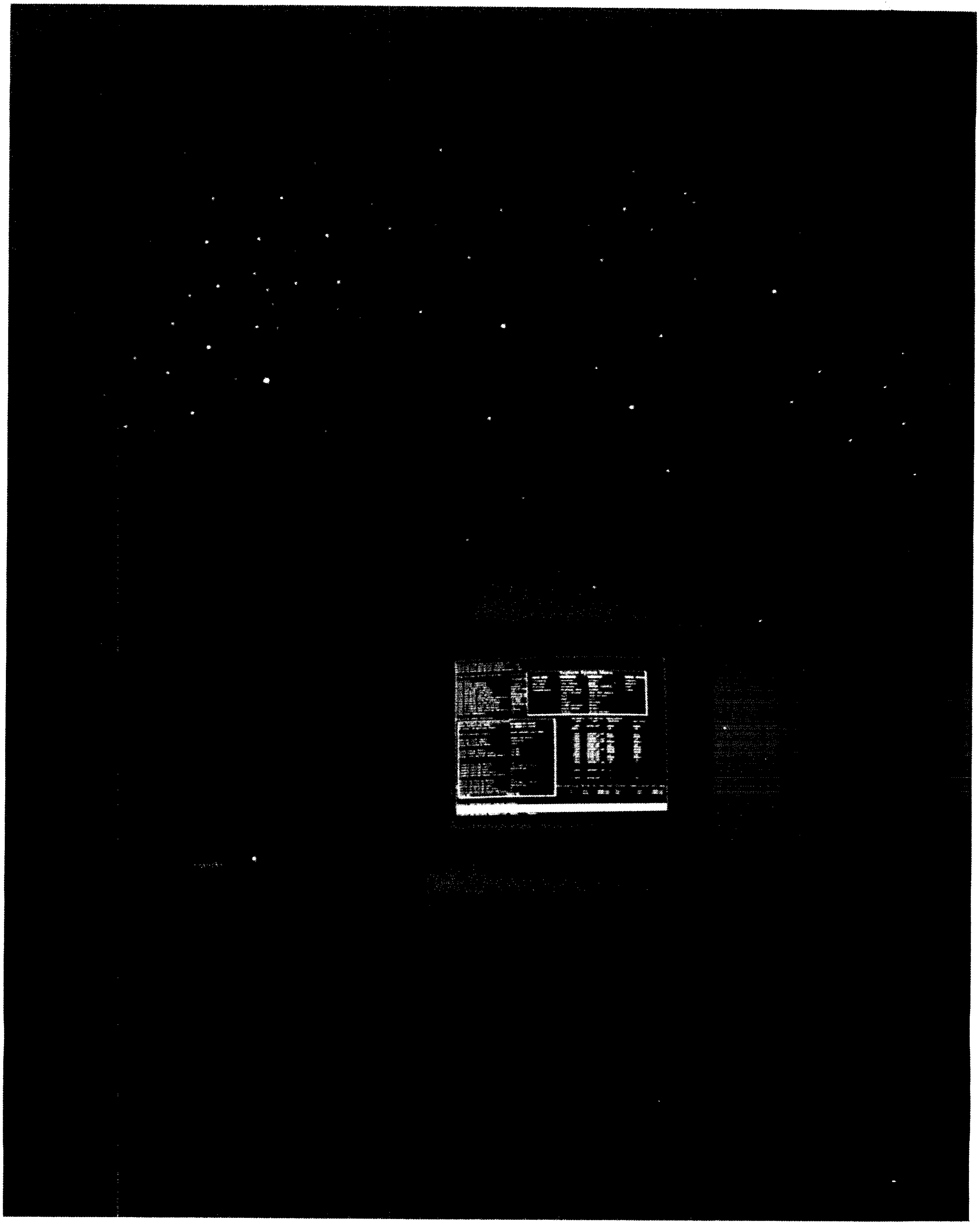
EXPLORER TECHNICAL SUMMARY



MANUAL REVISION HISTORY

Explorer Technical Summary (2243189-0001)

Original Issue	February 1985
Revision A	March 1986
Revision B	May 1986
Revision C	June 1987
Revision D	August 1988

The system-defined windows shown in this manual are examples of the software as this manual goes into production. Later changes in the software may cause the windows on your system to be different from those in the manual.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at 52.227-7013.

Texas Instruments Incorporated
ATTN: Data Systems Group, M/S 2151
P.O. Box 2909
Austin, Texas 78769-2909

CONTENTS

ABOUT THIS MANUAL

This summary introduces you to the Texas Instruments Explorer systems—the Explorer, the Explorer II, and the Explorer LX. These systems are designed specifically for high-speed Lisp processing in standalone environments and in combination with conventional and UNIX®-based environments. The Explorer systems offer a rich and highly productive programming environment for research and development of complex applications. Based on advanced 32-bit technology, the Explorer systems are packaged in compact, attractive enclosures, ideally suiting them for office operation.

The following terms are used in this manual:

- *Explorer* refers to the original Explorer system, a midrange, cost-effective computer.
- *Explorer II* refers to the Explorer II system, a high-performance computer based on the VLSI Explorer Lisp microprocessor.
- *Explorer systems* refers to both the Explorer and the Explorer II.
- *LX* refers to either an Explorer or an Explorer II system with an optional 68020 processor that includes UNIX-based system software.

This summary describes the major features, components, and design concepts of the Explorer systems. The summary also addresses the differences between the Explorer and the Explorer II. The summary is intended for managers, programmers, design engineers, and other technical people who require a broad understanding of system capabilities. The summary assumes you have a background in computer science and are familiar with conventional computer terminology and concepts.

For a general description of the product, read the first section. You can then refer to Sections 2 through 7 for more details on software features, and Sections 8 through 10 for hardware descriptions. Illustrations are used throughout the summary to highlight concepts. The screen formats give you an indication of the type of information displayed for the different system utilities. The actual contents of the screens vary considerably, depending on how you use the utilities, and the formats may vary according to the release of the software. This summary also includes an acronym list and a list of the software and hardware manuals that make up the Explorer systems documentation set.

1 EXPLORER SYSTEM OVERVIEW

The Explorer systems are advanced, single-user workstations that provide extensive support for development of knowledge-based applications, for rapid prototyping, and for research in new technologies, including artificial intelligence (AI). The Explorer systems are also affordable delivery vehicles for user applications that require symbolic processing, high-quality graphics, and special-purpose processors.

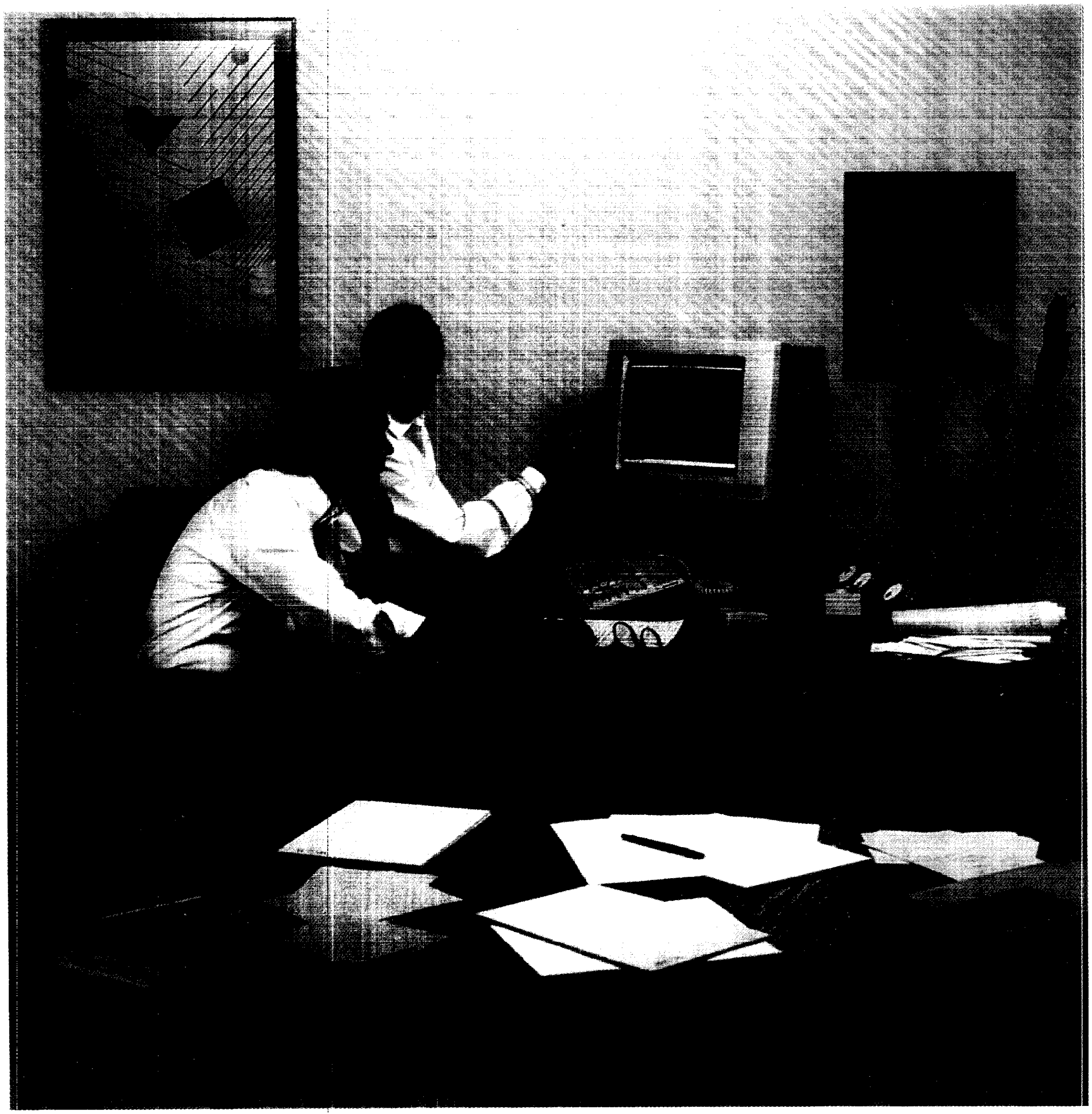
The Explorer systems offer a flexible, interactive, and highly productive programming environment. This environment permits you to use innovative programming techniques to solve difficult problems. This same environment provides run-time support for the execution of sophisticated, interactive programs. The Explorer systems offer you the following benefits:

- Natural manipulation of symbols that can represent abstract concepts using the Lisp language in a high-speed hardware architecture especially designed for symbolic processing
- Ability to handle ill-defined problems that have volatile types of data by using the tagged hardware and software support for run-time data type processing without compromising performance or safety
- Ease of program development using the extensive software provided, including a large number of high-level components and thousands of documented functions you can use as building blocks to construct programs (Software source code is part of the standard offering.)
- Improved programmer productivity in developing applications because of integrated programming tools that share a common environment
- Ability to create large programs without concern for memory allocation and deallocation, taking advantage of the Explorer systems' efficient incremental garbage collection and virtual memory management of up to 128 megabytes of virtual memory
- Transparent sharing of resources with a wide variety of computer systems through several networking facilities
- Access to a vast amount of information at one time with the high quality graphics and window system on the high-resolution display
- Instant feedback on exploratory programming because of incremental development support, essentially a what-you-see-is-what-you-get programming environment.

The Explorer system technology demonstrates innovative approaches in system architecture, processor design, and hardware packaging. NuBus technology provides device independence in system architecture, allowing configurations with multiple general- or special-purpose processors. The specially designed and microcoded Explorer systems processors directly implement the software run-time environment, providing fast execution of large programs without sacrificing the dynamic nature of Lisp. Hardware components are packaged in compact, attractive enclosures designed for easy maintenance and quiet, efficient operation in your office.

The open NuBus architecture allows Explorer systems to be available in several configurations, all of which are flexible and which can be expanded to meet your specific requirements. There are two primary models, the mid-range Explorer and the high-performance Explorer II. Each of these models is available with various amounts of memory and mass storage. The Explorer II combines the power of the a very large-scale integration (VLSI) processor chip with a cache, larger microcode store, and additional enhancements to provide significantly improved performance.

The NuBus allows for the addition of a 68020 microprocessor board for high-performance numeric processing from a UNIX-based System V environment. This processor is added to either of the primary systems to make Explorer LX or Explorer II LX systems. The LX systems provide for integration of Lisp with conventional software already written or applications being developed in C, FORTRAN, Pascal, and COBOL.



Explorer Software Environment

The requirements for very complex programs often cannot be specified in advance; you develop complex programs in incremental steps so that you can explore pieces of the problem and try different solutions before implementing the entire design. The Explorer software environment encourages this type of exploratory programming and allows you to interact freely with the system throughout all phases of the development cycle.

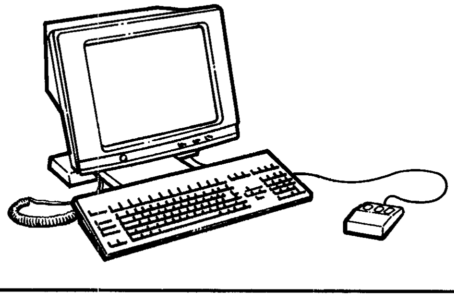
The software environment consists of the user interface, the Lisp language, and a rich set of integrated tools and facilities. The standard system provides all the tools necessary for program development and effective system management. Software options are available for specific networking protocols and for sophisticated features that you can incorporate into your applications.

Many features that contribute to the highly productive programming environment can also serve as major components of your application. You can use any of the system software, as well as the optional toolkits, as building blocks to rapidly develop a functional prototype. The environment provides tools to help you modify and enhance the prototype as it evolves into a finished product. The environment promotes development of a finished product that is optimized for performance but that still remains easily readable (and thus maintainable) source code. After the application is complete, the development tools are still available to update the application. In addition, you can use the patch facilities to upgrade or correct existing applications.

The software options include starter kits that teach you how to use a tool or that you can use as the base for an application. Your own application can, in turn, add functionality to the system and extend the environment to help you solve new, more complex problems.

User Interface

The user interface is a combination of hardware and software that gives you consistent, easy access to all system facilities. The monitor provides a high-resolution, bit-mapped display for viewing your input and system activities. The specialized keyboard and mouse provide quick access to items on the display. A sophisticated window system manages the display and lets you quickly and easily switch among activities. Help facilities offer a wide variety of online information at different levels of detail.



- **monitor, keyboard, mouse**
- **window system**
- **help facilities**

Lisp Language

- Common Lisp
- Extensions
  - Zetalisp
  - Flavors
  - Large function library

Lisp is an excellent programming language for applications that are dynamic or complex. Lisp run-time data type processing promotes rapid prototyping and easy adaptation to volatile inputs. The Explorer systems are written entirely in Lisp and support Common Lisp as well as extensions that encompass the full Zetalisp environment.

Flavors is provided as the object-oriented programming facility, a flexible mechanism that supports standard features of a message-passing, object-based paradigm.

A large library of functions comprise the system software. These functions are available in source code for you to review, change, and include in your own application.

Program Development Tools

- Zmacs Editor
- Lisp Listener
- Lisp compiler
- Debugging tools
- Inspector
- Flavor Inspector
- Universal Command Loop (UCL)
- Suggestions menus
- Tools for maintaining large systems
- Font Editor
- Namespace editor

The powerful Zmacs Editor provides context-sensitive editing for several languages and for text, with more than 600 commands available. Because all other tools are easily accessible from Zmacs and the tools are well integrated, most users operate from Zmacs as their base.

The Lisp Listener allows you to interact directly with the Lisp interpreter. The Lisp compiler allows programs to be incrementally defined; the compiler retains information needed for debugging. Thus, the program development cycle of edit/compile/debug can be repeated quickly and efficiently.

With the debugging tools, you can find and fix bugs quickly, even in complex programs. These tools are window-oriented and can operate directly on your source code. They include facilities for following and controlling the execution of your program, modifying its behavior, and inspecting the components of data structures and flavors.

The Universal Command Loop helps you construct a command interface for interactive applications.

Suggestions menus list the commands that are available to you in a particular utility. These menus are context-sensitive, changing as your activities change within a given utility.

Other facilities include tools for maintaining large systems, an editor for modifying and creating new fonts, and an editor for creating and modifying collections of named objects and classes.

Software Options

- TI Prolog
- Natural Language Menu (NLMenu) system
- Relational Table Management System (RTMS)
- Grasper network representation tool
- Explorer LX
- Personal Consultant Plus

The software options include toolkits designed to help you incorporate AI technology and other sophisticated features into your applications. They provide the basic framework for building components such as natural language interfaces, databases, and knowledge-based systems. These frameworks can be modified and customized easily to meet your needs.

TI Prolog has the advantage of full integration within the Lisp environment and always operates in a fully compiled mode.

Explorer LX provides software support for a conventional processor (68020) and the means to communicate between it and the Explorer systems processor. This multiprocessor system is uniquely suited to handling problems that require both symbolic and numeric processing.

Personal Consultant Plus provides a framework for building your own expert system.

In addition to the software options listed, Texas Instruments provides special products to suit particular customer needs.

A number of third-party suppliers also provide software for the Explorer systems.

System Facilities

- Memory management
- Microcode
- Performance monitoring
- Processes
- Stream I/O
- File system
- Backup and restore
- System status

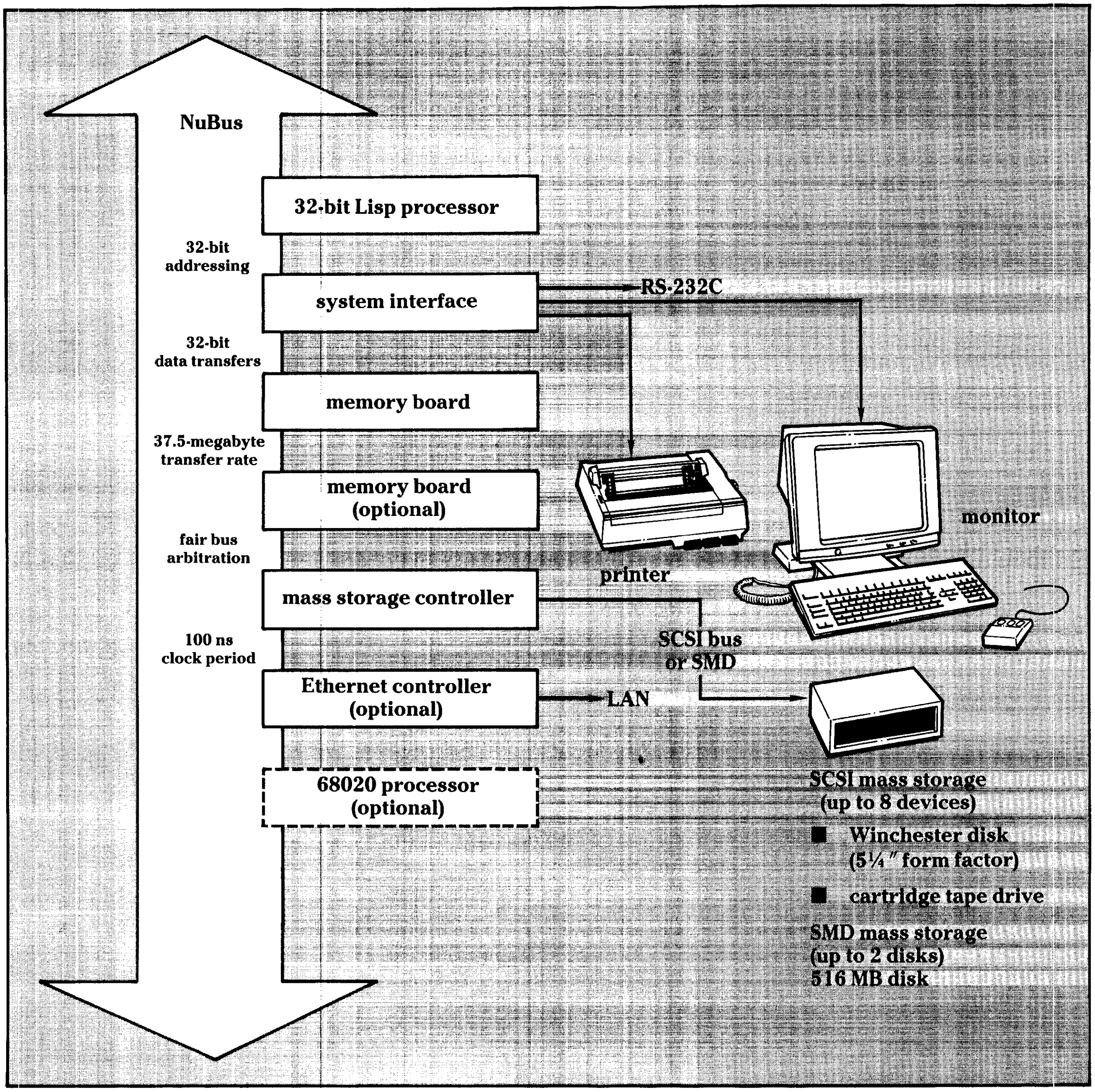
The Explorer systems provide a number of operating system facilities for run-time support and efficient implementation of Lisp programs. The memory management facilities automatically handle paging operations, dynamic storage allocation, and incremental garbage collection. A large number of the most frequently used functions are implemented in microcode to provide fast, efficient execution. The systems also provide facilities for performance monitoring, multitasking, stream input/output (I/O) to peripheral devices, a file system, backup and restore operations, and system status examination.

Networking Facilities

- Local area networks
- Ethernet interface
- Remote login
- Transparent file I/O
- Network mail

The Explorer systems give you the advantages of dedicated use as well as the flexibility to customize the environment. Networking facilities allow you to combine these advantages with the ability to share expensive resources and communicate with a wide variety of computer systems.

The Explorer systems support local area networks (LANs) using the industry-standard Ethernet interface. Facilities provided include remote login, transparent file I/O, and network mail using a variety of protocols—such as TCP/IP, NFS, DECnet and Chaosnet—to access mainframes, workstations, and personal computers. The SNA II option provides direct-line communication to IBM® mainframes.


NuBus
32-bit Lisp processor
32-bit addressing
system interface
RS-232C
32-bit data transfers
memory board
37.5-megabyte transfer rate
memory board (optional)
fair bus arbitration
mass storage controller
100 ns clock period
Ethernet controller (optional)
LAN
68020 processor (optional)
printer
monitor
SCSI bus or SMD
SCSI mass storage (up to 8 devices)
■ Winchester disk (5¼″ form factor)
■ cartridge tape drive
SMD mass storage (up to 2 disks) 516 MB disk

Explorer Systems Hardware Features

Explorer systems are used for large, complex software systems that demand high processor performance, large memory, and abundant disk capacity. The systems are centered around a backplane bus, called the NuBus, that provides high-speed exchange of data among processors, memory, mass storage, special devices, and remote computers (through networking). The NuBus processor-independent architecture and its multiprocessor support allow you to add new processors, memory, or devices to take advantage of new technologies or to tailor your system to meet specific application needs.

Processors and Memory

Explorer systems have a single-slot processor designed specifically for fast, high-level operation on Lisp data structures. Two processor types are available: Explorer and Explorer II. Either processor provides full 32-bit data paths and writable control store for microcode. The processors also provide hardware support for demand paging, garbage collection, and tagged data handling for better development performance than can be achieved by conventional processors. Additionally, the LX processor can be added to either system for smooth integration of a conventional processor.

The Explorer II processor is based on the Explorer Lisp microprocessor. Explorer II incorporates all the features of the Explorer processor with increased performance due to an increased clock rate and matched high-speed control store, architecture improvements to the processor design, and increased memory efficiency from a large cache, NuBus block transfer, and faster main memory. The Explorer II can be configured with 8, 16, or 32 megabytes of memory and can be expanded to 128 megabytes.

The LX processor is built around the Motorola 68020 microprocessor, which runs at 16.67 MHz. The processor has 2 megabytes of on-board memory and an optional 2-megabyte daughter board for expanded memory requirements. This processor operates in parallel with the Lisp processor, with high-bandwidth communications across the NuBus to allow independent or fully integrated processing.

The Explorer processor is closely coupled to system memory via a private, local memory bus. One or two memory boards—which are available in 4- or 8-megabyte capacities—can be attached to the Explorer local bus.

Mass Storage

The Explorer systems support mass storage in a variety of configurations through a single NuBus board. There are two mass storage controllers to choose from; both provide an interface to the industry-standard Small Computer Systems Interface (SCSI) bus, which supports both disk and tape devices. The MSC also provides a storage module drive (SMD) interface for larger high-performance disks. The mass storage devices are packaged separately from the system enclosure and can be daisy-chained to provide flexibility and opportunity for growth. Disk capacities range from 143 megabytes to 516 megabytes for a total system disk storage exceeding 2 gigabytes.

User Interface

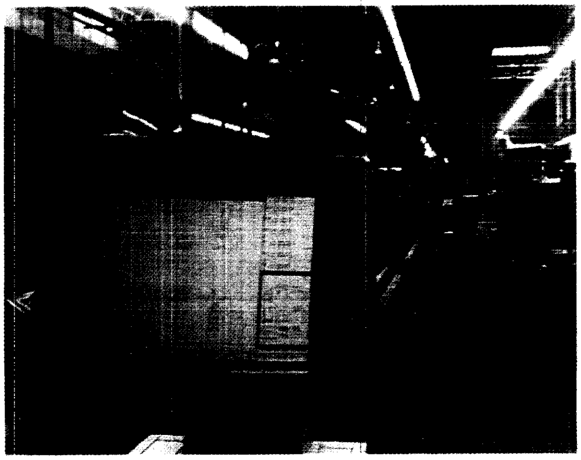
The system interface board provides a fiber-optic link to the monitor and both serial and parallel ports for printers and other devices.Logic on the monitor and system interface board includes graphics support for the high-resolution (1024 by 808 pixels) bit-mapped display.

Peripherals

The Explorer systems have peripherals that support specific requirements, including dot-matrix printers, laser printers, and special products such as a digital signal processor and a large-capacity chassis. (Special products are sold under special terms and conditions.)

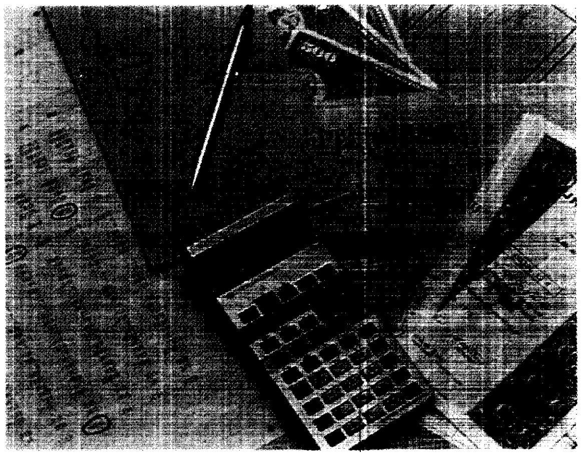
Explorer Applications

The high performance and powerful software environment of the Explorer systems have enabled an increasing number of companies to benefit from applications using AI technologies. Such technologies include knowledge-based systems in areas that aid in manufacturing operations, scheduling, equipment diagnostics and repair, financial planning, and the rapid prototyping of complex software systems.



Manufacturing

Explorer system-based applications are successful at improving productivity of operations in such areas as production scheduling and planning, inventory control, and the purchase of materials. Other applications address process quality, product design, and plant simulation. For example, operations departments are using Explorer systems for troubleshooting equipment problems while training personnel to repair and even to predict equipment failures before they become costly and critical.

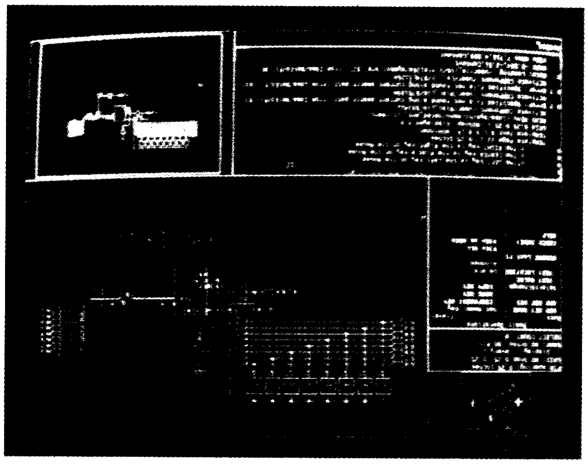


Business and Finance

Applications based on Explorer systems aid insurance companies with risk analysis of complex situations and with management of underwriting. Financial institutions are developing programs that assist with loan assessment and analysis, estate and portfolio management, and currency trading. Major wall street companies are developing applications that track stock prices and watch for insider trading, as well as assist in determining income possibilities. Major corporations are developing knowledge-based programs that assist in pricing and advertising for their products.

Defense/Aerospace

Knowledge-based systems are widely used in the defense and aerospace industries to provide advanced solutions to complex situations. Expert systems are used for design and verification of components for the space station and the space shuttle operations. Explorer-based systems are also used for information fusion, submarine tracking, helicopter failure diagnosis and repair, natural language front ends to large databases, and advanced avionics.



Research and Design

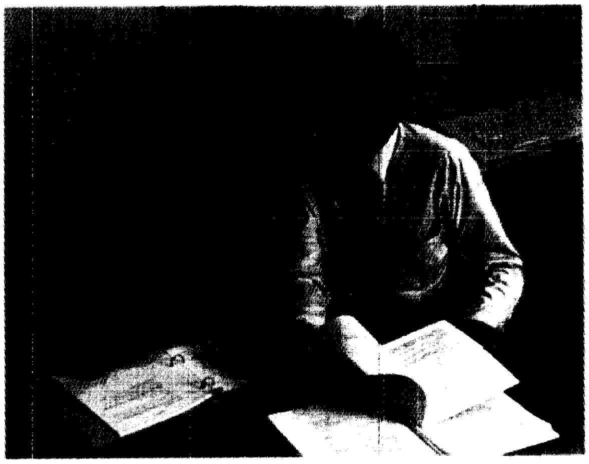
Explorer systems are ideal tools for research into areas such as neural networking, vision, speech, signal interpretation, and software development. The increased productivity and flexibility of the Explorer development environment meets research requirements for a powerful, fast, and reliable development system. For example, TI used the Explorer to aid in the VLSI design of the Lisp microprocessor for the Explorer II.

Service Industries

Explorer systems can be used to schedule expensive resources more efficiently. Airlines use Explorer systems for yield management, route planning, and scheduling. Service and repair industries are developing diagnostic systems to distribute expertise to distant locations. Other users are developing systems to control and diagnose phone switch gear and to predict communications problems.

Customer Support

The Explorer systems are backed by a network of customer support and service organizations to help you use your system effectively, develop applications, and maintain your system.



Knowledge Engineering Services

Texas Instruments provides Knowledge Engineers (KEs) who work with you to develop and customize your knowledge-based system. KE experience in design of expert systems and other complex applications combined with the specific knowledge of your domain expert can produce a unique program that you can amend and expand as changes and growth require. In addition, your application developers quickly gain hands-on experience by working closely with our KEs.

Knowledge Engineering Services have a wide variety of service offerings ranging from day-to-day consulting to multimonth contracts for large-scale projects. Prepriced packaged services include:

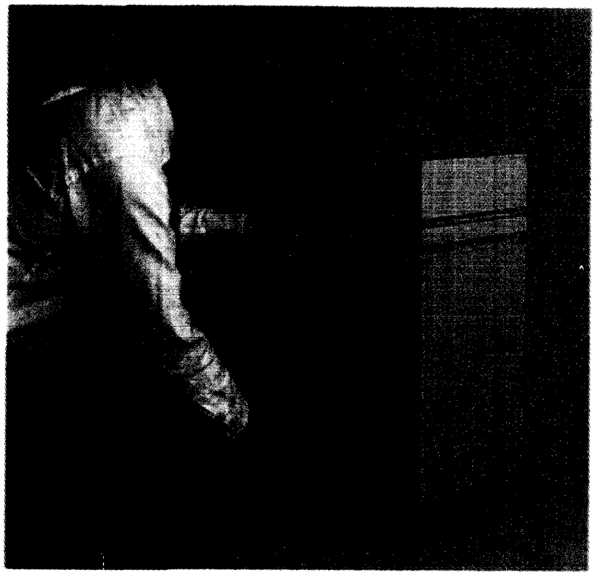
- Management assessment — A one- or two-month study exploring the application of AI technology to your company
- Application assessments — A multiweek study of various applications, resulting in a detailed report outlining their implementation
- Project proposals — A one- to four-month in-depth analysis of the resources required to complete a knowledge-based system on a specific application (A system prototype is often part of the proposal.)

Customer Engineering Support

The customer engineering group consists of system analysts, hotline personnel, and application engineers. They provide a technical interface between you and the product development group. The system analysts are trained in Lisp and AI techniques and are available in the field to provide you with technical information about the product and to help you find solutions to your problems. The hotline provides a direct factory contact. The application engineers maintain an awareness of vertical markets and evaluate customer requirements for future products and enhancements.

Training

The Texas Instruments Education Center offers live, hands-on training by qualified instructors. Training consists of introductory courses designed for managers and those curious about symbolic processing, as well as a series of courses designed for programmers and system designers who will be using Explorer systems. The series covers Lisp and the software environment, use of the Explorer toolkits, and other advanced topics. The courses are presented as lectures followed by practice sessions on the system. You can take these courses at a Texas Instruments site, or you can arrange to have them taught at your location.

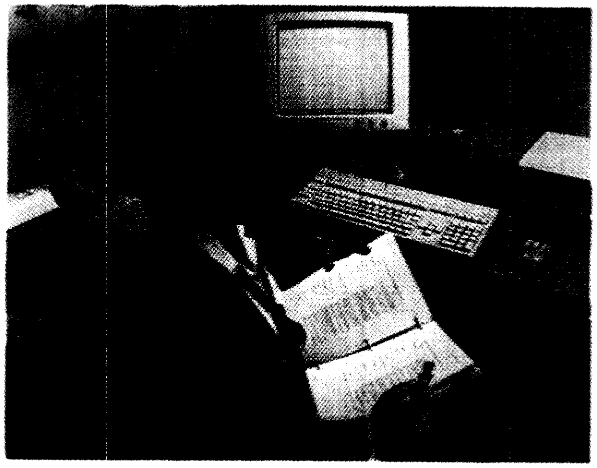


Field Service

For your convenience, Texas Instruments provides several types of onsite service by trained customer representatives in designated areas. These services include an installation service, a maintenance agreement service, and an on-call service.

Maintenance agreement service covers all routine maintenance defined in the contract except customer preventive maintenance responsibilities. It provides a variety of options. For example, extended coverage allows you to budget for maintenance costs and provides priority service over on-call service customers in both service scheduling and spare parts allocation during peak service periods.

Texas Instruments also provides a fixed-price repair service for those customers who stock their own spare parts and service their own equipment.



Documentation

In addition to online documentation, the Explorer systems include a comprehensive set of software and hardware manuals.

The software manuals cover all aspects of the Lisp environment and related software tools. The software manuals include tutorials and guides that help you become familiar with the system, and reference manuals that cover specific topics in detail. A glossary and an index to the complete manual set are also included.

The hardware manuals cover both the system level and the subassembly level of the various hardware configurations. They include information for preparing and operating your system, for explaining functional and circuit operations of the equipment, and for performing field maintenance.

USER INTERFACE

2

The Explorer systems user interface gives you quick, easy access to all system facilities. It consists of several hardware components and software to control them. The hardware components are collectively called the console (sometimes called the display unit) and include the following:

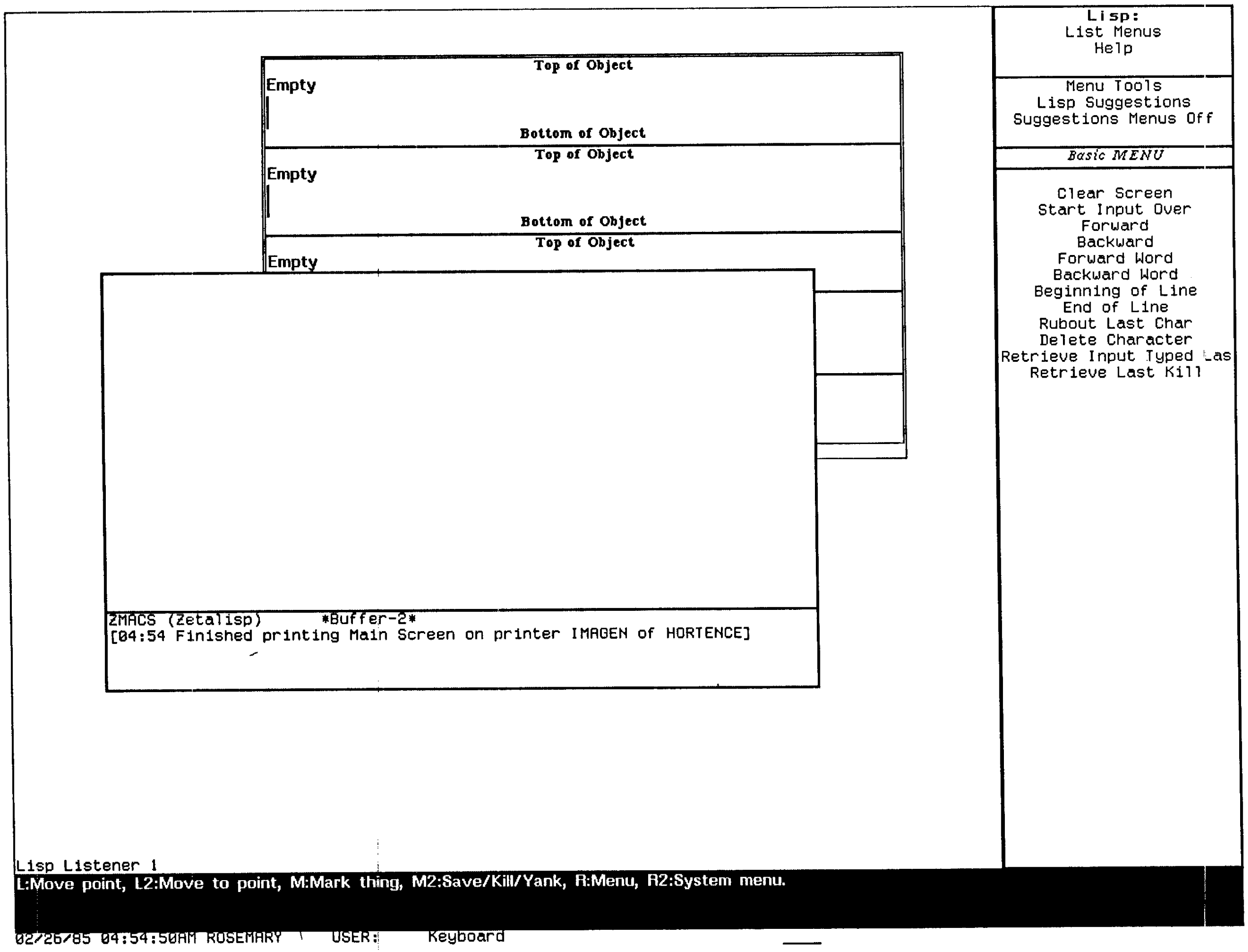
- A monitor that provides a high-resolution (1024 by 808 pixels), bit-mapped raster display in landscape orientation. The monitor supports high-quality graphics and multiple fonts.

- A keyboard with 111 keys, including character keys, function keys, and several levels of modifier keys. The software and the keyboard support chording of multiple keys, so you can invoke hundreds of commands with simple keystroke combinations.

- A high-resolution mouse that gives you quick access to items on the display. The mouse has three buttons that perform a variety of operations on the item to which the mouse cursor is pointing.

The software for the user interface is based on a sophisticated window system that manages the display of various activities on the screen and directs input from the keyboard or the mouse to the appropriate program. The Explorer systems also provide numerous online help facilities that guide you in using the system, promote the transition from novice to expert user, and provide quick reference to specific information.

The user interface is extremely flexible, letting you vary your method of interaction according to your preference or level of expertise. For example, you can invoke commands by using menus and the mouse, by typing command names, or by entering special keystroke combinations associated with the commands. The system also has a number of parameters you can modify to customize the user interface.



```
ZMACS (Zetalisp)      *Buffer-2*
[04:54 Finished printing Main Screen on printer IMAGEN of HORTENCE]
```

```
L:Move point, L2:Move to point, M:Mark thing, M2:Save/Kill/Yank, R:Menu, R2:System menu.
```

```
02/26/85 04:54:50AM ROSEMARY \    USER:     Keyboard
```

Window System

When you interact with the various system facilities, you do so through rectangular areas of the screen, called *windows*. A window is typically associated with one program and functions as a stream by providing a place for the program to write to and read from.

The window system controls the creation of windows and the allocation of display space among various programs. You can create any number of windows for different programs and split the screen to display several windows at one time. A window can vary in size or shape, overlap other windows, or completely cover them, much like papers on a desk. To help you identify the various types of windows and their location on the screen, most windows have borders and labels.

To interact with a program, you can either create a new window for it or select a window that already exists. You create or select windows by using menus and the mouse or by using keystrokes. When you select a window, it becomes visible and receives all input directed to it from the keyboard. You can select only one window for input at a time; however, you can shift rapidly back and forth among windows without losing their states or disrupting ongoing activities.

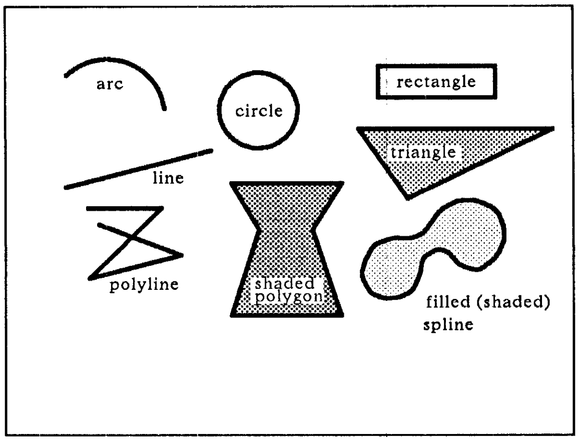
The window system keeps track of the mouse and the window currently in control of the mouse. As you move the mouse to point to different places on the screen, the shape of the mouse cursor may change to visually indicate its function in a particular place on the screen.. Certain items in a window are *mouse-sensitive*, which means they are highlighted in some way (usually a box appears around them) when you point to them. You can select a mouse-sensitive item by pointing to it and then clicking one of the mouse buttons.

Windows are arranged in a hierarchy, and each window has a superior and a list of inferiors. The top of the hierarchy is a screen; thus, a screen has no superiors. A window is an inferior of the screen, and a pane is an inferior of a window. Panes serve the same purpose as windows, and a group of related panes can be collected into a *frame*. You can manipulate a frame as a single object.

Some programs use windows to display the contents of a buffer, which may be too big to fit entirely in the window. In these cases, you see only a portion of the buffer at one time, and you can scroll it forward or backward using the mouse. Such windows contain scrolling zones that allow you to scan the buffer slowly. Some of them also have a scroll bar, which acts as a scale, allowing you to rapidly access any area of the buffer.

The window system represents a vast collection of software that provides a great deal of flexibility and functionality. It provides tools for creating windows interactively, graphics support, and menus and choice facilities.

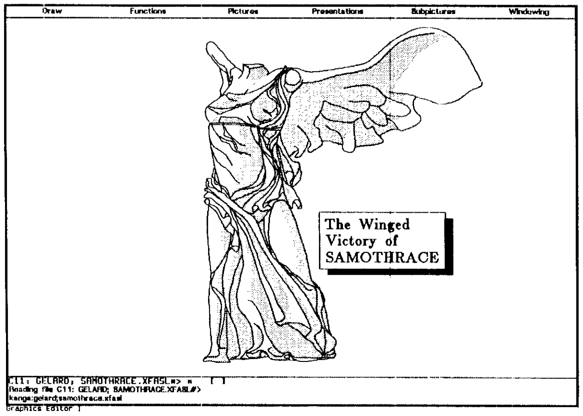
The window system can also be incorporated into your application as the basis for your user interface; you can customize windows to suit your application needs.



Graphics Primitives

You can incorporate graphics primitives into your application to draw a number of objects, including lines, splines, arcs, circles, polygons, and text. You can simply outline or fill in the objects for a shading effect. Once you have drawn them, you can manipulate these objects individually (such as copy, move, delete, or scale them), or you can collect them in a group and manipulate them as a unit.

You can combine, print, or store pictures created by the graphics primitives in buffers, files, or a database for later use. All applications that use the graphics window system can share pictures with one another.



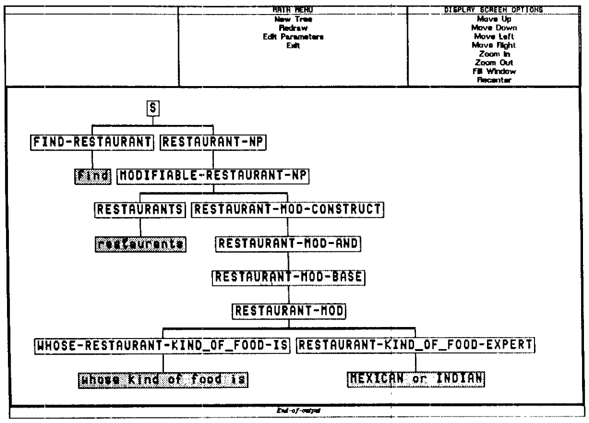
```
C11: GELARD; SAMOTHRACE.XFASL#> *    [ ]
Reading file C11: GELARD; SAMOTHRACE.XFASL#>
kanga:gelard;samothrace.xfasl
Graphics Editor ]
```

Graphics Editor

The Graphics Editor provides an interactive program for drawing pictures. All the graphics window primitives are available through menus of commands and mouse operations. For example, to draw an object, you simply select the appropriate command from a menu and use the mouse to set two or three points on the display that define the object.

The Graphics Editor also includes rulers and grids, which can overlay the display to specify points more accurately. The points you specify with the mouse are aligned on the nearest ruler or grid point. You can easily modify the spacing of ruler or grid points.

With the Graphics Editor, you can create pictures for presentations, reports, or manuals. Your applications can also use pictures or subpictures created with the Graphics Editor.

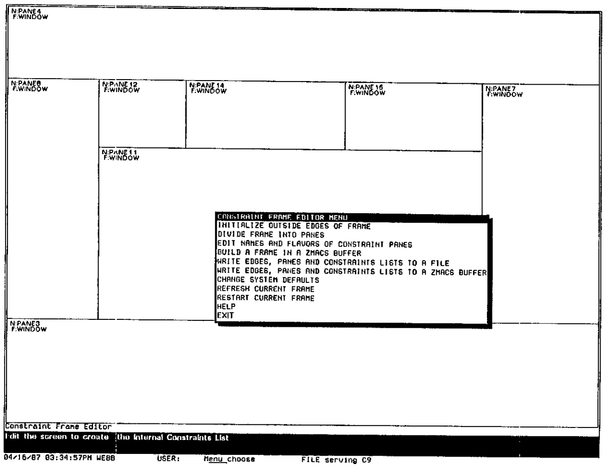


Tree Editor

The Tree Editor graphically displays any tree-structured entity. For example, you can display a hierarchy of combined flavors.

The Tree Editor provides commands that let you format the display either horizontally or vertically and pan or zoom in on the structure.

To use the Tree Editor for a specific application, you write simple interface routines that enable it to traverse the tree structure, to handle node selection using the mouse, and to provide editing functions on the tree structure or on the data in the nodes. The Tree Editor provides the flavors and methods to do the work.

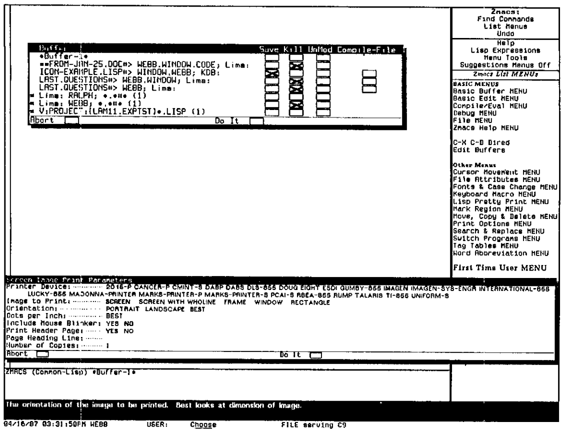

N:PANE4
F:WINDOW
N:PANE8
F:WINDOW
N:PANE12
F:WINDOW
N:PANE14
F:WINDOW
N:PANE16
F:WINDOW
N:PANE7
F:WINDOW
N:PANE11
F:WINDOW
N:PANE3
F:WINDOW
CONSTRAINT FRAME EDITOR MENU
INITIALIZE OUTSIDE EDGES OF FRAME
DIVIDE FRAME INTO PANES
EDIT NAMES AND FLAVORS OF CONSTRAINT PANES
BUILD A FRAME IN A ZMACS BUFFER
WRITE EDGES, PANES AND CONSTRAINTS LISTS TO A FILE
WRITE EDGES, PANES AND CONSTRAINTS LISTS TO A ZMACS BUFFER
CHANGE SYSTEM DEFAULTS
REFRESH CURRENT FRAME
RESTART CURRENT FRAME
HELP
EXIT
Constraint Frame Editor

Edit the screen to create the Internal Constraints List

04/16/87 03:34:57PM WEBB USER: Menu_choose FILE serving C9

Frame Editor

The Frame Editor is a menu-driven utility that generates code for constraint frames. A frame is a window that is divided into subwindows (panes), using the hierarchical structure of the window system. One kind of frame is the constraint frame, which adjusts the shapes of its panes automatically as its own shape is changed. The Frame Editor enables you to specify the position and size of panes in a constraint frame, specify the type and name of a pane, write the constraint frame code to a buffer or file, generate a sample constraint frame, and set the editing defaults for creating constraint frames.



Buffers	Save	Kill	UnMod	Compile-File
Buffer-1	□	□	□	
**FROM-JAN-25.DOC#> WEBB.WINDOW.CODE; Lima:	□	☒	□	
ICON-EXAMPLE.LISP#> WINDOW.WEBB; KDB:	☒	□	□	□
LAST.QUESTIONS#> WEBB.WINDOW; Lima:	□	☒	□	□
LAST.QUESTIONS#> WEBB; Lima:	□	☒	□	□
* Lima: RALPH; *.*#* (1)	□	□	□	
* Lima: WEBB; *.*#* (1)	□	☒	□	
* V:PROJECT:[LAM11.EXPTST]*.LISP (1)	□	□	□	

Abort □ Do It □

Screen Image Print Parameters

Printer Device: 2015-P CANCER-P CMINT-S DABP DABS DLS-855 DOUG EIGHT ESDI GUMBY-855 IMAGEN IMAGEN-SYS-ENGR INTERNATIONAL-855 LUCKY-855 MADONNA-PRINTER MARKS-PRINTER-P MARKS-PRINTER-S PCAI-S RBEA-855 RUMP TALARIS TI-855 UNIFORM-S
Image to Print: SCREEN SCREEN WITH WHOLINE FRAME WINDOW RECTANGLE
Orientation: PORTRAIT LANDSCAPE BEST
Dots per Inch: BEST
Include Mouse Blinker: YES NO
Print Header Page: YES NO
Page Heading Line:
Number of Copies: 1

Abort □ Do It □

ZMACS (Common-Lisp) *Buffer-1*

The orientation of the image to be printed. Best looks at dimension of image.

04/16/87 03:31:50PM WEBB USER: Choose FILE serving C9

Menus and Choice Facilities

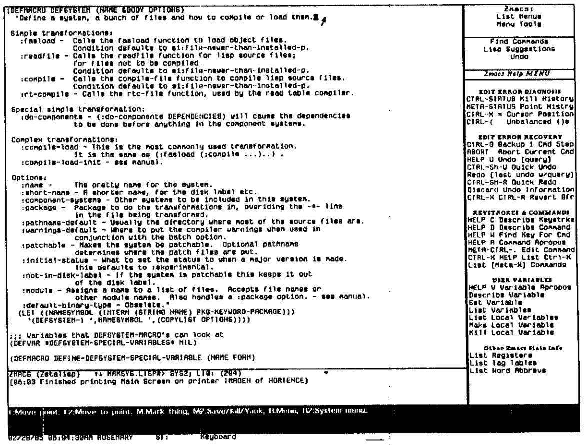
The system includes different types of menus, some of which apply to the system as a whole and some to specific activities.

Some menus are temporary; they remain invisible until you or a program calls them and then disappear as soon as you select an option from them. An example of a temporary menu is the System menu, which provides selections for each of the major utilities in the system.

The system also includes multiple-choice and choose-variable-values menus. A multiple-choice menu presents you with a list of items and several choice boxes next to each item. A choose-variable-values menu presents you with a set of variables and their current values. Some of the items list all of the valid values, with the current value shown in boldface. You can change a value by selecting the value with the mouse.

Help Facilities

Most of the utilities include special assistance features or commands to enhance ease of use. Such features complete partially entered commands and pathnames, provide a history of the last 60 commands or keystrokes used, and offer suggestions on what commands are available with the utility. Other help facilities provide a wide variety of information at varying levels of detail.



```
(DEFMACRO DEFSYSTEM (NAME &BODY OPTIONS)
  "Define a system, a bunch of files and how to compile or load them.
 Simple transformations:
   :fasload -  Calls the fasload function to load object files.
               Condition defaults to si:file-never-than-installed-p.
   :readfile - Calls the readfile function for lisp source files;
               for files not to be compiled.
               Condition defaults to si:file-never-than-installed-p.
   :compile -  Calls the compile-file function to compile lisp source files.
               Condition defaults to si:file-never-than-installed-p.
   :rt-compile - Calls the rtc-file function, used by the read table compiler.

 Special simple transformation:
   :do-components - (:do-components DEPENDENCIES) will cause the dependencies
               to be done before anything in the component systems.

 Complex transformations:
   :compile-load - This is the most commonly used transformation.
               It is the same as (:fasload (:compile ...)..) .
   :compile-load-init - see manual.

 Options:
   :name -     The pretty name for the system.
   :short-name - A shorter name, for the disk label etc.
   :component-systems - Other systems to be included in this system.
   :package -  Package to do the transformations in, overiding the -*- line
               in the file being transformed.
   :pathname-default - Usually the directory where most of the source files are.
   :warnings-default - Where to put the compiler warnings when used in
               conjunction with the batch option.
   :patchable - Makes the system be patchable.  Optional pathname
               determines where the patch files are put.
   :initial-status - What to set the status to when a major version is made.
               This defaults to :experimental.
   :not-in-disk-label - If the system is patchable this keeps it out
               of the disk label.
   :module - Assigns a name to a list of files.  Accepts file names or
               other module names.  Also handles a :package option. - see manual.
   :default-binary-type - Obselete."
  (LET ((NAMESYMBOL (INTERN (STRING NAME) PKG-KEYWORD-PACKAGE)))
    `(DEFSYSTEM-1 ',NAMESYMBOL ',(COPYLIST OPTIONS))))

;;; Variables that DEFSYSTEM-MACRO's can look at
(DEFVAR *DEFSYSTEM-SPECIAL-VARIABLES* NIL)

(DEFMACRO DEFINE-DEFSYSTEM-SPECIAL-VARIABLE (NAME FORM)
```

```
ZMACS (Zetalisp)   ↑↓ MAKSYS.LISP#> SYS2; L10: (284)
[06:03 Finished printing Main Screen on printer IMAGEN of HORTENCE]
```

L:Move point, L2:Move to point, M:Mark thing, M2:Save/Kill/Yank, R:Menu, R2:System menu.

02/28/85 06:04:30AM ROSEMARY   S1:   Keyboard

Suggestions

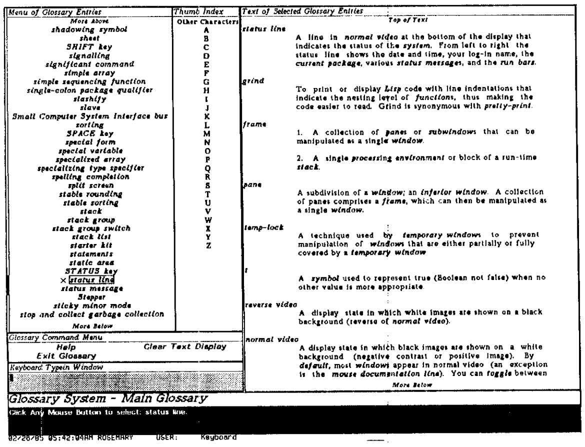
Each major system utility that has a large number of commands includes a Suggestions menu in one of its panes. The Suggestions menu lists the commands available to that utility, making operations readily accessible. When you point to an item on the Suggestions menu, the mouse documentation line gives a short description of the item as well as the keystroke associated with it. This allows you to access commands as a novice and learn the keystrokes for faster access. Once you become thoroughly familiar with the keystrokes, you can turn off the Suggestions menu to allow more room on the display for other panes.

Mouse Documentation Window and Status Line

The mouse documentation window (usually in reverse video) and the status line below it appear at the bottom of each main window. These lines are always visible; they can not be overlayed by other windows.

The mouse documentation window gives you a short description of currently available mouse operations.

The status line provides information on system status, such as the date and time, your login name, the current package, the state of the current process, the state of an open file, and the console idle time.



status line
A line in ***normal video*** at the bottom of the display that indicates the status of the ***system***. From left to right the status line shows the date and time, your log-in name, the ***current package***, various ***status messages***, and the ***run bars***.

grind
To print or display ***Lisp*** code with line indentations that indicate the nesting level of ***functions***, thus making the code easier to read. Grind is synonymous with ***pretty-print***.

frame
1. A collection of ***panes*** or ***subwindows*** that can be manipulated as a single ***window***.

2. A single ***processing environment*** or block of a run-time ***stack***.

pane
A subdivision of a ***window***; an ***inferior window***. A collection of panes comprises a ***frame***, which can then be manipulated as a single ***window***.

temp-lock
A technique used by ***temporary windows*** to prevent manipulation of ***windows*** that are either partially or fully covered by a ***temporary window***

t
A ***symbol*** used to represent true (Boolean not false) when no other value is more appropriate.

reverse video
A display state in which white images are shown on a black background (reverse of ***normal video***).

normal video
A display state in which black images are shown on a white background (negative contrast or positive image). By ***default***, most ***windows*** appear in normal video (an exception is the ***mouse documentation line***). You can ***toggle*** between

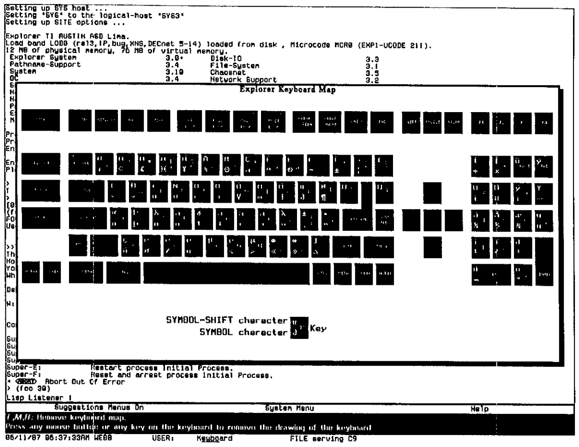
Glossary System - Main Glossary

Click Any Mouse Button to select: status line.

02/28/85 05:42:04AM ROSEMARY   USER: Keyboard

Glossary Utility

The Glossary utility provides online definition of terms specific to the Explorer systems and the Lisp environment. The utility allows you to scroll through the entire glossary or to access a particular definition. If the definition of one term contains words or phrases that are also in the glossary, the words are highlighted, and you can select them from within the definition by pointing to them and clicking the mouse. You can also add glossary files for your own applications.



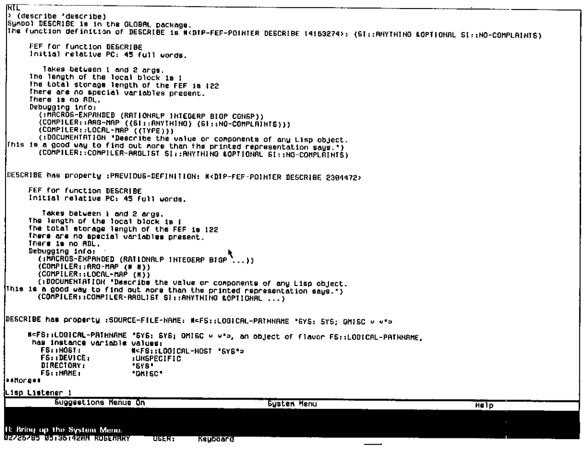
```
Setting up SYS host ...
Setting "SYS" to the logical-host "SYS3"
Setting up SITE options ...

Explorer TI AUSTIN ASD Lima.
Load band LOD8 (rel3,IP,bug,XNS,DECnet 5-14) loaded from disk , Microcode MCR8 (EXP1-UCODE 211).
12 MB of physical memory, 76 MB of virtual memory.
 Explorer System                3.0+    Disk-IO                3.3
 Pathname-Support               3.4     File-System            3.1
 System                         3.10    Chaosnet               3.5
 OC                             3.4     Network Support        3.2
```

```
Super-E:        Restart process Initial Process.
Super-F:        Reset and arrest process Initial Process.
+ <ABORT> Abort Out Of Error
> (foo 30)
Lisp Listener 1
```

```
L,M,R: Remove keyboard map.
Press any mouse button or any key on the keyboard to remove the drawing of the keyboard
06/11/87 06:37:33AM WEBB        USER:     Keyboard          FILE serving C9
```

HELP Key

The HELP key presents a menu of choices appropriate to the currently selected window. For many utilities, Basic Help displays a general description of the utility and provides information to get you started. Other Help items describe specific utility features or show a list of available commands and keystrokes. When used in combination with other keys (such as SYMBOL and TERM), Help describes the variety of options available with the other keys.



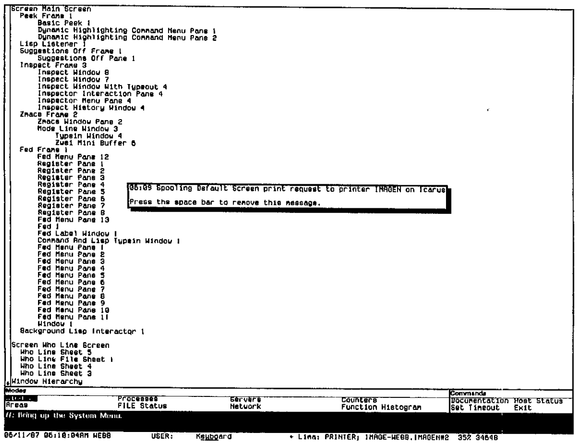
```
NIL
> (describe 'describe)
Symbol DESCRIBE is in the GLOBAL package.
The function definition of DESCRIBE is #<DIP-FEF-POINTER DESCRIBE 14163274>: (SI::ANYTHING &OPTIONAL SI::NO-COMPLAINTS)

     FEF for function DESCRIBE
     Initial relative PC: 45 full words.

        Takes between 1 and 2 args.
     The length of the local block is 1
     The total storage length of the FEF is 122
     There are no special variables present.
     There is no ADL.
     Debugging info:
       (:MACROS-EXPANDED (RATIONALP INTEGERP BIGP CONSP))
       (COMPILER::ARG-MAP ((SI::ANYTHING) (SI::NO-COMPLAINTS)))
       (COMPILER::LOCAL-MAP ((TYPE)))
       (:DOCUMENTATION "Describe the value or components of any Lisp object.
This is a good way to find out more than the printed representation says.")
       (COMPILER::COMPILER-ARGLIST SI::ANYTHING &OPTIONAL SI::NO-COMPLAINTS)


DESCRIBE has property :PREVIOUS-DEFINITION: #<DIP-FEF-POINTER DESCRIBE 2304472>

     FEF for function DESCRIBE
     Initial relative PC: 45 full words.

        Takes between 1 and 2 args.
     The length of the local block is 1
     The total storage length of the FEF is 122
     There are no special variables present.
     There is no ADL.
     Debugging info:
       (:MACROS-EXPANDED (RATIONALP INTEGERP BIGP ...))
       (COMPILER::ARG-MAP (# #))
       (COMPILER::LOCAL-MAP (#))
       (:DOCUMENTATION "Describe the value or components of any Lisp object.
This is a good way to find out more than the printed representation says.")
       (COMPILER::COMPILER-ARGLIST SI::ANYTHING &OPTIONAL ...)


DESCRIBE has property :SOURCE-FILE-NAME: #<FS::LOGICAL-PATHNAME "SYS: SYS; QMISC u u">

     #<FS::LOGICAL-PATHNAME "SYS: SYS; QMISC u u">, an object of flavor FS::LOGICAL-PATHNAME,
      has instance variable values:
       FS::HOST:             #<FS::LOGICAL-HOST "SYS">
       FS::DEVICE:           :UNSPECIFIC
       DIRECTORY:            "SYS"
       FS::NAME:             "QMISC"
**More**
Lisp Listener 1
```

```
R: Bring up the System Menu.
02/26/85 05:36:42AM ROSEMARY     USER:     Keyboard
```

Online Function Descriptions

With a single keystroke, you can see online descriptions for many commands and functions. Most commands and functions have only a short description, but more complicated ones have both short and long descriptions. Another keystroke gives you a list of function arguments and returned value(s).

Most utilities include apropos and describe facilities. The apropos facility searches the system for symbols that contain a specified substring. You can do a string search on command names, variable names, or Lisp functions. The describe facility displays information about a specified object, such as a symbol's value and its properties.



```
Screen Main Screen
  Peek Frame 1
      Basic Peek 1
      Dynamic Highlighting Command Menu Pane 1
      Dynamic Highlighting Command Menu Pane 2
  Lisp Listener 1
  Suggestions Off Frame 1
      Suggestions Off Pane 1
  Inspect Frame 3
      Inspect Window 8
      Inspect Window 7
      Inspect Window With Typeout 4
      Inspector Interaction Pane 4
      Inspector Menu Pane 4
      Inspect History Window 4
  Zmacs Frame 2
      Zmacs Window Pane 2
      Mode Line Window 3
          Typein Window 4
          Zwei Mini Buffer 6
  Fed Frame 1
      Fed Menu Pane 12
      Register Pane 1
      Register Pane 2
      Register Pane 3
      Register Pane 4
      Register Pane 5
      Register Pane 6
      Register Pane 7
      Register Pane 8
      Fed Menu Pane 13
      Fed 1
      Fed Label Window 1
      Command And Lisp Typein Window 1
      Fed Menu Pane 1
      Fed Menu Pane 2
      Fed Menu Pane 3
      Fed Menu Pane 4
      Fed Menu Pane 5
      Fed Menu Pane 6
      Fed Menu Pane 7
      Fed Menu Pane 8
      Fed Menu Pane 9
      Fed Menu Pane 10
      Fed Menu Pane 11
      Window 1
  Background Lisp Interactor 1

Screen Who Line Screen
  Who Line Sheet 5
  Who Line File Sheet 1
  Who Line Sheet 4
  Who Line Sheet 3
Window Hierarchy
```

```
06:09 Spooling Default Screen print request to printer IMAGEN on Icarus

Press the space bar to remove this message.
```

```
R: Bring up the System Menu.
06/11/87 06:10:04AM WEBB        USER:     Keyboard          + Lima: PRINTER; IMAGE-WEBB.IMAGEN#2  352 34648
```

Notification

When certain events occur that are unrelated to what you are currently doing with the selected window, the system notifies you with a beep and displays an explanatory message. Such an event might be an error in a process whose window is not exposed. Some windows accept notification from other windows and display the message enclosed in square brackets. Otherwise, a small temporary window containing the message pops up. Once you've read the message, you can use the mouse to remove this window from the display. Notifications are saved so you can look at them again if necessary.

3

THE LISP ENVIRONMENT

The Explorer supports a programming environment that is outstanding when the user wants to deal with complex problems, do incremental development, or do rapid prototyping of a difficult problem. The reason often cited is that the user has access to Lisp, which provides incredible flexibility. However, the Explorer offers more than just the Lisp language to provide the basis for these claims.

This section deals with three aspects of programming on Explorer systems:

- The general Lisp programming environment
- The Lisp language
- The flavors object-oriented programming paradigm supported on the Explorer

These form the heart of the user's language tools; however, these must be complemented by the integrated development tools described in the Section 4 and the underlying system facilities described in Section 5 to enable the programmer to really take advantage of their power.

Lisp Programming Environment

The Explorer systems, unlike many implementations of Lisp on other processors, have system software and applications software written completely in Lisp. An Explorer system is built for a single user, and that user controls the processes that are executing. The execution environment is an integrated one; there are no artificial walls between functions in one process (or application) and those in another. You need to know only the name of the function to be able to access it from anywhere; a single large linear address space contains all of the defined programs and data. Thus, there can be free access to the existing code anywhere in the system, allowing a user to invoke system-defined functions in applications and allowing for very flexible debugging tools. Because of the integrated base of software, debugging tools can be used to look at code currently being executed or to inspect values defined by any process or modify the current state of any structure.

Because all of the software executing is written in Lisp, definitions can easily be altered and those new definitions used immediately. A function definition is carried as one of the value slots of the function symbol, and a new definition can easily replace the old one. When the next call to that function is made, the new definition is used. There is no need to link together various function definitions to make use of them together; the appropriate current definition is always immediately available from the function symbol. The definitions in use can be compiled or they can be interpreted—users mix modes freely.

With the large linear address space (and large virtual memory support to handle it), the Explorer Lisp environment keeps available much auxiliary information about the code in memory to enable the use of powerful development tools. For example, each function carries a documentation string that describes its activity; the user can examine this string with a single keystroke to verify the function's syntax or check on allowable values for arguments. The compiler includes debugging information, such as names of the arguments of each function, along with its compiled form, as part of the function; this allows for rich debugging tools that can be used on compiled code. The function symbol also carries a field that tells the source file in which the definition appears; this allows the user to load the appropriate file for editing by using a single keystroke on the function name.

Programs written in Lisp generally use a lot of memory, most of it in structures of short life times. As in most Lisp systems, the Explorer Lisp system supports a garbage collection scheme to manage the reclaiming of memory no longer in active use. Unlike some systems, though, the Explorer has a nearly-invisible garbage collection mechanism that is very fast both because of its algorithms and because of the hardware support for tagged (data-typed) memory. With languages other than Lisp, users often have to do their own memory allocation and deallocation; with Lisp on the Explorer, all of the memory management is taken care of invisibly to the user. This is just one aspect of the many features provided by the system facilities described in a later section.

Lisp Language

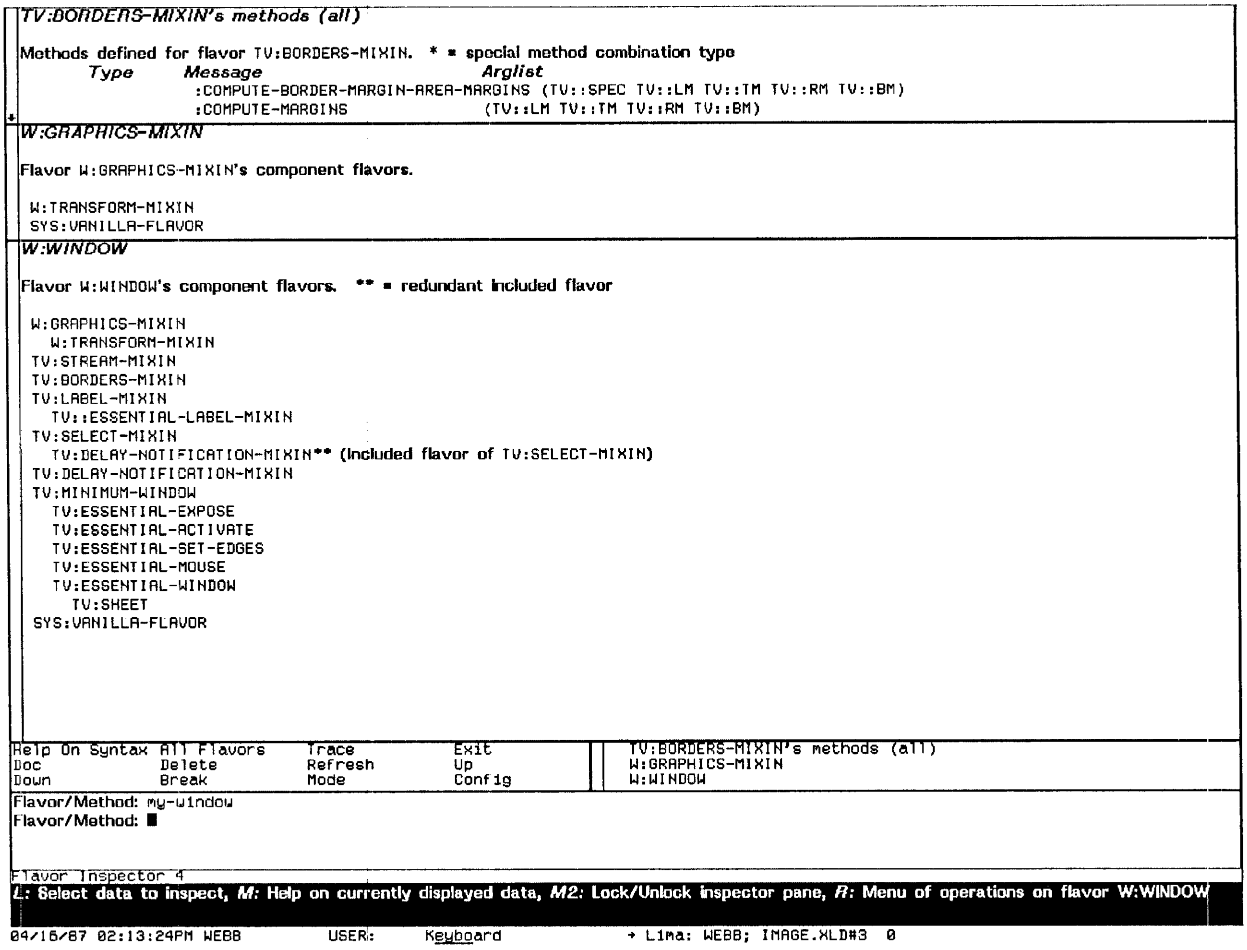
Lisp is a powerful, high-level language that has been widely used for AI research and is becoming increasingly popular for AI applications and general symbolic processing. It offers the expressive power needed for the implementation of complex systems. A number of features distinguish Lisp from most other programming languages:

- Symbolic processing — Lisp allows you to work with symbols and symbolic structures rather than just numbers and characters. Symbols are linked together in the form of list structures to represent any real-world object or abstract concept. Functions are provided to create and manipulate lists to form new and increasingly complex structures. Thus, Lisp provides an excellent foundation for representing, storing, and accessing knowledge in a form that can be processed.
- Flexibility — You can easily build programs that can construct other programs, a technique often used in AI programming, because Lisp program statements (functions) and Lisp data have the same form (lists). Storage is dynamically allocated and deallocated, allowing you to build structures of unpredictable sizes and shapes that evolve as the program executes. Lisp also allows you to define your own data types and functions and control their evaluation.
- Extensibility — Lisp allows you to build new and complex structures from existing ones. You can access user-defined functions and data types in the same way as system-defined functions and data types. This feature and the flexibility of Lisp make it easy for you to extend the language or create specialized languages closely related to the problems you want to solve.

The Explorer systems support Common Lisp, the industry standard for portability and consistency among different machines. The Common Lisp specification covers a rich set of data types, expressive control structures, macros, a package system for preventing name-space conflicts, and stream I/O.

In addition to Common Lisp, the Explorer systems support higher-level extensions to the language, including the flavor system, IEEE floating point representation, support for ISO characters, and handling of many types of pathnames and hosts. The extensions encompass the full Zetalisp environment.

Common Lisp is an evolving standard, and Texas Instruments plays an active role in the development of the standard. As the Common Lisp standard evolves, the Explorer systems implementation will conform to the emerging standard.



```
TV:BORDERS-MIXIN's methods (all)

Methods defined for flavor TV:BORDERS-MIXIN.  * = special method combination type
       Type       Message                      Arglist
                  :COMPUTE-BORDER-MARGIN-AREA-MARGINS (TV::SPEC TV::LM TV::TM TV::RM TV::BM)
                  :COMPUTE-MARGINS              (TV::LM TV::TM TV::RM TV::BM)
```

```
W:GRAPHICS-MIXIN

Flavor W:GRAPHICS-MIXIN's component flavors.

 W:TRANSFORM-MIXIN
 SYS:VANILLA-FLAVOR
```

```
W:WINDOW

Flavor W:WINDOW's component flavors.  ** = redundant included flavor

 W:GRAPHICS-MIXIN
   W:TRANSFORM-MIXIN
 TV:STREAM-MIXIN
 TV:BORDERS-MIXIN
 TV:LABEL-MIXIN
   TV::ESSENTIAL-LABEL-MIXIN
 TV:SELECT-MIXIN
   TV:DELAY-NOTIFICATION-MIXIN** (Included flavor of TV:SELECT-MIXIN)
 TV:DELAY-NOTIFICATION-MIXIN
 TV:MINIMUM-WINDOW
   TV:ESSENTIAL-EXPOSE
   TV:ESSENTIAL-ACTIVATE
   TV:ESSENTIAL-SET-EDGES
   TV:ESSENTIAL-MOUSE
   TV:ESSENTIAL-WINDOW
     TV:SHEET
 SYS:VANILLA-FLAVOR
```

```
Flavor/Method: my-window
Flavor/Method: █
```

```
Flavor Inspector 4
L: Select data to inspect, M: Help on currently displayed data, M2: Lock/Unlock inspector pane, R: Menu of operations on flavor W:WINDOW
04/16/87 02:13:24PM WEBB        USER:      Keyboard          → Lima: WEBB; IMAGE.XLD#3  0
```

Flavor System

The flavor system provides a powerful facility for object-oriented programming. This type of programming deals with *objects*, which are active entities representing real-world objects (such as ships or windows on the screen). An object encompasses data and the operations that can be performed on that data. Programs communicate with objects by sending messages to them. The flavor system hides the implementation of objects from programs that use them. Thus, programs using objects can be concerned only with what operations can be performed on the objects, not how those operations are coded. This technique reduces the complexity of large systems and makes them easier for you to use and maintain. It also allows you to reuse software developed for previous applications, a key to incremental prototyping.

A *flavor* is an abstract object that describes a whole class of similar objects. Thus, a flavor forms a conceptual model, and a specific object is an *instance* of the flavor. You can create any number of instances of a flavor. The flavor has *instance variables*, which specify attributes of the conceptual model. Each instance holds its own set of values for the instance variables. The flavor can establish default initial values for instance variables, or you can supply them explicitly when you instantiate a flavor.

A flavor also has attached *methods*, which are functions that define generic operations for any instance of that flavor. The methods can refer to any of the instance variables of that flavor. To operate on a specific object, you simply create an instance of the flavor and send the object messages naming the desired operations. The object reads each message, finds the appropriate method to handle the named operation, and returns the result.

The Explorer systems include a large number of predefined flavors, which you can instantiate and use in your programs. For example, the window system is based on flavors, and you can create instances of different types of existing window flavors. You can also define your own flavors. The flavor system provides many features to aid you in this process.

One of the most powerful features of the flavor system is the ability to combine existing flavors to form a new flavor. The new flavor inherits all the instance variables and methods belonging to the component flavors.

This ability to combine, or *mix*, flavors speeds development by allowing the easy integration of previously written and proven code into new programs.

PROGRAM DEVELOPMENT TOOLS 4

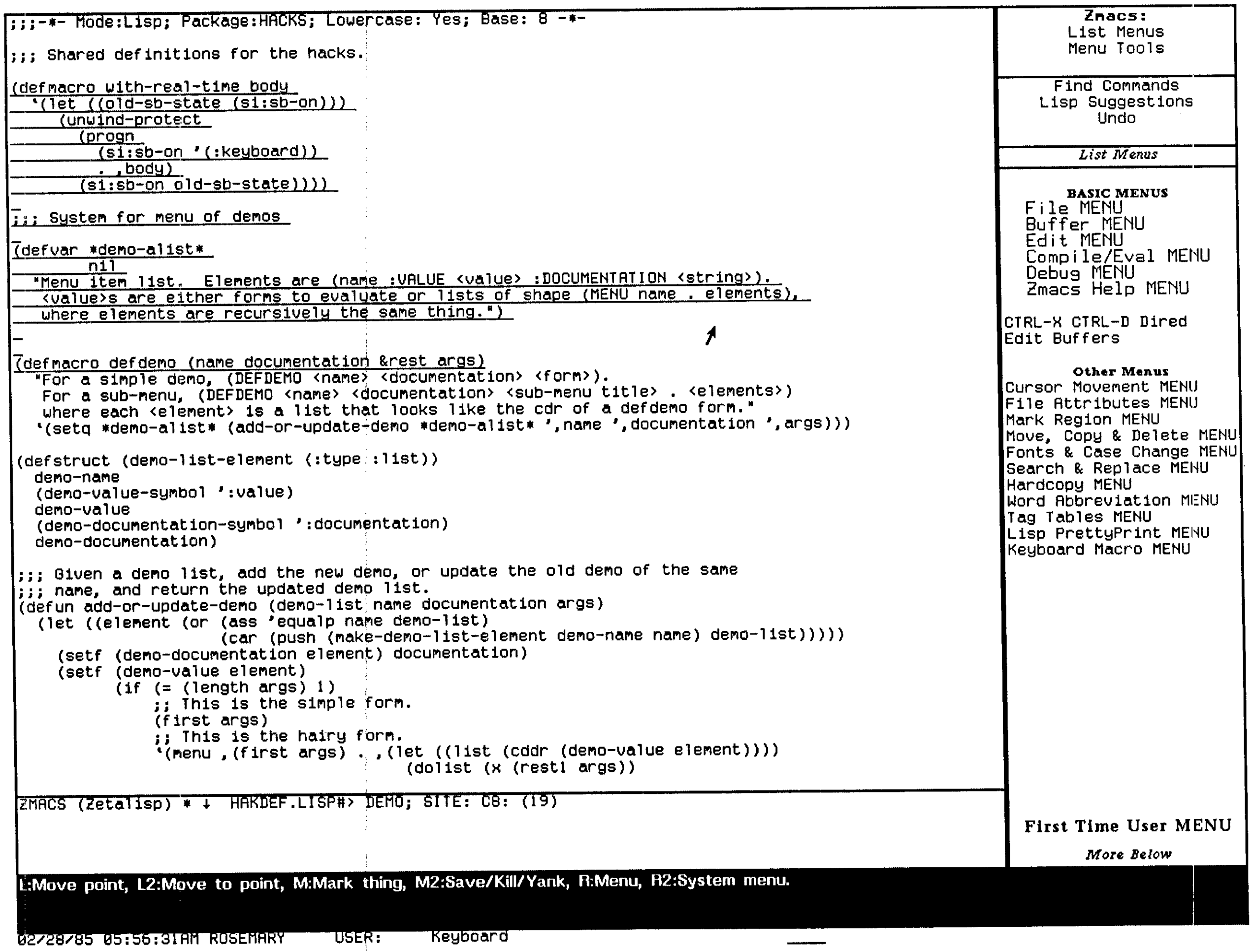
The Explorer systems provide a powerful and comprehensive set of tools for rapid program development. These tools enable you to execute partially developed programs. Because no explicit linking step is required, the Lisp system allows a program to be loaded even though some of its functions and variables are undefined. Thus, you can develop programs in pieces and rapidly move through the cycle of testing, debugging, and changing your program. You can start with simple tasks or existing code, and build on these until you are satisfied with the results.

Many of the tasks normally associated with program development are eliminated or simplified, increasing your productivity. You have access to the entire run-time environment during all phases of the development cycle. You can change a single function without having to recompile or relink the rest of your program; the function's callers will use the new definition. Memory is allocated, deallocated, and reclaimed automatically as needed.

The tools are highly integrated, so they can interact with each other and provide a unified setting for program development. For example, you can create, compile, execute, and debug Lisp programs without leaving the editor.

The program development tools consist of the following:

- Zmacs Editor
- Lisp Listener
- Lisp compiler
- Debugging tools
- Inspector
- Flavor Inspector
- Suggestions menus
- Universal Command Loop (UCL)
- Tools for maintaining large systems
- Font Editor
- Namespace Editor



```
;;;-*- Mode:Lisp; Package:HACKS; Lowercase: Yes; Base: 8 -*-

;;; Shared definitions for the hacks.

(defmacro with-real-time body
  `(let ((old-sb-state (si:sb-on)))
     (unwind-protect
       (progn
         (si:sb-on '(:keyboard))
         . ,body)
       (si:sb-on old-sb-state))))

;;; System for menu of demos

(defvar *demo-alist*
        nil
  "Menu item list.  Elements are (name :VALUE <value> :DOCUMENTATION <string>).
   <value>s are either forms to evaluate or lists of shape (MENU name . elements),
   where elements are recursively the same thing.")


(defmacro defdemo (name documentation &rest args)
  "For a simple demo, (DEFDEMO <name> <documentation> <form>).
   For a sub-menu, (DEFDEMO <name> <documentation> <sub-menu title> . <elements>)
   where each <element> is a list that looks like the cdr of a defdemo form."
  `(setq *demo-alist* (add-or-update-demo *demo-alist* ',name ',documentation ',args)))

(defstruct (demo-list-element (:type :list))
  demo-name
  (demo-value-symbol ':value)
  demo-value
  (demo-documentation-symbol ':documentation)
  demo-documentation)

;;; Given a demo list, add the new demo, or update the old demo of the same
;;; name, and return the updated demo list.
(defun add-or-update-demo (demo-list name documentation args)
  (let ((element (or (ass 'equalp name demo-list)
                     (car (push (make-demo-list-element demo-name name) demo-list)))))
    (setf (demo-documentation element) documentation)
    (setf (demo-value element)
          (if (= (length args) 1)
              ;; This is the simple form.
              (first args)
              ;; This is the hairy form.
              `(menu ,(first args) . ,(let ((list (cddr (demo-value element))))
                                        (dolist (x (rest1 args))
```

```
ZMACS (Zetalisp) * ↓  HAKDEF.LISP#> DEMO; SITE: C8: (19)
```

```
L:Move point, L2:Move to point, M:Mark thing, M2:Save/Kill/Yank, R:Menu, R2:System menu.
```

```
02/28/85 05:56:31AM ROSEMARY      USER:      Keyboard
```

Zmacs Editor

Zmacs is a real-time display editor, based on the M.I.T. Emacs Editor, that provides extensive support for writing Lisp and Prolog programs as well as other types of text. Zmacs is window-oriented; you can easily create any number of Zmacs buffers to edit different files and switch rapidly among them.

Zmacs offers hundreds of features not found in other editors. Commands manipulate text as units, including characters, words, lines, sentences, paragraphs, and regions. A region is any block of text that you define. Some of the operations you can perform on these units are move, copy, delete, transpose, indent, fill, justify, set fonts, and change to uppercase or lowercase.

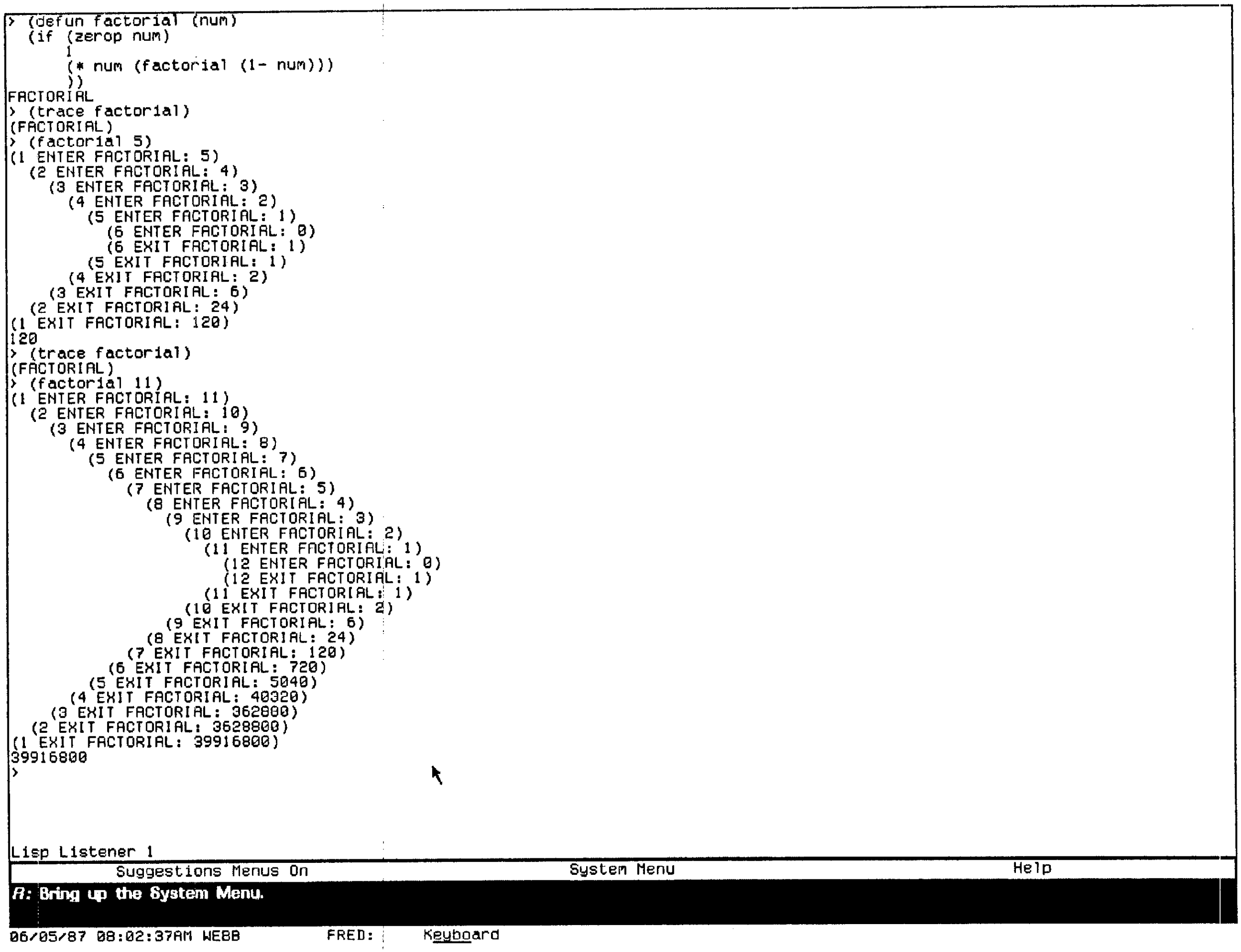
Zmacs is context sensitive; it provides modes that tailor the environment for editing a specific type of text. The mode in effect determines how Zmacs parses text and how commands operate. For example, in Common Lisp mode, Zmacs treats text according to the language syntax rules so you can move quickly through Lisp structures. Common Lisp mode also provides many features to help you write programs. These features include commands to perform the following:

- Match parentheses and check for unbalanced parentheses
- Properly indent and align code
- Evaluate or compile functions
- Locate and edit the source definition of Lisp objects (such as functions, macros, variables, and flavors)
- Locate and edit the callers of a function
- Locate and edit combined methods of a flavor

Zmacs is integrated with the general system help facilities. The Zmacs Suggestions menus list available commands for easy access and provide information about each command. Other help facilities include a history of the most recently invoked commands and completion on function names, command names, and pathnames.

Zmacs also provides facilities for creating and manipulating files and directories. A directory editor is embedded in Zmacs to help you maintain your directories and files.

Zmacs includes many features that allow you to tailor it to fit your needs. For example, you can set user variables to specify options, define your own modes and commands, or define keyboard macros. A keyboard macro allows you to define a sequence of keystrokes and invoke the sequence with a single command.



```
> (defun factorial (num)
  (if (zerop num)
      1
      (* num (factorial (1- num)))
      ))
FACTORIAL
> (trace factorial)
(FACTORIAL)
> (factorial 5)
(1 ENTER FACTORIAL: 5)
  (2 ENTER FACTORIAL: 4)
    (3 ENTER FACTORIAL: 3)
      (4 ENTER FACTORIAL: 2)
        (5 ENTER FACTORIAL: 1)
          (6 ENTER FACTORIAL: 0)
          (6 EXIT FACTORIAL: 1)
        (5 EXIT FACTORIAL: 1)
      (4 EXIT FACTORIAL: 2)
    (3 EXIT FACTORIAL: 6)
  (2 EXIT FACTORIAL: 24)
(1 EXIT FACTORIAL: 120)
120
> (trace factorial)
(FACTORIAL)
> (factorial 11)
(1 ENTER FACTORIAL: 11)
  (2 ENTER FACTORIAL: 10)
    (3 ENTER FACTORIAL: 9)
      (4 ENTER FACTORIAL: 8)
        (5 ENTER FACTORIAL: 7)
          (6 ENTER FACTORIAL: 6)
            (7 ENTER FACTORIAL: 5)
              (8 ENTER FACTORIAL: 4)
                (9 ENTER FACTORIAL: 3)
                  (10 ENTER FACTORIAL: 2)
                    (11 ENTER FACTORIAL: 1)
                      (12 ENTER FACTORIAL: 0)
                      (12 EXIT FACTORIAL: 1)
                    (11 EXIT FACTORIAL: 1)
                  (10 EXIT FACTORIAL: 2)
                (9 EXIT FACTORIAL: 6)
              (8 EXIT FACTORIAL: 24)
            (7 EXIT FACTORIAL: 120)
          (6 EXIT FACTORIAL: 720)
        (5 EXIT FACTORIAL: 5040)
      (4 EXIT FACTORIAL: 40320)
    (3 EXIT FACTORIAL: 362800)
  (2 EXIT FACTORIAL: 3628800)
(1 EXIT FACTORIAL: 39916800)
39916800
>
```

Lisp Listener

The Lisp Listener is a window-oriented program that allows you to interact directly with the Lisp interpreter. The Lisp Listener reads a Lisp form, evaluates it, and prints the results. This process is the *read-eval-print* loop used by a number of system facilities.

The Lisp Listener accepts forms as you type them and allows you to edit those forms. Input is complete and evaluation takes place when you finish typing or editing a complete Lisp form. Other features that help you develop programs in Lisp include the following:

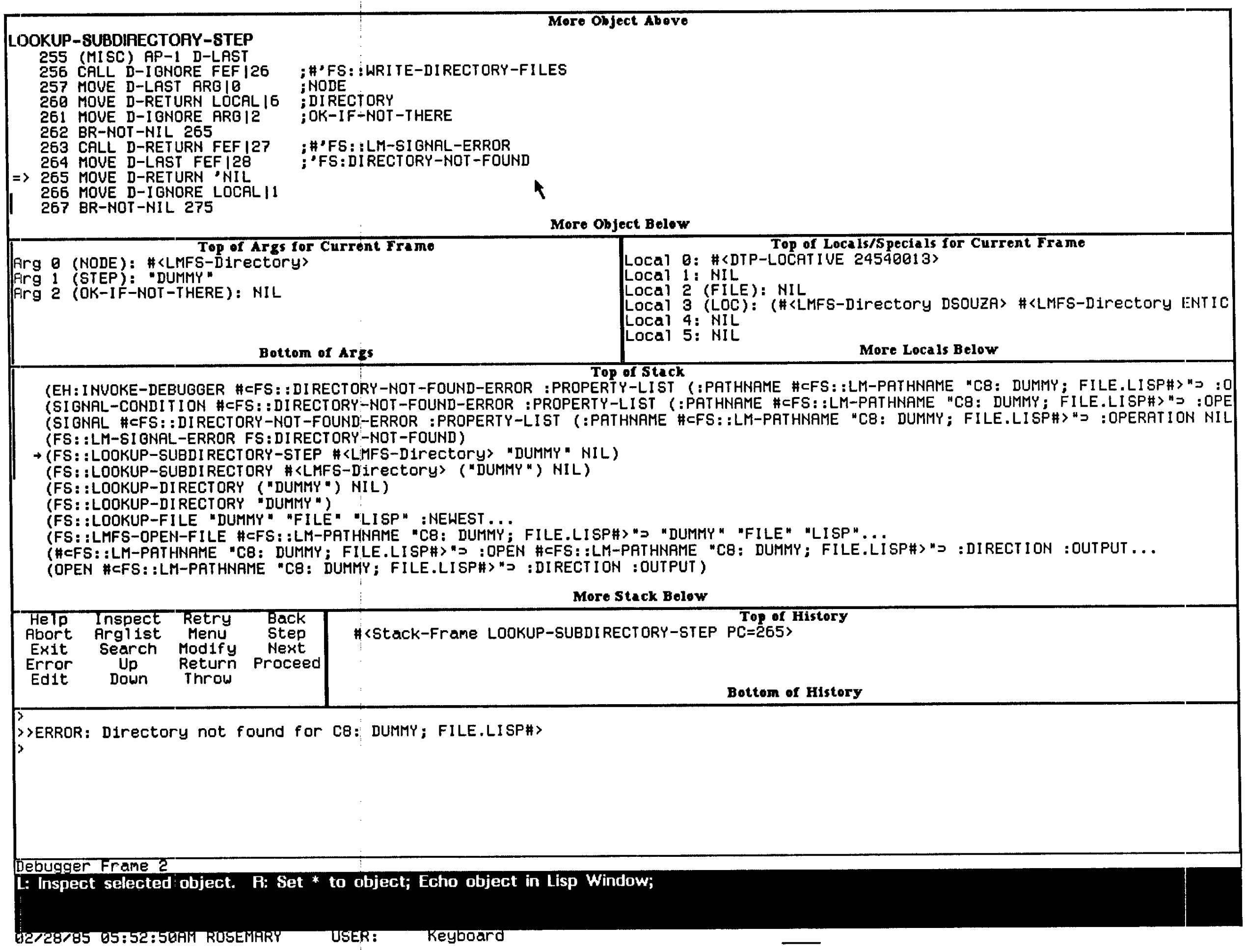
- An abbreviated form of message sending, which permits completion of method names and allows you to send messages implicitly to a flavor instance
- Online descriptions of function arguments and method arguments
- Completion commands for functions, methods, and symbols

Lisp Compiler

The compiler converts Lisp functions into machine code so they run faster and require less memory. Otherwise, compiled functions execute the same way as interpreted functions. The optimizations performed by the compiler include the following:

- Constant folding
- Source-level optimizations
- Peephole optimizations
- Branch optimizations and dead code elimination
- Tail recursion elimination
- Flavor combination optimizations
- Optional in-line expansion of function calls

You can compile or recompile portions of your code. The compiler checks for errors and issues warning messages, which are saved in a database. Zmacs includes commands to locate and edit files that contain compiler warnings.



```
LOOKUP-SUBDIRECTORY-STEP
   255 (MISC) AP-1 D-LAST
   256 CALL D-IGNORE FEF|26     ;#'FS::WRITE-DIRECTORY-FILES
   257 MOVE D-LAST ARG|0        ;NODE
   260 MOVE D-RETURN LOCAL|6    ;DIRECTORY
   261 MOVE D-IGNORE ARG|2      ;OK-IF-NOT-THERE
   262 BR-NOT-NIL 265
   263 CALL D-RETURN FEF|27     ;#'FS::LM-SIGNAL-ERROR
   264 MOVE D-LAST FEF|28       ;'FS:DIRECTORY-NOT-FOUND
=> 265 MOVE D-RETURN 'NIL
   266 MOVE D-IGNORE LOCAL|1
   267 BR-NOT-NIL 275
```

```
Arg 0 (NODE): #<LMFS-Directory>
Arg 1 (STEP): "DUMMY"
Arg 2 (OK-IF-NOT-THERE): NIL
```

```
Local 0: #<DTP-LOCATIVE 24540013>
Local 1: NIL
Local 2 (FILE): NIL
Local 3 (LOC): (#<LMFS-Directory DSOUZA> #<LMFS-Directory ENTIC
Local 4: NIL
Local 5: NIL
```

```
  (EH:INVOKE-DEBUGGER #⊂FS::DIRECTORY-NOT-FOUND-ERROR :PROPERTY-LIST (:PATHNAME #⊂FS::LM-PATHNAME "C8: DUMMY; FILE.LISP#>"⊃ :O
  (SIGNAL-CONDITION #⊂FS::DIRECTORY-NOT-FOUND-ERROR :PROPERTY-LIST (:PATHNAME #⊂FS::LM-PATHNAME "C8: DUMMY; FILE.LISP#>"⊃ :OPE
  (SIGNAL #⊂FS::DIRECTORY-NOT-FOUND-ERROR :PROPERTY-LIST (:PATHNAME #⊂FS::LM-PATHNAME "C8: DUMMY; FILE.LISP#>"⊃ :OPERATION NIL
  (FS::LM-SIGNAL-ERROR FS:DIRECTORY-NOT-FOUND)
 →(FS::LOOKUP-SUBDIRECTORY-STEP #<LMFS-Directory> "DUMMY" NIL)
  (FS::LOOKUP-SUBDIRECTORY #<LMFS-Directory> ("DUMMY") NIL)
  (FS::LOOKUP-DIRECTORY ("DUMMY") NIL)
  (FS::LOOKUP-DIRECTORY "DUMMY")
  (FS::LOOKUP-FILE "DUMMY" "FILE" "LISP" :NEWEST...
  (FS::LMFS-OPEN-FILE #⊂FS::LM-PATHNAME "C8: DUMMY; FILE.LISP#>"⊃ "DUMMY" "FILE" "LISP"...
  (#⊂FS::LM-PATHNAME "C8: DUMMY; FILE.LISP#>"⊃ :OPEN #⊂FS::LM-PATHNAME "C8: DUMMY; FILE.LISP#>"⊃ :DIRECTION :OUTPUT...
  (OPEN #⊂FS::LM-PATHNAME "C8: DUMMY; FILE.LISP#>"⊃ :DIRECTION :OUTPUT)
```

```
#<Stack-Frame LOOKUP-SUBDIRECTORY-STEP PC=265>
```

```
>
>>ERROR: Directory not found for C8: DUMMY; FILE.LISP#>
>
```

```
Debugger Frame 2
L: Inspect selected object.  R: Set * to object; Echo object in Lisp Window;
02/28/85 05:52:50AM ROSEMARY      USER:      Keyboard
```

Debugging Tools

The Explorer systems provide an interactive debugger as well as other tools to help you find and handle errors in your programs. The debugger allows you to examine the environment in which an error (or some other condition) is signaled, take corrective action and resume execution, or abort the program. When first entered, the debugger displays a message that describes the error and indicates where it occurred. Once inside the debugger, you have access to a wide variety of commands. For example, you can proceed from the error, examine and change the values of arguments or local variables, call the editor to change and recompile your source code, and evaluate a Lisp object or reinvoke a function within the context of the current environment.

A window-based debugger (example shown on the left) is also available. The window-based debugger has several panes, allowing you to view different types of information at one time.

Other debugging tools include the following:

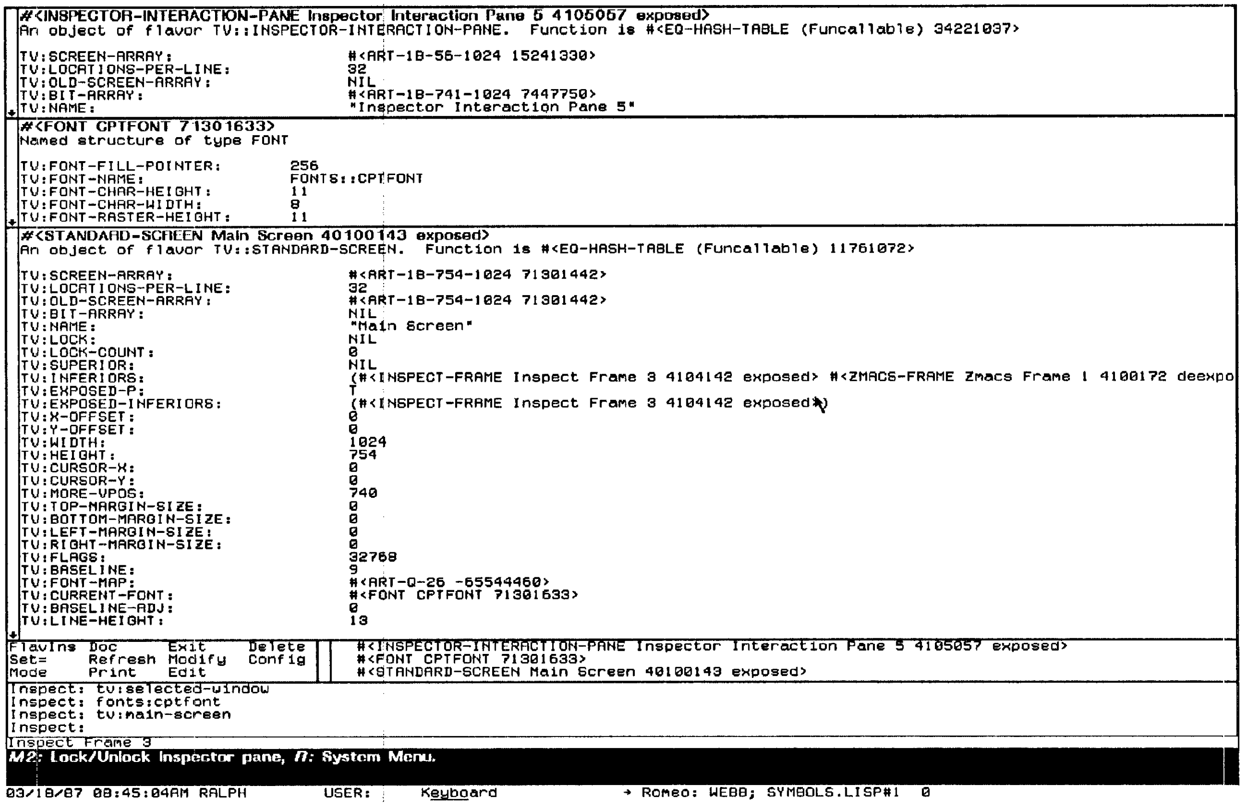
- A break facility, which halts execution of your program. You can specify a break point in your program or enter a break point from the keyboard any time during execution. You can also specify a break that invokes the debugger when a specified function is called or called under certain conditions.

- A trace facility, which allows you to trace specified functions. Each time you enter a traced function, a message containing the function's name and its arguments is optionally printed. When you exit the function, a second message is printed containing the function's name and its returned value(s). Options are available for you to specify other information to be printed, to invoke the step facility, or to specify trace conditions.

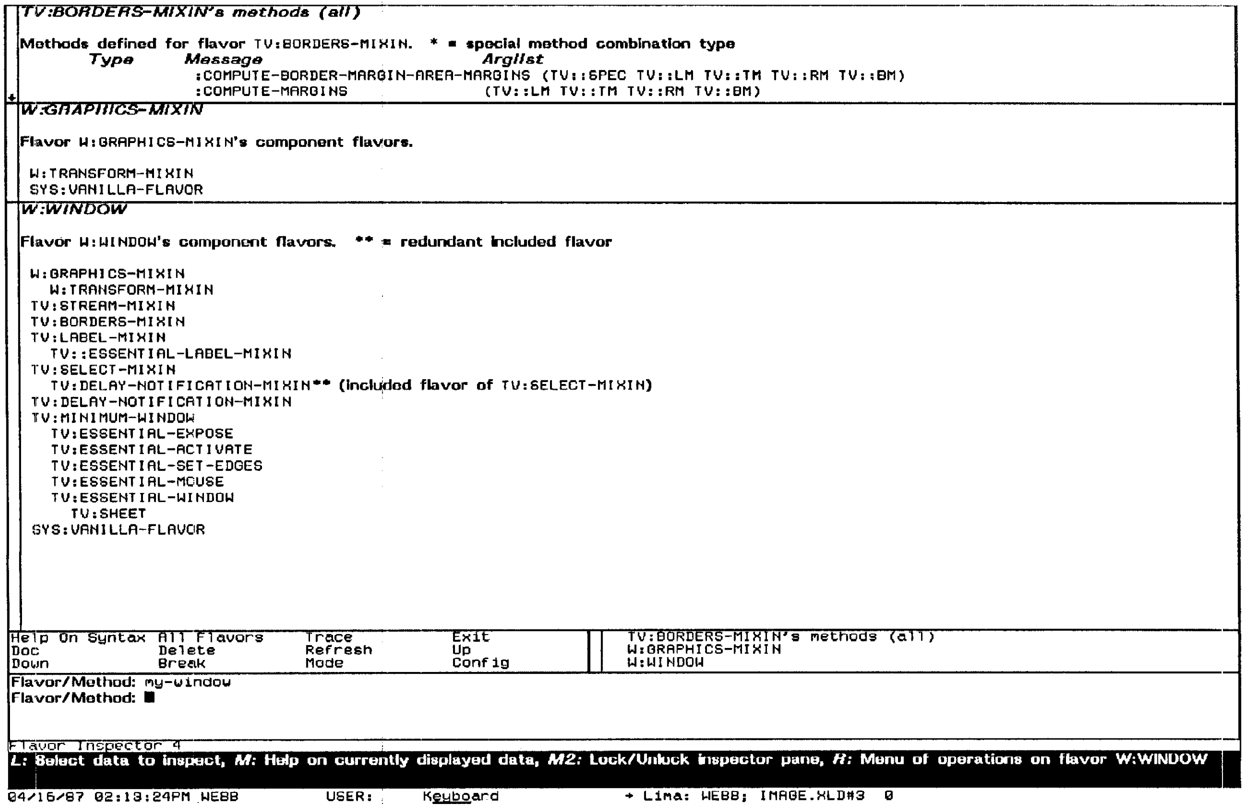
- A step facility, which allows you to follow each step in the evaluation of a Lisp form and examine what is occurring. This facility displays the next form to be evaluated and then waits for a command. The commands provide options on how to proceed to the next step. For example, you can continue to step through evaluations at the current level or go up to the next level in the stack.

- An advise facility, which allows you to modify a function's behavior by adding to it without actually changing its source definition. This facility is useful for testing a change to a function without committing to that change.

- Error or condition handlers, which allow you to specify error handling or specific conditions within your program. For example, you can create an error handler that ignores an error, proceeds from a condition, prints a message, or calls the debugger.



```
#<INSPECTOR-INTERACTION-PANE Inspector Interaction Pane 5 4105057 exposed>
An object of flavor TV::INSPECTOR-INTERACTION-PANE.  Function is #<EQ-HASH-TABLE (Funcallable) 34221037>

TV:SCREEN-ARRAY:                #<ART-1B-56-1024 15241330>
TV:LOCATIONS-PER-LINE:          32
TV:OLD-SCREEN-ARRAY:            NIL
TV:BIT-ARRAY:                   #<ART-1B-741-1024 7447750>
TV:NAME:                        "Inspector Interaction Pane 5"
#<FONT CPTFONT 71301633>
Named structure of type FONT

TV:FONT-FILL-POINTER:       256
TV:FONT-NAME:               FONTS::CPTFONT
TV:FONT-CHAR-HEIGHT:        11
TV:FONT-CHAR-WIDTH:         8
TV:FONT-RASTER-HEIGHT:      11
#<STANDARD-SCREEN Main Screen 40100143 exposed>
An object of flavor TV::STANDARD-SCREEN.  Function is #<EQ-HASH-TABLE (Funcallable) 11761072>

TV:SCREEN-ARRAY:                #<ART-1B-754-1024 71301442>
TV:LOCATIONS-PER-LINE:          32
TV:OLD-SCREEN-ARRAY:            #<ART-1B-754-1024 71301442>
TV:BIT-ARRAY:                   NIL
TV:NAME:                        "Main Screen"
TV:LOCK:                        NIL
TV:LOCK-COUNT:                  0
TV:SUPERIOR:                    NIL
TV:INFERIORS:                   (#<INSPECT-FRAME Inspect Frame 3 4104142 exposed> #<ZMACS-FRAME Zmacs Frame 1 4100172 deexpo
TV:EXPOSED-P:                   T
TV:EXPOSED-INFERIORS:           (#<INSPECT-FRAME Inspect Frame 3 4104142 exposed>)
TV:X-OFFSET:                    0
TV:Y-OFFSET:                    0
TV:WIDTH:                       1024
TV:HEIGHT:                      754
TV:CURSOR-X:                    0
TV:CURSOR-Y:                    0
TV:MORE-VPOS:                   740
TV:TOP-MARGIN-SIZE:             0
TV:BOTTOM-MARGIN-SIZE:          0
TV:LEFT-MARGIN-SIZE:            0
TV:RIGHT-MARGIN-SIZE:           0
TV:FLAGS:                       32768
TV:BASELINE:                    9
TV:FONT-MAP:                    #<ART-Q-26 -65544460>
TV:CURRENT-FONT:                #<FONT CPTFONT 71301633>
TV:BASELINE-ADJ:                0
TV:LINE-HEIGHT:                 13
FlavIns Doc     Exit    Delete      #<INSPECTOR-INTERACTION-PANE Inspector Interaction Pane 5 4105057 exposed>
Set=    Refresh Modify  Config      #<FONT CPTFONT 71301633>
Mode    Print   Edit                #<STANDARD-SCREEN Main Screen 40100143 exposed>
Inspect: tv:selected-window
Inspect: fonts:cptfont
Inspect: tv:main-screen
Inspect:
Inspect Frame 3
M2: Lock/Unlock Inspector pane, R: System Menu.

03/18/87 08:45:04AM RALPH          USER:     Keyboard              → Romeo: WEBB; SYMBOLS.LISP#1  0
```



```
TV:BORDERS-MIXIN's methods (all)

Methods defined for flavor TV:BORDERS-MIXIN.  * = special method combination type
        Type       Message                         Arglist
                   :COMPUTE-BORDER-MARGIN-AREA-MARGINS (TV::SPEC TV::LM TV::TM TV::RM TV::BM)
                   :COMPUTE-MARGINS                 (TV::LM TV::TM TV::RM TV::BM)
W:GRAPHICS-MIXIN

Flavor W:GRAPHICS-MIXIN's component flavors.

 W:TRANSFORM-MIXIN
 SYS:VANILLA-FLAVOR
W:WINDOW

Flavor W:WINDOW's component flavors.  ** = redundant included flavor

 W:GRAPHICS-MIXIN
   W:TRANSFORM-MIXIN
 TV:STREAM-MIXIN
 TV:BORDERS-MIXIN
 TV:LABEL-MIXIN
   TV::ESSENTIAL-LABEL-MIXIN
 TV:SELECT-MIXIN
   TV:DELAY-NOTIFICATION-MIXIN** (included flavor of TV:SELECT-MIXIN)
 TV:DELAY-NOTIFICATION-MIXIN
 TV:MINIMUM-WINDOW
   TV:ESSENTIAL-EXPOSE
   TV:ESSENTIAL-ACTIVATE
   TV:ESSENTIAL-SET-EDGES
   TV:ESSENTIAL-MOUSE
   TV:ESSENTIAL-WINDOW
     TV:SHEET
 SYS:VANILLA-FLAVOR

Help On Syntax All Flavors  Trace    Exit      TV:BORDERS-MIXIN's methods (all)
Doc            Delete       Refresh  Up        W:GRAPHICS-MIXIN
Down           Break        Mode     Config    W:WINDOW
Flavor/Method: my-window
Flavor/Method: ■

Flavor Inspector 4
L: Select data to inspect, M: Help on currently displayed data, M2: Lock/Unlock inspector pane, R: Menu of operations on flavor W:WINDOW

04/16/87 02:13:24PM WEBB           USER:     Keyboard              → Lima: WEBB; IMAGE.XLD#3  0
```

Inspector

The Inspector is a useful tool for examining complex data structures. It is a window-oriented program that allows you to view and modify the components of Lisp objects. The components displayed depend on the type of object. For example, the components of a list are its elements, and those of a symbol are its value binding, function definition, property list, package, and print name.

An Inspector window contains several different panes (as shown on the left). The interaction pane allows you to enter Lisp forms to be evaluated and then inspected. This pane also prompts you for input when you invoke certain menu operations.

The history pane, located above the interaction pane, maintains a list of all objects that you have inspected so far. Objects in the history pane are mouse-sensitive; you can select them for reinspection or deletion from the history.

The top three panes are inspection panes; you can view a different object in each of these. When you inspect an object, it appears in the large inspection pane below the two smaller inspection panes, and the previously inspected objects shift upward. The inspection panes show the printed representation and components of the objects being inspected. Certain objects in the inspection panes are also mouse-sensitive; you can select them for further inspection or modification.

You can adjust the Inspector window configuration to have one, two, or three inspection panes. In addition, you can lock down an inspector pane so that it stays resident while objects are shown in other panes.

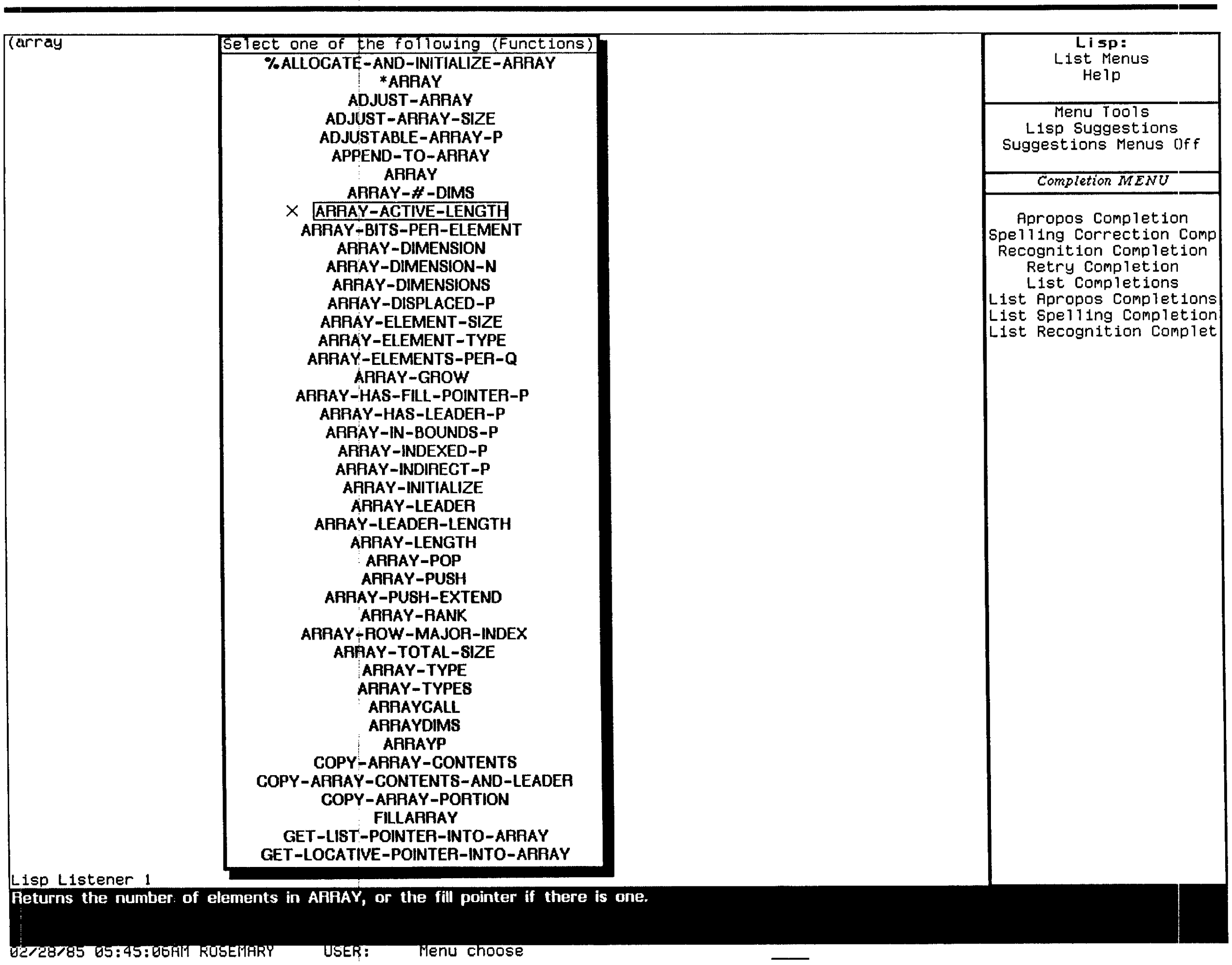
Flavor Inspector

The Flavor Inspector is a window-based utility that lets you observe and modify flavors, including their component flavors, instance variables, and methods. The Flavor Inspector works in much the same way as the Inspector.

A Flavor Inspector window contains several different panes. The command menu pane provides general Flavor Inspector commands.

The Lisp Listener pane allows you to enter flavor and method names to inspect, Lisp forms, and the names of the commands that are in the command menu panes.

The history pane and the inspection panes work like those of the Inspector.



```
(array
```

Returns the number of elements in ARRAY, or the fill pointer if there is one.

```
02/28/85 05:45:06AM ROSEMARY      USER:      Menu choose
```

Universal Command Loop

The Universal Command Loop (UCL) provides the basis for building a command interface for an interactive application. The UCL accepts various forms of user input (menu selection, mouse button, keystroke, or typed command name) and interprets the input as requests for command execution. The UCL also provides help facilities to accommodate both novice and experienced users.

The UCL is easily incorporated into your application; you simply supply the following:

- A set of command definitions, which includes all the information about each command, such as its name, the keystroke associated with the command, the code to be executed, short and long descriptions of the command, and a list of menus in which the command appears.
- One or more command table definitions, which organize a number of commands into a coherent set for name and key look-up by the UCL interpreter. With the command tables, you can set certain commands to be active. Several command tables can be defined for applications that need to alter the command context.
- A list of initial command tables that the interpreter uses to establish an initial command context. Your program can modify this list during execution to change the command context.

From the above information, the UCL generates a command interpreter with various help features that are based on the command definitions. The help features include Suggestions menus, display of keystrokes and descriptions, command completion, spelling correction, and string searches, and command completion. For example, command completion allows the user to specify part of a function call and then to request possible completions, as shown in the menu on the left.

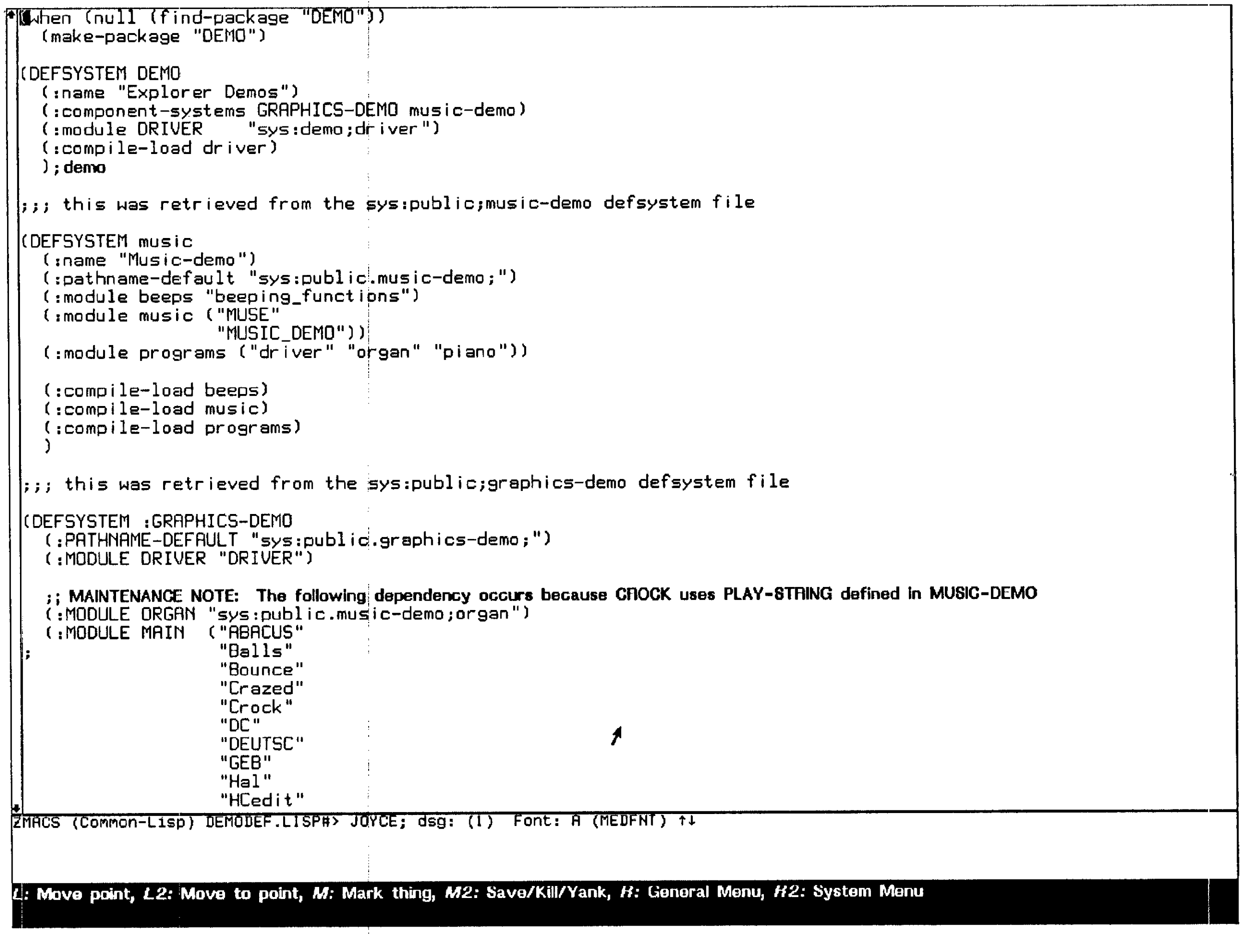
The UCL includes a number of features that allow you or a user to tailor the command interface to meet specific needs. A command editor is available for creating and modifying command definitions. User-defined macros allow you to enter a block of text with keystrokes or to create a command defined as a series of other commands.

Suggestions

The Suggestions system displays menus appropriate for different contexts of an application. These menus display lists of items that are mouse-sensitive, such as sub-menus, commands, Lisp symbols, information, and messages. You can even archive Lisp code in a menu for building commands.

Suggestions menus are displayed in a versatile window format that offers pop-up features, scrolling, and sub-menus. The Suggestions frame reapportions the window to display the Suggestions menu without interfering with the information displayed by the application. With the Suggestions facility, you can customize existing Suggestions menus to suit your needs or create new Suggestions menus for your applications.

The Suggestions system also provides a set of Lisp expressions menus that let you enter Lisp code from the Suggestions menus.



```
(when (null (find-package "DEMO"))
  (make-package "DEMO")

(DEFSYSTEM DEMO
  (:name "Explorer Demos")
  (:component-systems GRAPHICS-DEMO music-demo)
  (:module DRIVER    "sys:demo;driver")
  (:compile-load driver)
  );demo

;;; this was retrieved from the sys:public;music-demo defsystem file

(DEFSYSTEM music
  (:name "Music-demo")
  (:pathname-default "sys:public.music-demo;")
  (:module beeps "beeping_functions")
  (:module music ("MUSE"
                  "MUSIC_DEMO"))
  (:module programs ("driver" "organ" "piano"))

  (:compile-load beeps)
  (:compile-load music)
  (:compile-load programs)
  )

;;; this was retrieved from the sys:public;graphics-demo defsystem file

(DEFSYSTEM :GRAPHICS-DEMO
  (:PATHNAME-DEFAULT "sys:public.graphics-demo;")
  (:MODULE DRIVER "DRIVER")

  ;; MAINTENANCE NOTE:  The following dependency occurs because CROCK uses PLAY-STRING defined in MUSIC-DEMO
  (:MODULE ORGAN "sys:public.music-demo;organ")
  (:MODULE MAIN  ("ABACUS"
;                 "Balls"
                  "Bounce"
                  "Crazed"
                  "Crock"
                  "DC"
                  "DEUTSC"
                  "GEB"
                  "Hal"
                  "HCedit"
```

```
ZMACS (Common-Lisp) DEMODEF.LISP#> JOYCE; dsg: (1)  Font: A (MEDFNT) ↑↓
```

L: Move point, *L2:* Move to point, *M:* Mark thing, *M2:* Save/Kill/Yank, *R:* General Menu, *R2:* System Menu

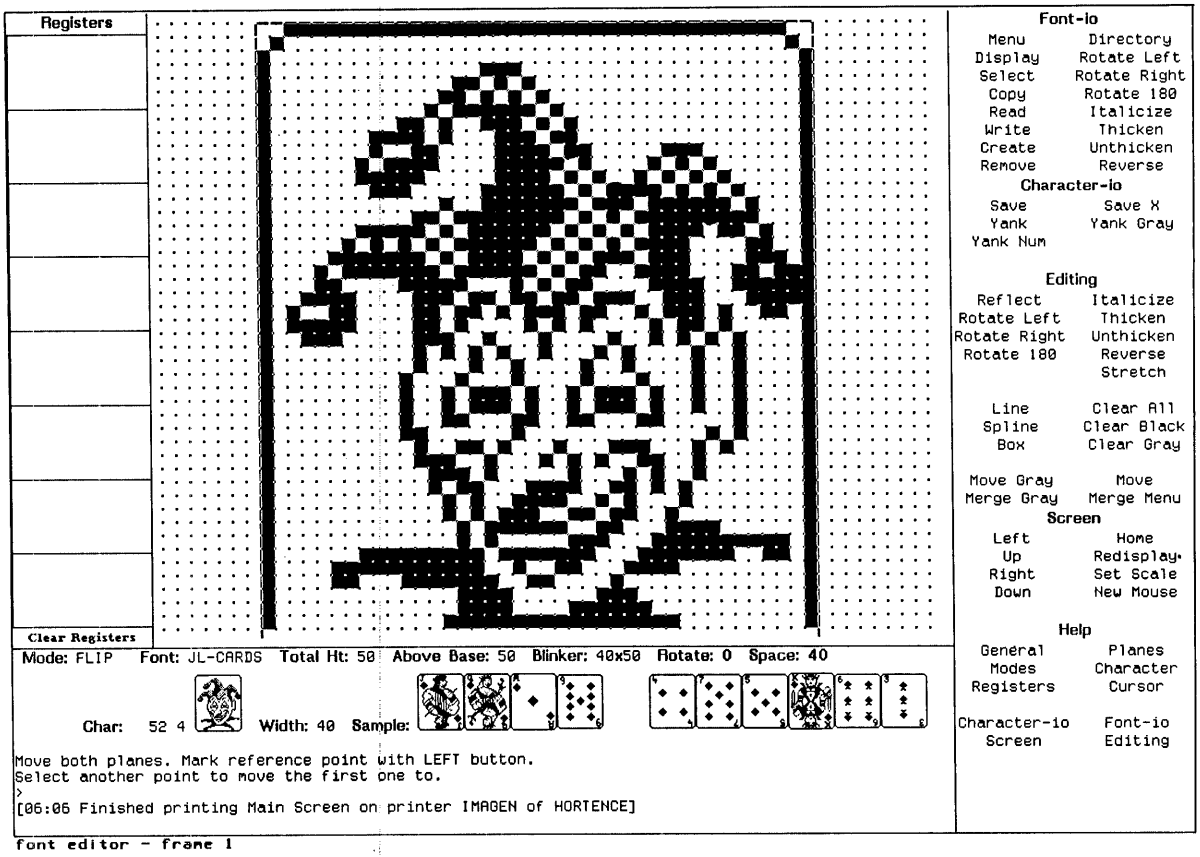
Tools for Maintaining Large Systems

The Explorer systems provide both a system definition facility and a patch facility to help you manage and maintain large programs. The system definition facility allows you to define the files comprising a program as a *system*. Once you define a system, the system definition facility handles the recompilation and loading of these files.

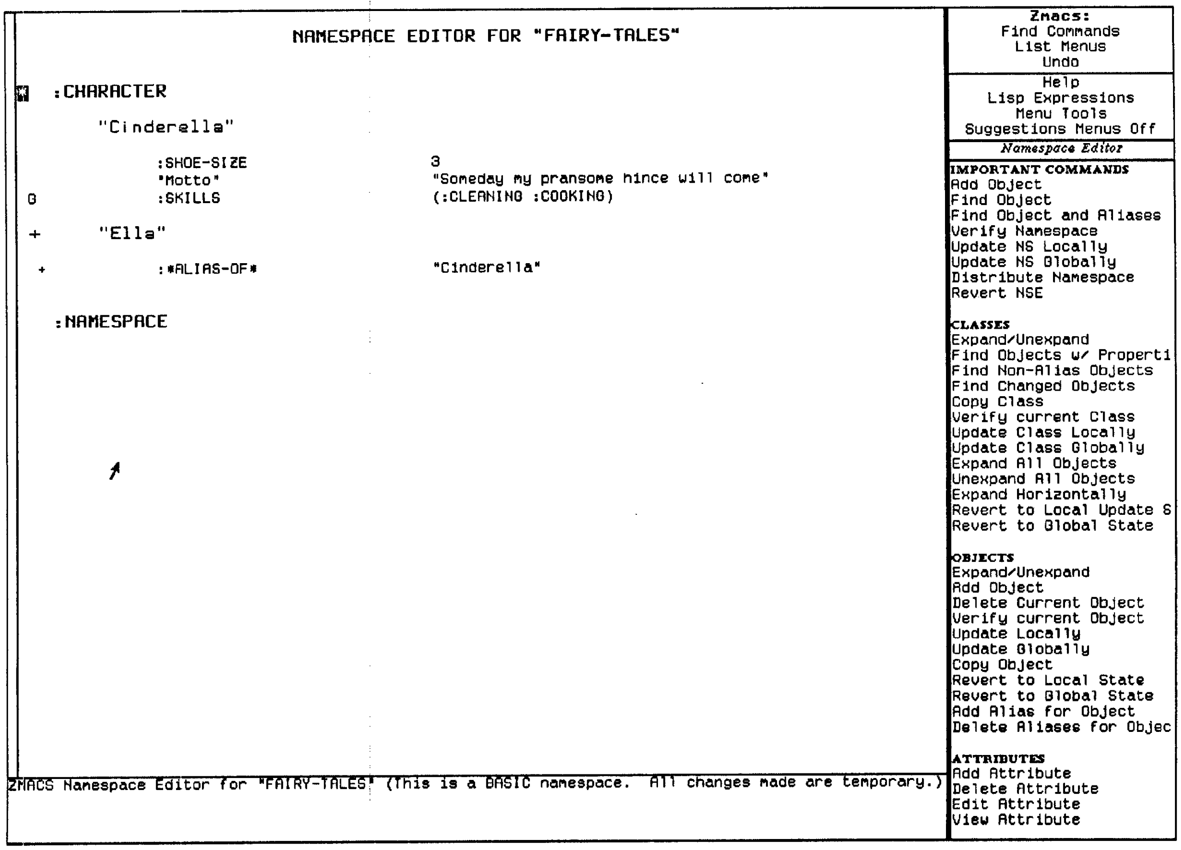
The system definition facility provides a number of options. The options allow you to specify properties of the system and transformations (operations such as compiling and loading) to be performed on files. You can also specify a condition for a transformation. For example, you can specify that a certain file must be loaded before another file is compiled. An example of such a specification is the DEFSYSTEM file for demo shown on the left.

The patch facility allows you to manage new releases of a system and issue small changes (patches) to update old versions. Once a system is present, you can ask for the latest patches to be loaded, ask which patches are already loaded, and add new patches. You can use Zmacs Editor commands to handle the creation, compilation, and disk storage of patch files. You can use Explorer system commands to load and review patch files. You can also create test patches, which will not be loaded in your released system.

You can maintain a history of multiple versions of a system. The facilities keep track of the different versions by labeling each one with a two-part number that consists of a *major* and *minor version number*. The minor version number is increased every time a new patch is made. The major version number is increased when the program is completely recompiled, and at that time, the minor version number is reset to zero.



```
Mode: FLIP   Font: JL-CARDS   Total Ht: 50   Above Base: 50   Blinker: 40x50   Rotate: 0   Space: 40

    Char:   52 4        Width: 40   Sample:

Move both planes. Mark reference point with LEFT button.
Select another point to move the first one to.
>
[06:06 Finished printing Main Screen on printer IMAGEN of HORTENCE]
font editor - frame 1
```



```
                    NAMESPACE EDITOR FOR "FAIRY-TALES"

  :CHARACTER

      "Cinderella"

              :SHOE-SIZE              3
              "Motto"                 "Someday my pransome hince will come"
 G            :SKILLS                 (:CLEANING :COOKING)

 +    "Ella"

 +            :*ALIAS-OF*             "Cinderella"

  :NAMESPACE

ZMACS Namespace Editor for "FAIRY-TALES" (This is a BASIC namespace.  All changes made are temporary.)
```

Font Editor

The Explorer systems supply many predefined fonts that you can use in the display or printing of text. You can use several different fonts in the same file. The Font Editor allows you to modify the existing fonts or create new ones.

The Font Editor is simple to use. It presents you with a grid of dots upon which, using the mouse, you draw magnified characters. You draw characters by setting or clearing these dots. A box shown on the grid is the *character box*. This box indicates the width and, along with the line below it, the height of a character.

The Font Editor has several menus from which you can select characters to be edited, font parameters, and commands. When you select a character, it is ready to be edited when the Font Editor places it in the character box. The font parameters determine the size of the character box. If you change these parameters, the Font Editor appropriately alters and redraws the character box. The commands provide many useful aids for drawing characters. For example, you can draw lines or curves, and stretch or rotate a character.

The Font Editor also displays a sample pane, which shows how the character you are editing will appear in normal size. You can define a string to be displayed in this pane so you can see how the character will look in context with others.

The Font Editor has a feature called the *gray plane*. Normally, the character being edited appears on the grid in black and is said to be in the *black plane*. A character in the gray plane appears in light gray *in back of* this character. Pixels that are common to both characters appear in dark gray. Editing performed on the black plane character does not affect the gray plane character. Thus, you can use the gray plane character as a guide to define a new character. Commands are available to move characters or pieces of characters between the two planes. Registers are provided to hold characters or pieces of characters while you are building or editing a font.

Namespace Editor

The Namespace Editor allows you to create a database for your network configuration in which you can specify items such as mailing lists, hosts (either logical or physical), printers, sites, and users on the network.

You can also use the Namespace Editor to create and modify other types of databases, such as databases for your personal use. These may be memory-resident temporary structures, or they may be disk-based namespaces. The Namespace Editor displays the database in an easy-to-read format and provides commands for a variety of operations. This editor allows you to change items in the database easily. In addition, a programmatic interface allows you to build and maintain the namespaces.

SYSTEM FACILITIES 5

The Explorer systems provide many facilities for run-time support and efficient implementation of Lisp programs. The tagged architecture provides efficient run-time checking and processing of data types. The architecture lets you create flexible programs quickly and efficiently. You need not use type declarations or restrict a program to handling only particular types of data. The Explorer systems perform the appropriate operations for the specific data in use, accessing the type of data along with its value.

The tagged architecture on the Explorer systems also provides automatic memory management. Built-in memory management and garbage collection frees you from handling the details of allocating, deallocating, and reclaiming memory. The large virtual memory enables you to handle very complex problems.

The operating system supports a variety of standard features, including the following:

- Microcode implementation of key functionality
- Metering system for performance monitoring
- Processes
- Stream input/output (I/O)
- File system
- Backup and restore utilities
- Utility for viewing system status

In addition, the operating system maintains a number of parameters that you can easily change to tailor the programming environment to suit your needs or preferences. You can specify these parameter values using the Profile utility. This utility builds or extends the initialization file, which is automatically loaded when you log in to the system. For example, you can set the default display fonts, initial Zmacs modes, and garbage collection parameters. An initialization file can contain any Lisp form. Thus, it can also contain functions you want defined at login.

Memory Management

The Explorer systems feature a demand-paged virtual memory with temporal garbage collection. The systems support up to 128 megabytes of virtual memory.

The garbage collector reclaims storage cells that are inaccessible and no longer required. A temporal (time-related) algorithm provides a nearly invisible garbage collector when you run with automatic garbage collection turned on. The algorithm is based on four levels of collectible storage--generations 0 through 3—with storage objects increasing in age from temporary to almost static. The younger storage objects are collected most frequently, with very rare collections among the objects that are almost static.

The systems support garbage collection in several modes. You can run with no garbage collection, with temporal garbage collection turned on, or in a batch collection mode. You use the batch mode to prepare a clean disk image of the system or to collect higher generations of garbage than would normally occur at a given time.

The Explorer systems allocate storage as objects are created. Storage is divided into *areas*; each area can contain related objects, independent of their types. You can specify which area(s) your program is to use for storage, or you can simply use the default area. By grouping related objects together in an area, the system performs better paging on your data.

A procedure is available to provide optimal placement of objects in memory, promoting fast access during use. You train the load band to place the program in low-level memory by going through the program's normal activities while using the Explorer garbage collection primitives. This training process moves frequently used software to the low end of memory. Once this version is saved to disk, you can load the version and do normal activities with very little paging because the optimized portion usually fits in physical memory.

Using specified areas also gives you additional control over garbage collection. You can declare certain areas to be static, which means that their contents seldom change and the garbage collector should not attempt to reclaim any space in them.

Microcode

Many of the Lisp language primitives and system functions are implemented in microcode to enhance system performance. These include the following:

- Device handling
- Virtual memory management
- Storage allocation
- Garbage collection
- Lisp data object support
- Lisp language kernel
- Screen drawing primitives
- Process switch primitives

Performance Monitoring

You can monitor your programs with the metering system to find out which parts of the program use the most time. When you run your program with metering, the system records each function call and return, along with the time it took place. Page faults and words consed are also recorded. After all the data is recorded, the metering system analyzes the records and reports how much time was spent executing each function.

By default, the metering system prints the amount of run time and real time spent executing each function that was called. Other options for analysis are also available. For example, you can print a list of callers to specified functions and how often each function was called.

A function histogram allows you to monitor function call activity for a set of processes. You can specify the processes to monitor and how often and how deeply to monitor, and you can view the result interactively or after the session completes. You can also save data for future review.

Timing macros are also available to obtain detailed performance information at any level of code. You specify data to be collected from choices of central processing unit (CPU) time, real time, paging time, page fault count, and consing count. You can also save data.

Processes

The Explorer systems provide facilities that allow you to create and control your own processes. You can execute several computations concurrently by placing each in a separate process. Each process has its own program counter, its own stack of function calls, and its own special-variable binding environment in which to execute.

All processes run in the same virtual address space and share the same set of Lisp objects. Unlike other systems that have special restrictive mechanisms for interprocess communication, the Explorer systems allow flexible communications between processes through shared Lisp objects. The Explorer systems also provide conventional mechanisms of atomic operations, critical sections, and interlocks.

A *scheduler* manages the set of active processes. Each active process is either currently running, trying to run, or waiting for some condition to become true. The scheduler repeatedly cycles through the active processes, determining whether each process is ready to run or is waiting to run.

When the scheduler finds a process ready to run, that process becomes the current process and executes. It remains the current process and continues to execute until either that process issues a wait or it uses up its time slice. At that time, control is returned to the scheduler. By default, a time slice occurs once each second.

Processes are implemented using the flavor system; thus, you can define your own flavors for different types of processes. You can also perform a number of operations on processes. For example, you can set the priority level of a process or specify the interval for its time slice.

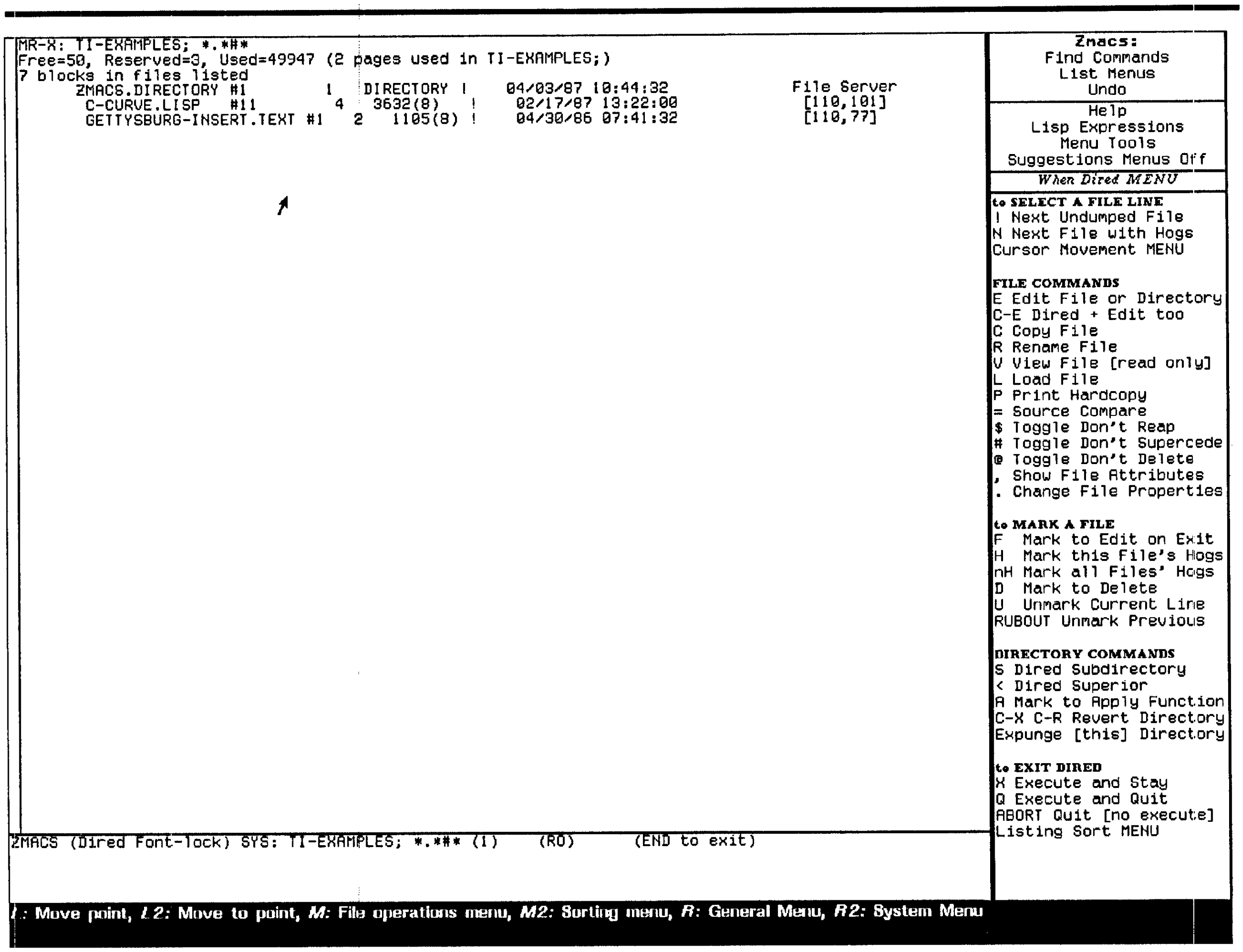
Stream I/O

The Explorer systems use data streams for performing input and output to peripheral devices. The stream concept allows you to write programs in a general way so that a program's input and output can be connected to any device.

A *stream* is a source and/or sink of characters or bytes. A set of generic operations is available with every stream; these include operations to output a character and input a character. Performing operations to a stream is the same for all streams. You can create a new type of stream simply by implementing the appropriate standard operations.

A stream can be input-only, output-only, or bidirectional. All input streams support all standard input operations, and all output streams support all standard output operations. All bidirectional streams support both input and output operations. In addition, special operations are provided for the following classes of streams:

- Windows, which take input from the keyboard and display output on the screen.
- File streams, which read or write the contents of a file.
- Networking streams, which are made from network connections. Data output to the stream goes out over the network; incoming data from the network is available as input from the stream.
- String streams, which read or write the contents of a string.
- Edit buffer streams, which read or write the contents of an editor buffer.
- The null stream, which can be passed to a program that asks for a stream as an argument. The null stream returns an immediate end-of-file if used for input and throws away any output.
- A cold-load stream, which performs I/O to the screen and the keyboard without using the window system.
- A serial stream, which allows access to the RS-232C port as a stream.



```
MR-X: TI-EXAMPLES; *.*#*
Free=50, Reserved=3, Used=49947 (2 pages used in TI-EXAMPLES;)
7 blocks in files listed
      ZMACS.DIRECTORY #1        1   DIRECTORY !    04/03/87 10:44:32          File Server
       C-CURVE.LISP   #11        4   3632(8)   !    02/17/87 13:22:00           [110,101]
       GETTYSBURG-INSERT.TEXT #1   2   1105(8) !    04/30/86 07:41:32           [110,77]

ZMACS (Dired Font-lock) SYS: TI-EXAMPLES; *.*#* (1)    (RO)      (END to exit)
```

L: Move point, *L2:* Move to point, *M:* File operations menu, *M2:* Sorting menu, *R:* General Menu, *R2:* System Menu

File System

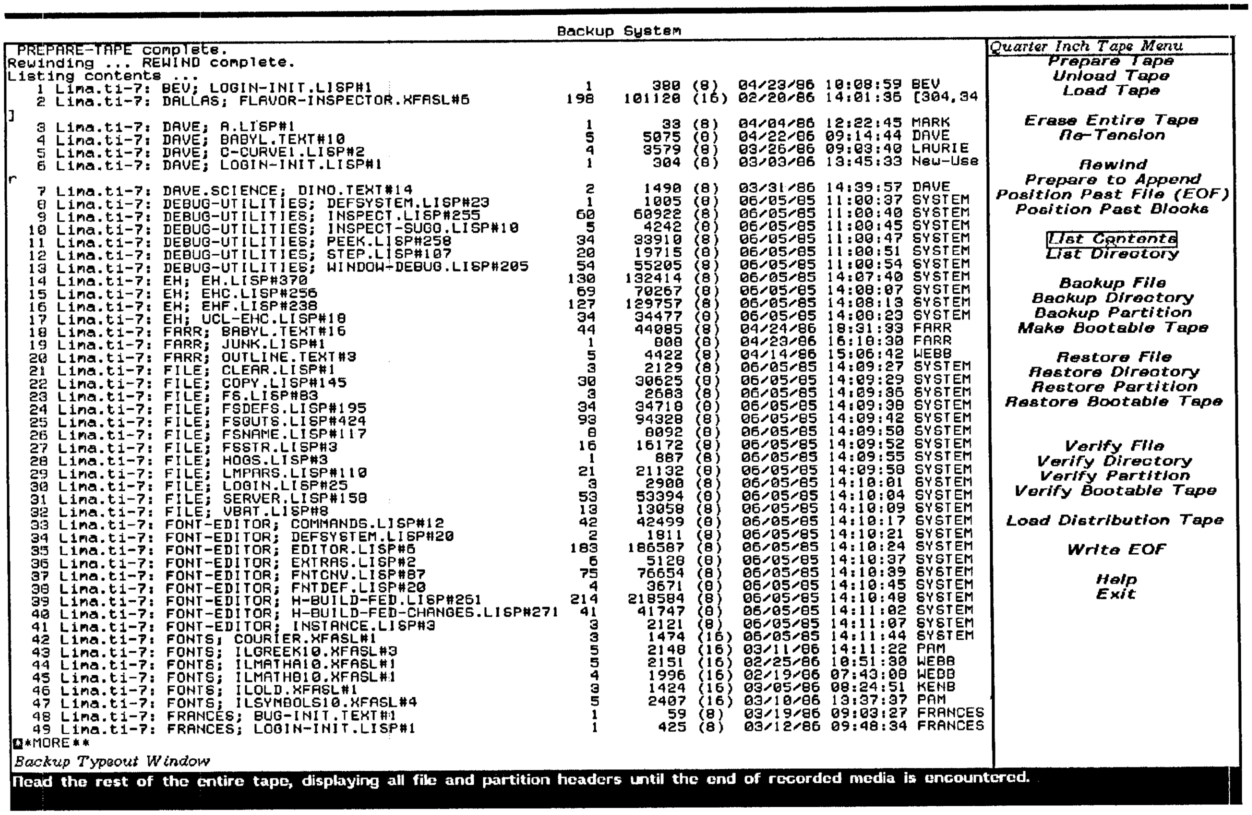
The Explorer file system provides a tree-oriented, multiple-level directory of files. Files carry attributes, allowing utilities and user programs to quickly check for the following:

- Type of file
- File owner
- Last update
- Other items

The Explorer systems handle version limiting (limiting the number of unique copies of a given file) on a directory basis.

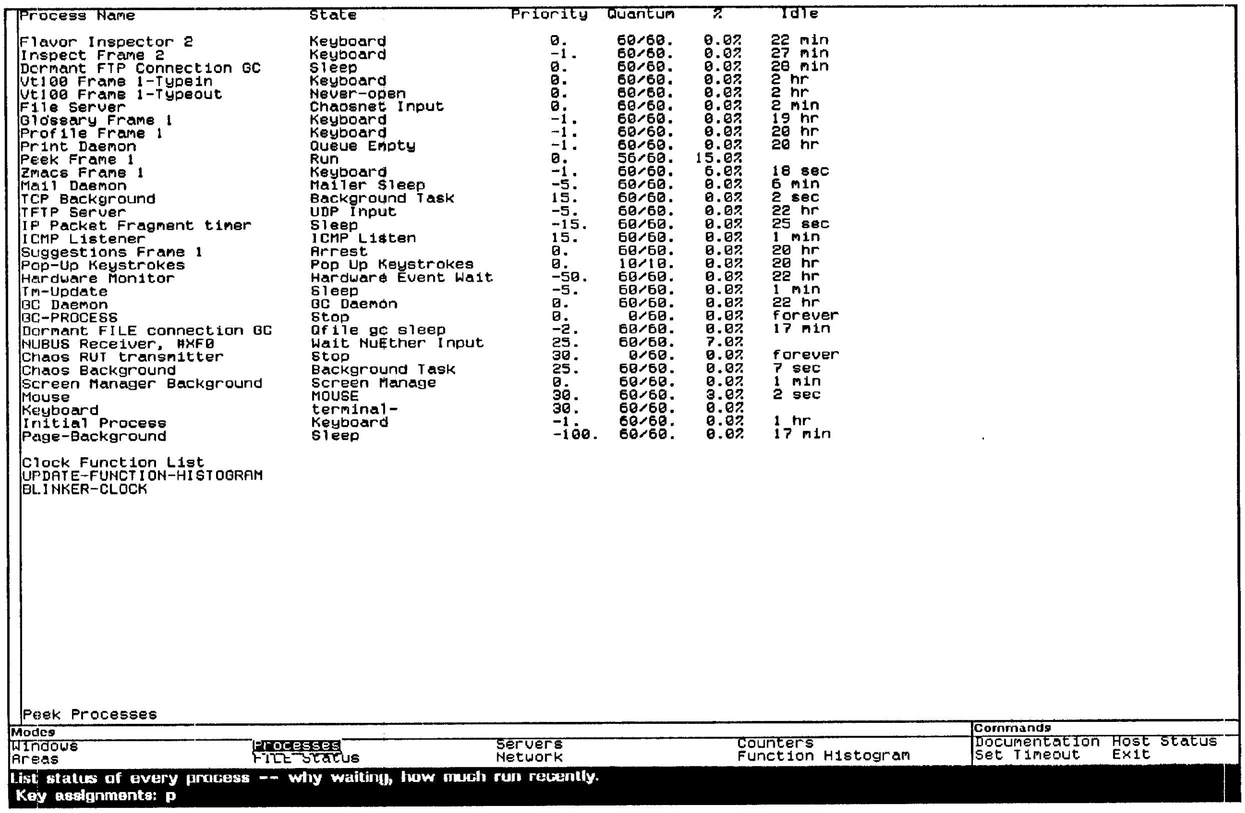
The file system supports a file partition that exists on one disk or spans several disks. You can expand the file partition if it exceeds the initially allocated space. A directory editor is available to perform the following functions on files in a directory:

- View
- Compare
- Delete
- Edit
- Print



```
 PREPARE-TAPE complete.
Rewinding ... REWIND complete.
Listing contents ...
   1 Lima.ti-7: BEV; LOGIN-INIT.LISP#1                          1      380 (8)  04/23/86 10:08:59 BEV
   2 Lima.ti-7: DALLAS; FLAVOR-INSPECTOR.XFASL#6              198   101120 (16) 02/20/86 14:01:35 [304,34
]
   3 Lima.ti-7: DAVE; A.LISP#1                                  1       33 (8)  04/04/86 12:22:45 MARK
   4 Lima.ti-7: DAVE; BABYL.TEXT#10                             5     5075 (8)  04/22/86 09:14:44 DAVE
   5 Lima.ti-7: DAVE; C-CURVE1.LISP#2                           4     3579 (8)  03/26/86 09:03:40 LAURIE
   6 Lima.ti-7: DAVE; LOGIN-INIT.LISP#1                         1      304 (8)  03/03/86 13:45:33 New-Use
r
   7 Lima.ti-7: DAVE.SCIENCE; DINO.TEXT#14                      2     1490 (8)  03/31/86 14:39:57 DAVE
   8 Lima.ti-7: DEBUG-UTILITIES; DEFSYSTEM.LISP#23              1     1005 (8)  06/05/85 11:00:37 SYSTEM
   9 Lima.ti-7: DEBUG-UTILITIES; INSPECT.LISP#255              60    60922 (8)  06/05/85 11:00:40 SYSTEM
  10 Lima.ti-7: DEBUG-UTILITIES; INSPECT-SUGG.LISP#10           5     4242 (8)  06/05/85 11:00:45 SYSTEM
  11 Lima.ti-7: DEBUG-UTILITIES; PEEK.LISP#258                 34    33910 (8)  06/05/85 11:00:47 SYSTEM
  12 Lima.ti-7: DEBUG-UTILITIES; STEP.LISP#107                 20    19715 (8)  06/05/85 11:00:51 SYSTEM
  13 Lima.ti-7: DEBUG-UTILITIES; WINDOW-DEBUG.LISP#205         54    55205 (8)  06/05/85 11:00:54 SYSTEM
  14 Lima.ti-7: EH; EH.LISP#370                               130   132414 (8)  06/05/85 14:07:40 SYSTEM
  15 Lima.ti-7: EH; EHC.LISP#256                               69    70267 (8)  06/05/85 14:08:07 SYSTEM
  16 Lima.ti-7: EH; EHF.LISP#238                              127   129757 (8)  06/05/85 14:08:13 SYSTEM
  17 Lima.ti-7: EH; UCL-EHC.LISP#18                            34    34477 (8)  06/05/85 14:08:23 SYSTEM
  18 Lima.ti-7: FARR; BABYL.TEXT#16                            44    44085 (8)  04/24/86 18:31:33 FARR
  19 Lima.ti-7: FARR; JUNK.LISP#1                               1      808 (8)  04/23/86 16:18:30 FARR
  20 Lima.ti-7: FARR; OUTLINE.TEXT#3                            5     4422 (8)  04/14/86 15:06:42 WEBB
  21 Lima.ti-7: FILE; CLEAR.LISP#1                              3     2129 (8)  06/05/85 14:09:27 SYSTEM
  22 Lima.ti-7: FILE; COPY.LISP#145                            30    30625 (8)  06/05/85 14:09:29 SYSTEM
  23 Lima.ti-7: FILE; FS.LISP#83                                3     2683 (8)  06/05/85 14:09:36 SYSTEM
  24 Lima.ti-7: FILE; FSDEFS.LISP#195                          34    34718 (8)  06/05/85 14:09:38 SYSTEM
  25 Lima.ti-7: FILE; FSGUTS.LISP#424                          93    94328 (8)  06/05/85 14:09:42 SYSTEM
  26 Lima.ti-7: FILE; FSNAME.LISP#117                           8     8092 (8)  06/05/85 14:09:50 SYSTEM
  27 Lima.ti-7: FILE; FSSTR.LISP#3                             16    16172 (8)  06/05/85 14:09:52 SYSTEM
  28 Lima.ti-7: FILE; HOGS.LISP#3                               1      887 (8)  06/05/85 14:09:55 SYSTEM
  29 Lima.ti-7: FILE; LMPARS.LISP#110                          21    21132 (8)  06/05/85 14:09:58 SYSTEM
  30 Lima.ti-7: FILE; LOGIN.LISP#25                             3     2900 (8)  06/05/85 14:10:01 SYSTEM
  31 Lima.ti-7: FILE; SERVER.LISP#158                          53    53394 (8)  06/05/85 14:10:04 SYSTEM
  32 Lima.ti-7: FILE; VBAT.LISP#8                              13    13058 (8)  06/05/85 14:10:09 SYSTEM
  33 Lima.ti-7: FONT-EDITOR; COMMANDS.LISP#12                  42    42499 (8)  06/05/85 14:10:17 SYSTEM
  34 Lima.ti-7: FONT-EDITOR; DEFSYSTEM.LISP#20                  2     1811 (8)  06/05/85 14:10:21 SYSTEM
  35 Lima.ti-7: FONT-EDITOR; EDITOR.LISP#6                    183   186587 (8)  06/05/85 14:10:24 SYSTEM
  36 Lima.ti-7: FONT-EDITOR; EXTRAS.LISP#2                      6     5128 (8)  06/05/85 14:10:37 SYSTEM
  37 Lima.ti-7: FONT-EDITOR; FNTCNV.LISP#87                    75    76654 (8)  06/05/85 14:10:39 SYSTEM
  38 Lima.ti-7: FONT-EDITOR; FNTDEF.LISP#20                     4     3671 (8)  06/05/85 14:10:45 SYSTEM
  39 Lima.ti-7: FONT-EDITOR; H-BUILD-FED.LISP#261             214   218584 (8)  06/05/85 14:10:48 SYSTEM
  40 Lima.ti-7: FONT-EDITOR; H-BUILD-FED-CHANGES.LISP#271  41    41747 (8)  06/05/85 14:11:02 SYSTEM
  41 Lima.ti-7: FONT-EDITOR; INSTANCE.LISP#3                    3     2121 (8)  06/05/85 14:11:07 SYSTEM
  42 Lima.ti-7: FONTS; COURIER.XFASL#1                          3     1474 (16) 06/05/85 14:11:44 SYSTEM
  43 Lima.ti-7: FONTS; ILGREEK10.XFASL#3                        5     2148 (16) 03/11/86 14:11:22 PAM
  44 Lima.ti-7: FONTS; ILMATHA10.XFASL#1                        5     2151 (16) 02/25/86 10:51:30 WEBB
  45 Lima.ti-7: FONTS; ILMATHB10.XFASL#1                        4     1996 (16) 02/19/86 07:43:08 WEBB
  46 Lima.ti-7: FONTS; ILOLD.XFASL#1                            3     1424 (16) 03/05/86 08:24:51 KENB
  47 Lima.ti-7: FONTS; ILSYMBOLS10.XFASL#4                      5     2407 (16) 03/10/86 13:37:37 PAM
  48 Lima.ti-7: FRANCES; BUG-INIT.TEXT#1                        1       59 (8)  03/19/86 09:03:27 FRANCES
  49 Lima.ti-7: FRANCES; LOGIN-INIT.LISP#1                      1      425 (8)  03/12/86 09:48:34 FRANCES
□*MORE**
```

Read the rest of the entire tape, displaying all file and partition headers until the end of recorded media is encountered.



```
Process Name                   State                 Priority  Quantum    %       Idle

Flavor Inspector 2             Keyboard                 0.      60/60.    0.0%    22 min
Inspect Frame 2                Keyboard                -1.      60/60.    0.0%    27 min
Dormant FTP Connection GC      Sleep                    0.      60/60.    0.0%    28 min
Vt100 Frame 1-Typein           Keyboard                 0.      60/60.    0.0%    2 hr
Vt100 Frame 1-Typeout          Never-open               0.      60/60.    0.0%    2 hr
File Server                    Chaosnet Input           0.      60/60.    0.0%    2 min
Glossary Frame 1               Keyboard                -1.      60/60.    0.0%    19 hr
Profile Frame 1                Keyboard                -1.      60/60.    0.0%    20 hr
Print Daemon                   Queue Empty             -1.      60/60.    0.0%    20 hr
Peek Frame 1                   Run                      0.      56/60.   15.0%
Zmacs Frame 1                  Keyboard                -1.      60/60.    6.0%    18 sec
Mail Daemon                    Mailer Sleep            -5.      60/60.    0.0%    6 min
TCP Background                 Background Task         15.      60/60.    0.0%    2 sec
TFTP Server                    UDP Input               -5.      60/60.    0.0%    22 hr
IP Packet Fragment timer       Sleep                  -15.      60/60.    0.0%    25 sec
ICMP Listener                  ICMP Listen             15.      60/60.    0.0%    1 min
Suggestions Frame 1            Arrest                   0.      60/60.    0.0%    20 hr
Pop-Up Keystrokes              Pop Up Keystrokes        0.      10/10.    0.0%    20 hr
Hardware Monitor               Hardware Event Wait    -50.      60/60.    0.0%    22 hr
Tm-Update                      Sleep                   -5.      60/60.    0.0%    1 min
GC Daemon                      GC Daemon                0.      60/60.    0.0%    22 hr
GC-PROCESS                     Stop                     0.       0/60.    0.0%    forever
Dormant FILE connection GC     Qfile gc sleep          -2.      60/60.    0.0%    17 min
NUBUS Receiver, #XF0           Wait NuEther Input      25.      60/60.    7.0%
Chaos RUT transmitter          Stop                    30.       0/60.    0.0%    forever
Chaos Background               Background Task         25.      60/60.    0.0%    7 sec
Screen Manager Background      Screen Manage            0.      60/60.    0.0%    1 min
Mouse                          MOUSE                   30.      60/60.    3.0%    2 sec
Keyboard                       terminal-               30.      60/60.    0.0%
Initial Process                Keyboard                -1.      60/60.    0.0%    1 hr
Page-Background                Sleep                 -100.      60/60.    0.0%    17 min

Clock Function List
UPDATE-FUNCTION-HISTOGRAM
BLINKER-CLOCK
```

List status of every process -- why waiting, how much run recently.
Key assignments: p

Backup and Restore

Backup facilities are available on the Explorer systems, allowing you to back up and restore data to tape. You can save files, directories, or disk partitions and then restore them to transport the data to another location or to restore the data in case of loss. (A partition is a logical segment on the disk, comprising a contiguous area of disk storage. For example, the file system is a single logical partition.)

An interactive interface to the tape system allows you to easily perform the following:

- Prepare the tape for backup
- Do the copy operations
- Verify the copy
- Restore any backed-up item

System Status

The Peek utility is a window-oriented program that displays the following different types of system activities and their current states:

- Processes
- Statistic counters
- Areas
- File system status
- Windows
- Network status
- Servers
- Function histogram

From the Peek menu, you can select the type of activity you want displayed. The displays are updated periodically, and you can specify the time interval between updates. Thus, you can observe the system's change over time as well as examine its current state.

The information that is displayed depends on the type of activity. For example, the area display shows all the currently defined areas, the size of each, and the percentage of available storage in each.

Items in the displays are mouse-sensitive. In some cases, selecting an item with the mouse displays additional information. In other cases, selecting an item brings up a menu of operations that can be performed on that item. For example, you can kill or reset an active process from the Peek window.

SOFTWARE OPTIONS 6

A set of software options for the Explorer systems enhances and complements the standard system tools to extend functionality of the software environment. These options provide high-level components to solve basic programming problems that occur in many applications.

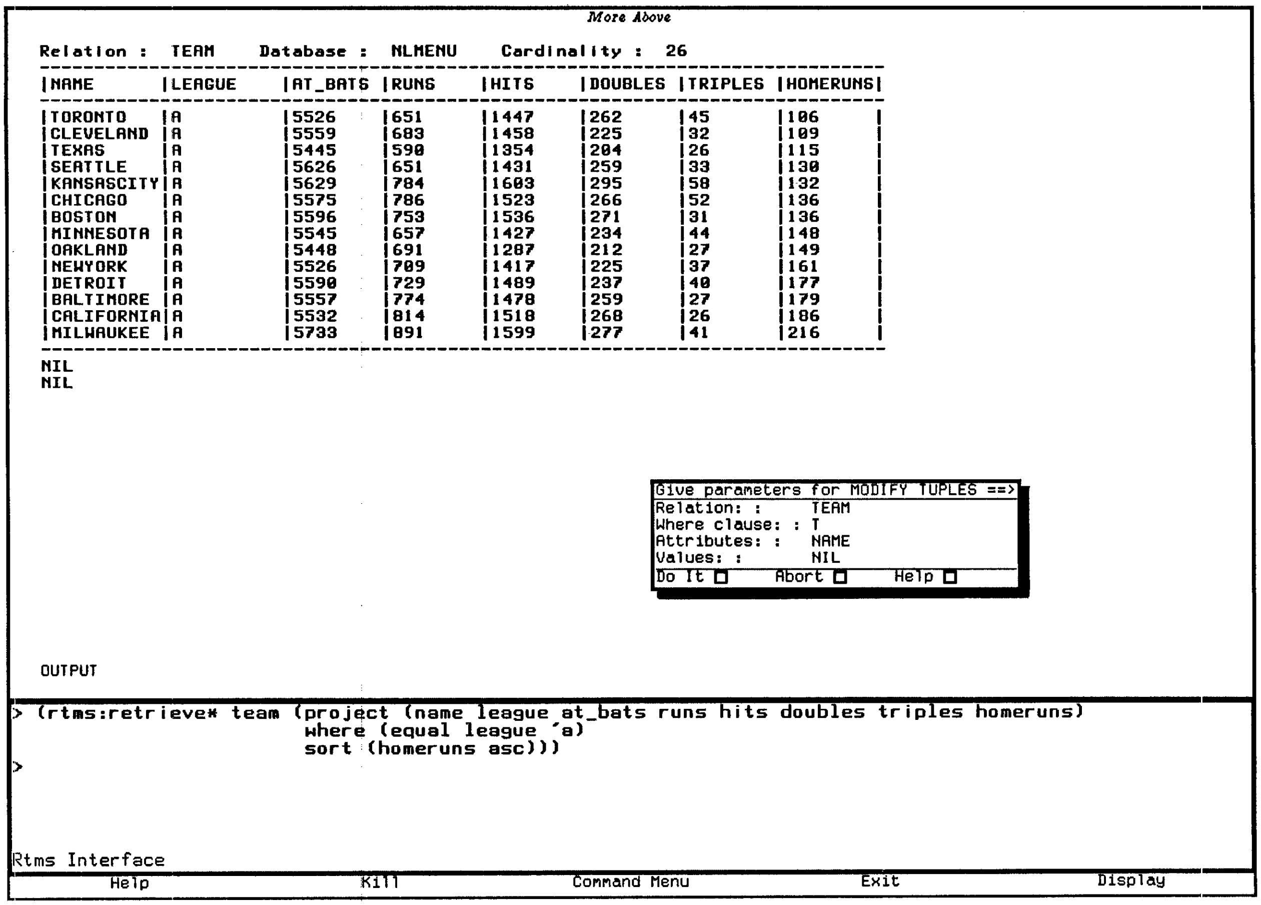
You can easily incorporate these components into your programs. The optional toolkits handle most of the programming details, allowing you to quickly build an application to meet your specific needs. They provide code that you can integrate into your application and build on. Using the toolkits also gives you consistency across applications.

Texas Instruments currently offers the following software options:

- Toolkits
  - Relational Table Management System (RTMS)
  - Natural Language Menu (NLMenu) system
  - Grasper network representation tool
  - Raster Technologies color graphics support
- TI Prolog
- Communications options (discussed in Section 7, Networking Facilities)
- Personal Consultant Plus for the Explorer
- Explorer LX multiprocessor support

Texas Instruments provides complete source code with the standard option package for all options but TI Prolog and Personal Consultant Plus.

Other products are also available for special market needs. See your customer representative for details.



More Above

```
Relation :  TEAM    Database :  NLMENU    Cardinality :  26
-------------------------------------------------------------------------------
|NAME       |LEAGUE     |AT_BATS |RUNS     |HITS     |DOUBLES |TRIPLES |HOMERUNS|
-------------------------------------------------------------------------------
|TORONTO    |A          |5526    |651      |1447     |262     |45      |106     |
|CLEVELAND  |A          |5559    |683      |1458     |225     |32      |109     |
|TEXAS      |A          |5445    |590      |1354     |204     |26      |115     |
|SEATTLE    |A          |5626    |651      |1431     |259     |33      |130     |
|KANSASCITY|A           |5629    |784      |1603     |295     |58      |132     |
|CHICAGO    |A          |5575    |786      |1523     |266     |52      |136     |
|BOSTON     |A          |5596    |753      |1536     |271     |31      |136     |
|MINNESOTA  |A          |5545    |657      |1427     |234     |44      |148     |
|OAKLAND    |A          |5448    |691      |1287     |212     |27      |149     |
|NEWYORK    |A          |5526    |709      |1417     |225     |37      |161     |
|DETROIT    |A          |5590    |729      |1489     |237     |40      |177     |
|BALTIMORE  |A          |5557    |774      |1478     |259     |27      |179     |
|CALIFORNIA|A           |5532    |814      |1518     |268     |26      |186     |
|MILWAUKEE  |A          |5733    |891      |1599     |277     |41      |216     |
-------------------------------------------------------------------------------
NIL
NIL
```

```
Give parameters for MODIFY TUPLES ==>
Relation: :       TEAM
Where clause: :   T
Attributes: :     NAME
Values: :         NIL
Do It □      Abort □      Help □
```

OUTPUT

```
> (rtms:retrieve* team (project (name league at_bats runs hits doubles triples homeruns)
                       where (equal league 'a)
                       sort (homeruns asc)))
>
```

Rtms Interface

Relational Table Management System

RTMS allows you to build, access, and manipulate relational tables within the Lisp environment. It provides the basic operations necessary for a database while allowing as much flexibility as possible. All manipulations of the database occur in virtual memory; however, you can save and load the database elements to and from disk. RTMS provides the following functions:

- Data definition functions to define or destroy databases, relations, views, and attributes. These also include functions to attach and detach relations to a database.

- Relation manipulation functions to retrieve, join, and perform set operations (union, difference, and intersection) on relations. The join function allows you to specify any type of join operation.

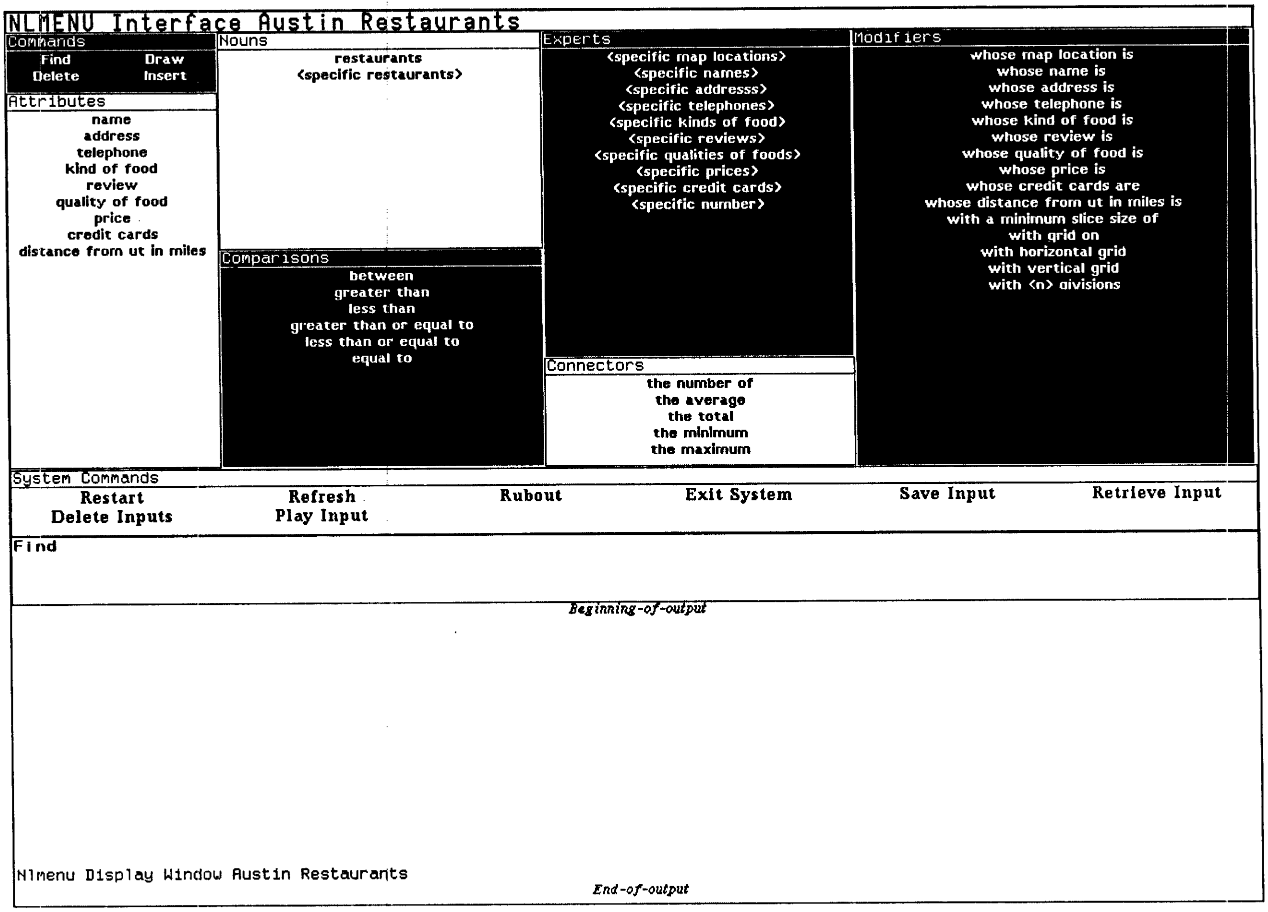
- Other functions to perform miscellaneous operations, such as saving and loading a database or relation; inserting, deleting, or modifying tables; and renaming a database, relation, or attribute.

You can define relations in any desired format (combination of implementation type and storage structure). RTMS includes accessor functions for relations implemented in any of nine system-defined formats. You can use other formats by defining your own accessor functions. RTMS supports several different domains for attribute values. You can also add new domains.

The relations in a database do not need to be stored in the same directory. The system automatically loads a relation when it is referenced. In addition, there is no expressed limit on the size or number of relations in a database, provided the database fits within virtual memory.

RTMS provides a set of global variables that control some of its features. You can easily change the values of these variables to select only the features you want at a particular time. For example, you can turn off validity checking, suppress messages, or have the system automatically save modified relations.

RTMS is integrated with the rest of the Lisp environment. You can invoke its functions from your application or from the Lisp Listener. An interactive, menu-based interface is provided to help you study or debug your database. The interface is window-oriented and consists of a scrollable output pane and a Lisp Listener pane. The output pane contains all output generated by RTMS during the session. Database objects in the output pane are mouse-sensitive, enabling you to select them and display information about them.


NLMENU Interface Austin Restaurants
Commands
Find
Draw
Delete
Insert
Attributes
name
address
telephone
kind of food
review
quality of food
price
credit cards
distance from ut in miles
Nouns
restaurants
<specific restaurants>
Comparisons
between
greater than
less than
greater than or equal to
less than or equal to
equal to
Experts
<specific map locations>
<specific names>
<specific addresss>
<specific telephones>
<specific kinds of food>
<specific reviews>
<specific qualities of foods>
<specific prices>
<specific credit cards>
<specific number>
Connectors
the number of
the average
the total
the minimum
the maximum
Modifiers
whose map location is
whose name is
whose address is
whose telephone is
whose kind of food is
whose review is
whose quality of food is
whose price is
whose credit cards are
whose distance from ut in miles is
with a minimum slice size of
with grid on
with horizontal grid
with vertical grid
with <n> divisions
System Commands
Restart
Delete Inputs
Refresh
Play Input
Rubout
Exit System
Save Input
Retrieve Input
Find
Beginning-of-output
Nlmenu Display Window Austin Restaurants
End-of-output

Natural Language Menu System

The Natural Language Menu (NLMenu) system helps you build powerful but friendly user interfaces to your applications. These interfaces allow users with little or no programming background or experience to construct complex input sentences, such as queries to a database.

An NLMenu interface presents users with a set of windows and menus that contain words or phrases in English or some other natural language. The user constructs sentences by using the mouse to select the appropriate words or phrases. Software included in the toolkit controls each step of the selection and sentence construction by allowing the user to make selections in active windows only. With each selection, the software examines the context of the sentence created so far and decides which windows to activate next and which items to put in the windows.

This technique ensures that the user constructs only valid sentences and eliminates the need for error checks. Once the sentences are constructed, the software translates user input into the target language and passes it to the system or application as required.

To tailor the interface to your application, you must provide the following data structures:

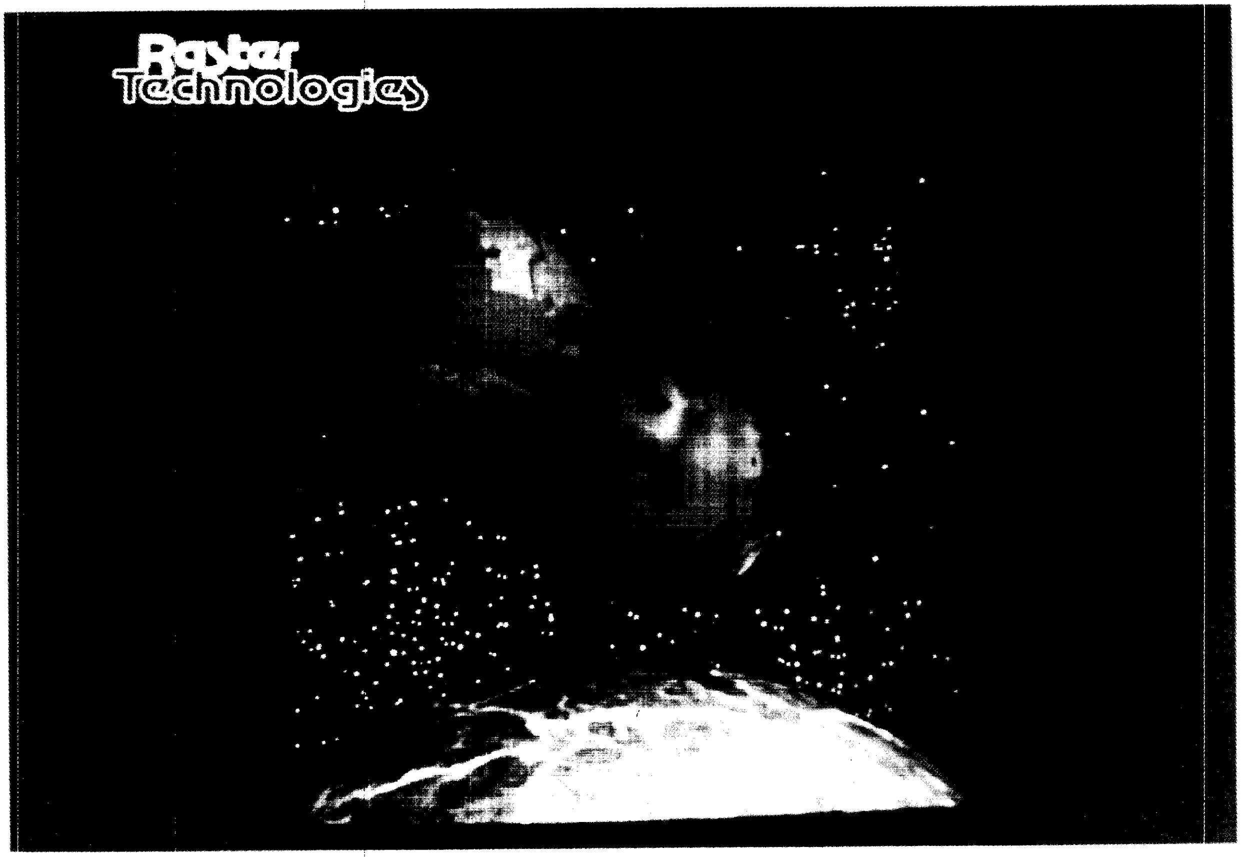
- A lexicon that specifies the words and phrases that can be used to construct sentences as well as their meaning within the sentence

- A grammar that indicates how the words and phrases can combine to form a sentence

- A screen description that indicates how menus and items are configured on the screen

The NLMenu system contains tools to help you write and debug these data structures.

The NLMenu option includes a starter kit, providing sample grammars and lexicons. In addition, a database interface generator provides a set of tools to create NLMenu interfaces to RTMS databases.



Raster
Technologies

Grasper

Grasper is a general network representation tool designed to efficiently encode and access a gamut of real-world knowledge that ranges from the interpretation of an English sentence to the modeling of a modern chemical factory. Implemented in Flavors, Grasper provides a means for you to easily represent your perception of a particular problem or situation, modify that representation, and then access and use the knowledge.

Grasper is designed to represent knowledge as a network of conceptual objects—objects that, when represented through symbolic programming, can represent pieces of a real-world application. Modeling these systems as networks of nodes and interconnecting edges makes it easy to visualize and organize a knowledge base; allowing you to work with concepts, ideas and relationships that cannot be represented in normal data processing terms.

Grasper software consists of a set of functions, flavors, and methods to create, delete, bind, evaluate, and show the existence of elements in a network. The network elements include the following:

- Nodes to represent specific objects
- Edges to represent relationships between objects
- Spaces to represent a collection of objects

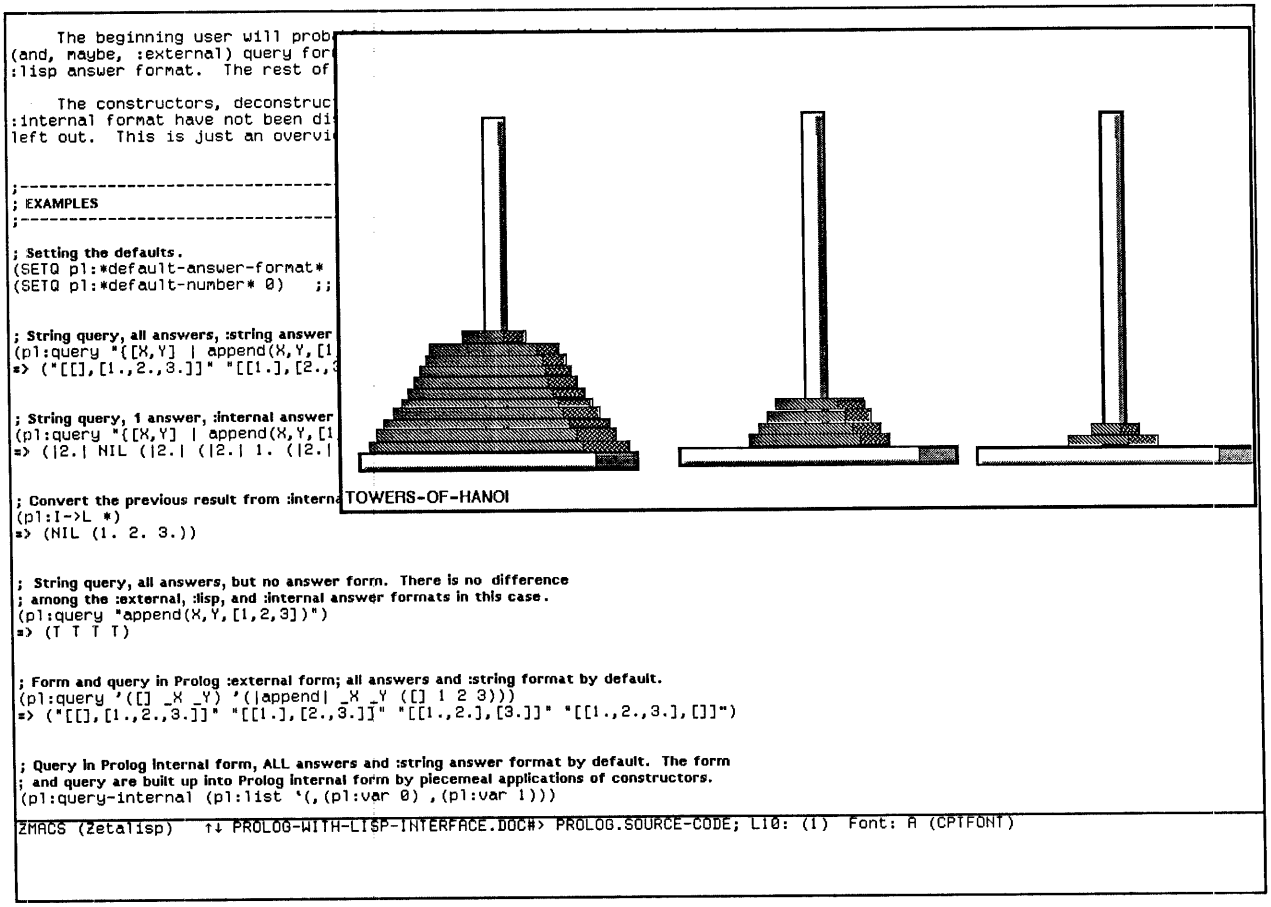
You can use Grasper primitives to describe your data, or you can combine these primitives into your own data description primitives.

Raster Color Graphics Support

The Raster Technologies system adds high-resolution color graphics capabilities to the Explorer systems. The Raster Technologies system consists of an intelligent frame buffer and color monitor driven by multiple microprocessors programmed to perform sophisticated graphics functions.

This system enhances the versatility of the Explorer systems in many areas, including C^3 (command, control, and communications), design, engineering, medical, simulation, and scientific applications.

The software supplied with the toolkit provides a Lisp library of about 200 Lisp functions that give you access to the Raster Technologies color graphics support routines. It includes a starter kit with a set of demonstration programs that illustrate the capabilities of the Explorer systems/Raster Technologies combination.



```
    The beginning user will prob
(and, maybe, :external) query for
:lisp answer format.  The rest of

    The constructors, deconstruc
:internal format have not been di
left out.  This is just an overvi


;----------------------------------
; EXAMPLES
;----------------------------------


; Setting the defaults.
(SETQ pl:*default-answer-format*
(SETQ pl:*default-number* 0)   ;;


; String query, all answers, :string answer
(pl:query "{[X,Y] | append(X,Y,[1
=> ("[[],[1.,2.,3.]]" "[[1.],[2.,3


; String query, 1 answer, :internal answer
(pl:query "{[X,Y] | append(X,Y,[1
=> (|2.| NIL (|2.| (|2.| 1. (|2.|


; Convert the previous result from :interna
(pl:I->L *)
=> (NIL (1. 2. 3.))


;  String query, all answers, but no answer form.  There is no  difference
; among the :external, :lisp, and :internal answer formats in this case.
(pl:query "append(X,Y,[1,2,3])")
=> (T T T T)


; Form and query in Prolog :external form; all answers and :string format by default.
(pl:query '([] _X _Y) '(|append| _X _Y ([] 1 2 3)))
=> ("[[],[1.,2.,3.]]" "[[1.],[2.,3.]]" "[[1.,2.],[3.]]" "[[1.,2.,3.],[]]")


; Query in Prolog internal form, ALL answers and :string answer format by default.  The form
; and query are built up into Prolog internal form by piecemeal applications of constructors.
(pl:query-internal (pl:list '(,(pl:var 0) ,(pl:var 1)))
ZMACS (Zetalisp)   ↑↓ PROLOG-WITH-LISP-INTERFACE.DOC#> PROLOG.SOURCE-CODE; L10: (1)  Font: A (CPTFONT)
```

TI Prolog

Prolog is a clean, simple, and direct programming tool for programming in logic. TI Prolog is a high-performance implementation of Prolog that is fully integrated into the Explorer systems environment.

TI Prolog is a superset of DECsystem-10™ Prolog, the definitive language standard for Prolog. Unlike many versions of Prolog, TI Prolog features an interactive compiler, which automatically compiles all clauses you enter without losing the flexible and incremental nature of its debugging tools. For interactive access to TI Prolog you can use the powerful Prolog Listener, which is analogous to top-level interpreters provided by some Prologs.

The Prolog Listener provides:

- Standard Explorer systems input editor facilities for ease of type-in
- Pathname completion
- Completion of system and user predicates
- Online help for all system predicates, with support to conveniently define help for your predicates
- Menu interface for predicates routinely used in program development. All debug information is dynamically displayed
- The same Universal Command Loop (UCL) interface found in other Explorer subsystems
- The compiler, which is automatically invoked on all clauses you enter

TI Prolog extends the Zmacs Editor to provide significant support for Prolog code. As for Lisp code, Zmacs supports incremental (re)compilation of Prolog clauses in arbitrary buffers and automatic lookup of user predicate definitions in defining files.

Debugging consists of the powerful DECsystem-10 style of tracing facilities, including spypoints and definition lookup (through Zmacs). Error-handling is done by the standard Explorer Lisp error handling systems.

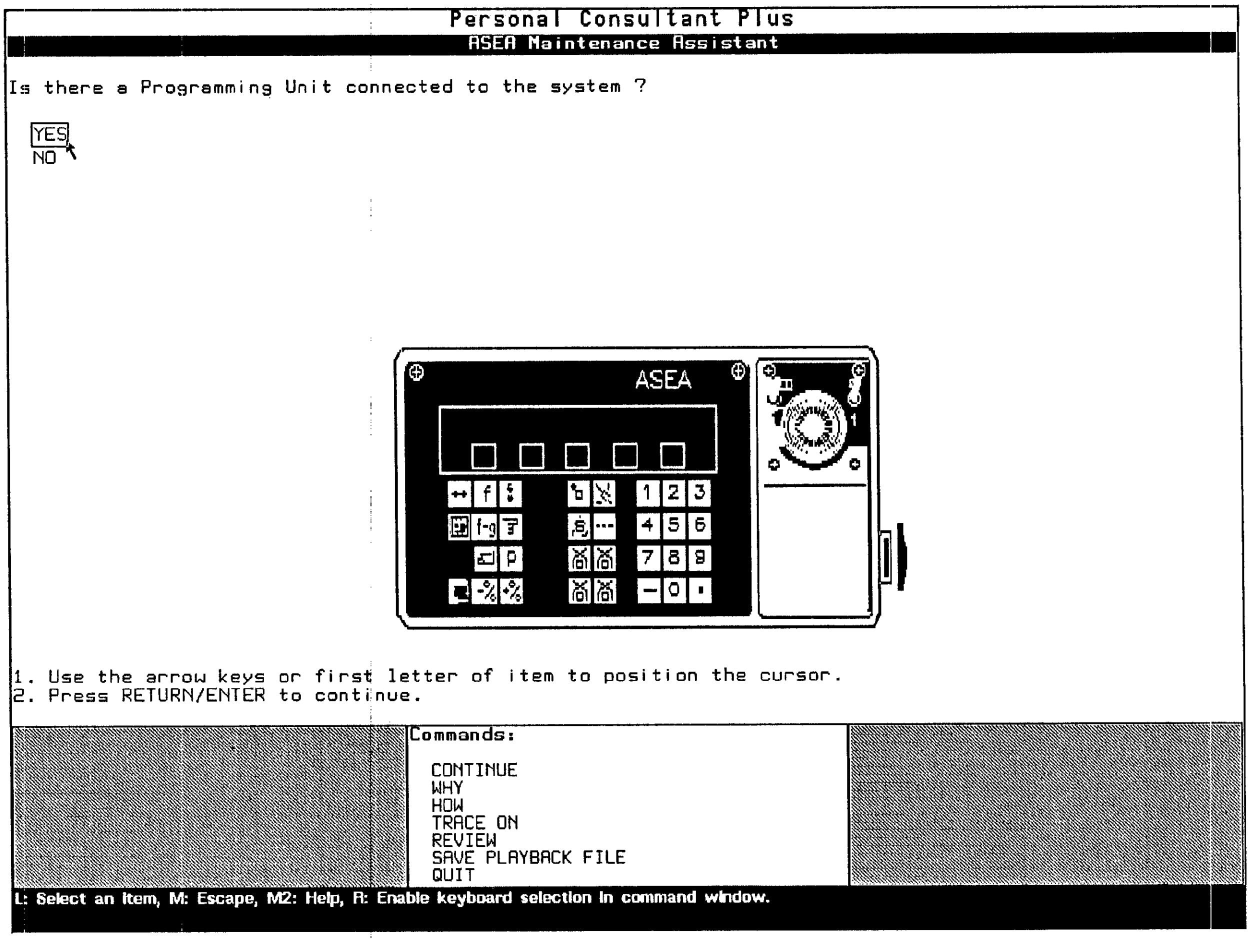
To provide complete integration of Lisp and Prolog, you can call Lisp from Prolog and Prolog from Lisp. Because TI Prolog shares data types with the Lisp system, the Lisp/Prolog interface is simplified. Because Lisp is easily accessed, TI Prolog offers complete access from Prolog to the Explorer window system and many other facilities.

The Common Lisp package system is available for Prolog code, providing modularization of user code into different namespaces, avoiding potential name conflicts in large systems.

TI Prolog comes with a number of example programs that illustrate its capabilities. For example, REST (rule-based expert system tool) is an expert system shell written entirely in Prolog.

To manage large Prolog programs, you can organize, compile, and load modules using the same mechanism you use to manage Lisp modules.

TI Prolog consists of microcode, Lisp, and Prolog support. Hardware modifications are not necessary.



Personal Consultant Plus

ASEA Maintenance Assistant

Is there a Programming Unit connected to the system ?

[✓ YES]
NO

1. Use the arrow keys or first letter of item to position the cursor.
2. Press RETURN/ENTER to continue.

Commands:

CONTINUE
WHY
HOW
TRACE ON
REVIEW
SAVE PLAYBACK FILE
QUIT

L: Select an Item, M: Escape, M2: Help, R: Enable keyboard selection in command window.

Personal Consultant Plus

Personal Consultant Plus (PC Plus) on the Explorer is a set of programming tools that facilitate the creation of expert system applications. An *expert system* is an advanced computer program that solves complex problems using artificial intelligence techniques. These techniques apply knowledge relevant to a problem domain to find a solution in a particular situation.

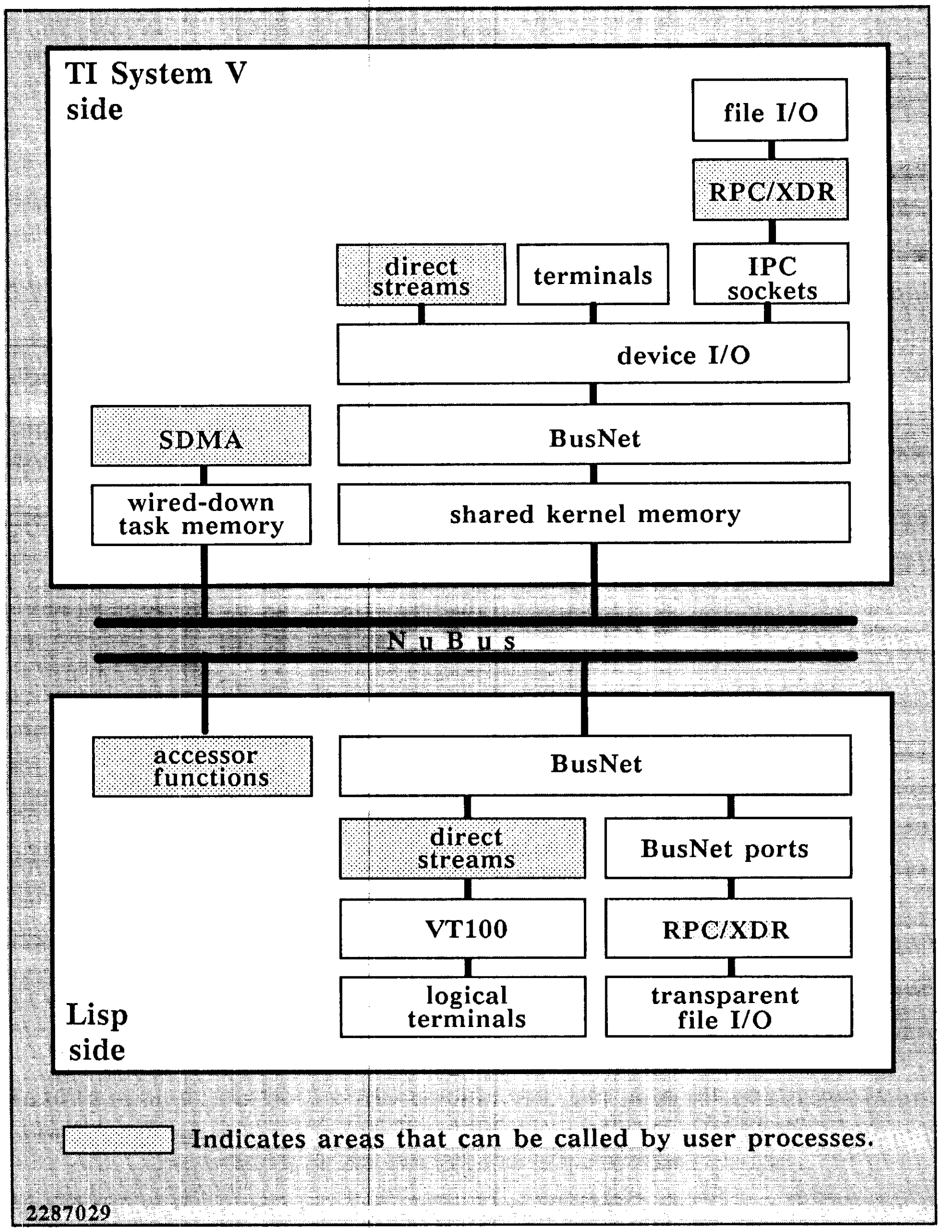
An expert system created with PC Plus has the following properties:

- When alternate paths are provided by the developer, it can make recommendations even when data is uncertain or missing and when the decision process is imprecise.
- It can explain in natural language sentences why data is needed or how conclusions were reached.
- It allows incorporation of qualitative decision factors as well as quantitative factors.
- It interacts with the user through a natural question and answer dialogue.

PC Plus improves the developer's productivity and range of problem solutions by offering the following advantages:

- A self-contained, highly-interactive generate/test environment including comprehensive rule entry language, integrated window-oriented editor, and extensive debugging and validation aids.
- Certainty factor input and calculation to handle uncertain or incomplete user input and/or imprecise knowledge.
- Rule prioritization capabilities to control the order in which rules are tried in the inference process.
- User explanation facilities to determine *why* a particular question is being asked and *how* information was determined, or to *review* previous responses for possible changes.
- Ability to create compiled, user-defined, Lisp functions to solve difficult problems or extensively customize an application.
- Frame descriptors with inheritance to organize a large, complex problem into a series of related subproblems for added simplification, modularity, and control.
- Meta rules to supply information on how to best apply the other forms of knowledge—particularly rules.
- External program interfaces to enable developers to couple their applications with other specialized programs such as communications packages.

PC Plus on the Explorer is a high-functionality member of the Personal Consultant Series. Programs on the Explorer are knowledge-base compatible with PC Plus on personal computers. This allows users to develop complex applications in the rich environment of the Explorer and deliver the application on personal computers. The compatibility also permits users to begin development on smaller systems and grow into the Explorer as the application demands.


TI System V side
file I/O
RPC/XDR
direct streams
terminals
IPC sockets
device I/O
SDMA
BusNet
wired-down task memory
shared kernel memory
NuBus
accessor functions
BusNet
direct streams
BusNet ports
VT100
RPC/XDR
logical terminals
transparent file I/O
Lisp side
Indicates areas that can be called by user processes.
2287029

Explorer LX

The Explorer LX is designed for applications that integrate symbolic processing programs written in Lisp with computation-intensive programs written in such conventional languages as C, FORTRAN, COBOL, or Pascal.

The Explorer LX can handle a variety of demanding applications characterized by a high degree of complexity, the need for expert system support, high-speed I/O throughput, and massive number-crunching in scientific, engineering, industrial, and defense settings. Because of its dual environment, the Explorer LX is an excellent computer system for process control, simulation and modeling, and other applications that benefit from coupling a high-performance Lisp processor and a high-performance conventional processor. In addition, TI System V can drive up to sixteen additional conventional terminals or data I/O lines using a communications carrier board (CCB) with plug-in cards that feature various interfaces.

Shared physical memory (often called shared direct memory access, or SDMA) is a unique feature of the Explorer LX. A TI System V task can lock address space so the operating system will always have a known address corresponding to a physical address. The TI System V program then transmits the address location to Lisp, which can then execute NuBus reads and writes to that address, allowing Lisp applications to monitor and modify the execution of TI System V applications.

For example, if a TI System V program is executing a simulation program, a Lisp-based expert system can monitor the process to determine if the simulation is proceeding within acceptable limits. If not, the expert system can modify parameters in the TI System V process to effect a change.

Explorer LX also provides shared file system facilities to support the shared memory capabilities. Lisp applications have transparent access to the TI System V file system and the Lisp file system. File transfer utilities move data between the Lisp file system and the shared TI System V file system.

In addition, Remote Procedure Call (RPC) and External Data Representation (XDR) mechanisms, based on Sun Microsystem's public domain implementations, are available on both TI System V and Lisp, letting you add new services to either processing environment. Programs running under Lisp can use RPC to call the 68020-based processor to run required routines such as mathematical operations in conventional languages and then return the data to the Lisp processor. Applications running on the 68020-based processor can also send queries written as Lisp expressions to the Lisp side. You can use this capability, for example, when an application program requires that an expert system make decisions for the application.

Different internal data representations in the Lisp and TI System V applications are automatically translated to the proper format by the XDR facility.

The system user interface includes multiple window capabilities, with multiple simultaneous Lisp and TI System V windows. TI System V terminal I/O can be directed to a window managed by Lisp. Mapping is transparent to the TI System V application program. This capability would, for example, allow a user to run a TI System V application in one window, a debugger in another, and a Lisp application in a third.

NETWORKING FACILITIES 7

The Explorer systems support local area networks (LANs) using the industry-standard Ethernet interface. A LAN combines the advantages of a single-user workstation with the ability to share expensive resources and communicate with other Lisp machines and a variety of other computers using standard protocols.

With the Explorer LAN, your programs can communicate with other programs through the use of data streams. A special open call allows any two programs on the network to establish a stream between them. With the exception of this special open call, data streams are manipulated by the standard Explorer I/O calls. The data streams can be unidirectional or bidirectional.

The Explorer software incorporates a Generic Network Interface. The details of the network protocol used to move the stream of information between machines are hidden from the application. The choice of protocol to be used (when more than one is available) is usually deferred until runtime.

Networking services provided by the Explorer LAN include the following:

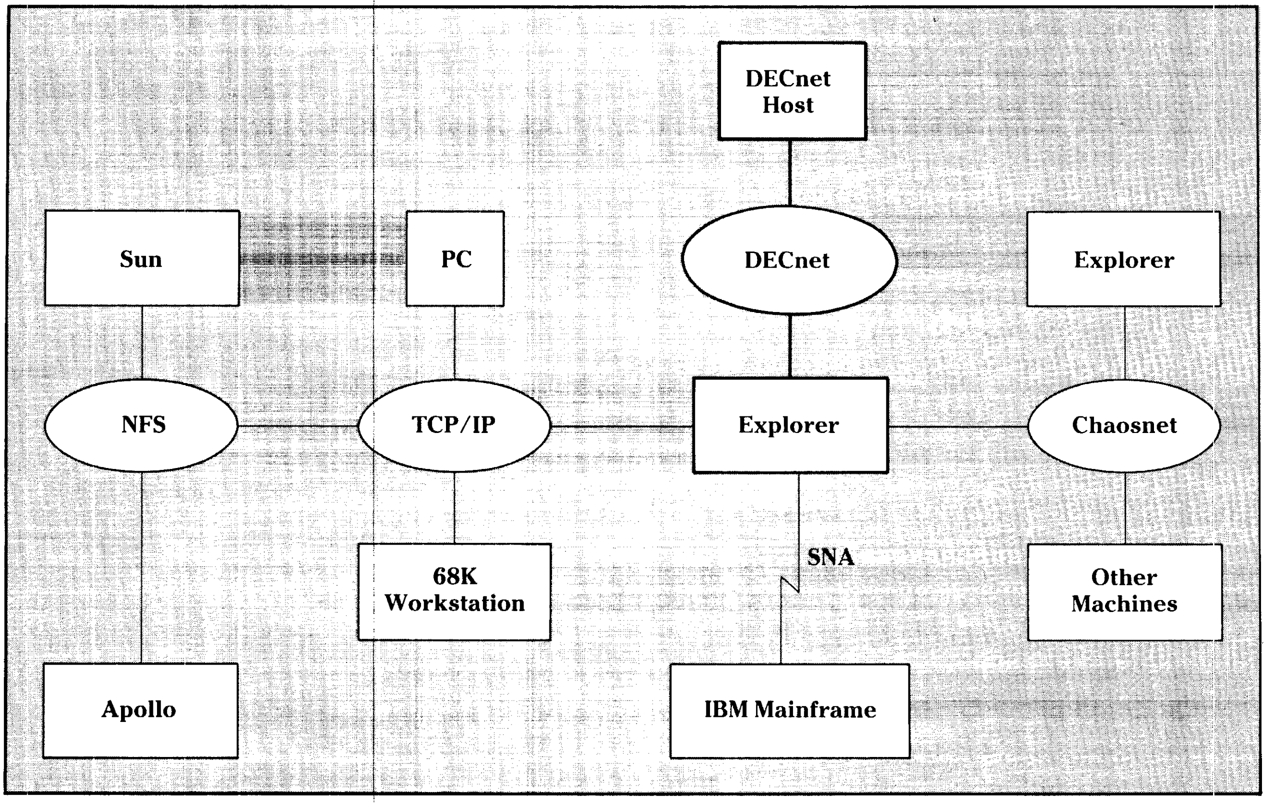
- Transparent file I/O — You can access all files, directories, and devices known to the network from any Explorer system as if that resource were located on the local system.

- Remote login — You can log in to hosts on the network by using your Explorer system as a virtual terminal to the remote host. The Explorer systems emulate both a standard ANSI terminal and most of the extended features of a VT100™ terminal.

- Task-to-task communication — You can write application programs that communicate directly among themselves over the network.

- Electronic mail — You can send and receive mail across the network and manage your mail files with a powerful user interface and message filing facility.

- Network name management — The Explorer software includes a unique system that provides a distributed, replicated, and highly robust database. You can use this system to manage network global information, such as machine names and addresses.


DECnet Host
Sun
PC
DECnet
Explorer
NFS
TCP/IP
Explorer
Chaosnet
SNA
68K Workstation
Other Machines
Apollo
IBM Mainframe

Communications

The NuBus Ethernet controller provides an interface between the Explorer systems and the LAN. The Ethernet controller is a single board that plugs into the backplane of the system enclosure. The Ethernet controller connects to a standard Ethernet transceiver and cable. Both thick and thin Ethernet cables are available.

The Explorer systems support several communications protocols:

- The Chaosnet protocol is bundled with the system software and uses the NuBus Ethernet controller. Chaosnet is used for communications typically between Lisp machines.

- The TCP/IP protocol is an add-on software package that provides communication to non-Explorer systems. TCP/IP is the standard communications protocol adopted by the Department of Defense as well as several important wide-area networks. It has been implemented for almost every machine that is currently widely available.

- NFS is an add-on software package that is used in conjunction with the TCP/IP protocol. NFS has become a standard among the UNIX community for transparent file access. The Explorer implementation extends this access into the Explorer file system.

- Explorer Interface to DECnet is an add-on-software package that provides access to systems produced by Digital Equipment Corporation, such as the VAX™ product line. This package allows transparent file access, inter-task communication, and remote login via protocols supported by Digital Equipment Corporation on their hardware.

- SNA II is an add-on software package that emulates IBM 3270-type equipment, providing access to systems that support IBM SNA protocols. A terminal interface is provided for remote login, and a program station control interface is provided for inter-task communication.

The protocols listed share diagnostic facilities (accessed via Peek, for example) that give you complete visibility into the flow of data from the network through the local machine.

Both TCP and Chaosnet utilize an implementation of Address Resolution Protocol (ARP) to provide dynamic binding of protocol addresses to physical addresses.

The Explorer implementation of the Generic Network Interface allows you to extend the system relatively easily to include additional network protocols.

Explorer SNA Communications

The Explorer System Network Architecture (SNA) Communications product provides a communications link using the IBM 3708 protocol converter that allows the Explorer to exchange data with IBM mainframe hosts. This product gives the user the ability to access a large number of existing applications that use the 3270 protocol interactively and to write programs that can interact with the 3270 protocol without user intervention. Explorer users can tap information residing on mainframe hosts, largely eliminating the need to develop their own interfaces or to input the data again. This facilitates more rapid development and implementation of expert systems and other AI applications in a variety of environments.

The Explorer communicates over a serial line to a 3708 protocol converter that connects to the IBM mainframe through an SNA/SDLC line. Explorer SNA emulates the IBM 3274 Model 51C or Model 61C control unit. Through a single converter, as many as nine Explorers can be connected to a single IBM host; alternately, eight Explorers can be connected to two IBM hosts. The 3708 converts EBCDIC data from the IBM host to ASCII format for the Explorer, and vice versa. The Explorer interfaces with the converter through VT100 terminal emulation. The Explorer SNA Communications product is a special product.

Explorer SNA II

Explorer SNA II, built around IBM's pervasive 3270 protocol, emulates the IBM 3274 Model 41C Control Unit or 3174 Model 1R Control Unit. Acting as a gateway between the IBM host and other Explorers connected over the Ethernet LAN, the SNA II package on an Explorer can provide up to 32 logical units per physical unit and 2 physical units per CCB in the Explorer. SNA II provides emulation of the 3278 Model 2 Display Unit with additional features such as support for a 25x80-character screen, user-definable characters, and user-definable key mapping. Program Station Control access, which enables programs to take the place of an operator interacting with the system, is available for all Explorer systems connected to the gateway Explorer. SNA II also includes support for leased lines and switched circuits with auto-dial capability.

DECnet

The Explorer system interface to DECnet provides you with several major capabilities. It supplies virtual terminal capabilities to allow the Explorer to serve as a terminal on another host system through the C-Term communications protocol.

You can initiate file transfers from either the Explorer or the DECnet host system. This capability is integrated into the file system to provide transparent access, that is, files are accessed via the same interfaces used to access local files with no additional effort.

DECnet also supports interprocess communication across the network. This allows Explorer users to take advantage of many of the capabilities of the DEC™ host and the Explorer systems, such as inferencing capabilities for expert systems.

Once installed, the Explorer Communications Interface to DECnet is fully integrated into the Explorer software environment, and can be used in conjunction with other Explorer software toolkits. The only hardware required for either the Explorer systems or the remote DEC system is the Ethernet interface hardware.

TCP/IP

The Explorer implementation of TCP/IP includes almost the full suite of protocols commonly implemented for access to, for example, Berkeley UNIX systems. This includes File Transfer Protocol (FTP) for access to files, Trivial File Transfer Protocol for access to files on small systems, Simplified Mail Transfer Protocol for electronic mail, Transmission Control Protocol (TCP) for reliable stream-oriented communication over unreliable networks, Universal Datagram Protocol for datagram interaction between systems, Internet Control Message Protocol for network management, and Internet Protocol (IP) for interfacing to low-level network facilities. FTP is highly integrated into the remote file system capability, providing fully transparent access. This implementation of IP supports the use of very large packets (via fragmentation and reassembly) and subnetting for dynamic management of larger network complexes. Many of the miscellaneous services available via Chaosnet are also implemented via TCP/IP.

NFS

The Network File System (NFS) provides Explorer system users transparent communications between Explorer and other UNIX workstations over Ethernet.

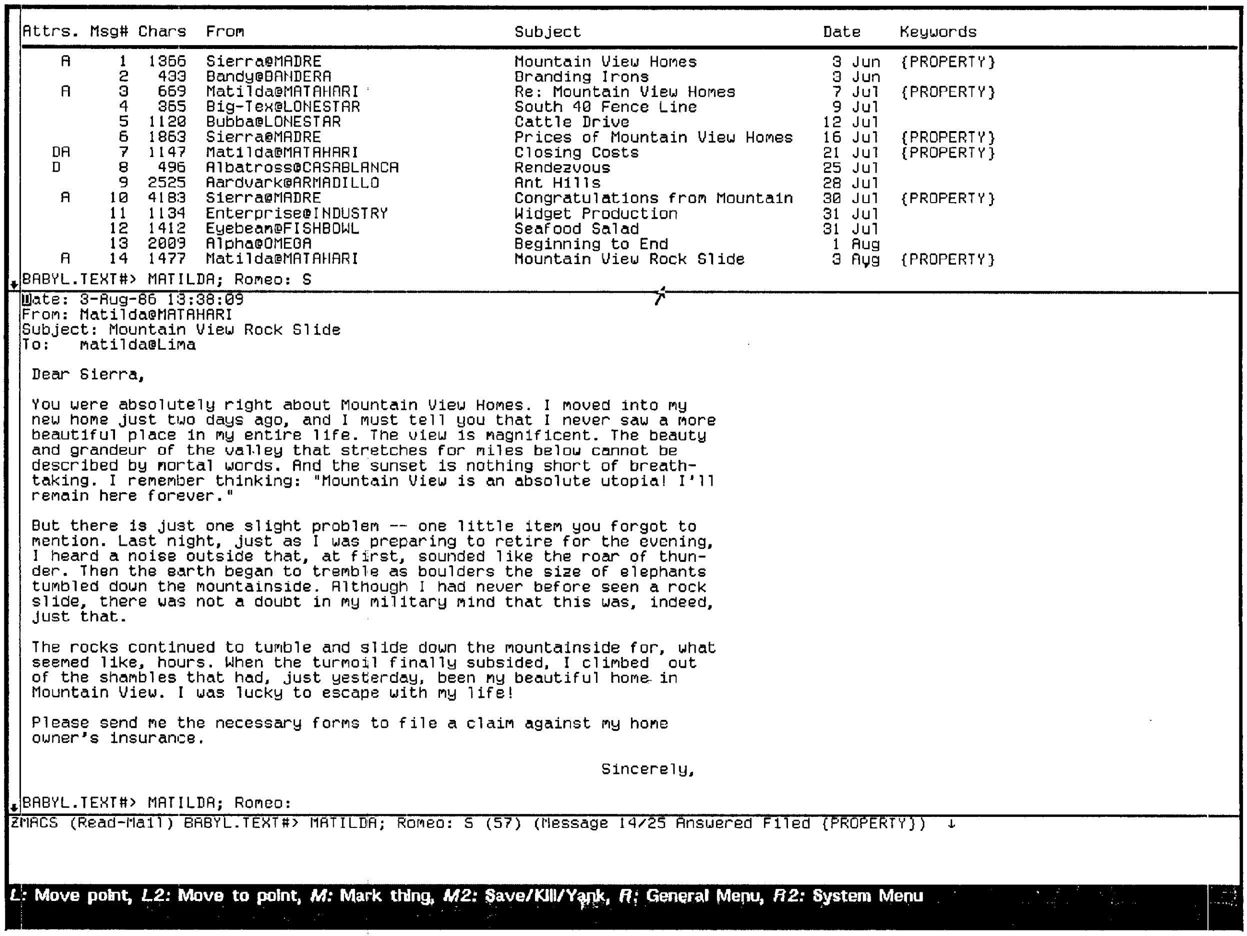
NFS is an independent network service that uses Sun's Remote Procedure Call (RPC) and External Data Representation (XDR) protocols. NFS provides the functionality necessary for each node to create a local file system made of parts of remote file systems. All accesses to files in this file system, whether local or remote, are transparent to the users and application programs. The local operating system determines whether a file is local or remote, and transparently uses the NFS protocols to fulfill requests for remote files. NFS thus allows a network to keep only a single file as a secure master to safeguard data and important files. Users do not need to know the composition, configuration, or topology of the network or of any specific machine in the network to gain the advantages of networking.

Since its introduction, NFS has gained increasing acceptance as a networking standard in the computer industry. NFS opens communications between Explorers and other computers—from personal computers to mainframes—and enables Explorer users to gain access to valuable data shared over the network.

Chaosnet

Chaosnet is an efficient protocol suite. It provides both connection-oriented and transaction-oriented modes of data transport, electronic mail, and a file access protocol that is integrated into the file system so that remote file access is almost completely transparent to most users.

The easy, high-speed integration of Chaosnet into user programs has allowed the development of networked facilities for resource management. Bulk transfer (that is, band transfer) over the network can be handled at very high data rates. Chaosnet provides a facility for quick messages, announcements, and dialogue over the net, and it provides the time and date. Chaosnet also provides services for examining the status and the logged-in user of each machine.



```
Attrs. Msg# Chars  From                          Subject                          Date    Keywords
     A     1  1366  Sierra@MADRE                 Mountain View Homes               3 Jun  {PROPERTY}
           2   433  Bandy@BANDERA                Branding Irons                    3 Jun
     A     3   669  Matilda@MATAHARI             Re: Mountain View Homes           7 Jul  {PROPERTY}
           4   365  Big-Tex@LONESTAR             South 40 Fence Line               9 Jul
           5  1120  Bubba@LONESTAR               Cattle Drive                     12 Jul
           6  1863  Sierra@MADRE                 Prices of Mountain View Homes    16 Jul  {PROPERTY}
    DA     7  1147  Matilda@MATAHARI             Closing Costs                    21 Jul  {PROPERTY}
    D      8   496  Albatross@CASABLANCA         Rendezvous                       25 Jul
           9  2525  Aardvark@ARMADILLO           Ant Hills                        28 Jul
     A    10  4183  Sierra@MADRE                 Congratulations from Mountain    30 Jul  {PROPERTY}
          11  1134  Enterprise@INDUSTRY          Widget Production                31 Jul
          12  1412  Eyebeam@FISHBOWL             Seafood Salad                    31 Jul
          13  2009  Alpha@OMEGA                  Beginning to End                  1 Aug
     A    14  1477  Matilda@MATAHARI             Mountain View Rock Slide          3 Aug  {PROPERTY}
↓BABYL.TEXT#> MATILDA; Romeo: S
Date: 3-Aug-86 13:38:09
From: Matilda@MATAHARI
Subject: Mountain View Rock Slide
To:   matilda@Lima

 Dear Sierra,

 You were absolutely right about Mountain View Homes. I moved into my
 new home just two days ago, and I must tell you that I never saw a more
 beautiful place in my entire life. The view is magnificent. The beauty
 and grandeur of the valley that stretches for miles below cannot be
 described by mortal words. And the sunset is nothing short of breath-
 taking. I remember thinking: "Mountain View is an absolute utopia! I'll
 remain here forever."

 But there is just one slight problem -- one little item you forgot to
 mention. Last night, just as I was preparing to retire for the evening,
 I heard a noise outside that, at first, sounded like the roar of thun-
 der. Then the earth began to tremble as boulders the size of elephants
 tumbled down the mountainside. Although I had never before seen a rock
 slide, there was not a doubt in my military mind that this was, indeed,
 just that.

 The rocks continued to tumble and slide down the mountainside for, what
 seemed like, hours. When the turmoil finally subsided, I climbed  out
 of the shambles that had, just yesterday, been my beautiful home in
 Mountain View. I was lucky to escape with my life!

 Please send me the necessary forms to file a claim against my home
 owner's insurance.

                                                   Sincerely,

↓BABYL.TEXT#> MATILDA; Romeo:
ZMACS (Read-Mail) BABYL.TEXT#> MATILDA; Romeo: S (57) (Message 14/25 Answered Filed {PROPERTY})  ↓

L: Move point, L2: Move to point, M: Mark thing, M2: Save/Kill/Yank, R: General Menu, R2: System Menu
```

Electronic Mail

The Explorer Mail utility is an electronic mail system that enables users to exchange mail messages via a computer network. This system makes it possible to communicate with people around the world. The system is composed of the mail reader, which allows you to receive messages from other network users, and the mailer, which sends messages over the network.

The mail reader provides two separate buffers that allow you to view and manipulate mail messages. These are called the message buffer and the mail summary buffer. The *message buffer* holds all your active messages, and provides templates that allow you to compose, send, forward, or reply to mail messages. The message buffer displays the text of each message you request. The *summary buffer,* which is the default viewing mode, contains one-line descriptions of all messages in the message sequence. The messages are marked to indicate whether you have viewed a message, replied to a message, marked it for deletion, and so on.

The mailer system provides facilities for transmitting and receiving mail messages over the network. These facilities include network mail protocols, a queuing system for transmission retries, handling of mailing lists and mail forwarding, and functions to parse and interpret mail addresses and message headers. The mailer is configurable for adaptation to a variety of mail environments. The Explorer systems provide a number of namespace objects and attributes that determine the protocol used to communicate with specific hosts and the routing used to deliver messages to a particular mail address.

A mailing list contains user mail addresses, enabling you to send messages to all users specified by the list. You simply specify the name of the mailing list in the To: field of the message template, and the mailer automatically sends the message to the members of the mailing list.

You can customize the Mail utility by defining your own filters and template buffers to supplement those defined by the system.

HARDWARE OVERVIEW 8

System Architecture

The Explorer system architecture and hardware provide high-performance processing, high-speed graphics, and system configuration flexibility at a minimum cost. The systems are centered on a high-speed backplane bus called the NuBus. The NuBus provides high bandwidth for exchange of data among all system boards. The NuBus architecture is processor independent, allowing future system growth and flexibility.

The Explorer processor directly implements Lisp for high-speed symbolic processing. The large main memory and virtual memory management allow fast execution of very large programs. The system interface board provides graphics logic, an interface to the console, and other system resources. The mass storage controller implements the industry-standard Small Computer System Interface (SCSI) bus for access to disk and tape drives as well as other commercially available devices. An optional mass storage controller provides a storage module drive (SMD) disk interface in addition to the SCSI bus. The NuBus Ethernet controller provides access to local area networks (LANs). Other NuBus boards may also be configured in Explorer systems to offer a variety of hardware options and features.

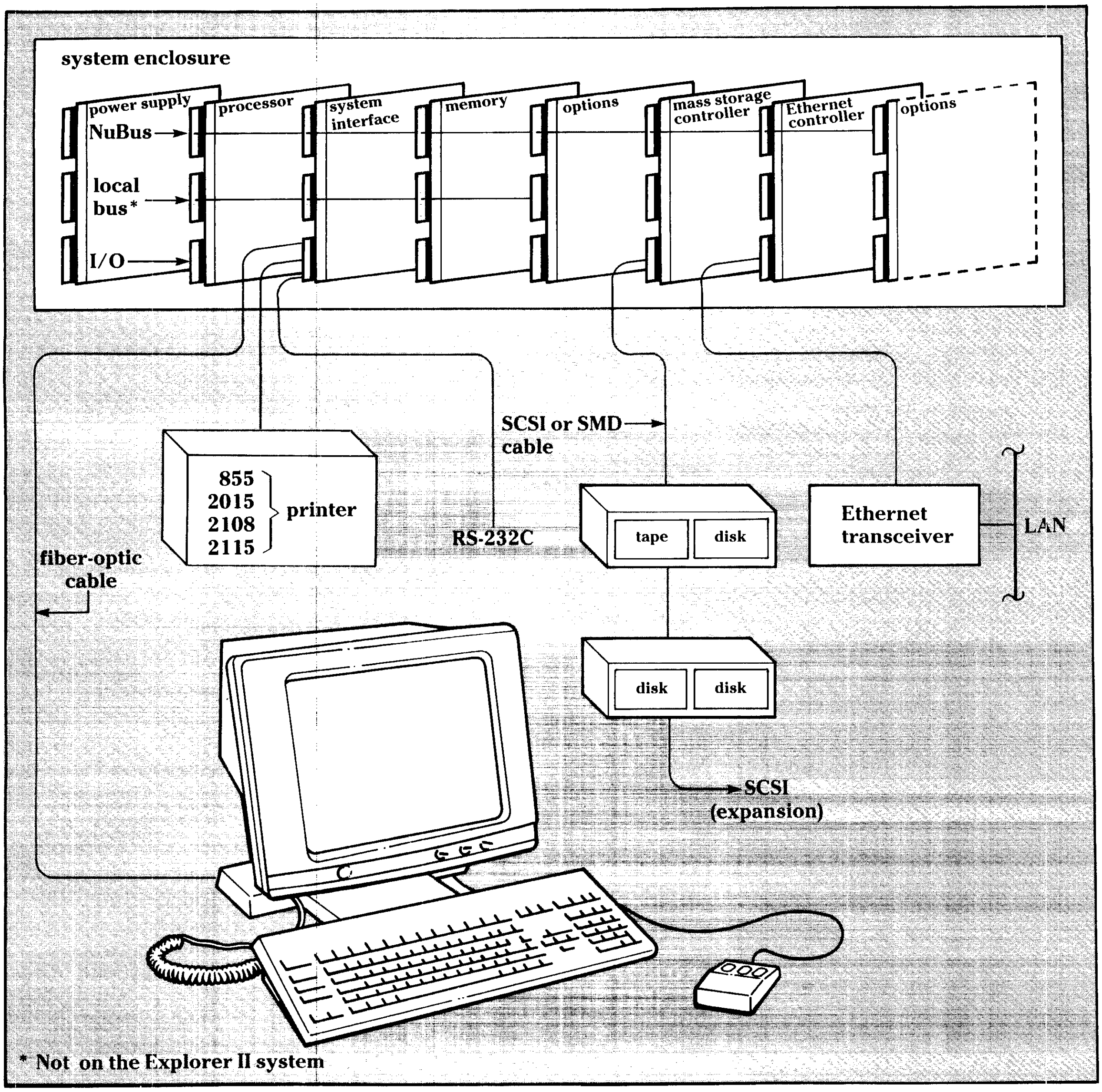
A 68020-based processor is an optional NuBus board used in the Explorer LX system, which enables you to integrate the symbolic processing of an Explorer system with the conventional processing available from TI System V. In addition, TI System V can drive up to sixteen additional conventional terminals or data I/O lines using a communications carrier board (CCB) with plug-in cards that feature various interfaces.

The processor, the system interface board, and the memory boards use state-of-the-art surface-mount technology, with which chip carrier circuits are soldered directly to the surface of the board to allow better packing density and increased performance without high costs.

All system boards use a standard three-high Eurocard format with three 96-pin DIN connectors that plug into the backplane. All cable connections are through peripheral cable adapter (PCA) boards that plug into the rear of the backplane, thus eliminating the need for front edge connectors. No special backplane wiring, jumpers, or switches are required, making the boards easy to install and remove for service or changes in system configuration.

Each system board contains a configuration read-only memory (ROM) that allows the system to configure itself automatically according to the boards installed. At power-up, the software reads each configuration ROM—slot by slot—to identify, test, and initialize the boards. Each board has a red fault indicator light-emitting diode (LED), mounted on the front edge, for quick recognition of a failed board if the monitor is unavailable for reporting error messages.

If you purchase an Explorer and later decide that your needs require the increased performance of the Explorer II, you can easily upgrade to the Explorer II by replacing the processor board.


system enclosure
power supply
NuBus
local bus*
I/O
processor
system interface
memory
options
mass storage controller
Ethernet controller
options
SCSI or SMD cable
855
2015
2108
2115
printer
RS-232C
tape
disk
Ethernet transceiver
LAN
fiber-optic cable
disk
disk
SCSI (expansion)
* Not on the Explorer II system

NuBus System Interface

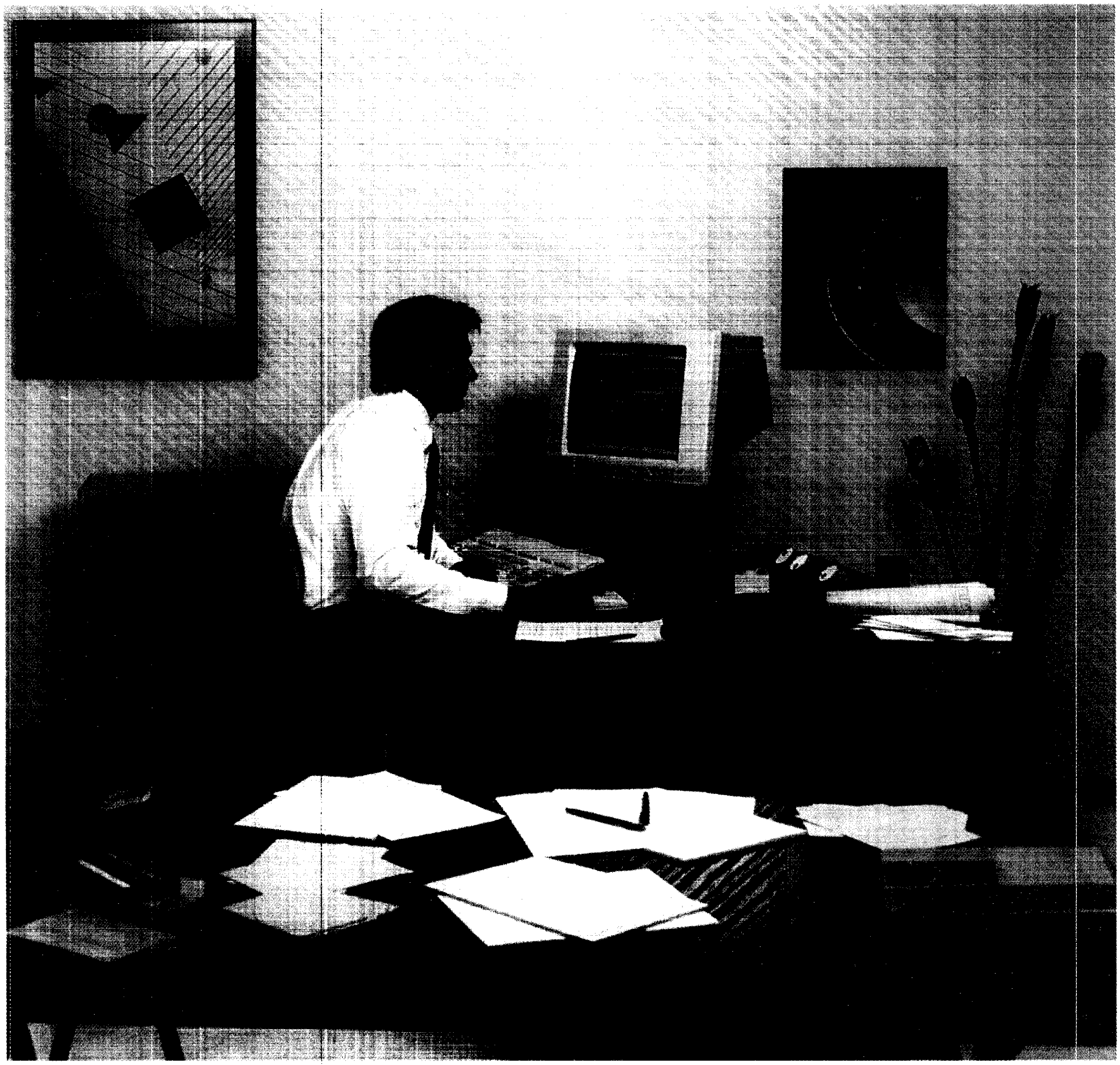
The NuBus system interface is a flexible bus structure. For each interaction, a device takes control of the NuBus interface, thus becoming a *master*, and addresses another device that becomes a *slave* for that interaction. The slave device, after decoding its address, acts on the command provided by the master. A simple handshake protocol between the master and the slave allows devices of different speeds to use the NuBus interface. Fair arbitration among potential bus masters gives each device an equal share of the bus bandwidth.

The NuBus interface supports direct addressability of more than four gigabytes of memory with a 32-bit address. The following features are included in the NuBus interface:

- Processor-independent architecture with multiprocessor support
- 32 address lines multiplexed with 32 data lines that support 8-bit, 16-bit, and 32-bit data transfers
- Single, linear address space
- Block transfers of multiple words
- 37.5-megabyte-per-second maximum transfer rate
- 10-megahertz bus clock
- Synchronous operation
- Simple bus protocols (memory reads and writes are the only bus operations)

The NuBus features a memory-mapped event mechanism rather than a hard-wired interrupt scheme. Interrupts are implemented as write operations and require no unique signals or protocols. Any device can interrupt any other device on the NuBus by writing into an area of the address space that is monitored by the other device. A predefined set of addresses is used to post hardware interrupts. Additional software interrupts can be posted at any available location in the address space. The priority of the interrupt is specified by the software according to its address.

NuBus boards from third-party companies have been developed, and designs for additional NuBus products are in progress. If you want to design your own processors or controllers, you can obtain a license from TI to design and manufacture NuBus-based boards.



Local Bus

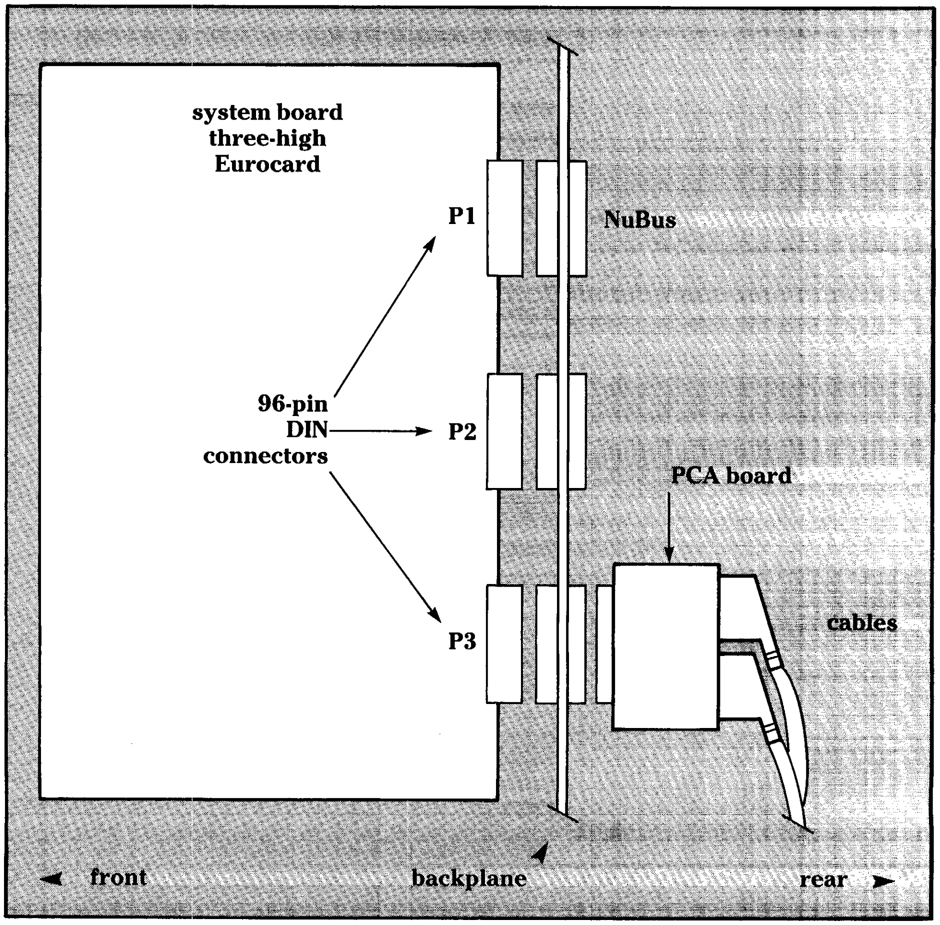
The local bus is a high-speed, 32-bit private bus for exclusive use by one Explorer processor. It is located on P2 and spans slots 3-6 on the Explorer backplane. The local bus provides the Explorer processor with access to main memory and the graphics bit map. It offers a high processor-to-memory bandwidth and leaves the NuBus available for system-wide operations. Use of the local bus is completely transparent to the software. Explorer II backplanes do not include the local bus. The Explorer II processor is tightly coupled to an on-board cache that is filled and emptied using NuBus block transfers.

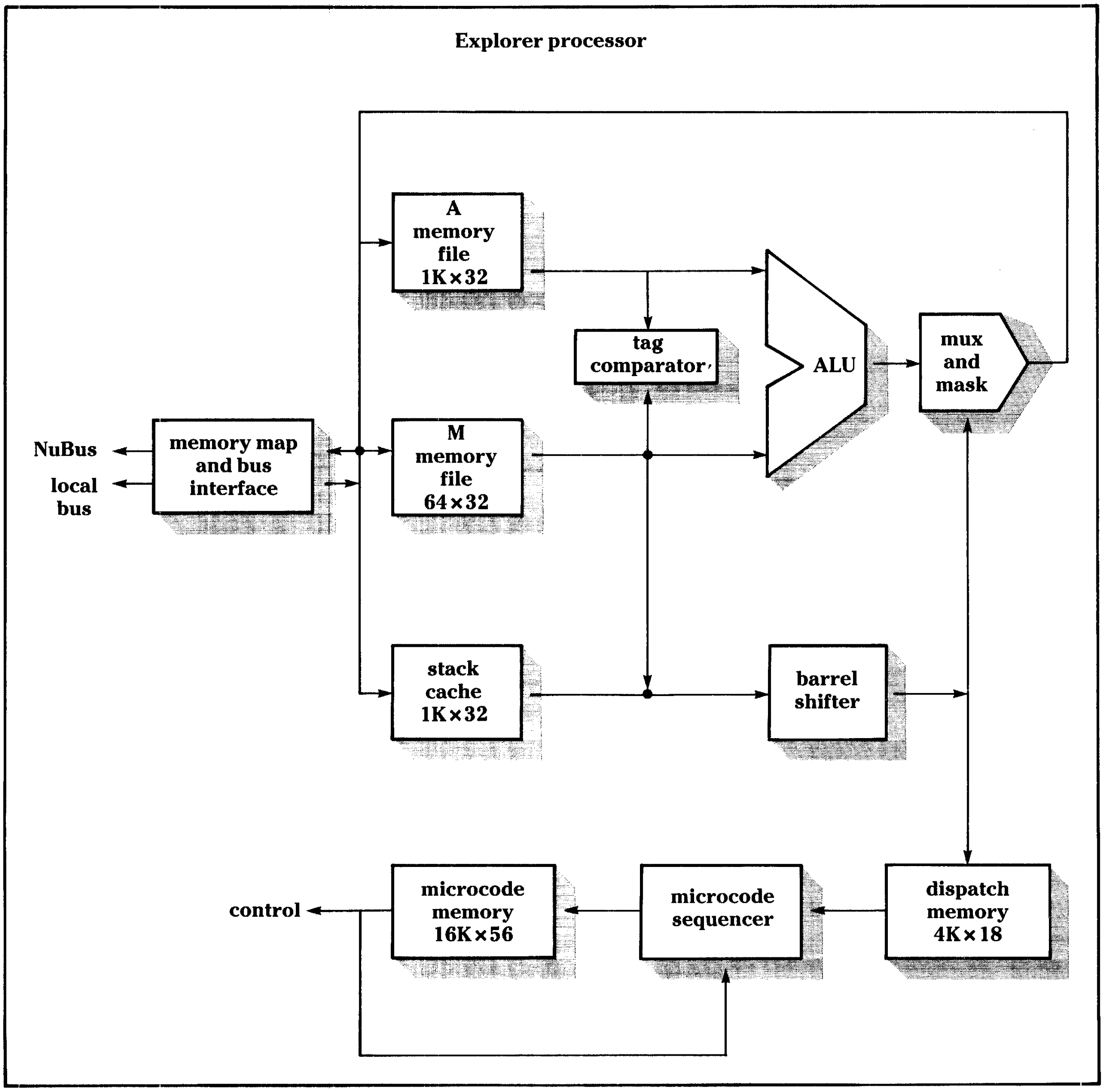
Backplane

The Explorer backplane allows a wide variety of boards to be installed in versatile configurations. It consists of a printed wiring board that contains eight slots. Slots 0 through 6 are for system boards, and slot 7 is reserved for the power supply. Slot 6 is specially designed for a high power dissipation load; the Explorer or Explorer II processor is installed in this slot.

Two versions of the backplane for the seven-slot enclosure are available, one with the local bus (for the Explorer) and one without (for the Explorer II).

The top row connectors (P1) are bused across seven slots to provide NuBus connections to all logic boards. The second and third row connectors (P2 and P3) are used for I/O signals defined by the particular board installed in that slot. PCA modules convert from the 96 DIN connectors on the backplane to the type of connector used on a given I/O cable.





Explorer processor

Processor Architecture

The Explorer processor is a microprogrammed processor specially designed for high-speed symbolic processing. Two versions of the Explorer processor are available: a very large-scale integration (VLSI) processor chip version (Explorer II) and a bit-slice processor version. Both use microcode designed for optimum Lisp performance.

Features common to both Explorer processors are:

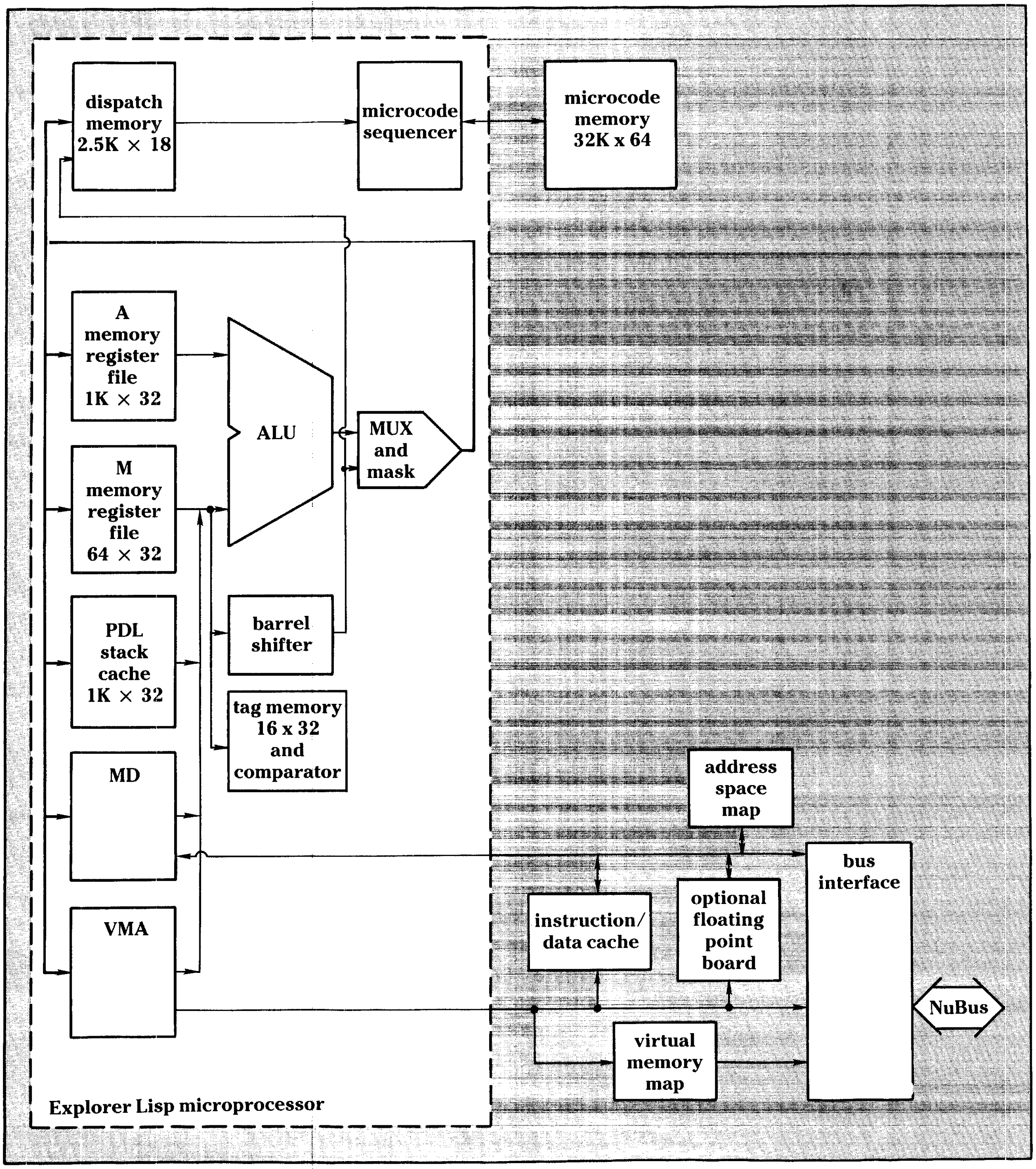
- 32-bit data path
- Bit-field hardware to manipulate complex data structures
- Tagged architecture for typed data
- Multibranching to facilitate control associated with symbolic processing
- Lisp instruction set decoding
- 128-megabyte virtual address space with demand-paged memory mapping
- Special hardware to facilitate garbage collection
- Large stack cache

Features unique to the Explorer II processor are:

- VLSI processor chip
- 128K-byte, two-way set-associative cache memory
- Autotransport mechanism to follow pointers in memory
- Higher microinstruction clock rate
- Floating point option

The Explorer II processor has a 32K by 63-bit writable control store for microinstructions. Microinstructions are pipelined and are fetched at the clock rate of the processor.

The Explorer has a 16K by 56-bit writable control store for microinstructions.


dispatch memory 2.5K × 18
microcode sequencer
microcode memory 32K x 64
A memory register file 1K × 32
ALU
MUX and mask
M memory register file 64 × 32
barrel shifter
PDL stack cache 1K × 32
tag memory 16 x 32 and comparator
MD
address space map
bus interface
VMA
instruction/ data cache
optional floating point board
NuBus
virtual memory map
Explorer Lisp microprocessor

Explorer II General Features

Major functional features of the Explorer II processor board include the Lisp processing performed by the Explorer Lisp microprocessor, on-board cache memory storage, virtual memory mapping, and floating point option board.

The cache memory logic and virtual memory mapping logic on the Explorer II processor board are an integral part of the processing system, providing a virtual-to-physical address translation for the processor as well as a direct high-bandwidth path to the private cache memory of the processor. This eliminates the need to arbitrate for the NuBus on a high percentage of memory read cycles. The Explorer II processor board contains the NuBus master logic that accesses external memory to perform cache-fills if the data is not in cache memory. The Explorer II processor also accesses the NuBus directly when the processor is in the physical addressing (unmapped) mode. The processor can post events to other devices, perform I/O operations, and access other memory not known to the virtual memory system.

The Explorer II processor logic is built around several major interconnecting buses:

- Virtual memory address (VMA) bus — Unidirectional 32-bit address bus whose source is the VMA register of the Explorer Lisp microprocessor.
- Data bus — Bidirectional interface to the Memory Data register of the Explorer Lisp microprocessor.
- Internal data bus — 32-bit bidirectional bus for NuBus resources.
- Internal address bus — Complement to the internal data bus.

Virtual Memory Mapper

The memory mapper provides virtual-to-physical address mapping for the processor. The memory mapper consists of two major blocks of logic—the virtual memory map and the address space map. The virtual memory system views the main memory as a 128-megabyte address space that is divided into 128K pages of 1K byte (256 words) each.

Cache

The cache is a two-way set-associative cache with 4K blocks of 4 words each, for a total capacity of 128K bytes. This cache is closely coupled to the memory mapper, relying on map status information to determine if a cache hit has occurred. The cache is driven by virtual addresses rather than physical addresses. A cache-option feature on the Explorer II processor board allows the cache to be selectively controlled in specific cases of prefetch cycles, autotransport cycles, instruction fetch cycles, and program data cycles. This cache-option feature also allows the total disabling of the cache.

Explorer Lisp Microprocessor

The Explorer Lisp microprocessor, a single chip condensed from the 32-bit Explorer processor, provides significantly increased performance to Explorer systems and also improves reliability by reducing the parts count. This VLSI chip contains over 550,000 transistors that form the arithmetic logic unit, control logic, and six different groups of random access memory (RAM), totaling 114,000 bits of RAM. The chip is 1 square centimeter, has 264 pins, and is constructed with a 1.2-micron, double-level metal CMOS process.

The VLSI processor implements a microcoded architecture that is compatible with the Explorer processor. Location of the 4000-byte register file and the 4000-byte top-of-stack cache on the VLSI chip greatly enhances the performance of the processor. The location of the RAM on the VLSI chip and the high clock speed greatly increase the microinstruction execution rate. Additional improvements in the control logic allow the VLSI chip to implement Lisp operations with fewer microinstructions, further improving the performance of the system.

Explorer Memory Board

The Explorer uses a large main memory for high performance in a demand-paged environment.

The Explorer memory board, which uses a local bus and a NuBus, is available in 4- and 8-megabyte configurations with byte parity error detection. Either one or two memory boards can be used for a range of from 4- to 16-megabytes of main memory. The memory board can perform individual 8-, 16-, or 32-bit write cycles from either the local bus or the NuBus. Memory access time via the local bus is less than 300 nanoseconds.

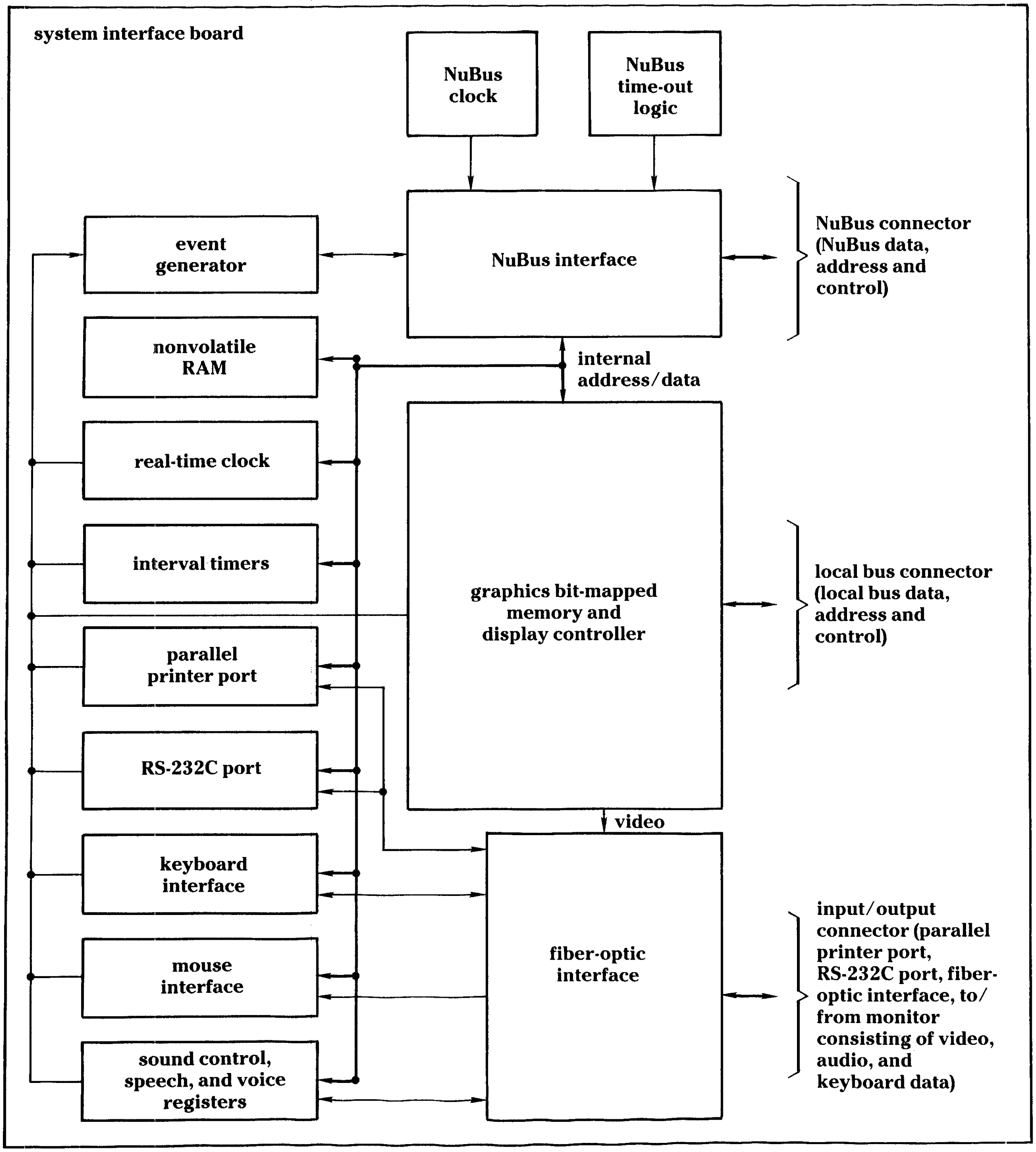
The memory board is a slave to the NuBus except when it is under the control of the local bus processor. When the local bus is present, the memory board is at all times a slave to the local bus. When two memory boards are used in a system with a NuBus and a local bus, one board can serve the NuBus at the same time that the other board is serving the local bus. Memory data block transfers are supported when the memory board is functioning as a NuBus slave. Block transfers are not supported on the local bus.

Explorer II Memory Board

The Explorer II memory board uses the NuBus. The Explorer memory board can also be used in the Explorer II. In this case, all accesses from the Explorer II processor are via the NuBus; the local bus port on the memory board automatically disables itself.

The Explorer II memory board supports only NuBus access and is available in 8-, 16-, and 32-megabyte configurations with error detection and correction (EDAC) logic to correct single-bit errors and flags dual-bit errors. The Explorer II (which does not contain the local bus) has no slot limitation on memory board locations, allowing configurations up to the full 128 megabytes of Explorer virtual address space. This NuBus slave memory board also supports byte, half-word, and 32-byte write cycles, as well as NuBus slave block transfers. Memory cycle times for block transfers after the first word are 100 nanoseconds.

The Explorer II memory board contains the control logic and memory chips to implement 32 megabytes of memory space using 1-megabit dynamic random-access-memory (DRAM) chips. Each stored word consists of 32 data bits and a 7-bit check word. Parallel EDAC logic on the board uses a modified Hamming code to generate the check word from the 32-bit data word during each memory write cycle.


system interface board
NuBus clock
NuBus time-out logic
NuBus interface
NuBus connector (NuBus data, address and control)
event generator
nonvolatile RAM
internal address/data
real-time clock
interval timers
graphics bit-mapped memory and display controller
local bus connector (local bus data, address and control)
parallel printer port
RS-232C port
video
keyboard interface
mouse interface
fiber-optic interface
input/output connector (parallel printer port, RS-232C port, fiber-optic interface, to/from monitor consisting of video, audio, and keyboard data)
sound control, speech, and voice registers

System Interface Board

The system interface board provides a wide variety of user and system services:

- Graphics control logic
- Interface to the monitor, keyboard, and mouse through a fiber-optic link
- System resources
- Parallel printer and RS-232C ports

The graphics controller uses a one-megabit bit map to store the high-resolution display image (1024 by 808 pixels). The bit map is implemented with high-speed RAM for instantaneous image update. The video display is completely refreshed 60 times per second to minimize flicker. Write operations to the bit map are interlaced with video display refresh cycles to eliminate any speckled effect on the display. The graphics controller has special hardware to accelerate window operations.

The fiber-optic link carries the display image, keyboard, and mouse data. The keyboard interface provides a special chord detector that monitors key-code sequences that can initiate a system reboot. Sound-control circuits provide program control of the sound/noise generator in the console.

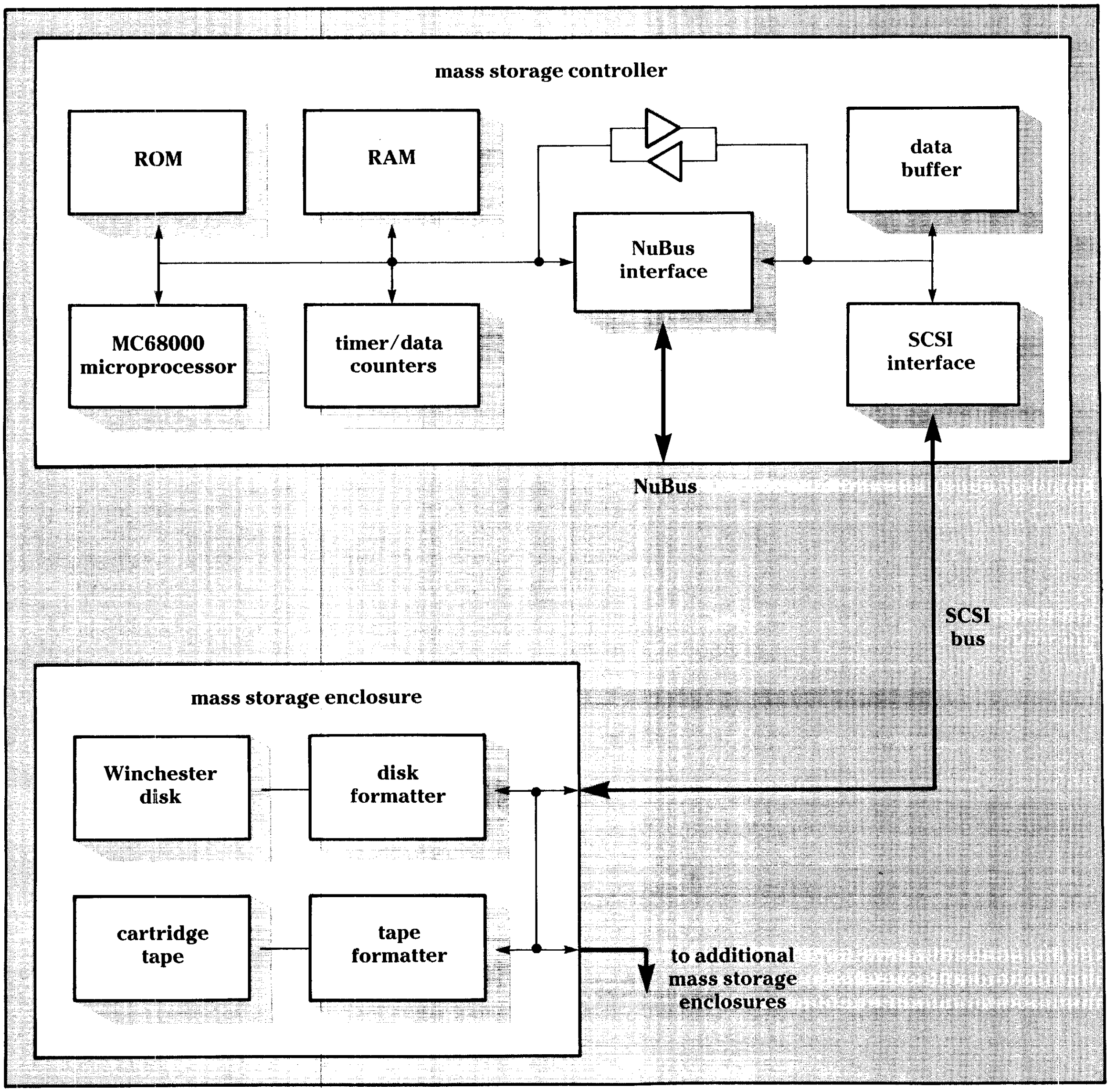
Other resources include clocks, timers, nonvolatile memory, and power failure event logic. A battery-powered real-time clock makes it unnecessary to reinitialize the system clock on power-up. Other interval timers can issue events at programmable intervals. The 2K-byte nonvolatile RAM maintains critical data through power-off, power failure, and board removal. Circuits monitor the power failure warning line from the power supply and generate power failure warnings to all boards in the system.

The parallel printer port provides a Centronics-compatible, low-cost peripheral connection. The RS-232C port provides a standard connection for a wide variety of devices.

Graphics Controller and Memory

The bit-mapped graphics display controller consists of an 8K-word by 64-bit bit-mapped memory, a NuBus interface, a local bus interface, address latches, a logical operation unit, a cathode-ray-tube controller, shift register, shift register latch, and video generator.

The bit-mapped memory stores an image of the information displayed on the video display. The memory uses 16 high-speed 8K by 8-bit static RAM devices to provide a single-plane video map of 1024 by 1024 pixels. Of these, 1024 by 808 pixels are displayed. The display controller must access the bit-mapped memory in order to refresh the video display. These accesses are shared with the processor memory accesses in such a way as to maximize processor access without sacrificing video display quality. The processor accesses the bit-mapped memory using either the local bus or the NuBus.


mass storage controller
ROM
RAM
data buffer
NuBus interface
MC68000 microprocessor
timer/data counters
SCSI interface
NuBus
SCSI bus
mass storage enclosure
Winchester disk
disk formatter
cartridge tape
tape formatter
to additional mass storage enclosures

Mass Storage Subsystem

The mass storage subsystem is available in a variety of configurations with various devices and enclosures. In a small form factor, 5 1/4-inch disks and tapes are housed two per mass storage unit (MSU) enclosure. Each intelligent formatter in the enclosure provides device-level control for the specific device(s) attached to it, 143-megabyte (unformatted) Winchester disks and 60-megabyte cartridge tapes. Table-top enclosures house 1/2-inch tapes, and floor-mount cabinets house larger SMD Winchester disks.

Two mass storage controllers are available. Both have a SCSI bus to support 5 1/4-inch disks, cartridge tapes, and 1/2-inch reel tapes:

- NuBus Peripheral Interface (NUPI), which provides a SCSI interface
- Mass Storage Controller (MSC), which provides an interface to either SMD or SCSI

The NUPI features a 10-megahertz MC68000 microprocessor that relieves the system of overhead associated with controlling multiple devices on the SCSI bus. The NUPI also allows data transfers to and from noncontiguous pages in main memory via a special scatter/gather mechanism. The data path in the NUPI is separate from the MC68000 microprocessor bus, allowing noninterleaved data transfers to and from disks. In addition, a special extended command set enhances demand-paging operations.

The MSC is similar to the NUPI. In order to handle the increased bandwidth required to support both SCSI and SMD devices, the MSC features a high-performance direct memory access (DMA) engine and two MC68010 microprocessors, as well as a NuBus master interface with a block-move feature that can sustain transfer rates of over 6 megabytes per second between on-board DMA and NuBus memory.

The SCSI bus interconnects up to seven formatters on a single daisy-chained bus. The SCSI bus uses high-level generic mass storage commands to support a mix of peripherals with different characteristics and various data transfer rates. SCSI signals are bused on a 50-pin cable, which includes an 8-bit data bus with parity, bus arbitration, and handshake signals. The NUPI controller supports asynchronous SCSI data transfers for rates up to 1.5 megabytes per second. The MSC supports both asynchronous and synchronous SCSI data transfers for up rates to 4 megabytes per second.

Mass Storage Controller Design Features

Each of the Explorer mass storage controllers share important design features—a command block software interface, data path independence from control paths, and SCSI bus implementation.

Command Block Software Interface

The mass storage controller command block software interface is fast and flexible because the Explorer system processor can build commands in NuBus memory and independently notify the controller of each command by its address in NuBus memory. The Explorer processor issues a command to the controller by building a command block in NuBus memory and then writing the address of this block into the controller command address register. The controller acknowledges receipt of the command block by setting the busy bit in the command block in NuBus memory. The controller then executes the command, performing scatter/gather on data in NuBus memory, queuing up commands to various devices, and so on. Unlike the NUPI controller, which is limited to one active command per device, the MSC provides internal command queuing for multiple commands per device. On completion of the command, the controller updates the command block in NuBus memory, resetting the busy bit and setting the command complete bit.

Independent Data Paths

Explorer system mass storage controllers are designed with independent paths for data and control. Data from peripheral devices pass through the controller on dedicated data paths, with DMA control for efficient transfer between devices and NuBus memory. This provides maximal sustained bandwidth for disk and tape data transfers.

Some mass storage controllers share microprocessor and device data buses, resulting in bottlenecks between control operations and data transfers. These bottlenecks usually result in a need to interleave disk data, so that a small burst of data is transferred with an intervening delay to the next block or sector of data. This greatly reduces disk data throughput because it requires multiple disk revolutions to transfer a single track of data. The Explorer mass storage controllers, however, maintain one-to-one interleaving of all disk data. This ensures optimal throughput in all Explorer I/O operations.

SCSI Bus

The SCSI bus provides the Explorer system with several advantages:

- It is an industry standard — The SCSI bus has a wider range of peripheral offerings than any other interface available, letting you configure tapes, disks, and optical disks in various shapes and sizes to your Explorer system.

- It allows device independence — The SCSI bus accommodates many different kinds of peripherals. Only one slot in the system enclosure is required to support both tapes and disks, as well as future mass storage upgrades.

- It combines performance with configuration flexibility — Because the SCSI bus allows multithreaded I/O with several active devices by means of disconnect/reconnect protocols and bus arbitration, both slow and fast devices can coexist on the bus, each meeting its bandwidth requirements. The NUPI and the MSC each can support up to seven SCSI devices.

Ethernet Controller

The Ethernet controller option provides support for a LAN. A LAN is a communication link between pieces of equipment that are close enough to link with a continuous cable but too far apart to link with a backplane or a parallel-format cable, generally a distance that ranges from 5 meters (16.5 feet) to 1000 meters (3300 feet). The upper distance limit is set by the power of the line driver devices and the signal losses in the transmission cable.

The NuBus Ethernet controller is a single board that plugs into the backplane of the system enclosure. The controller is driven by Explorer processor-based microcode. An I/O adapter board on the rear of the backplane and an adapter cable connect to a standard Ethernet transceiver and cable. Ethernet software support is described in Section 7, Networking Facilities.

The main logic groups on the NuBus Ethernet controller are:

- Intel 82586 Local Communications Controller
- Serial interface device (Manchester code converter)
- Buffer memory
- NuBus slave logic
- NuBus master (event) logic
- On-board data buses and control
- Adapter interface board

LX Hardware

The Explorer LX is a multiprocessor version of the Explorer systems that combines symbolic and numeric processors with a powerful set of software tools. The base Explorer LX includes a 68020-based processor board. You can add a CCB to expand the Explorer LX with terminals, and you can add other I/O devices.

68020-Based Processor and Memory

The 68020-based LX processor features the following:

- Virtual memory for sharing system operations, addressable to 4 gigabytes
- Three main buses for addresses and data
- 68020 microprocessor for system operation and control, operating at 16.7 MHz
- 68881 floating-point coprocessor for floating-point calculations, operating at 16.7 MHz
- 68851 paged-memory management unit for memory control
- 16K-byte cache memory for logical address, data, and instructions
- Main memory for software operations; 2 megabytes of error-correcting on-board DRAM with an optional 2 megabytes of DRAM in a piggy-back board
- Interrupt control chip for handling board and bus interrupts
- Three timers for a real-time clock, interval timing, and time accounting
- NVRAM for system configuration support
- ROM for board information and self-test

CCB

The CCB complements the other communications option hardware on the Explorer systems, offering a variety of configurations with plug-in cards and downloadable 68010-based code. Plug-in option cards include an eight-channel asynchronous interface and a multifunction card. The CCB supports any combination of up to two option boards, such as the 8-channel MUX and multifunction option, with appropriate adapter cards and cables that plug in the rear of the system enclosure backplane.

Terminals

Using the CCB and an RS-232C interface, you can install up to sixteen terminals to the TI System V side of an Explorer LX. Terminals typically installed include the Model 924 Video Display Terminal (VDT) and the Model 945 workstation.

Coprocessors

In addition to the Lisp processor available on the Explorer systems, two coprocessors are available as options—the 68020-based processor and the special product Odyssey board. (A special product is available under special terms and conditions.)

68020

The Explorer LX combines the Explorer Lisp processor with the 68020-based processor running a TI-implemented version of UNIX System V and specially-developed LX software to facilitate communication between the two processors. In addition, the LX processor board includes a 68881 floating-point coprocessor to handle numeric computations.

TI System V supports multiple processors. Thus, you can install two 68020-based processors in a standard system enclosure to produce a three-processor system with concurrent execution. With the larger chassis, you can install up to four processors. The larger chassis is a special product.

Odyssey

The special product Odyssey board, when installed on an Explorer system, adds high-performance digital signal processing and expandable multiprocessor capabilities. The Odyssey thus provides a powerful environment for the development and delivery of a wide range of applications that require the combination of high-speed numerical computations and symbolic processing. These applications include speech recognition and synthesis, signal understanding and interpretation, image understanding, real-time digital signal processing, and neural network simulation.

With appropriate speech recognition and synthesis software, Odyssey adds voice input and output capabilities to expert systems. The system also provides data acquisition and rapid signal analysis necessary for Explorer-based software to interpret various signals. For example, an expert system would have rapid access to results of a fast Fourier transform performed on vibration data.

Odyssey is based on the TI TMS32020 Digital Signal Processing (DSP) microprocessor, which features a 200-nanosecond instruction cycle time. Each Odyssey board contains four TMS32020 processor modules, providing a powerful dedicated signal-processing environment that performs independent parallel processing. In local-memory mode, each processor module can execute separate algorithms simultaneously. Each module consists of the TMS32020 processor, 64K x 16-bit words of DRAM for data memory, 8K x 16-bit words of high-speed static RAM for program memory, and control logic to access the Odyssey internal bus, which is essentially a modified NuBus. This internal bus communicates among modules on the same board or between boards. An additional communication/control module on the Odyssey board provides a two-way data path to the NuBus for communication with other boards within the Explorer system, such as the Lisp processor.

HARDWARE PACKAGING

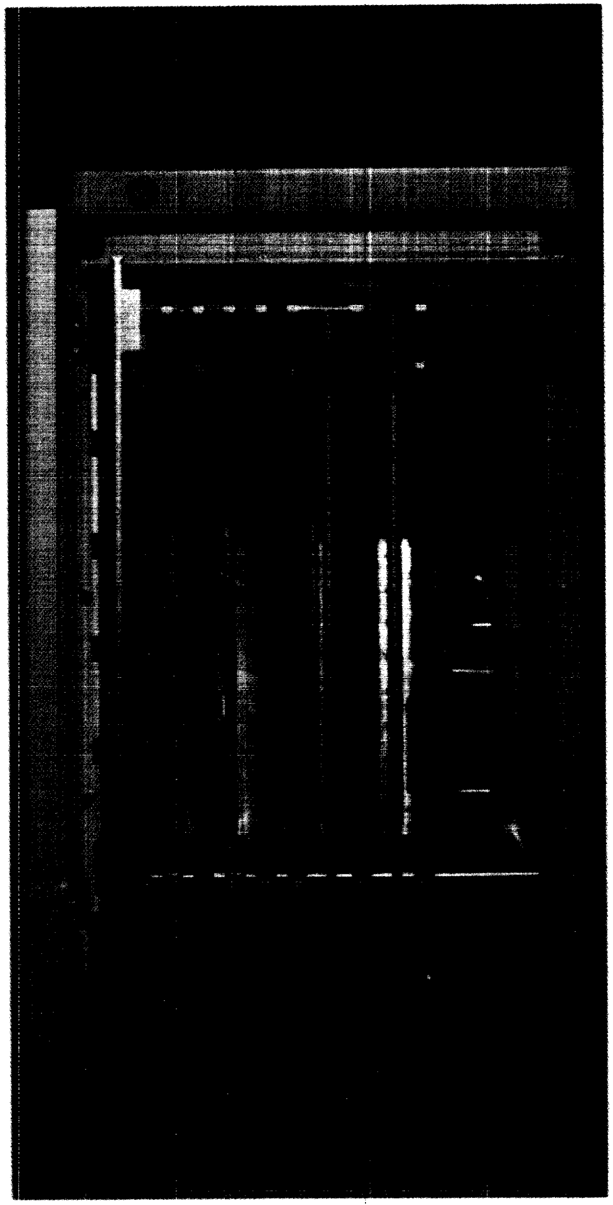
The Explorer systems use advanced technologies in the design and packaging of hardware components. The components are packaged in attractive, compact enclosures to make the system a pleasant and unobtrusive addition to your office. Included are the following design considerations:

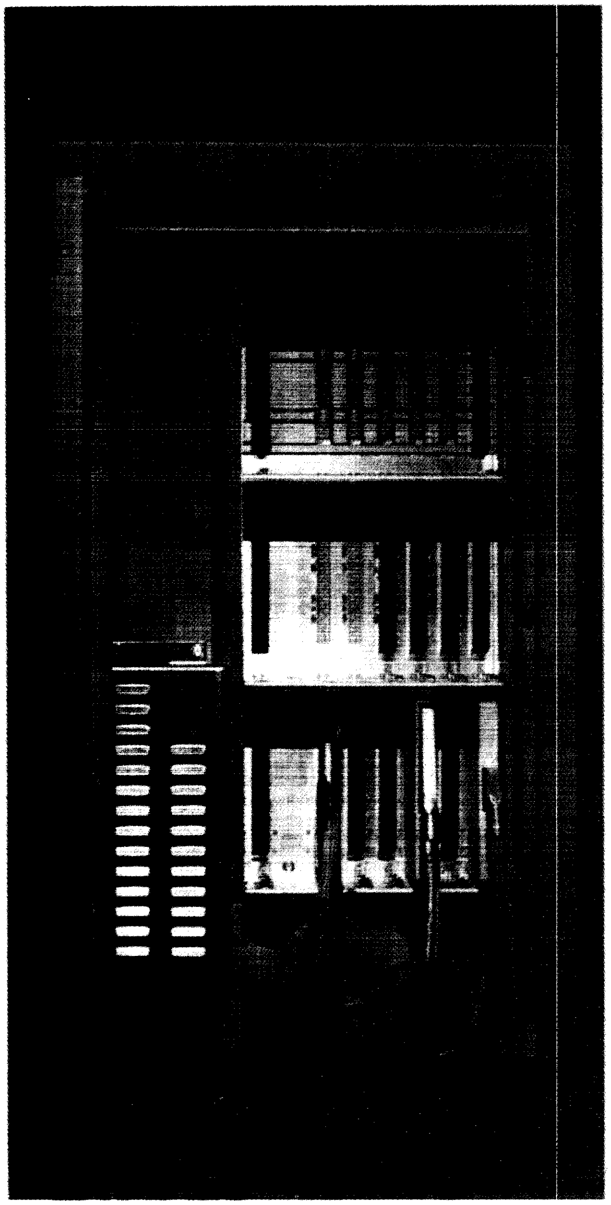
- Usability
- Reliability
- Quiet operation
- Ease of service

The design of the Explorer systems is influenced by the results of ergonomic studies to provide comfortable, efficient operation. System components are packaged in separate enclosures so that you can place them in convenient locations. Parts are readily accessible for service or quick replacement. All components and assemblies are built to exact standards and subjected to extensive quality checking. The system is designed to operate under normal office conditions and requires minimum site preparation. You can easily convert the system for international voltage and frequency requirements. Advanced fan and motor designs provide effective cooling with minimum noise.

The hardware components include the following:

- System enclosure
- System console (monitor, keyboard, and mouse)
- Mass storage enclosures
- Printers (optional)





System Enclosure

The system enclosure contains seven slots to hold circuit boards. The processor is always installed in the seventh slot because this slot provides a higher cooling capacity. The remaining six slots are used for the other system boards. In systems that contain the Explorer processor, the memory board and the system interface board are installed in slots that connect to the local bus. In Explorer II, the other system boards may be installed in any slot except the seventh.

The system enclosure opens from both the front and rear for easy access to the boards and cables. Interlock switches remove power from the enclosure whenever the front inner door or rear door is opened. The circuit boards are easily accessible from the front and connect to the backplane through three 96-pin DIN connectors. The cables are accessible from the rear and connect to the backplane through Peripheral Cable Adapter (PCA) boards. This design allows you to remove system boards without unplugging cables. All cable connections are neatly hidden inside the rear door.

A high-efficiency switching power supply is also accessible from the front and plugs into the backplane. The power supply reports fan failure, overtemperature, and power failure for data protection. Only conventional 120-volt ac, 15-ampere service is required, which reduces site preparation to a minimum.

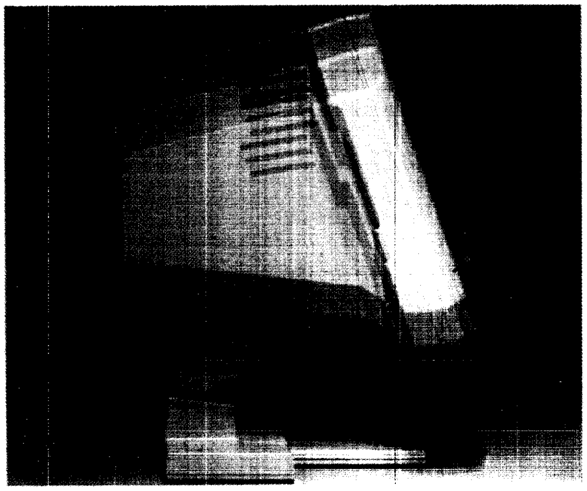
The cooling system provides a unique airflow design that takes in air from the top of the enclosure and exhausts it at the bottom, minimizing dust intake and annoying waist-high air movement. An air filter is located above the board slots for easy cleaning. The fan is well-embedded in the enclosure with extensive vibration-damping devices for low noise level.

The system enclosure is compact, weighing approximately 27.7 kilograms (60 pounds) without boards installed and 38.5 kilograms (85 pounds) when full. It measures 63.5 centimeters (25 inches) high, 33 centimeters (13 inches) wide, and 45.7 centimeters (18 inches) deep, allowing it to fit under most desks or tables.

An optional, larger system chassis is available as a special product. (A special product is available under special terms and conditions.) This larger chassis can contain up to sixteen boards driven by one NuBus board, or up to twenty-one boards when driven by two NuBus boards.

System Console

The system console consists of the monitor, keyboard, and mouse. A fiber-optic link connects the monitor to the system enclosure. Connections to the keyboard and mouse are also routed through the fiber-optic link. The fiber-optic link provides a flexible, lightweight cable connection that transfers data at the rate of 68 megabits per second. Cables are provided in standard lengths of 15.2 meters (50 feet) and optional lengths of 60.9 meters (200 feet). An optional plenum fiber optic cable is available in the 60.9 meters length.

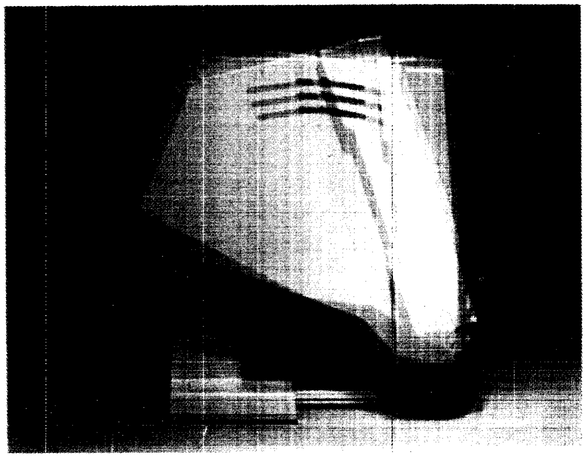


Monitor

The monitor houses the video display and contains cable connectors for the keyboard and mouse as well as a microphone and headset. The cable connectors are located on the front of the monitor base to eliminate long cables and minimize cable clutter. The monitor also includes a built-in speaker with volume control.

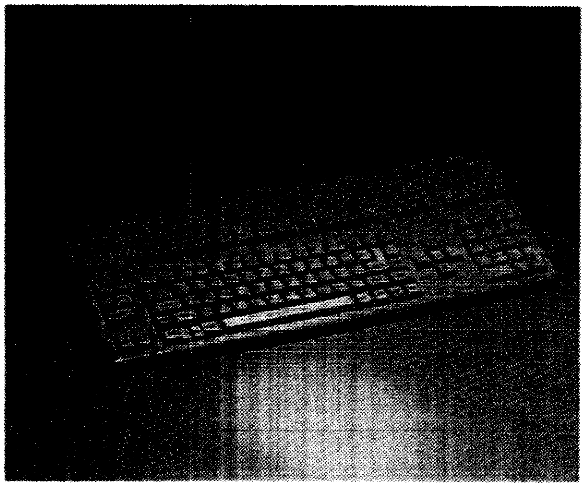
The 43-centimeter (17-inch) diagonal tube mounted in landscape orientation provides a high-resolution display with 1024 pixels horizontally and 808 pixels vertically. The video display is completely bit-mapped and uses a 60-hertz refresh rate with a noninterlaced raster to minimize flicker.

The display is monochrome (black and white) with simulated gray tones supported by software. The software supports display of normal and reverse video images. You can control both the brightness and contrast of the images with controls on the front of the monitor.



The monitor requires minimal desk space with a 39.6-centimeter (15.6-inch) by 31.2-centimeter (12.3-inch) base. The advanced ergonomic monitor stand has height, tilt, and swivel adjustment for a comfortable view of the screen. The display screen has a continuous height-adjustment range of 10.9 centimeters (4.3 inches), a continuous tilt range of 5 degrees forward and 20 degrees backward, and a continuous swivel range of 360 degrees.

The monitor weighs approximately 18.1 kilograms (40 pounds) and measures 39.8 centimeters (15.7 inches) high (at its lowest height adjustment), 43.9 centimeters (17.3 inches) wide, and 41.4 centimeters (16.2 inches) deep.



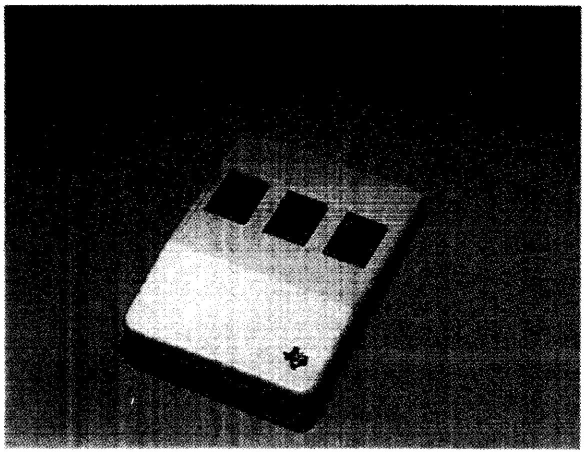
Keyboard

The low-profile keyboard contains 111 keys, including the standard alphanumeric keys, Lisp-specific keys, a numeric pad, and cursor control keys. The keyboard provides high-quality, snap-action, tactile feedback and silent operation. Software programmable features of the keyboard include:

- Key-click option to provide audio feedback if you desire.

- Auto-repeat option, which repeats the key function when you continue to hold a key down.

The keyboard is 3.5 centimeters (1.4 inches) high at minimum tilt (5 degrees), 52 centimeters (20.5 inches) wide, and 20.3 centimeters (8 inches) deep. You can adjust the slope anywhere between 5 degrees and 15 degrees for comfortable operation. The keyboard is attached to the monitor with a coiled cord, and stores under the monitor when not in use.



Mouse

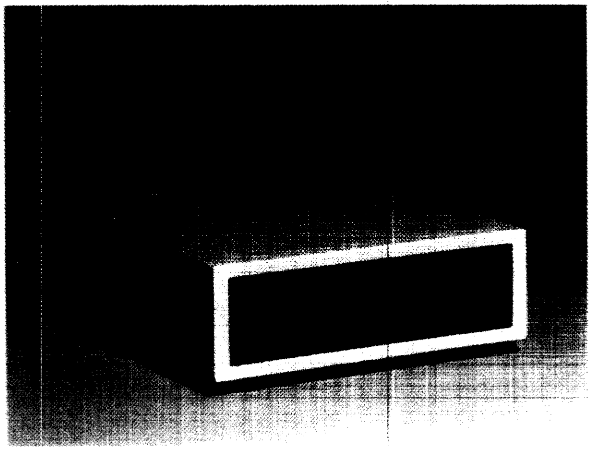
The mouse acts as a quick pointing device over the full range of the video display. As you move the mouse over a flat surface, a mouse cursor moves on the video display. The mouse tracks motion at rates up to 30 inches per second with a resolution of up to 320 events per inch. The mouse is designed for comfortable operation and has three buttons with which you can invoke operations without having to return your hand to the keyboard.

Mass Storage Enclosures

Three types of enclosures house mass storage devices separate from the main system enclosure—the mass storage unit (MSU); the Trimline disk cabinet; and the 1/2-inch tape enclosure. The MSU is designed for 5 1/4-inch form-factor disks and tape drives, and the Trimline disk cabinet houses larger capacity, half-rack SMD disks. Various combinations of these enclosures can be configured on an Explorer system to provide the mix and capacity of permanent data storage desired.

Each enclosure provides its own power supply. In addition, the MSU enclosures also report overtemperature and power interruption conditions to the system processor for data protection.

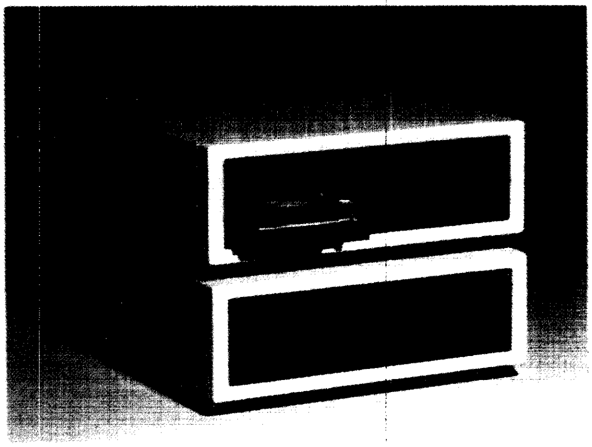
The MSU enclosures are designed to be stacked, either on top of the system enclosure (up to two) or on a desk or table. The 1/2-inch tape enclosure is designed for table-top use, and the Trimline disk cabinet is a free-standing enclosure that is typically placed on the floor next to the system enclosure.



5 1/4-inch Winchester Disks

Disk capacities start at 143 megabytes (unformatted). The average access times are 27 milliseconds or less, and the transfer rates are 5 megabits per second or faster.

The disk drives use a voice-coil actuator for lower power consumption and higher speed. The actuator is automatically locked on power-down for shipping and moving. Sealed disk units provide high reliability with no preventive maintenance.



1/4-inch Cartridge Tapes

The cartridge tapes provide low-cost disk backup and are the standard distribution medium for new software releases and patches. Each network of the Explorer systems requires at least one tape unit. The 0.6-centimeter (1/4-inch) tape cartridges are available in a 182.8-meter (600-foot) length with a 60-megabyte capacity. The tapes use the industry-standard QIC-24 format, making them interchangeable with other standard 60-megabyte tapes.

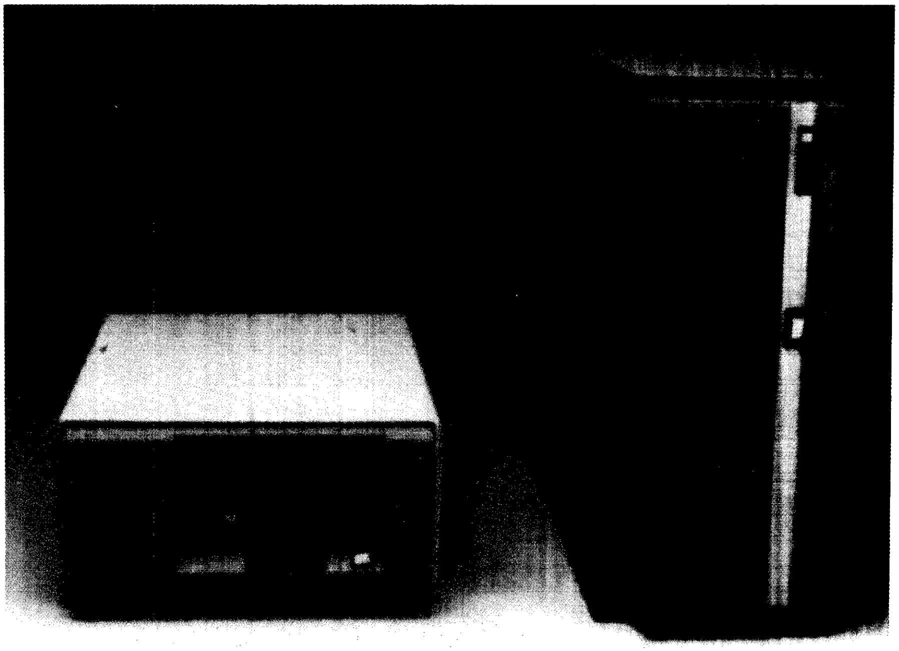
The tape drives provide 228-centimeter-per-second (90-inch-per-second) streaming at a 90-kilobyte-per-second transfer rate. Preventive maintenance for the tape drives is limited to a simple periodic head-cleaning procedure.

SMD Disks

The SMD Winchester disks provide high capacity and high performance for demanding applications. The disks feature 516-megabyte (unformatted) capacity with 18-millisecond access time and 15-megabits-per-second data transfer rate. The SMD disks are installed in a Trimline cabinet, up to two per cabinet.

1/2-inch Tape

The 1/2-inch tapes provide an interchange medium for transporting data between the Explorer systems and other types of computer systems, particularly traditional mainframes. The 1/2-inch reels provide up to 90-megabyte capacity in 3200 bits-per-inch mode, or 45-megabyte capacity in 1600 bits-per-inch mode. Internal caching optimizes performance on the Explorer systems to allow 100-inch-per-second streaming tape speed.



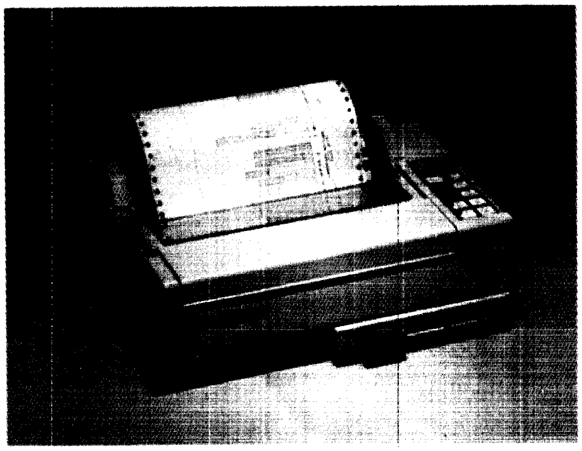
Printers

Several Printer options are available for Explorer systems. Printers can operate from either the serial port or the Centronics-compatible parallel printer port on the system interface board. You can attach a printer locally to your system, or you can direct output to a print-server station over a local area network (LAN). Printers currently supported by the Explorer systems include the following:

- Model 855 Printer
- OmniLaser™ 2015 Page Printer
- OmniLaser 2108/2115 Page Printer (with PostScript®)

These printers provide output for text files, graphics objects, and video display images.

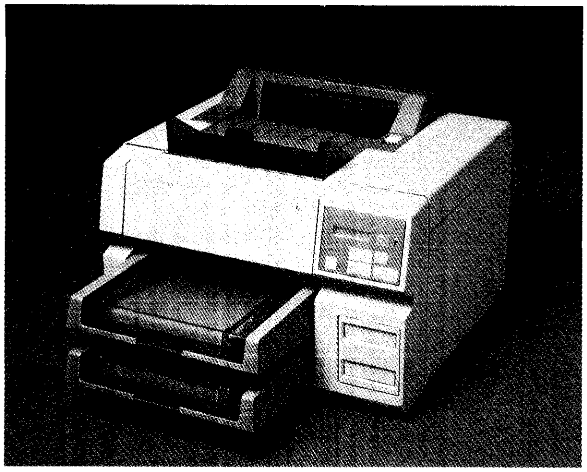
Additional printers are also supported; for details, contact your customer representative.

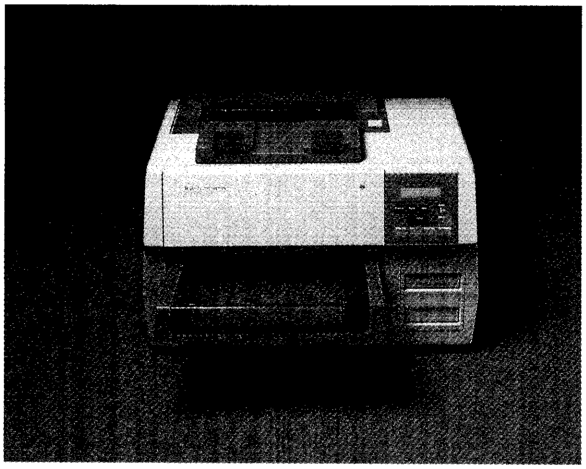


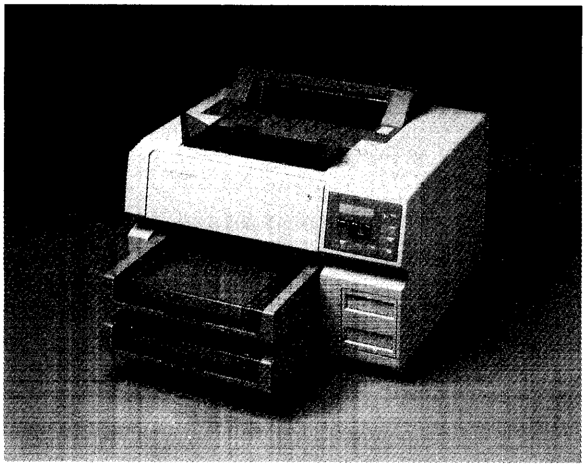
Model 855 Printer
The Model 855 Printer is a dot-matrix impact printer. Standard features include the following:

- Interchangeable font modules with choice of pitch and true descenders
- Bidirectional printing up to 150 characters per second
- Variable resolution up to 56.7 dots per inch (144 dots per inch)
- Mosaic graphics for borders, bar graphs, and patterns
- Raster graphics for pictures, graphs, and charts
- Friction paper feed
- 256-character buffer

Optional features include an adjustable-width tractor-drive assembly, paper-stacking tray for fanfold paper, paper-roll holder, and a 4000-character buffer.







OmniLaser 2015 Page Printer
The OmniLaser 2015 Page Printer is a second-generation laser printer designed for high duty-cycle, long-life use. Standard features include the following:

- Standard Courier, 10, 12, and 16.7 pitch fonts, plus optional plug-in font cartridges or downloaded fonts.
- Print speed up to 15 pages per minute
- 300x300 dots-per-inch resolution for graphics and screen dump outputs
- Standard RS-232C serial or Centronics parallel interface
- Emulation of TI 855 and other printers
- 512K memory
- Capacity to handle a variety of papers and forms with 500-sheet input/output capacity
- Quiet operation and user maintainability

OmniLaser 2108 and 2115 Page Printers
The OmniLaser 2108 and 2115 Page Printers are similar to the OmniLaser 2015, but include PostScript support. In addition to the features of the OmniLaser 2015, standard features of the OmniLaser 2108 and 2115 include the following:

- Standard Courier, Helvetica, Times Roman, and Symbol fonts on board plus plug-in cartridge fonts or downloaded fonts
- Font scaling under PostScript control
- Print speed up to 8 pages per minute (OmniLaser 2108) or up to 15 pages per minute (OmniLaser 2115)
- 2-megabyte memory (OmniLaser 2108) or 3-megabyte memory (OmniLaser 2115)
- 250-sheet input/output capacity (OmniLaser 2108) or 500-sheet capacity (OmniLaser 2115)
- Professional graphics and page layout capabilities with PostScript support (serial interface recommended for interactive development).

HARDWARE SPECIFICATIONS 10

Hardware System Architecture

Dual bus structure:

NuBus	32-bit general-purpose, multiprocessor bus 37.5-megabyte-per-second maximum bandwidth
Local bus	32-bit single-master, high-speed memory access port

Input/output:

SCSI	Industry-standard SCSI peripheral bus 1.5-megabyte-per-second maximum transfer rate MC68000 microprocessor-based controller
MSC	SMD disk interface, up to 3 megabytes-per-second Asynchronous SCSI I/O, up to 1.5 megabytes-per-second Synchronous SCSI I/O, up to 4 megabytes-per-second
CCB	8-channel RS-232C asynchronous serial I/O up to 19.2 kilobits-per-second Synchronous serial I/O up to 56 kilobits-per-second
Fiber-optic	68-megapixel-per-second transfer to display Dual 50-micron fiber-optic cable Also carries keyboard, mouse, and sound
Serial:	RS-232C synchronous and asynchronous communications to 19.2K baud
Parallel	Centronics-compatible parallel printer port
LAN	10-megabit-per-second Ethernet option

Explorer Processor

- Full 32-bit microprogrammed CPU
- 16K-word writable control store (56-bit microinstruction word)
- 7-MHz microinstruction cycle
- 4K-byte stack cache
- 128-megabyte virtual address space
- High-speed, demand-paged memory mapping hardware
- 1K-byte page size
- Microcode:
  - Device handling
  - Virtual memory management
  - Storage allocation and garbage collection
  - Lisp data object support
  - Lisp language kernel plus other functions
  - Graphics routines

Explorer II Processor

- 32-bit microprogrammed Explorer Lisp microprocessor
- 32K-word writable control store (63-bit microinstruction word)
- 4K-byte stack cache
- 128K-byte two-way set-associative cache memory
- 128-megabyte virtual address space
- High-speed, demand-paged memory mapping hardware
- Autotransport logic to follow tagged pointers in memory
- 1K-byte page size
- Microcode:
  - Device handling
  - Virtual memory management
  - Storage allocation and garbage collection
  - Lisp data object support
  - Lisp language kernel plus other functions
  - Graphics routines

Explorer Main Memory

- 4- and 8-megabyte boards (on local bus), up to 16 megabytes per system
- Less than 300-nanosecond access time; 800-nanosecond cycle time
- NuBus block (up to 16 words) transfers fully supported
- Byte parity error detection
- NuBus non-block transfers: 400-nanosecond read cycle time
300-nanosecond write cycle time
- NuBus block transfers: 300-nanosecond/word read and 200-nanosecond/word write cycle times following the first cycle
- Byte parity error detection

Explorer II Main Memory

- 8-, 16- and 32-megabyte boards providing up to the full 128-megabyte virtual address space on NuBus (up to 4 memory boards per system)
- NuBus non-block transfers: 500-nanosecond read cycle time
300-nanosecond write cycle time
- NuBus block transfers: 100-nanosecond/word cycle times following the first cycle for both reads and writes
- EDAC logic that corrects single-bit errors, detects double-bit errors

Explorer LX Processor

- 68020 microprocessor, operating at 16.7 MHz
- 68881 floating-point coprocessor operating at 16.7 MHz
- 68851 paged-memory management unit
- Virtual memory, addressable to 4 gigabytes
- 16-kilobyte cache memory for logical address, data, and instructions
- Main memory: 2 megabytes of error-correcting DRAM with an optional 2 megabytes of DRAM in a piggy-back board
- Three timers for a real-time clock, interval timing, and time accounting
- NVRAM for system configuration support
- ROM for board information and self-test

Odyssey Coprocessor

- Four TMS32020 processor modules:
  - TMS32020 processor (200-nanosecond instruction cycle time)
  - 64K x 16-bit words of DRAM for data memory
  - 8K x 16-bit words of high-speed static RAM for program memory
  - Control logic to access the Odyssey internal bus
- NuBus interface controller with 8K x 16-bit words of program memory and 2K x 16-bit words of program ROM
- Internal bus that is a modified NuBus
- Communication/control module for a two-way data path to the NuBus

Mass Storage

- Small form-factor mass storage unit (MSU):
  - Enclosure that holds two 5 1/4-inch devices
  - Up to four enclosures connected via one SCSI bus
  - SCSI transfers that average 1 megabyte-per-second
  - 5 1/4-inch Winchester disks:
    - 143-megabyte unformatted, 112-megabyte formatted capacity per drive
    - 27-millisecond average seek time
    - 5-megabit-per-second transfer rate
    - 182-megabyte unformatted, 155-megabyte formatted capacity per drive
    - 19-millisecond average seek time
    - 10-megabit-per-second transfer rate
  - Cartridge tapes:
    - 1/4-inch cartridge tape
    - 60-megabyte cartridge tape capacity
    - Streaming capability at 228.6 centimeters (90 inches) per second
    - 90K-byte-per-second transfer rate
  - 1/2-inch reel tapes:
    - Table-top unit that connects to SCSI bus
    - 3200 BPI (90MB) and 1600 BPI (45MB) modes
    - Streams of 100 inches-per-second in 1600 BPI mode
    - 160K-byte-per-second drive transfer rate
    - 1M-byte-per-second SCSI transfer rate
- SMD disks:
  - Up to two SMD disks per Trimline disk cabinet
  - 516-megabyte unformatted, 440-megabyte formatted capacity per drive
  - 18-millisecond average seek time
  - 15 megabit-per-second transfer rate

Monitor

43.2-centimeter (17-inch) diagonal monitor mounted in landscape orientation

60-hertz refresh, noninterlaced raster

Monochrome display

Single-plane, 1024-by-1024-pixel bit map with 1024 by 808 pixels displayed

Ergonomic tilt, height, and swivel adjustment

Less than 300-nanosecond bit map access and cycle time

Read-modify-write with logical operations in a single cycle

Currently supported cable lengths of 15.2 and 60.9 meters (50 and 200 feet)

Keyboard

111 keys, including Lisp-specific and cursor-control keys

Ergonomic, low-profile design with continuous precision tilt between 5 and 15 degrees

High-quality tactile snap-action key switches with selectable audio feedback

Flexible coiled cord that connects directly to the front of the monitor

Mouse

Padless mouse with new anti-static mechanical tracking and optical encoding for high reliability

320-events-per-inch resolution, 30-inch-per-second tracking

Ergonomically designed case with three buttons

Cable that connects directly to the front of the monitor

Miscellaneous Features

Battery-powered time and date clock

2K-byte nonvolatile RAM

Efficient 675-watt system power supply

Overtemperature warning and protection feature

Size and Weight

System enclosure — 63.5 centimeters (25 inches) high, 33 centimeters (13 inches) wide, 45.7 centimeters (18 inches) deep; 27.2 kilograms (60 pounds)

Mass storage enclosure — 13.2 centimeters (5.2 inches) high, 33 centimeters (13 inches) wide, 38.6 centimeters (15.2 inches) deep;14.9 kilograms (33 pounds)

1/2-inch Tape — 241 centimeters (9.5 inches) high; 508 centimeters (20 inches) wide; 699 centimeters (27.5 inches) deep; 41 kilograms (90 pounds)

Trimline disk cabinet — 747 centimeters (29.4 inches) high; 273 centimeters (10.73 inches) wide; 819 centimeters (32.25 inches) deep; 91 kilograms (201 pounds) one disk; 131 kilograms (288 pounds) two disks

Monitor — 39.8 centimeters (15.7 inches) high, 43.9 centimeters (17.3 inches) wide, 41 centimeters (16.2 inches) deep; 18.1 kilograms (40 pounds)

Keyboard — 3.5 centimeters (1.4 inches) high, 52 centimeters (20.5 inches) wide, 20.3 centimeters (8 inches) deep

Environmental Specifications

Meets FCC level A, VDE level A, UL-478, IEC 380, 435, and CSA 22.2 No. 134 standards

Operating temperature range — 10 to 35 degrees Celsius (50 to 95 degrees Fahrenheit)

Noise level — 50 dBA SPL for each enclosure

Input voltage range — 102 to 132, 187 to 264 Vac

Operating power requirements (47 to 63 hertz;
with a 1/2-inch tape drive, 49 to 61 hertz)

	120 Vac	**220 Vac**	**240 Vac**
System enclosure	12. amps	6.5 amps	6.0 amps
Mass storage enclosure	2.0 amps	1.0 amp	1.0 amp
1/2-inch tape	1.7 amps	.85 amps	.85 amps
Trimline disk cabinet (2 disks)	8.0 amps	4.6 amps	4.6 amps
Monitor enclosure	1.7 amps	0.9 amps	0.9 amps

SOFTWARE MANUALS

Mastering the Explorer Environment	Explorer Technical Summary	2243189-0001
	Introduction to the Explorer System	2243190-0001
	Explorer Zmacs Editor Tutorial	2243191-0001
	Explorer Glossary	2243134-0001
	Explorer Networking Reference	2243206-0001
	Explorer Diagnostics	2533554-0001
	Explorer Master Index to Software Manuals	2243198-0001
	Explorer System Software Installation Guide	2243205-0001
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
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 Prolog User's Guide	2537248-0001
	Programming in Prolog, by Clocksin and Mellish	2537157-0001
	Explorer Color Graphics User's Guide	2537157-0001
	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 DECnet User's Guide	2537223-0001
	Personal Consultant Plus Explorer	2537259-0001
System Software Internals	Explorer System Software Design Notes	2243208-0001
	Release Information, Explorer System Software	2549844-0001

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
	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 System Interface General Description	2243145-0001
	Explorer NuBus Peripheral Interface General Description (NUPI board)	2243146-0001
Display Terminal Publications	Explorer Display Unit General Description	2243151-0001
	CRT Data Display Service Manual, Panasonic code number FTD85055057C	2537139-0001
	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
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
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
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	2539348-0001
	OmniLaser 2108 Page Printer Technical Reference	2539349-0001
	OmniLaser 2108 Page Printer Maintenance Manual	2539350-0001
	OmniLaser 2115 Page Printer Operator's Manual	2539344-0001
	OmniLaser 2115 Page Printer Technical Reference	2539345-0001
	OmniLaser 2115 Page Printer Maintenance Manual	2539356-0001

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

ACRONYMS

AI	artificial intelligence
ARP	Address Resolution Protocol
CCB	communications carrier board
CPU	central processing unit
DMA	direct memory access
DRAM	dynamic random-access memory
DSP	digital signal processing
EDAC	error detection and correction
FTP	File Transfer Protocol
I/O	input/output
IP	Internet Protocol
IPC	Interprocess Communication
KE	knowledge engineer
LAN	local area network
LED	light-emitting diode
MSC	mass storage controller
MSU	mass storage unit
NFS	Network File Server
NLMenu	Natural Language Menu
NUPI	NuBus Peripheral Interface
PC Plus	Personal Consultant Plus
PCA	peripheral cable adapter
RAM	random access memory
REST	rule-based expert system tool for TI Prolog
ROM	read-only memory
RPC	Remote Procedure Call
RTMS	Relational Table Management System
SCSI	small computer system interface
SDMA	shared direct memory access
SMD	storage module drive
SNA	System Network Architecture
TCP	Transmission Control Protocol
UCL	Universal Command Loop
VDT	video display terminal
VLSI	very large-scale integration
VMA	virtual memory address
XDR	External Data Representation