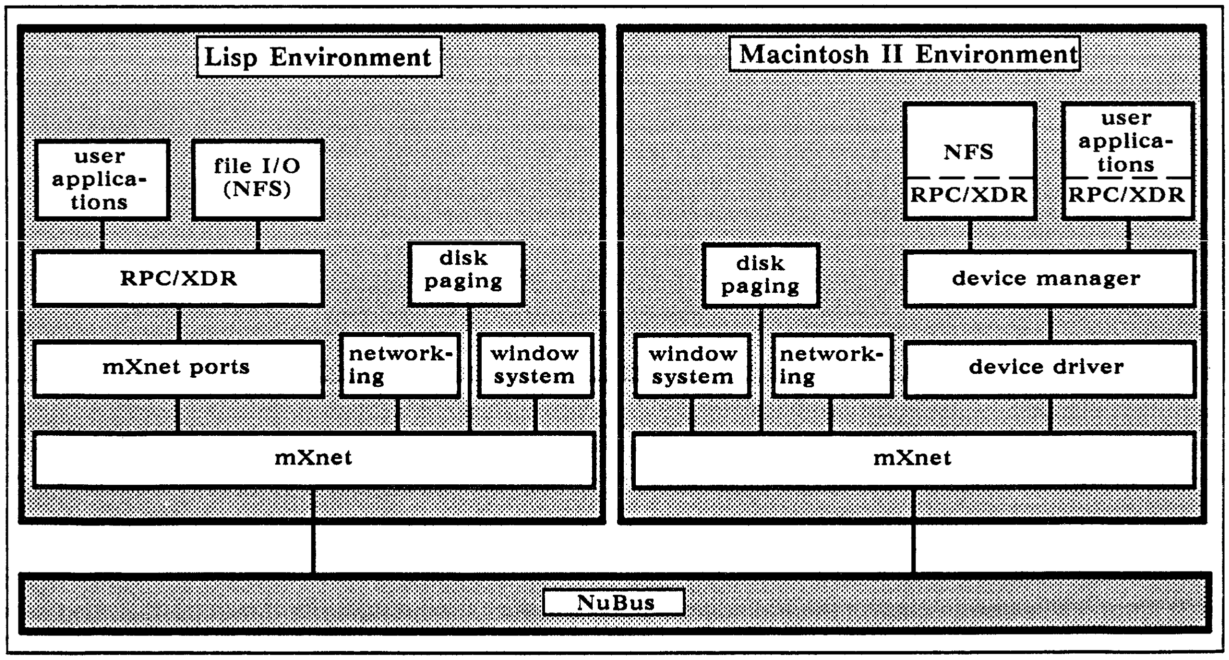
Introduction

12.1 This section first describes what makes communication between the Macintosh II processor and the microExplorer processor possible. The section next describes how you can use this communication ability to establish inter-environment communications for your own programs by activating the remote procedure call (RPC) protocol. RPC is the tool that enables two different processors to communicate.

NOTE: Before reading this section, you should be thoroughly familiar with the XDR and RPC information in the *Explorer Networking Reference* manual and the *Explorer NFS User's Guide*.

Figure 12-1 shows the software hierarchy that allows two different processor environments to communicate with each other over the NuBus.

Figure 12-1 Software Structure for Inter-Environment Communication



The common denominator between the microExplorer processor board and the Macintosh II processor is the NuBus. The NuBus is both a physical bus and the software for controlling data transfer over that bus.

The mXnet protocol acts as an interface between the NuBus software and either the Lisp environment of the microExplorer or the Macintosh operating system. Both processor environments have their own version of the mXnet protocol. The Macintosh mXnet provides the low-level data transfer between the NuBus software and the many subsystems on the Macintosh (window, disk, application, and so on). The microExplorer's mXnet protocol provides this same type of low-level data transfer between the NuBus software and the many subsystems in the Lisp environment.

All the software is provided for the operating systems and the existing utilities on the microExplorer and the Macintosh II to communicate. However, you may write an application program that uses a client/server relationship between the microExplorer processor and the Macintosh II (see paragraph 12.3, Client/Server Relationships). Your application does not need to interact with any of the intermediary software in Figure 12-1 except the RPC protocol.

The RPC library that you link with your Macintosh RPC applications contains a built-in Toolbox Server (TBServer). This particular code provides you with functionality to launch and kill the application from the microExplorer or from the Macintosh. Also, if the application is launched or killed from the Macintosh, the TBServer ensures that the microExplorer knows about it and performs the following functions:

- Disposal of the Lisp data structures and process
- Deallocation of the application channel associated with the Macintosh application

The TBServer also provides several application hooks that you can use to simplify writing a Macintosh RPC server. These features are explained in the following paragraphs.

NOTE: The debug window features found in the directories microexp:macsys:rpc-examples:call and :hooks are based upon a resource file licensed to Texas Instruments by Apple. If you use the templates from this directory to create applications for commercial distribution, you must either delete dependencies on the debug feature provided by Texas Instruments, or obtain a license agreement with Apple prior to distribution.

Minimum Requirements

12.2 Your Macintosh system must have the following items to create applications that execute remote procedure calls with the microExplorer:

- C and Pascal compilers (Macintosh Programmer's Workshop (MPW) 3.0 or greater). (The Pascal compiler is needed for the microExplorer debugging windows.)
- MPW shell (MPW 3.0 or greater)
- MultiFinder utility (Release 6.0 or greater)

Client/Server Relationships

12.3 As with any multiprocessor environment, client-server relationships can be mixed; that is, a particular service can be provided by one processor or the other, *or* both processors can provide the same service.

For the microExplorer to provide a service, you must write the service application and register it (using the **registerrpc** function) on the microExplorer. You then provide a client command for the Macintosh to invoke the microExplorer's service. This client command must issue a remote procedure call via the **callrpc** command, identifying the microExplorer's service in the call.

For the Macintosh to provide a service, you must write the service application and register it on the Macintosh. You then provide a client command for the microExplorer to invoke the Macintosh's service. This client command must issue a remote procedure call via the **callrpc** function, identifying the Macintosh's service in the call.

If both the microExplorer and the Macintosh are to provide the same service, you must perform the general procedures described in both of the preceding paragraphs.

General Steps for Implementing an RPC Application

12.4 To implement an RPC application, perform the following general procedures:

1. Design the application — Identify which environment (microExplorer or Macintosh) will supply the service and which will request the service.

2. Write the application.

3. Prepare a makefile (or **defsystem**) — If the Macintosh side provides the service, prepare a makefile to be used in the build process which follows. If the microExplorer side provides the service, you can prepare a **defsystem** to be used with a **make-system** (this is optional).

4. Build (or make) the application — If the Macintosh side provides the service, select the Build command from the menu in the MPW shell. If the microExplorer side provides the service, and you have prepared a **defsystem**, execute a **make-system** to build the application.

5. Debug the application (if necessary).

As you may have noticed, you need not build the RPC capability for the microExplorer environment because it already has RPC built into it. You do have the option, however, of creating a **defsystem** for the microExplorer-based application. You can then build the application simply by running **make-system** on it.

The rest of this section tells how to apply these generalized steps. Details are frequently explained by comparison to a Date and Time example server for both the microExplorer and the Macintosh.

Date and Time Example

12.5 The date and time example in this section provides a template for coding an RPC client/server relationship. In the example, the server provides a date and time service. When the server has been registered with the PortMap server, a client can access (call) the server. The example shows how to register the server on both the Macintosh side and the Lisp side. The example also shows how the client calls the server from both the Macintosh side and the Lisp side.

When executing in the microExplorer environment, the client uses RPC to obtain the current date and time from the Macintosh server. It then prints the results inside the window from which the client was executed.

When executing in the Macintosh environment, the client uses RPC to obtain the current date and time from the microExplorer server. It then prints the results inside the client's own debug window.

The date and time example is based on coded templates found in these files on the {boot} volume:

- :MICROEXP:MACSYS:RPC-EXAMPLES:REGISTER:MAKEFILE
- :MICROEXP:MACSYS:RPC-EXAMPLES:REGISTER:REGISTER.C
- :MICROEXP:MACSYS:RPC-EXAMPLES:CALL:MAKEFILE
- :MICROEXP:MACSYS:RPC-EXAMPLES:CALL:CALL.C
- :MICROEXP:EXPSYS:RPC-EXAMPLES:CALL:CALL.LISP
- :MICROEXP:EXPSYS:RPC-EXAMPLES:REGISTER:REGISTER.LISP

The REGISTER files contain the server code, and the CALL files contain the client code. To execute an example, compile and run the REGISTER file (server) on the Macintosh side or the microExplorer side. Next, compile and run the corresponding CALL file (client) on the other side. (That is, if you compile and run the REGISTER file on the Macintosh side, you would compile and run the corresponding CALL file on the microExplorer side.)

You should use these same coded templates and the HOOKS example (paragraph 12.8, HOOKS Example) to establish your own RPC clients and servers.

Establishing a Macintosh Server

12.6 A Macintosh server is an application program that provides some service on the Macintosh processor to some client on another processor; in this case, the microExplorer.

Macintosh Program File

12.6.1 Your program file can reside in any directory. However, you must link the library files in the MICROEXP:MACSYS: directory with your application to gain access to the RPC functionality.

After you have the application working properly, you then register it for access by the microExplorer or other processor.

Registering the Macintosh Server

12.6.2 To register your program file for RPC access, use the MICROEXP:MACSYS:RPC-EXAMPLES:REGISTER.REGISTER.C file as a template. The file contains RPC initializations that are required to register any server application, as follows:

```
#define program_number       0x20118650
#define version_number                1
#define procedure_number              1

application_init_hook()
{
   if(registerrpc(program_number,
                  version_number,
                  procedure_number,
                  get_current_time,
                  xdr_void,
                  xdr_int) != 0) {
    printf("Error in registerrpc\n");
   }
}
```

APPLICATION_INIT_HOOK is one of the TBServer's application hooks that you can define and use in your applications. Because this particular RPC server runs in background and has no window or menu bar, you only need to define this hook with a call to REGISTERRPC. When the application is launched, the TBServer calls your hook after initializing the lower layers of RPC. The call to REGISTERRPC within the hook registers your server for access by clients running on the microExplorer. Refer to paragraph 12.8, HOOKS Example, for further information on application hooks.

The remainder of the file contains the actual server functions that get called when the client requests this particular service from RPC, as follows:

```
pascal void GetDateTime(time)
    int *time;
    extern;

int * get_current_time()
{
   static int current_time;

   GetDateTime(&current_time);
   return(&current_time);
}
```

Writing the Registration Makefile

12.6.3 The register makefile compiles REGISTER.C and links it with the library files in MICROEXP:MACSYS:MX-LIB: to create the REGISTER application. The register makefile resides in the following file:

MICROEXP:MACSYS:RPC-EXAMPLES:REGISTER:MAKEFILE

You can use this makefile as a template for designing the makefile for your own Macintosh RPC servers. In order to write a server, you must link your source files with the same files that REGISTER links with. For more information, see the *Macintosh Programmer's Workshop (MPW) Reference* manual.

Building the Registration Makefile

12.6.4 After you have written the registration makefile, you must then build it. To do so, you must use the Build utility supplied as part of the MPW application.

Writing the microExplorer Client Code

12.6.5 When the microExplorer acts as a client, it attempts to access the date and time service provided by the Macintosh. Little is expected of the microExplorer client code other than to issue the remote procedure call to the Macintosh side with the **callrpc** command. The file that is named MICROEXP:EXPSYS:RPC-EXAMPLES:CALL:CALL.LISP has a template for establishing the remote procedure call from the microExplorer:

```
(defconstant *program-number*     #x20118650)
(defconstant *version-number*     1)
(defconstant *procedure-number*   1)

(defun calltest ()
 "This function tests to make sure callrpc on the Lisp side and
    registerrpc on the Macintosh side both work."
  (callrpc 'mac
           *program-number*
           *version-number*
           *procedure-number*
           :xdr-void
           nil
           :xdr-integer)
)
```

Note that before your first use of the **callrpc** function, you should issue the following form in order to avoid a delay while the call times out and is resubmitted:

```
(rpc:clear-port-map-cache :micronet)
```

If you are unfamiliar with the arguments to **callrpc**, see the *Explorer Networking Reference* manual. Assuming that you have performed the necessary tasks previously described, load the code listed above. At that point, you can get the date and time from the Macintosh by executing the **calltest** function from a microExplorer window.

Establishing a microExplorer Server

12.7 Similar to its counterpart on the Macintosh side, a microExplorer server is an application program that provides some service on the microExplorer processor to other processors, in this case, the Macintosh.

microExplorer Program File

12.7.1 The example date and time function is called **get-current-time-in-universal-time**. It resides in the following file:

MICROEXP:EXPSYS:RPC-EXAMPLES:REGISTER:REGISTER.LISP

Once your application is working properly in the Lisp environment, you then register it for access by the Macintosh or another processor.

Registering the microExplorer Server

12.7.2 A template file showing the Lisp-side registration procedure resides in MICROEXP:EXPSYS:RPC-EXAMPLES:REGISTER:REGISTER.LISP. For debugging the Lisp-side RPC, you can set the **rpc:rpc-msg-trace-p** variable to one of two values: ***query-io*** (for interactive notification) or to **t** (for a background stream typeout window).

The Lisp code for registering the example appears as follows:

```
(defconstant *register-program-number*    #x201180650)
(defconstant *register-version-number*              1)
(defconstant *register-procedure-number*            1)

(defun register-test ()
  "This function tests to make sure registerrpc on the Lisp side
     and callrpc on the Macintosh side both work."
  (registerrpc *register-program-number*
               *register-version-number*
               *register-procedure-number*
               'get-current-time-in-universal-time
               :xdr_void
               :xdr_integer
               :protocol :micronet
               :name "Get-Universal-Time"
               :server-id 'register-test)
)

(defun get-current-time-in-universal-time()
  (get-universal-time)
)
```

Writing the Call Makefile

12.7.3 The call makefile compiles CALL.C and links it with the library files in MICROEXP:MAC:MX-LIB: to create the CALL application. The call makefile resides in the following file:

MICROEXP:MACSYS:RPC-EXAMPLES:CALL:MAKEFILE

You can use this makefile as a template for designing the makefile for your own Macintosh RPC clients. In order to write a client, you must link your source files with the same files that CALL links with. For more information, see the *Macintosh Programmer's Workshop (MPW) Reference* manual.

Building the Call Makefile

12.7.4 After you have written the call makefile, you must build it. To do so, use the Build utility supplied as part of the MPW application.

Macintosh Client Code

12.7.5 When the Macintosh acts as a client, it attempts to access the date and time service that is provided by the microExplorer. Therefore, little is expected of the Macintosh side of your application other than to issue the remote procedure call to the microExplorer side with the **callrpc** command.

The MICROEXP:MACSYS:RPC-EXAMPLES:CALL:CALL.C file has a template for establishing the remote procedure call from the microExplorer.

The template is not as simple as its microExplorer-client counterpart: a large portion of the code is devoted to the debugging code provided for the microExplorer environment. The portions of code which are responsible for activating the remote procedure call appear as follows:

```
switch(error = mac_application_init())
   {
    case 0:
          /* Success. */
          break;

    case NO_APPLICATION_NAME:
     .
     .
     .
```

MAC_APPLICATION_INIT initializes the structures that are needed by RPC and the TBServer before your client can request a service from the microExplorer. MAC_APPLICATION_INIT is called from the function MAIN along with the other necessary Macintosh initializations. The switch/case statement (only partially shown above) checks for the various errors that can occur, as follows:

```
if((stat = callrpc("mx",
                   program_number,
                   version_number,
                   procedure_number,
                   xdr_void,
                   0,
                   xdr_int,
                   &current_time)) != 0)

    printf("error: callrpc stat = %d\n", stat);
```

CALLRPC issues the service request to the RPC server running on the microExplorer. Once the request has been issued, the Macintosh client code waits for a reply for the microExplorer side. The application decides what to do with the result returned from CALLRPC.

HOOKS Example

12.8 The HOOKS RPC example shown in the following paragraphs consists of a Macintosh RPC server and a microExplorer RPC client. The RPC service provided to the client is a simple four-function calculator. In addition to the RPC example, the file HOOKS.C (in the directory microexp:macsys:rpc-examples:hooks:) demonstrates the application hooks available in TBSERVER: APPLICATION_INIT_HOOK, APPLICATION_RUN_HOOK, and APPLICATION_CLEANUP_HOOK.

These hooks allow you to "hook into" the TBServer's functionality, which can simplify writing simple Macintosh applications. The hooks example also demonstrates how to properly launch and quit Macintosh applications from Lisp.

Building the Necessary RPC Filters

12.8.1 The calculator server running on the Macintosh needs three arguments: the operator and two operands. Because RPC allows only one input argument, you have to pass in a structure containing the three arguments — in this case, an integer (representing the operator) and two floating-points. Because there is no predefined RPC structure of this type, you have to define your own, as follows:

```
struct mystruct {
   int    operator;
   float  operand1;
   float  operand2;
};

bool_t xdr_mystruct(xdrs, mystruct_ptr)
XDR *xdrs;
struct mystruct *mystruct_ptr;
{
   return (xdr_int(xdrs, &mystruct_ptr->operator) &&
                   xdr_float(xdrs, &mystruct_ptr->operand1) &&
                   xdr_float(xdrs, &mystruct_ptr->operand2));
}
```

MYSTRUCT is a local C structure that the incoming argument will be converted into by the XDR_MYSTRUCT filter function. In RPC terminology, the incoming argument is DESERIALIZED into an argument of type MYSTRUCT. The deserializing function is built from predefined XDR filters.

The client must define a similar structure and filter function to convert the three arguments it passes to the calculator server (converting from its native language into the universal XDR language). The necessary Lisp code is as follows:

```
(defstruct mystruct
  (operator 0   :type integer)
  (operand1 0.0 :type single-float)
  (operand2 0.0 :type single-float))

(defun xdr-mystruct (stream obj)
  (rpc:default-and-resolve obj mystruct make-mystruct)
  (send stream :xdr-integer
   (locf (mystruct-operator obj)))    ; operator
  (send stream :xdr-float
   (locf (mystruct-operand1 obj)))    ; operand 1
  (send stream :xdr-float
   (locf (mystruct-operand2 obj)))    ; operand 2
```

The TBServer Application Hooks

12.8.2 When you build a Macintosh-side RPC server, the code is linked with the TBServer, giving your application all the functionality of a TBServer. Basically, this functionality includes the ability to launch and quit the Macintosh application from Lisp, and to make toolbox calls from Lisp. Additionally, the TBServer provides the following three application hooks that allow you to "hook into" the TBServer's functionality:

- APPLICATION_INIT_HOOK
- APPLICATION_RUN_HOOK
- APPLICATION_CLEANUP_HOOK

APPLICATION_INIT_HOOK is called once after all of the TBServer initializations are complete, but before the main program loop is entered. The TBServer's initialization includes all of the toolbox device managers and RPC. The Macintosh-side register example used the TBServer hook APPLICATION_INIT_HOOK to register the server procedure via **registerrpc** (see paragraph 12.6.2, Registering the Macintosh Server). The HOOKS example contains a window, menu bar and RPC server, and initializes each piece using APPLICATION_INIT_HOOK as follows:

```
application_init_hook()

{

  WWInit() ;
  printf("Register Calc Server From Mac\n");
  printf("Calc: waiting for calls.\n");

  SetUpMenus();

  if(registerrpc(program_number,
                 version_number,
                 procedure_number,
                 my_calculator,
                 xdr_mystruct,
                 xdr_float) != 0) {
  printf("Error in registerrpc \n");
  }
}
```

APPLICATION_RUN_HOOK provides a hook into the TBServer's main loop. Code that goes in the application's main event loop should be included in this hook. The TBServer calls the hook each time through its main loop. The following partial definition shows how this example uses APPLICATION_RUN_HOOK:

```
void application_run_hook()

{
  if(WaitNextEvent(everyEvent, &event, RPC_SLEEP_TIME, NULL))
        {
          switch(event.what)
                 {
                 case mouseDown:



   .
   .
   .
}
```

APPLICATION_CLEANUP_HOOK is called from the TBServer's Exit function on normal shutdown of the application and on certain fatal error conditions. You should include any application cleanup code in this hook. The following form shows how APPLICATION_CLEANUP_HOOK is used by HOOKS to unregister the calculator service when the application terminates; error_code is the value that was passed to Exit() by the calling function:

```
void application_cleanup_hook(error_code)

       int error_code;
{
       svc_unregister(program_number, version_number);

}
```

Launching and Quitting Macintosh Applications From Lisp

12.8.3 When you launch a TBServer application from the Macintosh or from Lisp, a Lisp process (in which your Lisp top-level function runs) is started up. Most TBServer applications are written using the Lisp toolbox interface and have a Lisp top-level main-event-loop. Although HOOKS is written in C and executes on the Macintosh side, you still need a dummy top-level loop for the associated Lisp process. This is provided by the HOOKS-LOOP function. When HOOKS starts up, HOOKS-LOOP runs in a loop that basically does nothing. When HOOKS is quit, the Lisp associated structures are deallocated and the process is killed. The following is an example of HOOKS-LOOP:

```
(defun hooks-loop()
   (setf (send si:current-process :priority) -5)
   (with-lock (*token* :whostate "just sleep")
))
```

DEFINE-MAC-APPLICATION is used to define, in Lisp, a Macintosh application that contains a TBServer. DEFINE-MAC-APPLICATION declares the name of the application, where it can be found, and the associated Lisp top-level function. The following form is the one used by this example:

```
(tb: define-mac-application hooks nil
          (:server-name "hooks"
           :lisp-function ′hooks-loop
           :directory ("microexp:macsys:rpc-examples:hooks:"))
)
```

Once the application has been defined by executing the DEFINE-MAC-APPLICATION form, the application can be started from the Macintosh by double-clicking the application's icon, or from Lisp by executing the LAUNCH-MAC-APPLICATION function. When the application is launched, a MAC-APPLICATION flavor instance is created and a micronet channel is allocated to the application. The function used to launch the HOOKS application is as follows:

```
(defun launch-my-server ()
  (setf *myserver*
   (tb:launch-mac-application 'hooks))
)
```

The application can be shutdown from the Macintosh or from Lisp. In either case, the application's Lisp data structures, process and application channel are deallocated. When the application is shutdown from Lisp, the application is sent an ExitToShell command via the toolbox-interface built into the application's TBServer. The function used to shutdown the HOOKS application is as follows:

```
(defun quit-my-server ()
  (when *myserver*
    (send *myserver* :kill-server)
    (send *myserver* :kill)
    (setf *myserver* nil))
)
```

HYPERLISP 13

microExplorer HyperCard Interface

13.1 HyperLisp is the microExplorer HyperCard interface. It allows HyperCard applications to pass Lisp forms to the Explorer environment for evaluation. When evaluation is complete, the result is passed back to HyperCard. This interface allows you to write application programs that use HyperCard in the Macintosh environment as the user interface to Lisp applications running in the microExplorer environment.

The descriptions of HyperCard features in this section are introductory and assume a working knowledge of HyperCard. For more information on HyperCard, refer to the *Macintosh HyperCard User's Guide*.

Requirements

13.2 HyperCard requires that HyperCard version 1.2 or later be running on a Macintosh System version 6.0 or later. Earlier versions of either will cause the interface to fail.

To use the HyperLisp interface, both the HyperCard application and the microExplorer must be running. Since both applications require a large amount of Macintosh memory, your Macintosh system must have a minimum of 3M bytes of memory for HyperLisp use.

Installation

13.3 The HyperLisp support software can be found in a folder called HyperLisp inside the microExp folder. The following pieces are included:

- TI-Lisp HyperCard stack
- HyperLisp file, which contains an external function (XFCN) resource
- ResEdit (resource edit) application

In order to use HyperLisp from HyperCard, you must make the TI-Lisp HyperCard stack accessible to HyperCard. You can do this by copying the TI-Lisp stack into a folder where your HyperCard will search for stacks (such as the default HyperCard Stacks folder).

CAUTION: It is a good idea to make a backup copy of the TI-Lisp stack since you can accidentally alter it using HyperCard.

Entering the TI-Lisp Stack

13.4 The first step in using HyperLisp is to go to the TI-Lisp stack. You can do this by invoking the Open Stack command from HyperCard's File menu and using the standard Macintosh directory dialog to select the TI-Lisp stack wherever it is located on your system.

To facilitate entering the TI-Lisp stack, a selector button has been provided on the second card of the TI-Lisp stack. By following the instructions below, you can install this button on your Home card. Afterwards, just clicking on the button from the Home card will go to the TI-Lisp stack.

1. Go to the TI-Lisp stack as described previously.
2. Go to the second card of the TI-Lisp stack by using the right arrow. You will see a TI icon with the label `TI HyperLisp` and two buttons, one labeled `Install TI HyperLisp on Home Card` and the other labeled `Delete TI HyperLisp from Home Card`.
3. Click on the button labeled `Install TI HyperLisp on Home Card`. This causes the Home card to appear along with the Message box.
4. The text in the Message box instructs you to click the mouse on the spot on the Home card where you want to place the TI-Lisp button. When you complete these instructions, the TI HyperLisp button will be installed on your Home card.

Once the TI HyperLisp button is installed on the Home card, you can go to the TI-Lisp stack by simply clicking on the button. If you later wish to reposition the TI HyperLisp button, you can do so by just deleting the button from the Home card then reinstalling it by using the procedure above. To delete the TI HyperLisp button from your Home card, simply click on the button labeled `Delete TI HyperLisp from Home Card` found on the second card of the TI-Lisp stack.

Using the HyperLisp Listener

13.5 The first card of the TI-Lisp stack contains the HyperLisp Listener. When the microExplorer is running, you can access the Lisp environment by simply typing Lisp forms into the HyperLisp Listener and pressing RETURN. The Lisp expression that you type is sent to a special micrcExplorer eval server for evaluation, and the result is displayed in the HyperLisp Listener window. Note that the microExplorer must be running before the HyperLisp Listener will work properly (see paragraph 13.7, Launching the microExplorer).

HyperTalk™ and the Message Box

13.6 The Message box is a command interface to HyperTalk, the programming language for HyperCard. To display the Message box, invoke the Message command from the Go menu or press Apple-M. You can now type HyperTalk commands and press RETURN to execute them. HyperTalk commands such as the ones described later in this section can be executed either from the Message box or from HyperCard scripts.

Launching the microExplorer

13.7 The microExplorer must be running before HyperLisp can be used. You can launch the microExplorer from HyperCard by using the following HyperTalk command:

```
open "microexplorer"
```

If you have not previously executed this command, a directory dialog will be displayed in order to locate the microExplorer application to launch. Once the microExplorer is booted, you can return to HyperCard via the Apple menu.

If the microExplorer is already running, you can use the following HyperTalk command to select it:

```
domenu "microexplorer"
```

Accessing Lisp From HyperTalk

13.8 HyperLisp is implemented as a HyperCard XFCN called **lispcmd** that accesses the microExplorer environment via the mXnet protocol. The syntax for **lispcmd** in HyperTalk is as follows:

lispcmd("*some Lisp expression*")

Note that the double quotes are required around the expression. For example, `lispcmd("(car '(a b c))")` would return `A`. HyperTalk allows you to set variables to the result of commands such as **lispcmd** and manipulate these results in a programmatic way.

lispcmd, like any HyperTalk command, can be executed from a HyperTalk script. The HyperLisp Listener is an example interface that uses **lispcmd**. If you examine the HyperLisp Listener's script, you can see how such interfaces might be implemented. To view the HyperLisp Listener script, first go to the card containing the HyperLisp Listener, then select the Card Info... item from the HyperCard Objects menu. Clicking on the Script... button of the dialog box that appears will display the HyperLisp Listener script.

If an error is encountered while trying to evaluate the Lisp expression, the result of **lispcmd** is an error description beginning with the notation >> ERROR. Programs performing result validation should look for that substring.

Since **lispcmd** takes a string as its argument, the HyperTalk **quote** command must be used to pass a string as part of the Lisp expression to be evaluated. For example, to display the string `"hi"` on the microExplorer's Lisp Listener window, you use **lispcmd** as follows:

```
lispcmd("(print " & quote & "hi" & quote & " w:initial-lisp-listener)")
```

The length of the Lisp expression in **lispcmd** and the length of the value returned are limited by the TI HyperLisp implementation to slightly less than 32K characters. In practice, you will encounter HyperCard-imposed field-length limitations before you reach this size.

lispcmd is normally only available in the TI-Lisp stack. If you try to invoke **lispcmd** from another stack, a warning message informs you that HyperCard `"can't understand lispcmd"`. In order to access **lispcmd** in another stack, you must either programmatically make the TI-Lisp stack available to the other stack in that stack's script or copy the **lispcmd** XFCN to the other stack or to the Home stack. If the **lispcmd** XFCN is in the Home stack, it can be accessed from any stack. Since the entire **lispcmd** XFCN is simply a resource, the ResEdit (resource edit) application can be used to copy the XFCN resource to the target stack from either the TI-Lisp stack or the HyperLisp file in the HyperLisp folder.

HyperLisp as an Application Front-End

13.9 In many cases, the user need not be aware of the microExplorer environment underlying HyperLisp. In these cases, simply returning a value from the Explorer environment is sufficient. In other cases, HyperCard can be used as a front-end to an Explorer application that in turn requires some user interaction. This type of application is implemented by opening or re-opening the microExplorer application following the **lispcmd** call by using the HyperTalk command `open "microexplorer"` or `domenu "microexplorer"`. Either of these commands exposes the microExplorer window in the foreground of your screen.

Because HyperCard is single-threaded, the **lispcmd** call must return a value before the **open** command will be executed. The easiest way to do this is to embed your Lisp function call inside a **process-run-function** form. **process-run-function** always returns a value immediately (even though the value itself is not usually used.) The value is ignored but that allows HyperCard to do the **open**. For example, the following two lines of a HyperTalk script start an Explorer application called `expert-system-shell` and then expose the microExplorer window so that you can type information for the expert system.

```
get lispcmd("(process-run-function (string 'shell) #'expert-system-shell)")
open "microexplorer"
```

HyperLisp Eval Server

13.10 HyperLisp eval serving is implemented by passing a string to the microExplorer, reading an expression from the string, passing the token read to **eval**, and then printing the result to a string that is returned as a result. The syntax for invoking eval serving was previously covered in the discussion of **lispcmd** in paragraph 13.8, Accessing Lisp From HyperTalk. In addition to this standard mechanism, the eval server on the microExplorer also contains a mechanism that allows custom evaluation of certain expressions passed from HyperCard. Most applications need not be concerned with this alternate mechanism; it exists only to accommodate special-purpose needs.

If the first token of the string passed from HyperCard is a symbol that has an **:eval-serve-handler** property that is a function, this function will be called with two arguments: the entire string and the index into the string where the first read terminated. The **:eval-serve-handler** function's result will be returned as the result of **lispcmd**.

For example, you can define the following function in the microExplorer environment:

```
(defun (:property silly :eval-serve-handler) (mystring index)
    ;; Return everything in the string except the token already parsed.
    (subseq (the string mystring) index))
```

Then, the command `lispcmd("silly silly me")` returns the result `silly me` rather than the result of evaluating the symbol **silly**. Since this mechanism shadows the evaluation of symbols that have an **:eval-serve-handler** function, such symbols should generally be isolated in a separate package. Evaluating a `(pkg-goto` *my-package*`)` expression could precede the start of such special-purpose commands.

TROUBLESHOOTING A

This appendix describes a number of procedures that you can use to correct any problems that you might encounter with the microExplorer hardware or software.

In the event you need outside help to resolve a problem, please call Texas Instruments Field Service at 1-800-572-3300 for hardware problems or your Customer Service representative for software problems. Be prepared to give your microExplorer serial number (from the label on the back of your computer) to the service representative when requesting assistance.

Following this introduction is a list of symptoms, causes, and solutions associated with possible microExplorer problems that you may encounter.

In the first set of problems, references are made to the fault indicator light-emitting diode (LED). This LED resides on the microExplorer processor board. The board turns the LED on when self-tests begin and then off when self-tests have successfully completed. Depending on which slot the microExplorer processor board is in, the LED may or may not be visible. If the microExplorer processor board is in slot 13 the LED can be seen through the center ventilation slot on the top of the Macintosh main unit. Slot 13 is the chassis slot labeled hexadecimal D on the Macintosh motherboard. The microExplorer LED is about one inch from the front edge of the ventilation slot.

Table A-1 Post-Installation Problems With the microExplorer Processor Board

Symptom	Cause/Action
The Macintosh does not function properly.	Incorrect set-up or malfunction in the Macintosh. Consult the Macintosh documentation or other appropriate manuals for troubleshooting the Macintosh system.
The microExplorer fault indicator LED remains lit during installation.	The microExplorer board is not properly seated in its slot. Power down the Macintosh and carefully follow the installation instructions to reinstall the memory expansion connector and board, checking that all connectors are exactly aligned and seated. If the fault indicator LED again stays on after power-up, power down the Macintosh, remove the expansion memory board from the microExplorer processor board, and reinstall the processor board without the expansion memory board attached to it. If the fault indicator LED goes off after you apply power again, then the expansion memory board is faulty. Call Texas Instruments Field Service at the number indicated at the beginning of this section. If the fault indicator LED again stays on after power-up, then the microExplorer processor board itself is bad. Call Texas Instruments Field Service.

Table A-1 Post-Installation Problems With the microExplorer Processor Board (Continued)

Symptom	Cause/Action
The microExplorer fault indicator LED stays on at installation.	Another board in the system could be disturbing execution of the microExplorer board self-tests. Power down the Macintosh, and remove all other add-in boards to temporarily eliminate them as sources of the disturbance. Reapply power to the Macintosh. If the LED on the microExplorer processor board turns off, one of the add-in boards is the source of the problem. To identify which add-in board is causing the problem, reinstall one add-in board at a time, powering up the Macintosh to check the fault indicator LED on the microExplorer processor each time. When you have identified the problem board, contact the manufacturer's representative for that board. If the fault indicator LED remains on with all the other add-in boards removed from the Macintosh, then the microExplorer processor board is faulty. Contact Texas Instruments Field Service.

Table A-2 Problems Installing the microExplorer Software

Symptom	Cause/Action
One of the System Files diskettes icons or the Host Driver diskette icon does not appear after you insert the diskette.	The icon is hidden behind an application window or is not visible in your directory listing. Move the application windows or scroll your directory listing to find the icon. If you do not find the item, remove the diskette by pressing Apple-Shift-1 and reinsert. If that does not work, obtain a replacement diskette from your TI customer representative.
The HDBackup utility fails to recognize one of the Development System Software diskettes.	Eject and reinsert the diskette. If that does not work, start again at the beginning of the procedure. If that fails, the diskette is probably damaged. Call your TI customer representative.
The *xxxx*.load file does not appear on the desktop after using the HDBackup utility.	The hard disk directory listing window may need updating. Close the window representing the directory contents of your hard disk; then reopen it.

Table A-2 Problems Installing the microExplorer Software (Continued)

Symptom	Cause/Action
The *xxxx*.load file appears as 1.*xxxx*.load.	The hard disk directory listing window may need updating. Close the window representing the directory contents of your hard disk; then reopen it.
After reopening the hard disk directory listing window, the *xxxx*.load file does not appear on the desktop after using the HDBackup utility.	The *xxxx*.load file may be hidden behind your hard disk System folder or another icon in your hard disk window. Use the Clean Up Window item on the Finder's Window menu to rearrange the icons in your hard disk directory listing window.
When you attempt to run the MakePFiles utility, you receive a −34 error.	The value you specified for the length of the partition-file exceeds the room you have on your disk. MakePFiles will still create a new partition file or resize an existing partition-file, allocating as much space as it can find for that file.
When you attempt to run the MakePFiles utility, you receive a −51 error.	The name you supplied for your page partition-file has tabs, blanks, or other invisible characters in it. Correct the name of the partition-file, deleting any invisible characters, and rerun the MakePFiles utility.

Table A-3 Problems During the microExplorer Launch

Symptom	Cause/Action
An alert message appears, indicating that the processor board for microExplorer could not be found or that it is not installed.	Either the ROM revision level on your Macintosh is earlier than 1.2 *and* your microExplorer processor board is in a slot other than slot 13 (hexadecimal D), or your Startup file has incorrect information in its forslot line. Correct the information in the Startup file. See paragraph 2.9, The Startup File, earlier in this manual for information about the Startup file. Check the ROM revision level of your Macintosh.
A message appears, indicating that the microExplorer processor failed to reset, a NuBus test failed, or that no communication was possible with the microExplorer processor board.	The microExplorer processor board is not seated properly in its slot. Power down the Macintosh and reseat the microExplorer processor board according to the instructions in the *microExplorer User's Guide*.

Table A-3 Problems During the microExplorer Launch (Continued)

Symptom	Cause/Action
After carefully reseating the microExplorer processor board, the same message appears indicating that the microExplorer processor failed to reset, a NuBus test failed, or that no communication was possible with the microExplorer processor board.	Another board in the system is disturbing execution of the microExplorer board self-tests. Power down the Macintosh, and remove all other add-in boards (except one video interface board) to temporarily eliminate them as sources of the disturbance. Reapply power to the Macintosh. If you can then launch the microExplorer application, one of the add-in boards is the source of the problem. To identify which add-in board is causing the problem, reinstall one add-in board at a time, powering up the Macintosh each time until the launch again fails. When you have identified the problem board, contact the manufacturer's representative for that board. If the launch fails even after all add-in boards have been removed from the Macintosh, the microExplorer processor board is faulty. Contact Texas Instruments Field Service.
The Macintosh system freezes shortly after the `Resetting` message appears on the Cold Load Stream. The Macintosh mouse does not track across the screen and the system is totally unresponsive.	A microExplorer application was already active in the system. Only one microExplorer application can be running at a time. Reboot the Macintosh (this may require powering it down first). Verify that the microExplorer is not already active by examining the list of running applications on the Apple menu. If the microExplorer is not active, attempt to launch it.
An alert message appears, indicating that the load or microcode partition-file is missing.	The Startup file has the wrong name for the missing partition-file. Correct the information in the Startup file. See paragraph 2.9, The Startup File, for information about the Startup file.
The microExplorer boot process stops shortly after displaying information about which microcode is being booted.	The software version levels of the microExplorer application and the microcode are not compatible. Consult the software level compatibility chart in Table A-4. Note that this chart may be updated in your system's latest Release Information document.
Erratic behavior or system locks up when launching the microExplorer.	MacroMaker is installed in the System folder. Move MacroMaker to another folder.

Table A-3 Problems During the microExplorer Launch (Continued)

Symptom	Cause/Action
The microExplorer boot process stops shortly after displaying information about which load band is being booted.	The system version level of the load band is not compatible with the version levels of the microExplorer application and the microcode. Consult the software level compatibility chart in Table A-4. Note that this chart may be updated in your system's latest Release Information document.
The Cold Load Stream display indicates that the load band or microcode booted is not the one expected.	The Startup file contains the wrong information, or the Startup file actually being used is located on a different disk volume. Consult the microExplorer's Cold Load Stream display to determine which Startup file was used when booting. Verify that the Startup file contains the desired boot source information.
An alert message appears, indicating that the MultiFinder utility cannot be found.	If you are running Macintosh operating system version 5, the MultiFinder is not the startup application. Enable the MultiFinder utility by making it the startup application. This procedure is described in paragraph 2.3, Enable the Multifinder. For information about the MultiFinder utility, see the *MultiFinder User's Guide*.
The Macintosh system freezes shortly after information appears about which load band is being booted. The Macintosh mouse does not track across the screen, and the system is totally unresponsive.	If your Macintosh operating system is version 6.x, the MultiFinder is not the startup application. Enable the MultiFinder utility by making it the startup application. This procedure is described in paragraph 2.3, Enable the Multifinder. For information about the MultiFinder utility, see the *MultiFinder User's Guide*.
When the TI icon is displayed on the initial Lisp Listener screen, the `Texas Instruments Explorer™` text beside it appears in a thin, scraggly-looking font.	One or more Macintosh fonts needed by the microExplorer are not installed in the Macintosh system resource file. Install the required Macintosh fonts. Consult Section 6, Fonts, for more information about required fonts and how to install them.

Table A-3 Problems During the microExplorer Launch (Continued)

Symptom	Cause/Action
Late in the microExplorer boot process, a notification appears indicating that no swap space exists.	You may have no page partition-file in the Macintosh file system, or the page partition-file is misnamed, or you need more paging space than is currently allocated. If no page partition-file yet exists, create one according to the instructions in Section 9, Partition-Files; then relaunch the microExplorer application. If you have a page partition-file, verify that it has the proper period-page (.page) suffix in its name. Correct the name if needed, then relaunch the microExplorer. If the page partition-file exists, and its name is correct, either expand the existing partition-file, or create another page partition-file. Be sure to relaunch the microExplorer application afterwards.
The microExplorer unexpectedly quits when printing a file or screen.	This problem can occur when PrintMonitor downloads fonts to the printer and the fonts are installed on the microExplorer. Remove all FONT and FONT resources except MOUSE from the microExplorer application and install them in the system file using the DA/Font Mover utility.

Table A-4 Software Compatibility Chart

Load Band Version	Microcode Version[1]	microExplorer Application Version[2]	Macintosh Operating System Version
Release 4.0	MX33	1.0	5.x
Release 4.1.1	MX39-MX41	1.28-1.30	5.x or 6.x
Release 5.0	MX96-MX98	2.x	5.x or 6.x
Release 6.0	M138-M143	3.x	6.x
Release 6.1	M195 or higher	6.x	6.x

Notes:

[1] The microcode version level is contained in the .MCR file name.

[2] If the microExplorer application is active, its version number can be obtained by selecting the About microExplorer item from its Apple menu. When the microExplorer application is not active, its version can be obtained as follows:

1. In the Finder, select the microExplorer icon by clicking on it once.
2. Select the Get Info item from the Finder's File menu.

The microExplorer version level is listed in the Version lines.

Table A-5 Problems During a microExplorer Session

Symptom	Cause/Action
The display stops being updated during a microExplorer session.	Look at the state window on the bottom of the microExplorer virtual screen. If it says `Crashed`, the microExplorer system has crashed. You should first attempt to warm boot as described in the following paragraphs. If the microExplorer state is something other than `Crashed`, wait for at least two minutes before taking any further action. If the display remains frozen, proceed as described in the following paragraphs. If you can still move the mouse cursor, drag down the Special menu and select the Warm Boot command. If the microExplorer successfully warm boots, *save your files immediately*. Next, execute the **sys:shutdown** function, answering `yes` to its prompt. After the function completes, quit the microExplorer application by dragging down the File menu and selecting the Quit command. Then, relaunch the microExplorer application. If the warm boot fails, drag down the Special menu and select the Force Crash command. This action preserves a crash file that may aid your TI Customer Service representative to correct your problem. Attempt to warm boot the microExplorer again. If this fails, quit the microExplorer application and attempt to relaunch the microExplorer. After any microExplorer system crash, forced crash, or warm boot, you should use the **report-last-shutdown** or **report-all-shutdowns** function to obtain analysis information that may be of use to TI analysts. Both of these functions allow you to write the analysis to a file. For maximum information, you should execute these functions in the warm-booted environment. If the display has stopped updating and the mouse cursor will not move, the Macintosh has crashed. Manually reset the Macintosh. One way to reset is to reboot the Macintosh by using the Programmer's switch. If your Macintosh has a Programmer's switch, reboot by pressing the System-Reset button on the Programmer's switch. The Programmer's switch is located on the right side of the Macintosh (as you are facing it). The System-Reset button is the one closest to you. (The farthest one is the Debug button.) If your Macintosh has no Programmer's switch, use the Power On button on the back of the Macintosh (on the right-hand side if you are facing the front).
A notification message appears indicating that you are running out of swap space.	You must increase the size of your existing page partition-file or add another page partition-file to create a page band. See Section 9, Partition-Files, for details on how to create or expand a page partition-file.

Table A-5 Problems During a microExplorer Session (Continued)

Symptom	Cause/Action
You cannot successfully perform a garbage collection (GC).	If the microExplorer crashes during GC, you may have run out of swap space or virtual memory. Reboot the microExplorer, and use the **report-last-shutdown** function to view the shutdown record. If the shutdown description is `Out of swap space`, you had insufficient paging storage. Create additional page partition-files by using MakePFiles (if the microExplorer is not running) or by using the sys:**add-page-band** function in the microExplorer Lisp environment. If the shutdown description is `Virtual Memory Overflow while traps disabled`, there was insufficient address space for the Garbage Collector's copy operations. You may need to invoke garbage collection earlier while there is virtual memory available or use the Extended Address Space (EAS) facility. If the shutdown description is something else, save it to a file and contact your TI Customer Service representative, or submit it in a bug report to Texas Instruments. Refer to your system's latest Release Information document. For more information on garbage collection, refer to the *Explorer Lisp Reference*.
The Macintosh freezes or gets a bus error immediately after launching the TBSERVER.	If your Macintosh ROMs are earlier than version 1.2, your microExplorer board must be in slot D and your startup file FORSLOT value must be 13. Correct the information in the Startup file. Check the ROM revision level of your Macintosh. Check that your microExplorer board is in slot D.

Table A-6 Performance Issues

Symptom	Cause/Action
The microExplorer is slow paging virtual memory. The disk produces a chattering noise.	The microExplorer may not be the selected application. Hence, sufficient processing time may not be allocated to perform paging adequately. Determine if the microExplorer is the selected application. The title bar of the selected application is highlighted, its application name has a check beside it in the Apple menu application list, and its icon appears on the far right of the menu bar. If the microExplorer is not the selected application, select it by clicking on its window or by choosing it from the Apple menu applications list.
Screen output, especially scrolling, seems slow when you have more than two colors or grays selected.	The Macintosh drawing routines are slower when multi-bit color is enabled, even if the running application does not use color. Using the Monitors facility of the Control Panel, decrease the number of color bits that can be used or switch to monochrome mode.
An alert message appears, indicating that the bit-array cache cannot hold any more screens.	Too many virtual screens have been created. Kill some of the screens by clicking on their close boxes, or increase the size of the microExplorer application. For details about this procedure, see Section 4, Memory and Disk Requirements.
Switching between windows often causes a delay of several seconds during which the screen is not updated and the cursor does not blink.	The microExplorer application is not sized sufficiently large for the size of your monitor or the number of virtual screens and windows in your environment. As a result, the microExplorer must spend a great deal of time moving window bit arrays between Macintosh and microExplorer memory. Increase the memory size of the microExplorer application. See Section 4, Memory and Disk Requirements, for details on this subject.

CHARACTER CODES AND DISPLAYS

B

Both the Explorer and the Macintosh use ASCII character codes for the 96 printable ASCII characters. Both, however, support many additional characters for which the internal codes are different. In addition, each use a different range of values for control codes.

Within the microExplorer processor, characters are always represented using the Explorer character codes (for compatibility with Explorer object files and for communicating with Explorers over a network). Before a character is written to a Macintosh file or displayed on the Macintosh screen, the microExplorer must first convert it to the Macintosh code for a character similar in appearance.

Similarly, when the microExplorer reads a character file from the Macintosh disk, it converts the characters from Macintosh character codes to the Explorer character codes. By using Macintosh character codes for files on the Macintosh disk, the microExplorer preserves a consistent appearance when the file is viewed or printed using Macintosh utilities. The microExplorer performs this conversion automatically, so that you usually need not be aware of it. If you are programming, you only need to know that Lisp functions such as **char-code** and **code-char** always use the Explorer character codes. Table B-1 shows a conversion table from the Explorer character codes to their equivalent Macintosh character codes.

Table B-1 Translations for NFS and the Times/Helvetica Fonts (Lisp to Macintosh Equivalents)

00	A5	20	20	40	40	60	60	80	00	A0	CA	C0	CB	E0	88
01	FE	21	21	41	41	61	61	81	01	A1	C1	C1	E7	E1	87
02	F0	22	22	42	42	62	62	82	02	A2	A2	C2	E5	E2	89
03	FD	23	23	43	43	63	63	83	03	A3	A3	C3	CC	E3	8B
04	F6	24	24	44	44	64	64	84	04	A4	DB	C4	80	E4	8A
05	F7	25	25	45	45	65	65	85	05	A5	B4	C5	81	E5	8C
06	CF	26	26	46	46	66	66	86	06	A6	DA	C6	AE	E6	BE
07	B9	27	27	47	47	67	67	87	07	A7	A4	C7	82	E7	8D
08	C4	28	28	48	48	68	68	88	08	A8	AC	C8	E9	E8	8F
09	C3	29	29	49	49	69	69	89	09	A9	A9	C9	83	E9	8E
0A	C6	2A	2A	4A	4A	6A	6A	8A	0A	AA	BB	CA	E6	EA	90
0B	A0	2B	2B	4B	4B	6B	6B	8B	0B	AB	C7	CB	E8	EB	91
0C	E0	2C	2C	4C	4C	6C	6C	8C	0C	AC	C2	CC	ED	EC	93
0D	B7	2D	2D	4D	4D	6D	6D	8D	0D	AD	D0	CD	EA	ED	92
0E	B0	2E	2E	4E	4E	6E	6E	8E	0E	AE	A8	CE	EB	EE	94
0F	B6	2F	2F	4F	4F	6F	6F	8F	0F	AF	F8	CF	EC	EF	95
10	D2	30	30	50	50	70	70	90	10	B0	A1	D0	DE	F0	7F
11	D3	31	31	51	51	71	71	91	11	B1	B1	D1	84	F1	96
12	BD	32	32	52	52	72	72	92	12	B2	E2	D2	F1	F2	98
13	C9	33	33	53	53	73	73	93	13	B3	E3	D3	EE	F3	97
14	DF	34	34	54	54	74	74	94	14	B4	AB	D4	EF	F4	99
15	CE	35	35	55	55	75	75	95	15	B5	B5	D5	CD	F5	9B
16	B8	36	36	56	56	76	76	96	16	B6	A6	D6	85	F6	9A
17	D1	37	37	57	57	77	77	97	17	B7	E1	D7	FA	F7	D6
18	DC	38	38	58	58	78	78	98	18	B8	FC	D8	AF	F8	BF
19	DD	39	39	59	59	79	79	99	19	B9	F5	D9	F4	F9	9D
1A	AD	3A	3A	5A	5A	7A	7A	9A	1A	BA	BC	DA	F2	FA	9C
1B	D7	3B	3B	5B	5B	7B	7B	9B	1B	BB	C8	DB	F3	FB	9E
1C	B2	3C	3C	5C	5C	7C	7C	9C	1C	BC	D4	DC	86	FC	9F
1D	B3	3D	3D	5D	5D	7D	7D	9D	1D	BD	E4	DD	D9	FD	FF
1E	C5	3E	3E	5E	5E	7E	7E	9E	1E	BE	D5	DE	FB	FE	AA
1F	F9	3F	3F	5F	5F	7F	BA	9F	1F	BF	C0	DF	A7	FF	D8

If you write an application that uses RPC to pass data between the Macintosh and the microExplorer processor, then you may need to perform character conversion yourself.

To translate an Explorer character code to its equivalent Macintosh character code, use the **tv:explorer-to-mac-char-code** function, supplying as its argument an Explorer character code. Conversely, to translate a Macintosh character code to its equivalent Explorer character code, use the **tv:mac-to-explorer-char-code** function, supplying as its argument a Macintosh character code. The argument and returned value for both of these functions is an integer from 0 to 255.

The set of glyphs (graphical representations — symbols in non-Lisp sense) supported by the Macintosh fonts has some differences from those included in the Explorer fonts, so that the microExplorer cannot display all characters the same as they would appear on an Explorer.

The Explorer software uses several characters which are not supported by the standard Macintosh fonts (such as ←, →, ⊂, and ⊃); therefore, the default Explorer font cptfont is mapped to a custom font on the Macintosh which supports all of the traditional Lisp Machine character glyphs. To accomplish this, cptfont uses a different character mapping table, shown in Table B-2.

Table B-2 cptfont Translations (Lisp to Macintosh Equivalents)

00	F3	20	20	40	40	60	60	80	*	A0	CA	C0	CB	E0	88
01	D9	21	21	41	41	61	61	81	*	A1	C1	C1	41	E1	87
02	F4	22	22	42	42	62	62	82	*	A2	A2	C2	41	E2	89
03	A7	23	23	43	43	63	63	83	*	A3	A3	C3	CC	E3	8B
04	5E	24	24	44	44	64	64	84	*	A4	E0	C4	80	E4	8A
05	C2	25	25	45	45	65	65	85	*	A5	B4	C5	81	E5	8C
06	F5	26	26	46	46	66	66	86	*	A6	E2	C6	AE	E6	BE
07	B9	27	27	47	47	67	67	87	*	A7	A4	C7	82	E7	8D
08	F6	28	28	48	48	68	68	88	*	A8	AC	C8	45	E8	8F
09	F7	29	29	49	49	69	69	89	*	A9	A9	C9	83	E9	8E
0A	F8	2A	2A	4A	4A	6A	6A	8A	*	AA	BB	CA	45	EA	90
0B	DA	2B	2B	4B	4B	6B	6B	8B	*	AB	C7	CB	45	EB	91
0C	B1	2C	2C	4C	4C	6C	6C	8C	*	AC	C2	CC	49	EC	93
0D	EE	2D	2D	4D	4D	6D	6D	8D	*	AD	C9	CD	49	ED	92
0E	B0	2E	2E	4E	4E	6E	6E	8E	*	AE	A8	CE	49	EE	94
0F	B6	2F	2F	4F	4F	6F	6F	8F	*	AF	D0	CF	49	EF	95
10	EF	30	30	50	50	70	70	90	*	B0	A1	D0	FC	F0	FD
11	F0	31	31	51	51	71	71	91	*	B1	B1	D1	84	F1	96
12	F1	32	32	52	52	72	72	92	*	B2	E5	D2	4F	F2	98
13	F2	33	33	53	53	73	73	93	*	B3	E6	D3	4F	F3	97
14	EA	34	34	54	54	74	74	94	*	B4	AB	D4	4F	F4	99
15	EB	35	35	55	55	75	75	95	*	B5	B5	D5	CD	F5	9B
16	E1	36	36	56	56	76	76	96	*	B6	A6	D6	85	F6	9A
17	DB	37	37	57	57	77	77	97	*	B7	F3	D7	EC	F7	D6
18	DC	38	38	58	58	78	78	98	*	B8	2C	D8	AF	F8	BF
19	DD	39	39	59	59	79	79	99	*	B9	E4	D9	55	F9	9D
1A	AD	3A	3A	5A	5A	7A	7A	9A	*	BA	BC	DA	55	FA	9C
1B	D7	3B	3B	5B	5B	7B	7B	9B	*	BB	C8	DB	55	FB	9E
1C	B2	3C	3C	5C	5C	7C	7C	9C	*	BC	E7	DC	86	FC	9F
1D	B3	3D	3D	5D	5D	7D	7D	9D	*	BD	E8	DD	59	FD	79
1E	DE	3E	3E	5E	5E	7E	7E	9E	*	BE	E9	DE	FA	FE	FB
1F	DF	3F	3F	5F	5F	7F	BA	9F	*	BF	C0	DF	F9	FF	D8

Note:
* Indicates a character that cannot be displayed on the Macintosh screen.

Other fonts map to standard Macintosh fonts, which lack many of the Lisp Machine mathematical symbols, but have more of the ISO international letters. Table B-3 shows which Explorer characters can be displayed using the Macintosh Times and Helvetica font families. (The Monaco font family supports fewer of the ISO characters.)

Table B-3 Macintosh Characters Display Based on NFS and Times/Helvetica Translations

00 •	20	40 @	60 ‘	80 ✱	A0	C0 À	E0 à
01 ✱	21 !	41 A	61 a	81 ✱	A1 ¡	C1 Á	E1 á
02 ✱	22 "	42 B	62 b	82 ✱	A2 ¢	C2 Â	E2 â
03 ✱	23 #	43 C	63 c	83 ✱	A3 £	C3 Ã	E3 ã
04 ✱	24 $	44 D	64 d	84 ✱	A4 ✱	C4 Ä	E4 ä
05 ✱	25 %	45 E	65 e	85 ✱	A5 ¥	C5 Å	E5 å
06 ✱	26 &	46 F	66 f	86 ✱	A6 ✱	C6 Æ	E6 æ
07 Π	27 ´	47 G	67 g	87 ✱	A7 §	C7 Ç	E7 ç
08 ✱	28 (	48 H	68 h	88 ✱	A8 ¨	C8 È	E8 è
09 ✱	29)	49 I	69 i	89 ✱	A9 ©	C9 É	E9 é
0A ✱	2A *	4A J	6A j	8A ✱	AA ª	CA Ê	EA ê
0B ✱	2B +	4B K	6B k	8B ✱	AB ≪	CB Ë	EB ë
0C ✱	2C ,	4C L	6C l	8C ✱	AC ¬	CC Ì	EC ì
0D ✱	2D -	4D M	6D m	8D ✱	AD …	CD Í	ED í
0E ∞	2E .	4E N	6E n	8E ✱	AE ®	CE Î	EE î
0F ∂	2F /	4F O	6F o	8F ✱	AF ✱	CF Ï	EF ï
10 ✱	30 0	50 P	70 p	90 ✱	B0 °	D0 ✱	F0 ✱
11 ✱	31 1	51 Q	71 q	91 ✱	B1 ±	D1 Ñ	F1 ñ
12 ✱	32 2	52 R	72 r	92 ✱	B2 ✱	D2 Ò	F2 ò
13 ✱	33 3	53 S	73 s	93 ✱	B3 ✱	D3 Ó	F3 ó
14 ✱	34 4	54 T	74 t	94 ✱	B4 ’	D4 Ô	F4 ô
15 ✱	35 5	55 U	75 u	95 ✱	B5 µ	D5 Õ	F5 õ
16 ✱	36 6	56 V	76 v	96 ✱	B6 ¶	D6 Ö	F6 ö
17 ✱	37 7	57 W	77 w	97 ✱	B7 ✱	D7 ✱	F7 ÷
18 ✱	38 8	58 X	78 x	98 ✱	B8 ✱	D8 Ø	F8 ø
19 ✱	39 9	59 Y	79 y	99 ✱	B9 ✱	D9 Ù	F9 ù
1A ≠	3A :	5A Z	7A z	9A ✱	BA º	DA Ú	FA ú
1B ◇	3B ;	5B [	7B {	9B ✱	BB ≫	DB Û	FB û
1C ≤	3C <	5C \	7C |	9C ✱	BC ✱	DC Ü	FC ü
1D ≥	3D =	5D]	7D }	9D ✱	BD ✱	DD ✱	FD ✱
1E ✱	3E >	5E ^	7E ~	9E ✱	BE ✱	DE ✱	FE ✱
1F ✱	3F ?	5F _	7F ∫	9F ✱	BF ¿	DF ß	FF ÿ

Note:
✱ Indicates no exactly corresponding character in the Macintosh character set.

THE EXPLORER SYSTEM MANUALS

C

Introduction

C.1 You will refer to several of the Explorer system manuals for information about programming, input/output, inter-environment communication, networking, and so on. Although the bulk of the material in those manuals applies to both the microExplorer and the Explorer system environments, some differences do exist. This section describes only those differences.

Each of the following primary-level paragraphs (C.2, C.3, C.4, and so on) describes the differences to be found in a particular Explorer manual. Where enough differences exist in a manual, secondary-level paragraphs (C.4.1, C.4.2, C.4.3, and so on) break the discussion into a section-by-section basis.

Explorer Lisp Reference

C.2 Table 4-1 of the *Lisp Reference* manual is subject to the restrictions found in Appendix B of this microExplorer manual.

Explorer Zmacs Reference

C.3 The List All Directory Names command in Zmacs does not apply in the microExplorer environment. To find which volumes are available on the local microExplorer, you can use the **sys:get-all-volume-names** function in a Lisp Listener or Break window.

Explorer Tools and Utilities

C.4 The *Explorer Tools and Utilities* manual has a number of differences of which you should be aware before attempting to use the manual with the microExplorer.

Section 1: Visidoc

C.4.1 This section of the *Explorer Tools and Utilities* manual is only applicable if you have purchased the microExplorer Development System Software source option to go with your microExplorer. If you have purchased this option, the following differences exist when applying this section to the microExplorer environment.

- Paragraph 1.2, Installing the Visidoc Client Software — You need not perform a make-system. The client software is saved in the kernel band.

- Paragraph 1.6, Making a Host a Visidoc Server — Refer to the release notes for the microExplorer source option for relevant information.

Section 3: Login Initialization File

C.4.2 This section of the *Explorer Tools and Utilities* manual has the following differences when applied in the Explorer or the microExplorer environment:

- Paragraph 3.1: **inhibit-init-file-p** —The login-init file is taken from the directory of the same name as user-name on the default device specified in the Startup file.
- Paragraph 3.3.3: Creating Logical Pathnames — See paragraph 8.3.3, Logical Pathnames.

Section 7: UCL Programmer Interface

C.4.3 Paragraph 7.1: Introduction — While the UCL application will work in the microExplorer environment, the microExplorer has only one beep. This beep maps to the Macintosh system beep function.

Section 8: Using Suggestions

C.4.4 This section is not applicable in the microExplorer environment.

Section 9: Programming Suggestions

C.4.5 This section is not applicable in the microExplorer environment.

Section 10: Graphics Editor

C.4.6 This section of the *Explorer Tools and Utilities* manual has the following differences when applied in the Explorer or the microExplorer environment:

- Paragraph 10.3.1: Window Layout — The icon menu does not have the same icons shown in the documentation so the menu in this paragraph is of little use. The icons appearing in the menu do not represent their capabilities.
- Paragraph 10.3.4.3: Using the Icon Menu — Do not use the icon menu.
- Paragraph 10.4: Objects — All of these documented objects are available for use with the graphics editor. Due to the mapping of Explorer icons to Macintosh icons, the icons that appear when you mark a single point for a given object differ from those shown in the documentation. Spline points are the only items that should be complicated by this difference. There is no Explorer icon that maps to the microExplorer; therefore, no visual mark is made for points that are selected.
- Paragraph 10.4.1.2: Color of Objects — The grey shades (stipple patterns) do not map identically from the Explorer to the microExplorer. The patterns making up the grey shades on the microExplorer are more complex and distinct that those of the Explorer.
- Table 10-2 and all other references to color systems are not applicable because the microExplorer software does not currently support color hardware.
- Paragraph 10.4.11.2: Drawing a Spline — No Explorer icon maps to the microExplorer; therefore, no visual mark is made for points that are selected.

- Paragraph 10.4.12: Text — Not all of the fonts documented for the font menu are currently available. Those that are not available automatically map to the default Macintosh system font, as defined by the **mac:*mac-font-translation-table*** variable described in Section 6, Fonts.
- Paragraph 10.5.1: Selecting Objects — Instead of the pointing finger icon used on the Explorer system to select objects, the microExplorer uses an open parenthesis icon: (.
- Paragraph 10.6.6: Printing Pictures — Graphics printing is not currently supported on the microExplorer. If you try to print a picture, a message appears stating that graphics printing is not supported.

Section 12: Font Editor

C.4.7 This section is not applicable in the microExplorer environment.

Section 24: Crash Analysis

C.4.8 This section of the *Explorer Tools and Utilities* manual has the following differences when applied in the Explorer or the microExplorer environment:

- Paragraph 24.3: Preparing NVRAM — This paragraph is not applicable in the microExplorer environment.
- Paragraph 24.6: Hardware Crash Descriptions and Troubleshooting —
  - NuBus Crashes: The microExplorer can detect Parity-limit-exceeded errors, and sometimes memory parity and NuBus timeout errors.
  - Processor Fault Crashes: The microExplorer can detect these errors.
  - Power Fail Crash: The microExplorer cannot detect these errors.
  - Mass Storage Subsystem Crashes: The microExplorer cannot detect these errors.
  - Troubleshooting NUPI Device and Controller Error Crashes: These errors are not applicable in the microExplorer environment.
- Paragraph 24.7: The Force Crash Keychord — META-CTRL-META-CTRL-C does not work in the microExplorer environment. Instead, select the Force Crash command from the Macintosh Special menu.

NOTE: The screen does not display in reverse-video when you select the Force Crash command. Instead, the microExplorer writes the following two items to the Cold Load Stream: `Force Crash` and `Wrote Crash Data`.

You may be unable to view the light-emitting diode (LED) on the processor through the ventilation slots on the Macintosh's main unit, depending on the placement of the microExplorer processor in the Macintosh main unit.

Section 27: Performance Tools

C.4.9 Paragraph 27.2.1: Setting Up a Meter Partition — This paragraph requires some qualifications before being applied to the microExplorer environment. You can use either the MakePFiles utility or the **sys:add-partition** Lisp function (as described in Section 9, Partition-Files). A good partition name for a meter partition is METR; be sure to use the period-metr (.metr) suffix (so an example meter partition-file name would be METR.METR). If you use the **sys:add-partition** function, specify the value **:metr** for the **:partition-type** keyword.

Section 28: Telnet

C.4.10 This section of the *Explorer Tools and Utilities* manual is only applicable if you have purchased the networking option to go with your microExplorer. If you have purchased this option, no differences exist in this section when applied to the Explorer or the microExplorer environment.

Section 29: VT100™ Emulator

C.4.11 This section of the *Explorer Tools and Utilities* manual is only applicable if you have purchased the networking option to go with your microExplorer. If you have purchased this option, notice that the microExplorer VT100 window contains no LEDs, and that VT100 graphics characters are not implemented on the microExplorer.

Section 30: Converse

C.4.12 This section of the *Explorer Tools and Utilities* manual is only applicable if you have purchased the networking option to go with your microExplorer. If you have purchased this option, no differences exist in this section when applied to the Explorer or the microExplorer environment.

Section 31: Mail

C.4.13 This section of the *Explorer Tools and Utilities* manual is only applicable if you have purchased the networking option to go with your microExplorer. If you have purchased this option, no differences exist in this section when applied to the Explorer or the microExplorer environment.

Section 32: Namespace Utilities

C.4.14 This section of the *Explorer Tools and Utilities* manual is only applicable if you have purchased the networking option to go with your microExplorer. If you have purchased this option, the only difference you should notice is in paragraph 32.2.5, Namespace Pathnames. The default pathname for a microExplorer will include a device component which is equal to the default device specified in the Startup file.

Section 33: Miscellaneous Network Functions

C.4.15 This section of the *Explorer Tools and Utilities* manual is only applicable if you have purchased the networking option to go with your microExplorer. If you have purchased this option, no differences exist in this section when applied to the Explorer or the microExplorer environment.

Section 34: Color Map Editor

C.4.16 This section is not applicable in the microExplorer environment.

Appendix A: Explorer Fonts

C.4.17 Fonts are represented and handled differently in the microExplorer environment than they are in the Explorer system. On the microExplorer, the **mac:*mac-font-translation-table*** maps most, but not all, of the fonts normally distributed on an Explorer system. Refer to Section 6, Fonts, for details on microExplorer font usage.

Appendix B: Command Tables

C.4.18 The following command tables are not applicable in the microExplorer environment:

- Font Editor Keystroke Assignments
- Color Map Editor Commands

Explorer Input/Output Reference

C.5 The description of serial and parallel streams in Section 1, entitled Streams, does not apply in the microExplorer environment.

Section 2: Pathnames

C.5.1 The version pathname component description in this section does not apply to Macintosh files. However, if you have the networking option, they are still relevant when referencing files on other types of hosts.

Section 3: Accessing Files

C.5.2 The description of file attribute lists and file properties in this section is considerably different in the microExplorer environment. See Section 8, Sharing File System Resources, for more information.

Other Sections

C.5.3 The following sections and appendixes are not applicable to the microExplorer environment:

- Section 6 — Maintaining a Disk
- Appendix C — International Considerations

Explorer Window System Reference

C.6 The *Explorer Window System Reference* manual has the following differences of which you should be aware before attempting to use the manual with the microExplorer.

Section 5: Visibility and Exposure

C.6.1 Paragraph 5.5: Pixels — The functions **w:black-on-white**, **w:white-on-black**, and **w:complement-bow-mode** are not applicable in the microExplorer environment.

Section 9: Fonts

C.6.2 Paragraph 9.8: Format of Fonts — All fonts used on the microExplorer have a **w:font-char-width-table**. No font has a **w:font-left-kern-table** or any bitmaps for the characters' glyphs, so the following functions have no meaning in the microExplorer environment: **w:font-raster-height**, **w:font-raster-width**, **w:font-rasters-per-word**, and **w:font-indexing-table**.

Section 11: The Mouse

C.6.3 The following paragraphs reflect differences in the microExplorer and the Explorer system environments.

- Paragraph 11.2: Mouse Variables and Functions — Variables dealing with mouse speed are meaningless in the microExplorer environment.
- Paragraph 11.3: Mouse Parameters — The variables **w:mouse-bounce-time** and **w:mouse-double-click-time** are not used.
- Paragraph 11.6: How Windows Handle the Mouse — The following methods and function are not applicable in the microExplorer environment: **:set-mouse-cursorpos**, **:set-mouse-position**, and **w:mouse-set-blinker-cursorpos**.
- Paragraph 11.8.1: — The microExplorer's scroll bars are Macintosh-style by default. The function **w:set-scrolling-style**, with a single argument of **:explorer** or **:mac**, can change the scrolling style to be that of the Explorer or back to that of the Macintosh.

Section 12: Graphics

C.6.4 All drawing is done using the Macintosh's QuickDraw Toolbox. The graphics model employed by QuickDraw differs in fundamental ways from that employed by the Explorer. A thorough knowledge of both models is needed to appreciate all the differences, but in general, the following items are true:

- For normal windows and text operations there is no visible difference.
- For drawing, especially filled regions, the two graphics systems behave differently.
- Applications using GWIN will be affected by the differences in the imaging models of the original Explorer and the Macintosh. Saved images (that is, pictures) in the graphics editor may be redrawn with slight differences on the microExplorer. The appearance of graphic output generated with GWIN may be different when an application is executed on a microExplorer versus an Explorer.

Section 18: Miscellaneous Features

C.6.5 Paragraph 18.3: Creating and Recording Sounds — The microExplorer supports only the beep sound (function and method).

Section 19: Using Color

C.6.6 The microExplorer does not support color as used in the Explorer window system.

SLE X Window System Programmer's Reference

C.7 Appendix B of the *SLE X Window System Programmer's Reference* manual describes the Explorer X11 Monochrome Server (X11M), which is not available for the microExplorer.

PORTING APPLICATIONS

D

Introduction

D.1 Because their environments are so similar, you can port applications between traditional Lisp machines and the microExplorer. When doing so, you must have the microExplorer Development System Software in order to take advantage of Zmacs, the compiler, and the microExplorer debug tools.

Porting an application from an Explorer system to run on the microExplorer normally requires that you follow a sequence similar to the following:

1. Transfer the application's object files to the Macintosh with the microExplorer configuration.

2. Load the files, and see how well they work. Many applications require no changes in order to run in both the Explorer and the microExplorer environments.

3. If you encounter problems with the application, transfer the application's source files to the Macintosh with the microExplorer configuration.

4. Modify the source files, excising any dependencies in the code from the source Lisp machine (as described in the following paragraphs). Also make any other necessary changes.

5. Compile your source files, then load and test your code. Testing should be performed *while still in the Development System Software environment*.

6. Continue to modify and test your code until your application is stable.

7. When all is ready, relaunch the microExplorer software required by your application, either the Delivery configuration (also called the System Software) or the Development System Software. Test the environment, load your application, and perform a disk save. Because the microExplorer uses files rather than partitions for its executable environment, the **disk-save** function saves your application and its Lisp environment to a file rather than to a partition.

NOTE: The microExplorer Delivery configuration is normally used with applications that require no window system interface of their own. These applications frequently use the interface provided by the Macintosh.

If your application has none of the dependencies described in the following paragraphs, it will probably run without modification. You do not even need to recompile the application. By following the guidelines in the following paragraphs, you can develop applications on an Explorer system to be run on a microExplorer.

Applications for Multiple Lisp Environments

D.2 If your application is to be run on both the microExplorer and on Explorer systems, you may need a predicate function that allows the application to discover for itself which system it is currently executing on. This predicate function is the **sys:mx-p** function. If the value returned by **sys:mx-p** is **t**, then your application is executing in the microExplorer environment. Otherwise, it is executing in the Explorer environment.

Window System Differences

D.3 Some minor differences exist between windows on the Macintosh and windows on the microExplorer, which are derived from the Explorer system.

For information on font differences, refer to Section 6, Fonts.

Display Size Variations

D.3.1 Due to the varying sizes of Macintosh II screens, your application may need to instantiate its windows on an Explorer screen sized anywhere from 424x608 pixels (or smaller) to 808x1024 pixels and beyond. By basing your windows' sizes on percentages of their superiors' sizes and using constraint frames, this variability in Macintosh II screen sizes can be handled nicely. Most Explorer system software windows are sized in this way. Not following this principle by using hard-coded window sizes will keep your product from executing on some Macintosh II configurations.

NOTE: If you have modified the window system in the environment under which you operate your Explorer system, the microExplorer Development System Software may be incompatible with that window-system environment.

Also, a virtual screen's initial appearance may not be what you expect if you acquire a load partition-file from a machine where someone has disk-saved an environment with different screen sizes.

Window Superiors

D.3.2 The microExplorer will support multiple virtual Explorer screens with each virtual Explorer screen mapping to a different Macintosh window. In this environment, the concept represented by the Explorer system variable **tv:main-screen** is no longer as straightforward as in an Explorer setting.

Use **tv:default-screen** as the superior for any top-level window being created, not **tv:main-screen**. **tv:default-screen** always refers to the Explorer virtual screen mapped to the topmost Macintosh window. **tv:main-screen** probably points to the first virtual Explorer screen created.

Selecting Screens

D.3.3 Because the microExplorer supports virtual screens, you may want your application to manage the selection of screens rather than requiring the user to change screens by using the Screens menu. Selecting a different virtual screen can be done by sending a **:select-screen** message to a window on that screen. The **:select-screen** operation has an effect similar to a **:select** message but first brings the screen to the front. Note that **:select-screen** is not supported in the regular Explorer window system environment.

Pixel AREFs

D.3.4 Direct references to the pixels on a screen (via the **aref** function), or to arrays that are displaced to the screen are not supported in the microExplorer environment.

Mouse Differences

D.4 Several differences exist between mouse usage on the Macintosh and mouse usage on the microExplorer (as derived from the Explorer system).

Clicking

D.4.1 The most obvious difference between the Macintosh and microExplorer mouse environments concerns the user interface. For example, how does a user click right, as the user of the Explorer system frequently must do, when the Macintosh mouse has only one button? The microExplorer Development configuration software has mapped the various Explorer three-button mouse clicks to equivalent mouse click/keystroke combinations. These mappings, which are shown on the keyboard overlay that comes with the microExplorer package, are as follows:

Table D-1 **Mouse Click Equivalents**

Documented microExplorer Mouse Click	Action on the Apple Mouse
L (click left)	Click the button (press the button once and release it).
M (click middle)	Click while holding down the Option key.
R (click right)	Click while holding down the Apple key.
L2 (double-click left)	Click the button twice quickly. (Press the button, release it, and press it again quickly.) *
M2 (double-click middle)	Click the button twice quickly while holding down the Option key.
R2 (double-click right)	Click the button twice quickly while holding down the Apple key.
LHOLD MHOLD RHOLD	Press the specified button/key combination and hold it down.

Note:
* If you double click too fast, the system sees only one click, If you double click too slowly, the system sees two single clicks.

Warping **D.4.2** Another area affected by mouse differences between the two environments concerns *mouse warping*; that is, changing of the mouse cursor's position by program control. The Macintosh does not support mouse warping; you can only move the Macintosh mouse cursor by moving the mouse itself.

Because mouse warping is not available on the Macintosh, several Explorer functions behave somewhat differently on the microExplorer. For example, when the system menu is brought up, the mouse cursor does not move to the center menu item. When a window is to be moved, the mouse cursor does not automatically jump to the upper left corner of the window's outline. When an application uses **tv:with-mouse-grabbed-on-sheet**, mouse cursor movement is not restricted to the specified sheet (although button clicks are delivered to the application as though the mouse had never left the sheet).

If your user will be bothered by the non-intuitive mouse behavior that results from the use of these functions on a microExplorer, you must change your code. Otherwise, no changes are required, since the mouse will function normally in all other respects.

A typical example of this behavior occurs in menus requiring you to choose from a number of items. In this situation, warping is used to preselect an item for default selection when a menu is displayed. Your application cannot perform this same preselection in the microExplorer environment, although the user can interactively select an item from the menu without hindrance.

Click Timing **D.4.3** Any attempt to time mouse clicks in microExplorer applications will fail, because mouse tracking, button clicks, and keyboard actions are all handled by the Macintosh (with messages passing between the Macintosh and the microExplorer to relay the information to the microExplorer). The Explorer window system's double click detection has been moved to the Macintosh; therefore, application-level double and triple click detection cannot work.

Cursor Glyphs **D.4.4** The microExplorer lets the Macintosh manage the mouse cursor as much as possible. Any Explorer mouse blinker of type **:character-blinker** is handled by the Macintosh. When **tv:mouse-set-blinker-definition** is called with a type of **:character-blinker**, the upper-left 16x16 pixels of the specified character's glyph are sent to the Macintosh and made the current Macintosh cursor. A Macintosh cursor mask is automatically generated for the glyph by the microExplorer. Drawing and erasing of such an Explorer mouse blinker are done entirely by the Macintosh, yielding the best possible mouse tracking.

Any Explorer mouse blinker of type **:bitblt-blinker** not larger than 16x16 pixels is handled just like mouse blinkers of type **:character-blinker**. When the mouse blinker is changed to a type other than **:character-blinker** or to a **:bitblt-blinker** whose glyph is larger than 16x16 pixels, the microExplorer gives the Macintosh a blank mouse cursor and proceeds to manage the drawing and erasing of the mouse blinker itself. Several commands must be sent to the Macintosh to erase and redraw such a blinker each time the mouse moves, yielding visibly slower mouse tracking.

Explorer mouse blinkers of type **:character-blinker** are handled as follows. The symbols **mac:mac-glyphs** and **mac:mac-mask-descriptors** contain arrays, which taken together, describe the Explorer mouse blinkers that the Macintosh can handle. The *n*th element of these arrays describe the *n*th glyph of the **fonts:mouse** font. Whenever **tv:mouse-set-blinker-definition** is called and the **fonts:mouse** property of the **mac:mac-glyphs** symbol is **nil**, these two arrays are initialized as follows. If the *n*th character in the **fonts:mouse** font has a glyph, then the *n*th entry of **mac:mac-glyphs** is loaded with a 16x16 bit array containing the glyph's image. The *n*th entry of **mac:mac-mask-descriptors** is loaded with a 16x16 bit array containing the glyph's mask. By adding a new glyph to an unused slot of **fonts:mouse**, setting the **fonts:mouse** property of **mac:mac-glyphs** to **nil**, and calling **tv:mouse-set-blinker-definition**, a new mouse blinker glyph may be made useable by the Macintosh.

It is possible to use a font other than **fonts:mouse** as a **:character-blinker** font if the following procedure is used. To use a character from the font named `fonts:my-font` as a mouse cursor glyph, first load the Explorer font file into the environment, then execute the following Lisp form:

```
(send w:default-screen :parse-font-descriptor fonts:my-font)
```

This form sets up a microExplorer font object as the value of the `fonts:my-font` symbol and stores the original Explorer font object on the **:explorer-font** property of the `fonts:my-font` symbol's property list. Note that only character codes for which the *Explorer* font has glyphs can be used as mouse blinkers.

It is also possible to use an existing Macintosh cursor as the Explorer's mouse blinker. If the *n*th entry in **mac:mac-glyphs** is a fixnum *x* instead of a bit-array, and **tv:mouse-set-blinker-definition** is called with an operation of **:set-character *n***, then the microExplorer has the Macintosh call GetCursor with an argument of *x*+16 to obtain a cursor to use. You must have registered your cursor in the microExplorer's resource file as resource number *x*+16 to use it this way.

Keyboard-Related Differences

D.5 Any time you map one keyboard to another, some differences are bound to occur both in the actual key-to-key correspondence and in the programming that is associated with keystroke sequences between the parent system of each keyboard. For details about key-to-key correspondences (and the microExplorer keyboard overlay) see Section 5, User Interface. For programmatic character code information, see Appendix B.

Limitations on Key Chording

D.6 Key chording on the microExplorer is constrained because the Macintosh has only three chording keys (Control, Option, and Apple/Command) whereas the Explorer has five chording keys (HYPER, SUPER, META, CTRL, and SYMBOL). The Explorer chording keys are mapped onto the Macintosh's chording keys as follows:

Explorer	Macintosh
CTRL	Control
META	Option
SUPER	Apple/Command
HYPER	Option *and* Apple/Command (simultaneously)
SYMBOL	Control *and* Apple/Command (simultaneously)

If you hold down all three of the Macintosh's chording keys (Control, Option, and Apple), there are three possible Explorer meanings for this key chord: SUPER-META-CTRL, META-SYMBOL, and HYPER-CTRL. This key chord is actually interpreted as HYPER-CTRL. Thus, the Explorer chords of META-CTRL, HYPER-CTRL, and SUPER-CTRL are possible on the microExplorer. However, the remaining 24 possible chordings of the five Explorer chording keys (HYPER-SUPER, HYPER-META, SUPER-META-CTRL, and so forth) are not supported.

NETWORK OPTION

E

Requirements

E.1 You need the following items to use the communications capabilities of the Network Option:

- Apple EtherTalk™ card with a revision greater than F. EtherTalk cards Revision F and prior require an upgrade to resolve data corruption problems. Contact your Texas Instruments or Apple Computer representative for instructions.

- Apple EtherTalk software version 1.1 or greater. You must inspect the diskette containing the EtherTalk software packaged with your EtherTalk card to determine the version before installation. You can verify the version of this software after installation as follows:

  1. In the Finder, open the System Folder.

  2. Select the icon labeled Ethertalk.

  3. Press Apple-I and a dialog box will appear. One of the lines within the box is labeled Version. If it does not say "Ethertalk 1.1", contact your Apple dealer for an upgrade.

If you previously installed patch release 5.0.1 onto your system, you should reinstall your EtherTalk driver after deleting the "Ethertalk" file from your System folder. This will avoid intermittent problems in booting your machine.

DECnet Communication Interface

E.2 When you boot your new band, if your namespace specifies a DECnet™ address for your host, you will see a warning that the microExplorer software is about to permanently modify your Mac system software. Note that this warning will actually appear as an error that you must proceed from.

If you confirm this choice with RESUME, the driver will place a new resource in the resource fork of your System file. You will not be able to use DECnet until you have rebooted your Macintosh and rebooted the microExplorer. After you have selected and completed this procedure, your System file includes a new resource containing an Ethernet™ address (which is determined by your DECnet address). The Ethernet address will be used by your EtherTalk board on every subsequent boot of your Macintosh.

WARNING: IT IS CRITICALLY IMPORTANT THAT THE MODIFIED SYSTEM FILE ON YOUR DISK *NEVER* BE COPIED OR MOVED TO ANOTHER MACINTOSH. If you do this, the modified Ethernet address will be used on the other system. Having two systems using the same Ethernet address at the same time leads to serious and difficult to diagnose problems. Always use Apple's Installer to set up a new System on a Macintosh.

Network File System (NFS)

E.3 Do not delete the LM:*default-disk*:NFS-FHANDLE: directory. Network NFS™ uses this directory to record information on file handles (fhandles) and their associated pathname objects. When a file or a directory on the Explorer is accessed from a Sun™ or an Apollo™ machine, a file handle is created if it is not in the cache. The **nfs:*record-fhandle-processor*** will then open a file in the LM:*default-disk*:NFS-FHANDLE: directory with *fhandle* as its file pathname and store *fhandle's* associated pathname object as the **:fhandle-pathname** property. This provides the capability to restore file handles between Explorer system boots. Use the function **nfs:clear-fhandle-table** to clear caches and delete entries in the NFS-FHANDLE directory.

You do not have to mount a Sun directory to be able to access that directory. The Explorer NFS software will mount all the file systems, for which you have access rights, to the Sun's export list (/etc/exports file) when you access the Sun the first time after the microExplorer system boot.

When mounting a microExplorer directory from a Sun, set the timeout to at least 50 (that is, timeo=50) to avoid multiple entries. The default value is 7.

INDEX

Alphabetization Scheme

The alphabetization scheme used in this index ignores package names and nonalphabetic symbol prefixes for the purposes of sorting. For example, the **sys:add-page-band** function is sorted under the entries for the letter A rather than under the letter S.

Information Technology Group - Austin
Documentation Questionnaire

microExplorer Development Software User's Guide

Do you use other TI manuals? If so, which one(s)?

______________________ ______________________

______________________ ______________________

______________________ ______________________

How would you rate the quality of our manuals?

	Excellent	Good	Fair	Poor
Accuracy	____	____	____	____
Organization	____	____	____	____
Clarity	____	____	____	____
Completeness	____	____	____	____
Overall design	____	____	____	____
Size	____	____	____	____
Illustrations	____	____	____	____
Examples	____	____	____	____
Index	____	____	____	____
Binding method	____	____	____	____

Was the quality of documentation a criterion in your selection of hardware or software?

☐ Yes ☐ No

How do you find the technical level of our manuals?

☐ Written for a more experienced user than yourself

☐ Written for a user with the same experience

☐ Written for a less experienced user than yourself

What is your experience using computers?

☐ Less than 1 year ☐ 1-5 years ☐ 5-10 years ☐ Over 10 years

We appreciate your taking the time to complete this questionnaire. If you have additional comments about the quality of our manuals, please write them in the space below. Please be specific.

____

____

____

____

____

Name ______________________ Title/Occupation ______________________

Company Name ____

Address ______________________ City/State/Zip ______________________

Telephone ______________________ Date ______________________

Manual Part No. 2552702-0001*C

TAPE EDGE TO SEAL

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS PERMIT NO. 7284 DALLAS, TX

POSTAGE WILL BE PAID BY ADDRESSEE

TEXAS INSTRUMENTS INCORPORATED
INFORMATION TECHNOLOGY GROUP
ATTN: PUBLISHING CENTER
P.O. BOX 149149 M/S 2146
AUSTIN, TEXAS 78714-9149

FOLD

2552702-0001*C

TEXAS INSTRUMENTS

PRINTED IN U.S.A.