64000

HP64000 Logic Development System

Model 64100AF/AT System Overview

HEWLETT PACKARD

CERTIFICATION

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Bureau of Standards, to the extent allowed by the Bureau's calibration facility, and to the calibration facilities of other International Standards Organization members.

WARRANTY

This Hewlett-Packard system product is warranted against defects in materials and workmanship for a period of 90 days from date of installation. During the warranty period, HP will, at its options, either repair or replace products which prove to be defective.

Warranty service of this product will be performed at Buyer's facility at no charge within HP service travel areas. Outside HP service travel areas, warranty service will be performed at Buyer's facility only upon HP's prior agreement and Buyer shall pay HP's round trip travel expenses. In all other cases, products must be returned to a service facility designated by HP.

For products returned to HP for warranty service. Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country.

HP warrants that its software and firmware designated by HP for use with an instrument will execute its programming instructions when properly installed on that instrument. HP does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HP SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HP SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

ASSISTANCE

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales and Service Office.

CW&A 2/81

Printing History

Each new edition of this manual incorporates all material updated since the previous edition. Manual change sheets are issued between editions, allowing you to correct or insert information in the current edition.

The part number on the back cover changes only when each new edition is published. Minor corrections or additions may be made as the manual is reprinted between editions.

First Printing.....February 1982

System Overview

Table of Contents

Table of Contents (Cont'd)

Chapter 5: 64000 System Modules

List of Illustrations

Chapter 1
General Information

Introduction

The HP 64000 Logic Development System is designed to aid both software and hardware development by providing the capabilities for emulation, analysis, and debugging. In short, it is a totally integrated solution to the task of microprocessor-based design.

Its unique architecture makes the system unexcelled in the way it integrates the emulator and/or analyzer with software development while providing both control and a window to observe system operation. During every task, from editing to assembling, linking, emulating, and analysis, the 64000 operates with the speed that significantly improves designer productivity.

A feature of the 64000 System is the concept of localized, rather than centralized, architecture. Each station has a powerful HP-developed, 16-bit dedicated processor. If you choose a networked configuration, as more stations are added, so is processing power. The advantage is that system response time becomes a function of the task at hand and is not determined by the system's workload, as is the case in a time-sharing situation. As a result, development time is accelerated. A designer can perform emulation at one station, while a programmer debugs software at another. Software and hardware development may proceed in parallel because designers can perform different tasks simultaneously, while sharing a common data base. You benefit from smoother running projects and faster more efficient microprocessor development.

Because up to six development stations can share system peripherals, it is much easier to justify higher performance units than when each user has a dedicated set. Users enjoy not only the benefits of high-performance peripherals, but also the ability to develop software jointly, sharing the same data base.

Because the microprocessor is only one piece of a complete system, it represents a software design situation unlike most computer systems. The processor is usually an integral part of some hardware that has nothing to do with computation. In some cases it is simply being used either as a programmable logic element or for control of the human interface with some process. These differences make the conventional tools for generating and debugging hardware and software incomplete for the task facing the microprocessor system designer. The 64000 Logic Development System was meant to provide a complete solution to this task in one package, and to make significant contributions to the efficiency of designer's time.

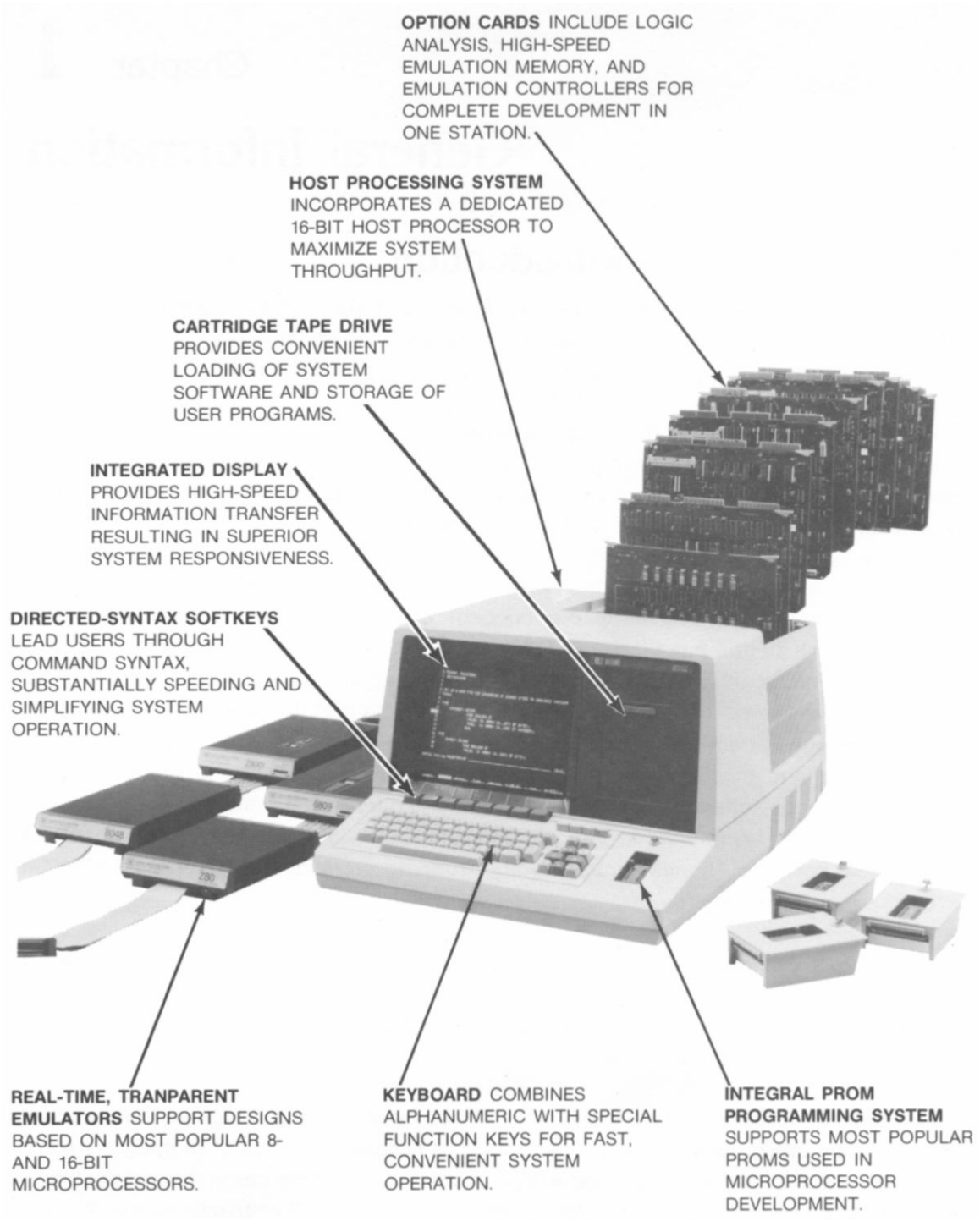


Figure 1-1. The 64000 Logic Development System

Chapter 2
System Configuration

Introduction

The HP 64000 Logic Development System can range in size from a single, stand-alone, development station with flexible disc-based storage and software, all the way up to a clustered system with six development stations, a hard disc, and printer. A wide selection of software packages, emulation, and analysis options are available for any configuration to provide complete design capability for virtually any microprocessor-based system.

The minimum 64000 System consists of the Model 64110A Mainframe with application modules in a stand-alone configuration. This configuration is designed primarily for those customers who want emulation, software development, or software and hardware analysis in a portable, programmable system along with the capability to expand by adding various options to their systems.

The cluster-based development system typically consists of at least one development station with a local mass-storage unit installed, compatible HP hard disc and printer. Adding emulator options with independent emulation memory adds the download function through emulation, which is the standard tool for exercising, debugging, and integrating hardware and software in the early development phases. Assistance in troubleshooting the target system is gained by adding state and timing analysis. As program modules are completed, they may be mapped into the target system's random-access memory, or with the PROM Programming System, they can be downloaded into one of many widely used programmable read-only memories (PROMS). The system may be expanded to accomodate larger design teams or multiple design efforts by adding up to five more development stations (see figure 2-1). For details, refer to the Installation and Configuration Manual.

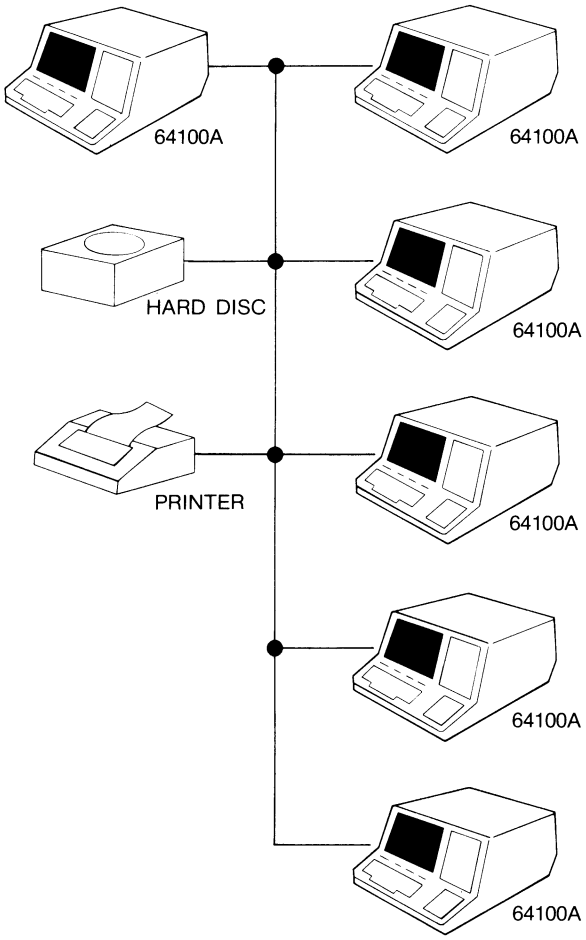
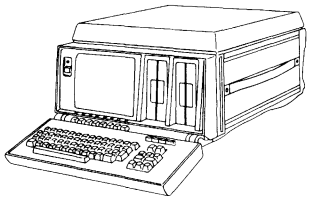


Figure 2-1. Cluster 64000 System

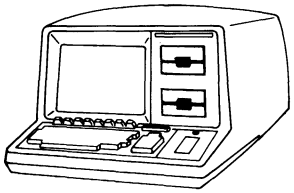
Development Stations

There are two development stations (Models 64100A and 64110A) which may be configured either in a cluster or as stand-alone stations depending on the particular design and/or analysis needs. The 64110A has an integral, dual, 5 1/4-inch flexible disc unit installed as standard equipment. The 64100A has an optional flexible disc or a tape cartridge unit for local mass storage (see figure 2-2). To operate stand-alone, the 64100A must have the flexible-disc option installed. Any combination of the two stations (with any combination of local mass storage units in the 64100A) may be configured in a cluster. However, the cluster is limited to a maximum of six stations, one of which must have local mass storage to load system software.

Integral dual flexible disc unit for local mass storage. Stand-alone or cluster configuration.



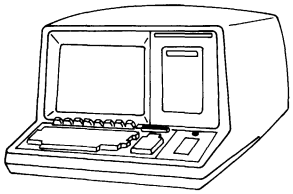
HP 64110A



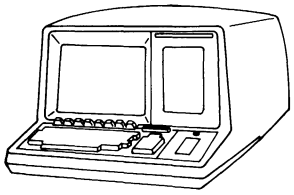
HP 64100A

Integral dual flexible disc unit for local mass storage. Stand-alone or cluster configuration.

Tape unit for local mass storage. Cluster configuration.



HP 64100A



HP 64100A

Add-on station for cluster configuration.

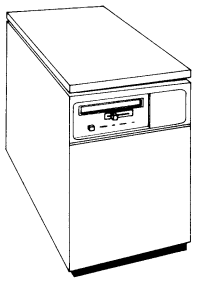
Figure 2-2. 64000 System Development Stations

Cluster Configuration

A typical 64000 cluster system consists of three 64100A development stations, each with a local mass storage device installed. The system would also include the HP 2631B Printer and an HP 7908P Disc Drive that provides 16.5 megabytes of disc-based storage plus an integral tape cartridge that provides 16.7 megabytes of backup capability. Selection of system peripherals, however, is not limited to a particular disc or printer. The 64000 System supports a wide range of HP hard discs and hard-copy devices to provide the specific performance and capacity that the design environment requires.

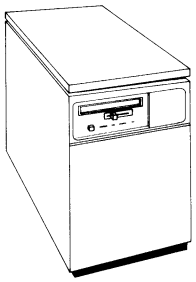
Optional disc drives are shown in figure 2-3. Multiple disc drives may be configured on the same cluster for increased storage capacity. (Certain configurations may require less than six development stations on the System Bus.)

Printer options are shown in figure 2-4. Depending on the required performance and hard-copy load, the choice ranges from a high-speed line printer to a lower-cost thermal printer.



HP 7911P

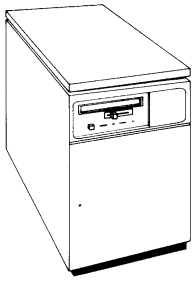
16.5-megabyte Winchester with tape backup.



HP 7908P

27-megabyte Winchester with tape backup.

64-megabyte Winchester with tape backup.



HP7912P

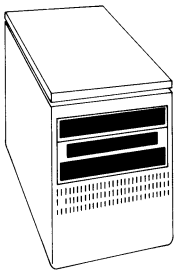
NOTE

The discs shown above may be mixed on a single system. (Each disc takes up one of eight System Bus addresses.) These discs may not be mixed with the discs shown in figure 2-3B.

Figure 2-3A. 64000 System Disc Drives

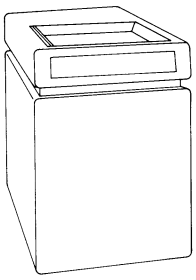
NOTE

The discs shown below may be mixed on a single system. (Discs may share a common controller which takes up only one System Bus address.) These discs may not be mixed with the discs shown in figure 2-3A.



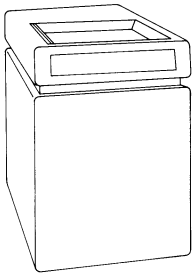
HP 7906M

20-megabyte with removable (10-megabyte) cartridge.



HP 7920M

50-megabyte with removable disc-pac.

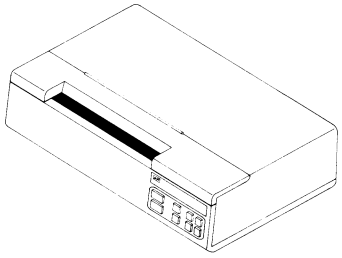


HP 7925M

120-megabyte with removable disc-pac.

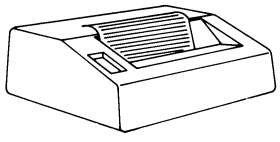
Figure 2-3B. 64000 System Disc Drives

120-cps thermal graphics printer.

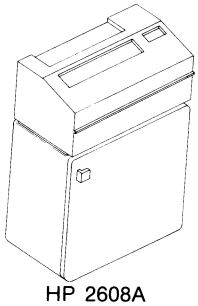


HP 2673A

180-cps dot-matrix printer.



HP 2631B



HP 2608A

400-lpm line printer.

Figure 2-4. 64000 System Printers

Stand-alone Configuration

When configured as a stand-alone unit, a development station provides the full range of design and analysis capabilities available in a clustered configuration. The portability and lower cost of a stand-alone configuration places it in many environments inappropriate for a cluster 64000 System. For example, analysis activities may be carried out in a field-service situation, emulation debugging may be done at a remote site, or software development may be finished up at a designer's home.

The fact that the physically identical station (without any hardware modifications) may operate in a cluster, as well as a stand-alone, provides for flexible and creative use of the 64000 system. Various processes may be done on a stand-alone station with all of the data and measurements fully available and executable on that station (or another station) configured in a cluster.

The integral HP-IB port allows a stand-alone station to function as an instrument in an HP-IB configuration (see figure 2-5). For example, a controller may be connected to initiate analysis in the station, and to post-process the data upon completion. All of this may be done unattended. The station may also operate as a "talk only" device on the HP-IB, providing the capability to dump data directly to a printer without the need for a controller.

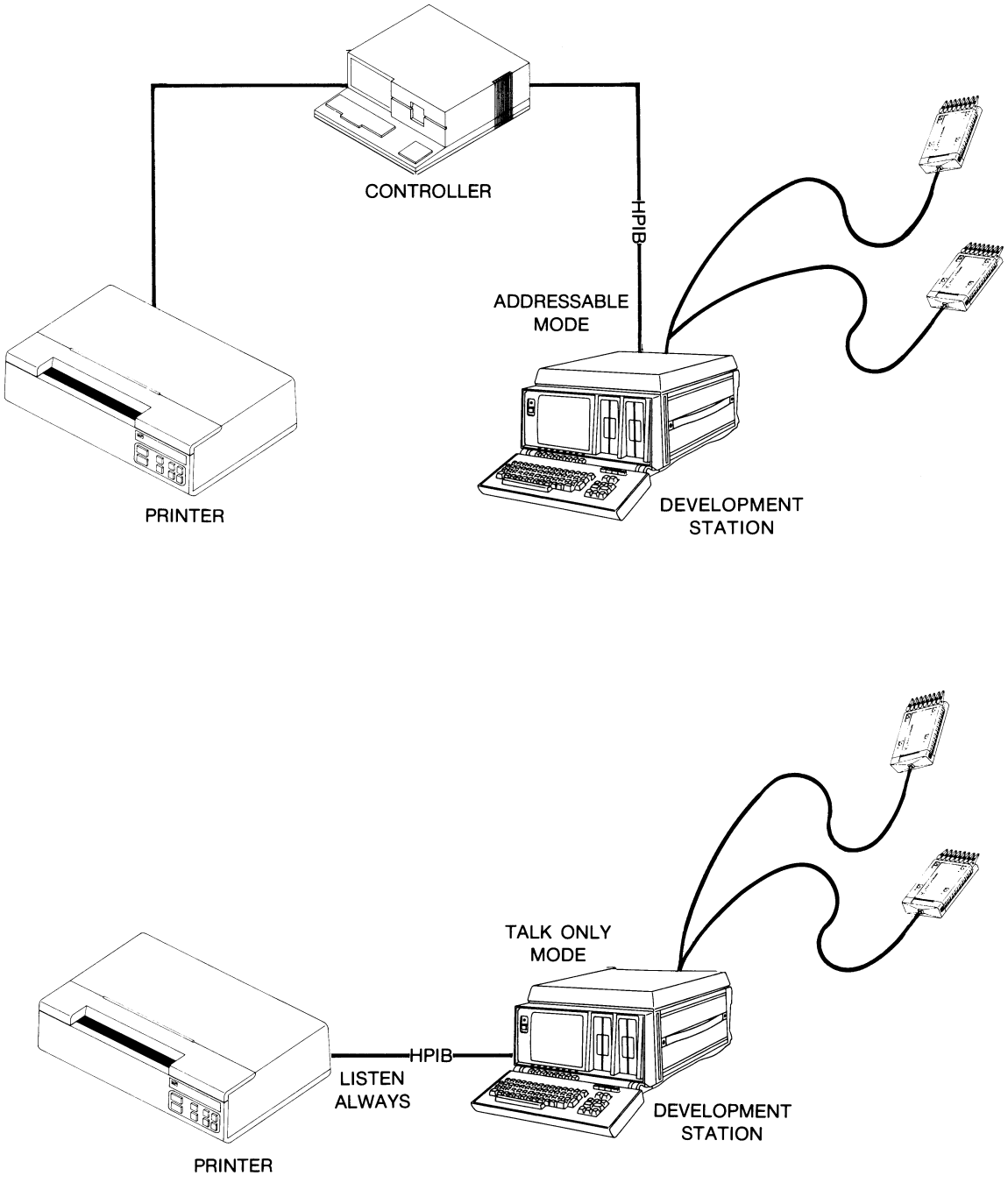


Figure 2-5. Stand-alone Configurations

RS-232-C

The 64000 System provides for communication to external devices. Each station contains an integral RS-232-C interface. This capability enhances both clustered and stand-alone configurations.

Typical uses for a cluster include the link of multiple clusters. This may be done either directly between two stations on different clusters, or through the common use of a central computer (see figure 2-6). Large design teams requiring access to a shared data base would find uses for connected clusters. Use of the terminal mode (provided with each 64000 Operating System) allows a station to function transparently as a terminal to the interfaced computer. With such a network, a designer quickly turns his lab-bench station into a complete work station for computing activity.

Stand-alone stations may likewise be networked to a central computer to exchange data or act as terminals. Remote data acquisition may be performed with the data transferred back via a modem or direct link. All of this may be done unattended at the computer location (see figure 2-7).

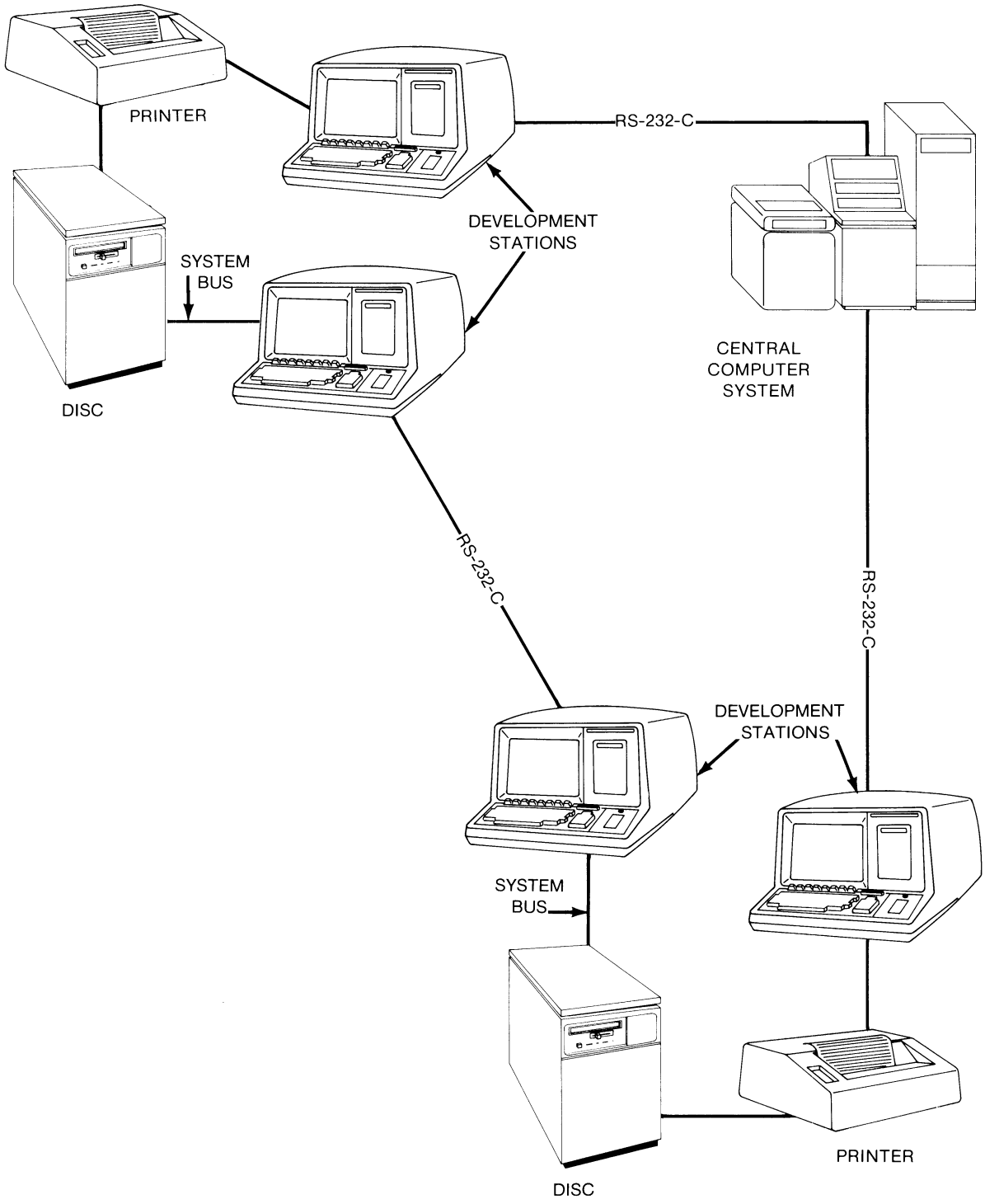


Figure 2-6. RS-232-C Linking 64000 Clusters

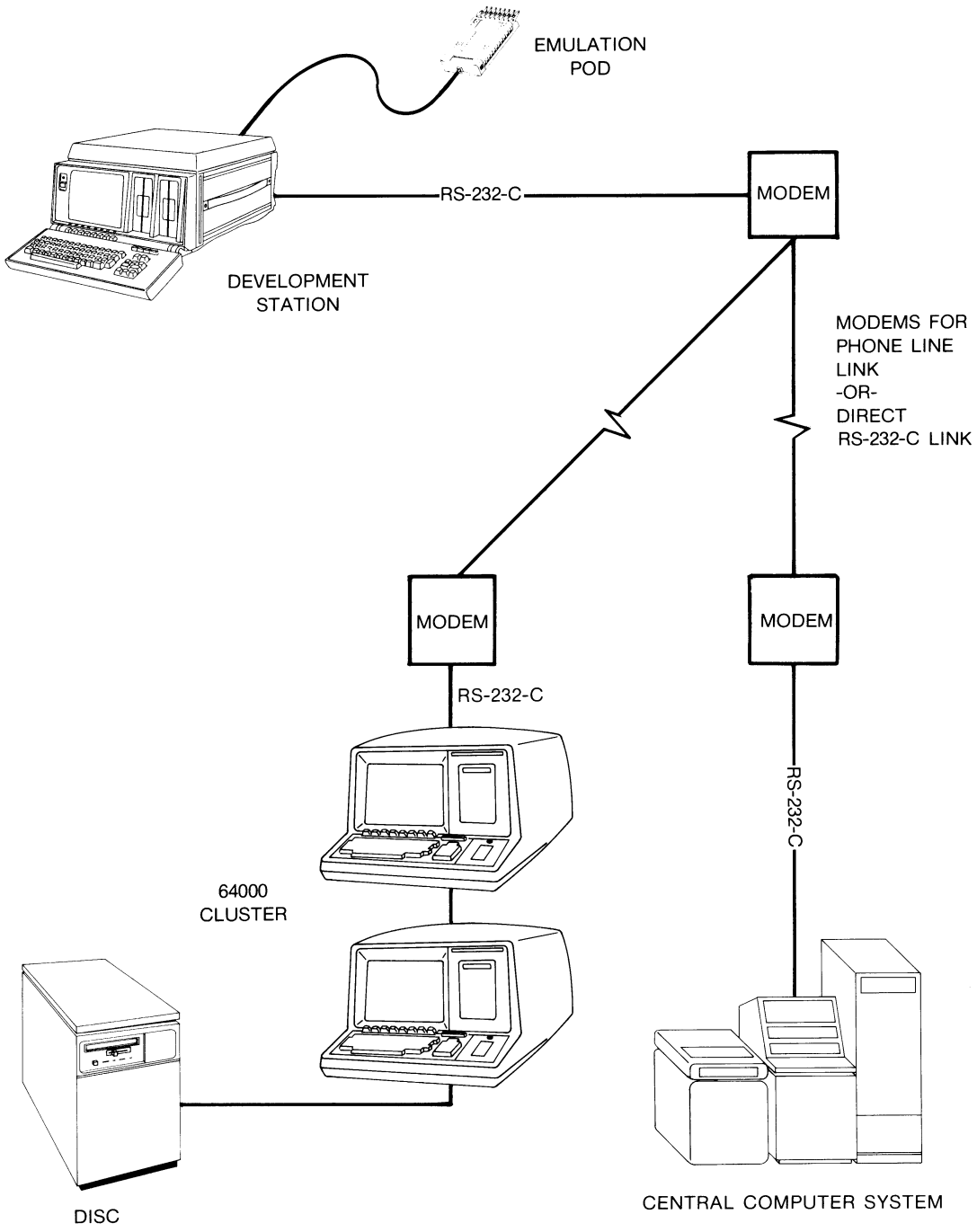


Figure 2-7. RS-232-C Linking a Remote Development Station

Chapter 3
System Architecture

Development Stations

The development station keyboard and display provide the interface between the operator and the logic development system. Operating systems, input/output, keyboard, display, and the development station bus are managed by the independent host processor and memory.

A choice of two development stations exists in the 64000 System: the Model 64100A and the Model 64110A. The architecture of both models is essentially the same with the following differences:

Differences	**64100A**	**64110A**
packaging	bench-top	portable
power supply	400W	150W
option card slots	10	5
host system	3 cards	2 cards

The easily-accessed card cage has slots to house the circuitry for the various system options. The first slots of the card cage are occupied by the cards of the host system, leaving the remaining slots available for option cards. The development station bus is universal, and options may be placed in any slot. The development station bus carries address, data, and control signals between the host processor system and option card positions.

Each option card can identify itself to the host processor. Thus the option software is self-configuring. Communication with the options is via a 32k-byte memory address space window. When a card is addressed by the host, one of three bank switch modes is also set, thereby expanding this window to an effective 96k bytes per option card.

Figure 3-1 shows a map of the entire 138k-byte address space of the host processor including the 32k-byte window. The display memory is an integral part of the program RAM, making possible the rapid display update required for such things as tracking softkeys and a screen-mode editor. The ROM space in the system is used for the bootstrap programs, for some frequently used utilities, and for the mainframe self-test software. In the current version of the 64000 System, 16k bytes of ROM are used. All of the operating software resides in the RAM area and is segmented so that only code for the current task is in memory.

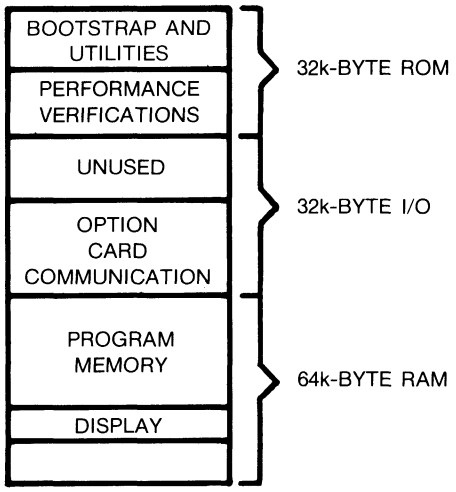
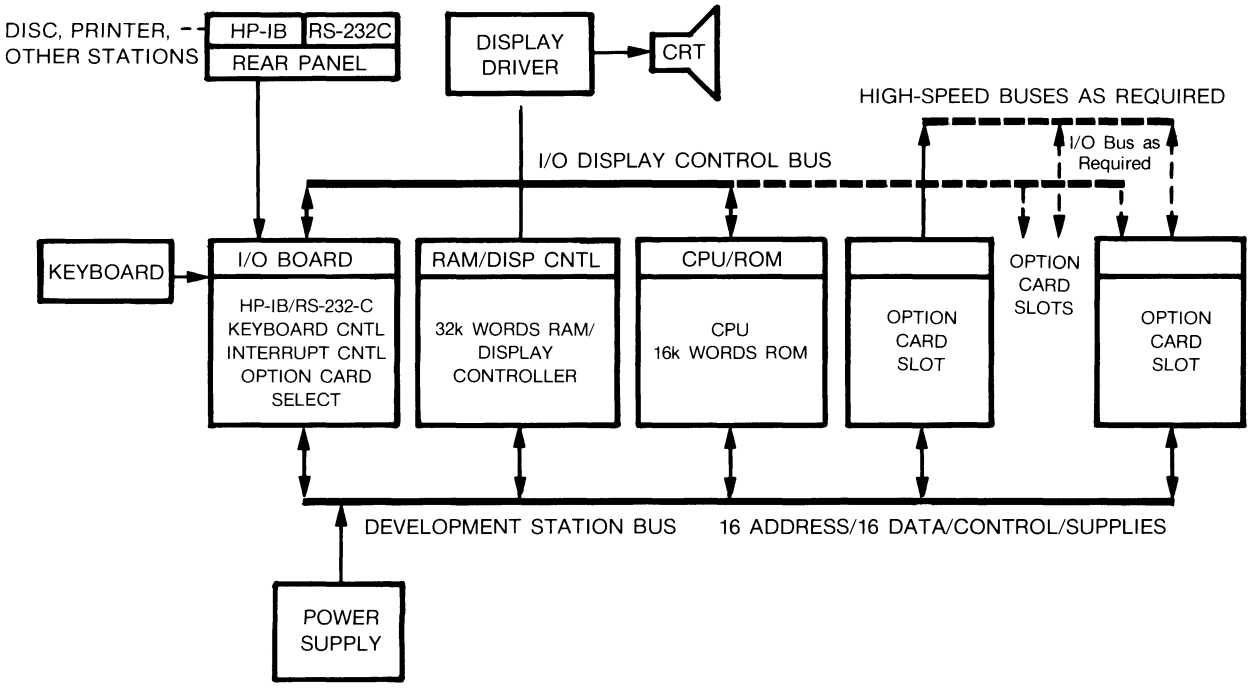


Figure 3-1. Host Processor Memory Map

High-speed buses (isolated from the mainframe bus) provide the communication and control link within subsystems and between subsystems (see figure 3-2). For example, the emulation subsystem uses a separate emulation bus between emulation control, emulation memory control, and analysis cards. A second high-speed bus connects the emulation memory to the memory controller. This complete isolation from the host environment allows passive monitoring of the execution of software without stopping the process. Because of this separation, it is also possible to continue emulation or analysis while software development is occurring on the same station, thus potentially doubling the use. The two buses are so independent that the prototype containing the emulator can be powered down and then up without affecting the host system. Even the data stored in the emulation memory remains unchanged, and the processor simply goes through its normal power-up sequence.



NOTE: This diagram illustrates the 64100A. The 64110A is physically different, but functionally equivalent.

Figure 3-2. Bus Architecture for 64100A

Another important benefit of this subsystem architecture is the future expandability of option modules. In terms of emulation, the host processor system puts no restrictions on the speed or word length of the processor being emulated. Future microprocessors will certainly be faster and more powerful, so it is important to allow for this to preserve the capital investment in the development system. Likewise, the analysis and PROM programming options are not limited.

Operation

At power-up the host processor interrogates a rear-panel switch to determine the ROM program to execute. There are several selectable modes: System Bus, local mass storage, and performance verification. The performance verification mode exercises all mainframe hardware, including the memory, flexible disc or tape drive, RS-232-C port, and HP-IB interface. The other modes are bootstrap programs. The normal mode of operation is to boot from either the hard disc on the System Bus or locally from the floppies. If booting from the hard disc, the program that is loaded performs a poll to determine all of the devices on the bus, configures the software I/O drivers based on that poll, and displays a system map. After booting from bus or local disc, eight softkey labels are displayed at the bottom of the display indicating the various functions available. The system is now awaiting a command and a status message indicates that state. To perform an assembly of a source file, for example, the softkey labeled "assemble" is pressed, followed by the name of the file to be assembled. The editor, compiler, and linker all use this same softkey-directed syntax.

Software

Just as important as the hardware architecture in a complete solution is the software package. Available software products include: operating system software (including file management, terminal mode, and text editor), assemblers, compilers, emulation software, analysis software, and PROM programmer software.

Since users of the system can range everywhere from the expert digital software designer to one with no previous software experience, the 64000 System is designed to provide considerable capability for the experienced software designer, and through the use of the directed-syntax softkeys, to give the new user access to the full capability of the system, not just the subset that is frequently used and remembered. To further enhance the convenience of the system, an effort was made to provide a uniform syntax and feature set in all aspects of the development tools. The rules for variable names are the same for the assembler, compiler, linker, and emulator. The feature set for all of the above modules also remains the same for each microprocessor so that the learning time for a new processor is much shorter. In some cases the same person has to work with more than one processor type simultaneously, so this approach becomes essential to reduce confusion.

With these features combined with the performance of a 16-bit processor per user and a high-speed hard disc, the turnaround cycle for changes is substantially reduced. As an example, it is possible to edit a file to make corrections, assemble that file, link it to other modules, and then execute the new code on the emulator in one minute. This level of performance encourages proper maintenance of source programs instead of memory patching to fix a problem.

File Management

The heart of all modern software development tools is the file management system. While automatic space allocation is a part of almost all systems, in the 64000 System this facility is significantly expanded to include the ability to recover accidently purged files or previous copies of an edited file up to the time when the space is needed for new files. A further enhancement aimed at managing the increased number of files being used is the user identification added to file names. By entering a user ID at the beginning of a session, all operations will be carried out on files under that name. The directory list defaults to listing only the files under that ID.

Resource Sharing in the 64000 System

Development stations in the 64000 System may be operated in either stand-alone or cluster configurations (see Chapter 2). Emulation, analysis and software modules are available for either configuration. A cluster system consists, at a minimum, of one development station, a disc memory, and a local mass storage unit. A maximum of six development stations, a printer, and eight disc drives can be connected on the system bus. The resource sharing provided by this configuration is very advantageous in most design environments.

The operating system software executing in the host processor of each station is implemented as a single-tasking system, responding to its keyboard inputs independently of any other station, except when two or more stations require access to a shared resource simultaneously (e.g., a disc memory or the printer). The use of these shared resources must be coordinated. The sharing protocol is simple, minimizing overhead in the operating system and reducing the number of operations that must be recovered in case of a system fault. Specifically, the shared resources are:

1. Access to a disc memory (excludes directory)
2. Access to read or modify a disc directory
3. Access to the printer.

The mechanical and electrical protocol used on the 64000 System Bus is derived from the HP Interface Bus or HP-IB (IEEE Standard 488-1978). However, in the 64000 System cluster context, messages are restricted to those needed for system operation. For example, I/O drivers and message protocols that would allow direct user control of interface message generation are not available. Therefore, only supported disc memories, printers and other stations may be connected to a development station which is installed in a cluster configuration.

The HP-IB standard was selected because of the existence of compatible disc memories and printers and a related family of reliable components (integrated interface electronics, connectors, and cables).

Each station can operate on the System as an active controller, talker, or listener. The current active controller monitors the state of the network --- that is, which stations are using or are waiting to use a shared resource. The active controller has the exclusive right to use the System Bus until control is passed to another station. However, a resource reserved by another station may not be used. Disc accesses not involving a disc directory access may be made by the active controller without restriction. Directory and printer accesses are the only two resources that must be reserved. Use of these resources is regulated by queues resident in the active controller for each function.

On each 64000 System, one and only one station is designated as master controller. This unit is responsible for initiating system activity by becoming the first active controller when the system is powered-on. Only this unit may assert the Interface Clear message, and therefore it is responsible for restarting a system that has experienced a partial power failure or a disruptive hardware or software fault.

This architecture makes it easy to change the number of development stations, the number and/or type of disc drives, and the printer. To effect a change, the system is powered off, reconnected and powered back on. No user-directed change in software is needed.

Fault Recovery

Recovery features have been implemented to lessen the effects of system faults. For example, it would be undesirable if low power on one station aborted an edit session on another station. All I/O operations have time-outs assigned, with appropriate recovery procedures in the event of malfunction. Disc operations that can't complete are retried. If a pass of control doesn't complete within the allotted time, the process is aborted and the previous active controller resumes control status.

Chapter 4
System Software and File Management

Introduction

The 64000 software, currently available, includes the following products (available on both DC100 tape cartridges and 5 1/4-inch flexible discs).

- Operating System Software
  - system monitor
  - editor
  - file manager
  - terminal mode
- PROM Programmer
- Assemblers/Linkers
- Compilers
- Emulator Software
- State Analyzer Software
- Timing Analyzer Software

The editor, assembler, compiler, linker, emulator, state analyzer, timing analyzer, PROM programmer, and terminal mode are covered briefly in this manual in Chapter 5. Each has a dedicated manual. Refer to the Manual Map at the front of this manual for the specific manual name.

Display Format

The Model 64000 software package is oriented around the use of the development station CRT display as the sole system control device. The display is divided into four areas as shown in figure 4-1.

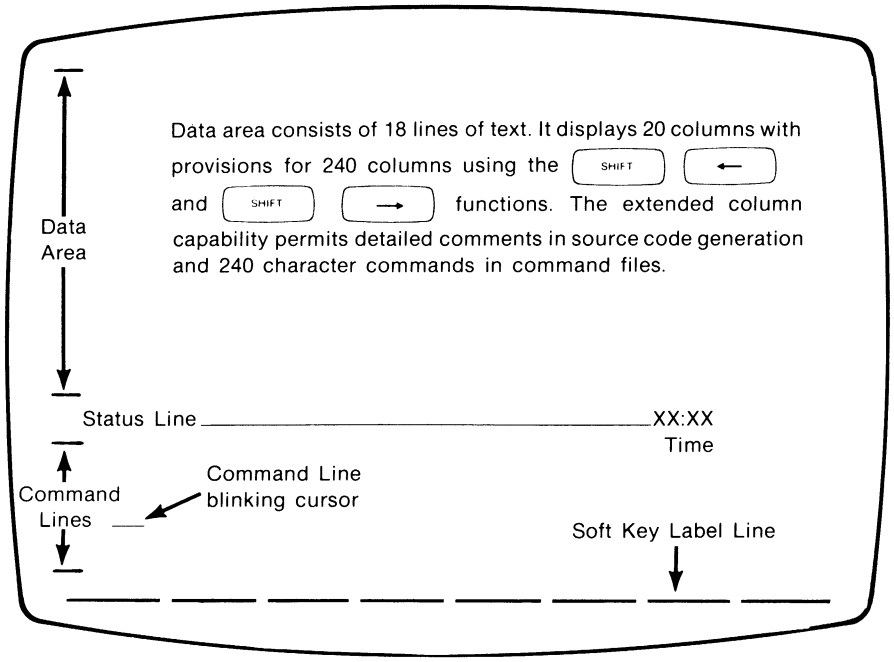


Figure 4-1. CRT Display

The status line displays system status, explanatory messages, error messages, and time.

The command area consists of a line that wraps across three screen lines. Entered commands and interrogation responses are displayed in this line.

A blinking underline cursor is present in the top left-hand corner of the command line, just under the status line. The cursor indicates the current character position on the line.

Directed Syntax Softkeys

Eight unmarked keys immediately below the CRT are labeled by the softkey label line just below the command line. These first-level softkeys (see figure 4-2) indicate the complete set of allowable entries and they change with each key stroke to reflect the next expected keyword or data in a command. If the user enters only that information prompted by the softkeys, the syntax will always be correct. Conversely, any entry not shown in the softkey labels will result in a syntactical error.

NOTE

It is also possible to type the commands (in lower-case characters) from the keyboard. This provides an alternative to the softkeys for the touch typist.

Figure 4-2. Typical First-level Softkeys

To further enhance the use of the system, a uniform syntax/feature set is provided. For example, numeric constants can be specified in decimal, hexadecimal, octal, or binary in the assembler, compiler, linker, emulator, monitor, and editor.

As each softkey is pressed, the cursor moves to the right providing reference for the inputs.

If an error is made on the input, the cursor can be backspaced to the point of error. As the cursor moves back, the softkey label line information changes presenting the syntax choices available. If the cursor is backspaced to one of the key words, the key word can be replaced by pressing another softkey.

NOTE

A word can be replaced by placing the cursor anywhere under that word and pressing another softkey. No other information in the command line is replaced or changed.

Figure 4-3(a) shows an example using the directory command, which can consist simply of the keyword "directory" or several options. In figure 4-3(b) the directory softkey has been pushed and the next allowable alternatives are shown:

<FILE>	user file name
all_files	all disc files
rec_files	all recoverable files
tapefiles	all tape files
listfile	specify an alternate listing file

In figure 4-3(c), the all_files option is selected and the labels again change to reflect other options. The complete command shown in figure 4-2(d) calls for all of the files modified after August 28, 1980 to be listed on the printer.

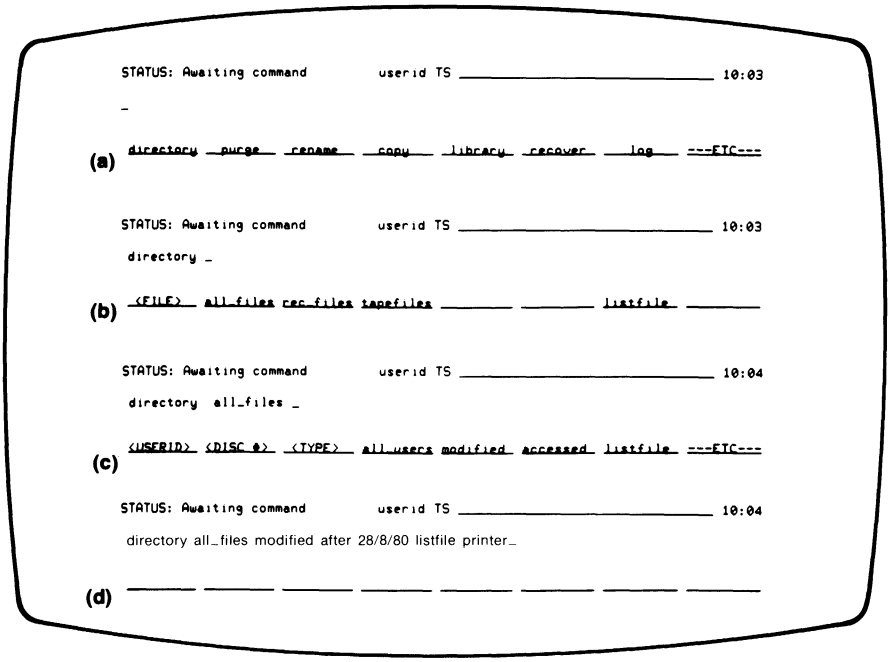


Figure 4-3. Constructing a Command Using the Softkeys

Files and File Management

Files

A file is defined as data stored in the system which is identified in a unique way. In the 64000, files are identified by four attributes: the name, userid, logical unit (LU) number, and file type. No two files in the system are identical across all four attributes.

All user-accessible files have a similar format. The data is stored in variable-length records. The number of words of data in a record is placed in the bytes immediately preceding and following the data. This arrangement allows for bidirectional access. It also provides a means for insuring the integrity of the file data. If the two lengths of a record are not the same, then a data read or write error is assumed.

Program modules such as the editor, assembler, and linker are called by the 64000 monitor using a system of overlays. When a module has been selected by the user (or the currently running module), an operating system routine is called to bring the correct file from the disc. Files of this sort are kept in a special non-record format. They are stored as memory images that can be read directly into the location in memory where they will be executed.

Command Files

The 64000 provides the user with a method of setting up a series of commands that must be executed in a specific sequence. This method is referred to as building a command file. Command files are source files (created in the edit mode or by using the log command) consisting of valid system commands to be executed in sequence. The use of command files for common housekeeping chores, etc., eliminates much of the required interaction between the user and the system. In addition, command files provide a convenient method of passing parameters by using a parameter declaration line preceding the commands in the command file. When a command file is called, the system will query the user for current values of the listed parameters.

File Protection

The user may protect his files by assigning a userid password to them. Protected userids are not accessible from any other userid. Although protected userid files will be displayed with other files in a directory command, they will only be displayed with "all_userid" specifiers. The files can only be accessed after the default userid has been set to the protected userid. Once entered, the password on the protected userid cannot be later identified. It can be changed, however, while operating under the protected userid by reentering the userid command.

File Management

The file manager (a module of the operating system) is responsible for the organization and control of files. File management functions such as purge, recover, and rename are accomplished by the file manager.

As files are accessed (viewed) or modified on the 64000, the file manager automatically corrects the modified and access dates. For dates to be valid, the date and time command must be used correctly.

Unique to the 64000 is the ability to recover purged files. The recoverable files directory is limited to a maximum number of recoverable files. When a file is purged, its space is linked to the available disc space. As more disc space is needed, the purged file will eventually be written over. Also, if the space reserved for a recoverable file is needed for an active file, the number of recoverable files will be reduced. The purge operation works on a first-in, first-written-over basis.

When a file is purged, its disc space is linked to the available disc space and deleted from the active file disc space. A recoverable directory entry will be made to allow listing of all available recoverable files. The direction of disc space use is such that all available unwritten space is used before the purged files are written over. On the recover command, the purged file is linked to the active disc space and that file's entry is removed from the recoverable directory.

Each time a file is edited, the previous version is also linked to the available disc space. It is therefore possible to recover all versions of a file that have not been written over. Since each recoverable file has a last access/modified date, the proper previous version may be quickly selected.

Page Structure

The file management system has a linked list structure. Each file consists of a block of sectors referred to as pages. The number of sectors per page is constant for a given storage media but may vary to optimize certain file management operations. The pages of a file are linked in both forward and backward directions (see figure 4-4).

When a file is being updated, the same sectors on the storage media are used. If the size of the file is increased, the file manager allocates another page to the file, linking it to the end of the last page.

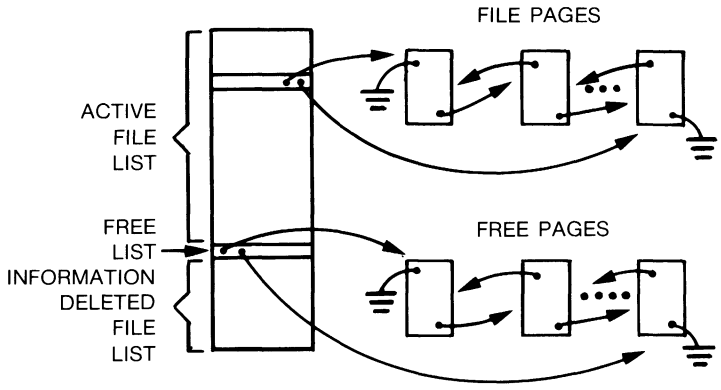


Figure 4-4. 64000 File Structure

Directory Format

As with most file management systems, the 64000 contains a directory which is organized as a hash coded list. Hash coding minimizes the amount of searching required to locate the directory entry for a given file. The hashed value of the file name indicates the directory sector on which the file information is most likely to reside. The data on that sector will indicate if the file exists or if another directory sector should be searched.

Each entry in the directory gives the name, user identification, and type of the file. Each entry contains pointers to the first and last pages of the file. In addition, two dates and times are kept for each file. One is the date/time that the file was last accessed. This is modified (with the system date/time) whenever the file is opened. The other is the date/time that the file was modified.

User Libraries

Libraries are a collection of relocatable modules that are stored on the system disc and can be referenced by the linker.

If a library file name is given as a response to the "object files?" query by the linker, all the relocatable modules in the library file will be relocated and linked. If a library file name is given as a response to the "library files?" query by the linker, only those relocatable modules that define the unsatisfied externals will be relocated and linked. The remaining relocatable modules in the library file are ignored. It is possible to combine relocatables into a library by using the system library command.

Refer to the System Software Reference manual for additional information concerning software operation.

Chapter 5

64000 System Modules

Editor

Introduction

The 64000 Editor is the means for creating and modifying existing source files. It allows the user to interactively perform block manipulation, multiple line operations, line insertion, deletion, and revision. It also provides high visibility of text through the use of multiline display format. Most single line edits are performed within this display, giving an accurate picture of the file.

The editor has a "three-in-one" (command/insert/revise) mode of operation. This is reflected by inverse video softkey labels. Commands are entered to the display command line through the keyboard. System commands are composed of keywords (lower case) and user-defined data. System commands are displayed at the bottom of the display immediately above the softkeys (see figure 5-1) and are entered by pressing the respective softkey. Softkey labels will change while entering commands to the system.

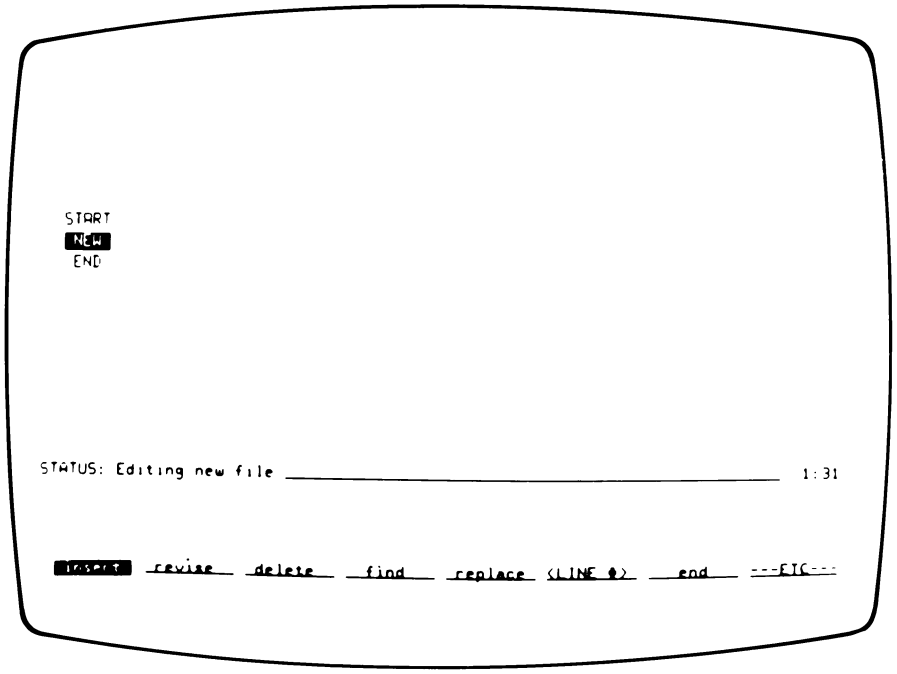


Figure 5-1. Editor Format

Operational Description

All commands available in the editor command mode may be executed in the insert and revise modes. The commands available for editing include: delete, find, replace, <LINE#>, end, merge, copy, extract, retrieve, insert, list, renumber, repeat, tabset, range, and autotab.

In the 64000 there are no artificial constraints on file size or workspace used. Positioning can be performed by rolling the text up or down, selecting a specific line number, or searching for a character string in the forward or reverse direction. All operations involving a group of lines, such as deleting, copying, extracting, listing or performing character replacement are done starting with the line containing the cursor through or until a line number, a character string, the start of a file, the end of a file, or the entire file.

When an edit session is started, two scratch files are created. These files serve as temporary storage for text that will not fit in RAM memory. When the original source file is opened, enough lines to fill the display buffer are placed on the CRT screen (see figure 5-2). More of the source file is read into Queue A. The amount of text read is limited to produce a reasonable response time. Only for very short files is the entire file read before the user is allowed to issue commands.

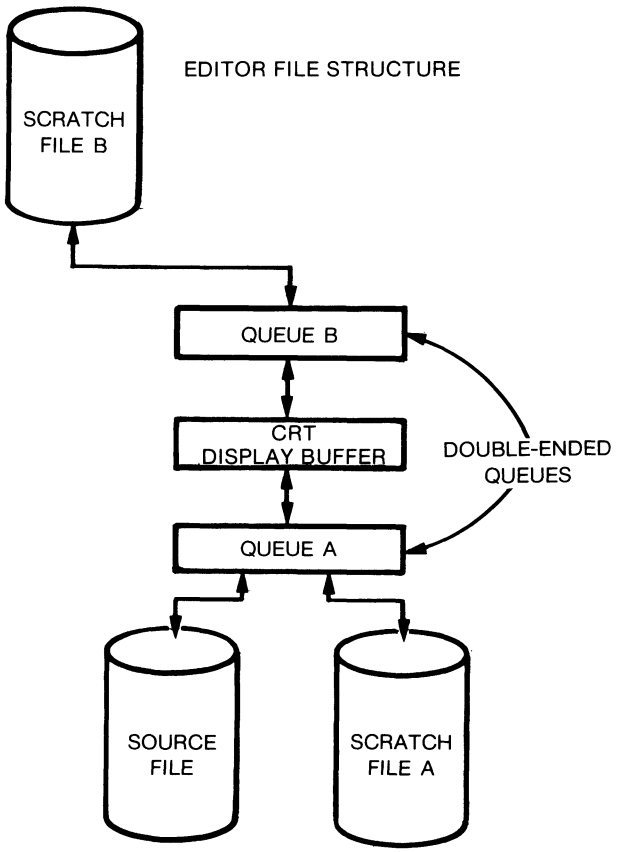


Figure 5-2. Editor Memory Space

As various commands cause more of the source file to be read, the data is brought into the display buffer and shuffled between the two double-ended queues. When the display buffer space is filled, records are written to scratch file B in the forward direction. Should a command require moving to an earlier line of text the records are written to scratch file A and read from scratch file B. The original source file is never overwritten.

When the end command is issued, a destination file is created. The text is written from scratch file B, the display buffer space, scratch file A, and the source file into this destination file. The original source file is then purged and the destination file renamed as source. The original source file is placed in a recoverable file list by the 64000 file manager and can be recovered if necessary. When the scratch files are closed they are deleted from the disc directory by the file manager.

For a detailed description of the editor, refer to the Editor Reference Manual.

Assembler

Introduction

The HP 64000 assembler is a table-driven assembler that converts the user's source program into a relocatable data structure which then can be linked into executable machine language. The assembler is capable of producing code for virtually any microprocessor and its main functions are the same regardless of the microprocessor. Additional information is added for specific microprocessors in the form of tables. The tables are used to interpret processor-dependent instructions and mnemonics.

The common functions of the assembler cover the interaction required with the host system. This includes reading and parsing the source program. The assembler handles all of the input and output file operations required by not only the source program but the resulting relocatable and list files as well. It also:

a. Parses each line of the source program and identifies the instruction for the specified processor.

b. Maintains a symbol table whose content contains file symbols along with their associated values and symbol types.

c. Checks operand fields and flags errors if the syntax and/or address rules are not followed.

Assembler Operation

The 64000 assembler reads the first line of the source file and looks for a directive that tells it which processor language is in the file that follows. It then reads another file that contains the table code for the indicated processor.

The assembler makes two passes through a source file to develop the machine code required by the microprocessor (see figure 5-3). On pass-1 the assembler looks for user-created symbols and stores them in a symbol table. On pass-2 the assembler recognizes the microprocessor mnemonics and once an opcode is identified, the assembler checks to see if it is an instruction that requires table code to understand the operand. If so, control is transferred to the special routines that use the table code to control the assembler. The tables instruct the assembler how to parse the operand field, what value to expect, how to generate the object code, and what error messages to generate, if any.

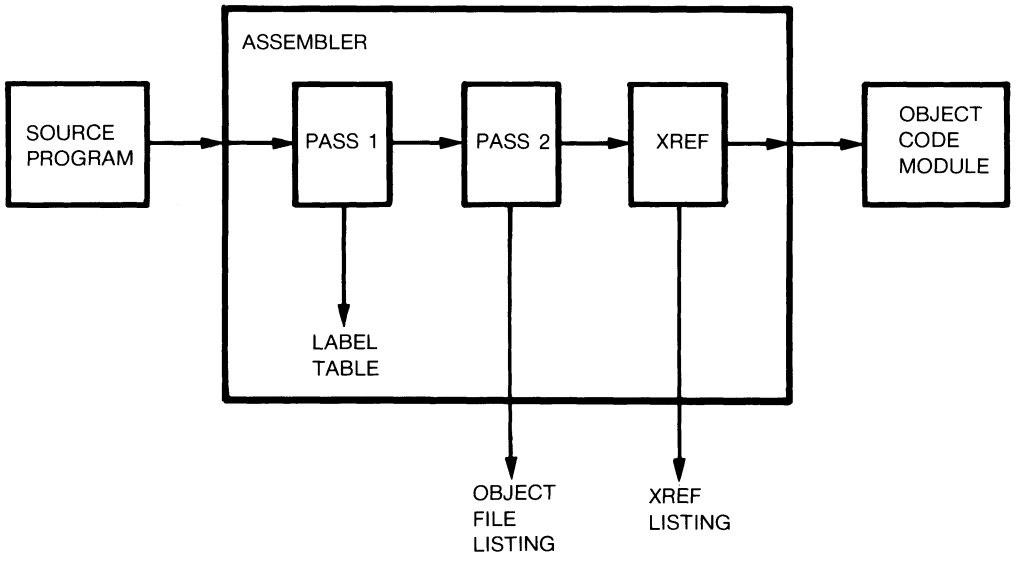


Figure 5-3. Assembler Two-pass System

The part of the assembler that handles the table code is a simple interpretor itself. It takes the specially coded table information and decodes it into instructions for the assembler. These instructions call up assembler functions, such as expression handlers, and object code generation. They also allow for arithmetic operations and testing for Boolean results.

User-definable Assembler

Since a set of tables is the only requirement necessary for the assembler to recognize different microprocessor languages, a program was developed to make this capability available to the user. The user-definable assembler program allows the user to generate an assembler for a unique chip or bit-slice processor. It can also be used to enhance existing assembler tables with user-specified instructions. To generate a customized assembler, the user must describe the syntax of each instruction and how to generate the object code. The assembler will handle all system overhead. It will generate relocatable files that can be used by the system linker and will produce list files like any other system assembler.

For additional information concerning the assembler, refer to the Assembler/Linker Reference Manual.

Pascal/64000 Compiler

Introduction

The 64000 Compiler Module provides for two types of compilation depending on the application: 1) generation of a relocatable object file to run on the target microprocessor, or 2) generation of an absolute pseudo-machine-code file to run on the development stations.

Currently available is Pascal/64000, an implementation of a subset of the Pascal programming language "standard", defined by Jensen and Wirth in PASCAL User Manual and Report (second edition) published by Springer & Verlag, 1976. The language has been enhanced to improve its utility as a tool for microprocessor system programming.

The Pascal/64000 compiler normally uses a three-pass compilation process to translate source programs directly into relocatable code for the target microprocessor (see figure 5-4). Relocatable files for a particular microprocessor may be linked together to produce an absolute program file. Then, by using the emulator, the absolute file can be loaded into emulation memory and executed in the proper microprocessor environment.

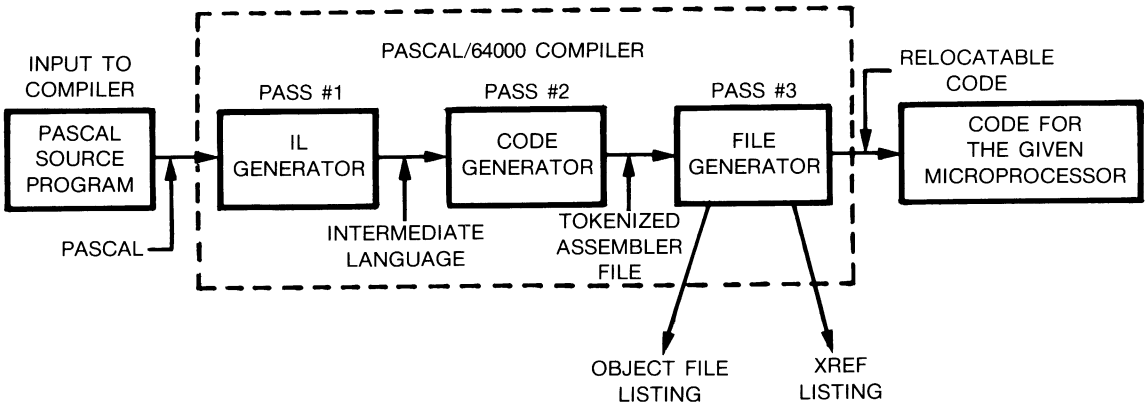


Figure 5-4. Pascal/64000 Three-pass Compiler

Pascal/64000 Subset

Since Pascal/64000 is a subset of Jensen and Wirth "standard" Pascal, it is compatible with this "standard" except for the following:

a. Real numbers and their associated standard functions, such as SIN, COS, EXP, LN, and SQR, are not currently implemented.

b. Strings are limited to a maximum of 255 characters. Strings and their associated standard operations are not implemented in certain microprocessors (refer to the Pascal/64000 Reference Manual for a list of specific microprocessors).

c. Files and their associated standard procedures, such as GET, PUT, READ, WRITE, and RESET are not implemented.

d. Packing of data is not carried below the byte level; the keyword, PACKED, is ignored by the compiler, except when defining a string, and the standard procedures PACK and UNPACK are not implemented. Bit packing may be achieved in 8-, 16-, or 32-bit data items by using the SET construct, "*" (AND), "+" (OR), and "–" (set difference), the predefined SHIFT function, and functional type change operations.

e. In most cases, sets are limited to 256 elements or less of any ordinal type. Set expressions with integer elements will be interpreted as being members of the SET OF 0..255 unless specifically qualified. Subsets are allowed, but the value of the maximum element of the SET may not exceed 255, i.e., SET of 240..255 (for exceptions, refer to the Pascal/64000 Reference Manual).

f. Procedures and functions are not allowed as parameters at this time.

g. Integers are 16-bit signed numbers in the range, –32768..32767. 32-bit signed integers are implemented for some microprocessors (refer to the Pascal/64000 Reference Manual for a list of specific microprocessors).

h. Set subranges are not implemented.

Pascal/64000 Extensions

Pascal/64000 contains enhancements that provide more versatility for microprocessor programming. These enhancements include the following:

a. The CASE statement can have an OTHERWISE clause.

b. Constant-valued expressions are allowed in places where a constant is allowed in standard Pascal.

c. CONST, TYPE and VAR declaration sections may be in any order after the optional LABEL declaration section, and any number of CONST, TYPE, or VAR declaration sections is allowed.

d. Identifiers may be of any length, but only the first 15 characters are significant. "Standard" Pascal uses only the first 8 characters.

e. Dynamic memory procedures: MARK and RELEASE may be used in the same program using the standard memory management procedures: NEW and DISPOSE.

f. Separate compilation of modules allows procedures and variables to be declared EXTERNAL (or GLOBAL) in a Pascal module so they may be defined (or accessed) in other Pascal or assembly language modules.

g. Program code and constants may be compiled to a separate relocatable area from data and variables allowing the design of ROM and RAM memory systems.

h. In addition to external linking, variables may be assigned to absolute memory locations permitting easy access to memory mapped I/O addresses.

The following extensions may be enabled with the $EXTENSIONS$ option:

i. Expressions or variables are permitted to have their implicit type changed by being included as the parameter of a "function" call of any named TYPE. Standard Pascal allows this by the use of variant type records, but the explicit functional type change greatly improves readability of the source program.

j. The built-in function ADDR returns the integer value of the address of the "parameter". "Standard" Pascal never allows the user access to address information.

k. Binary, octal, and hexadecimal constants may be used where constants are allowed in standard Pascal.

l. The predefined data type BYTE presents an 8-bit signed integer.

m. The predefined functions SHIFT and SHIFTC can be used to perform logical and circular shifting of 8-bit and 16-bit variables.

For additional information on the Pascal/64000 compiler, refer to the Pascal/64000 Compiler Reference Manual.

Host Pascal

HOST Pascal is a subset of UCSD Pascal. It is used to provide application programs on development stations for executing repetitive tasks, performing mathematical calculations, and solving problems.

The HOST compiler accepts as input a sequence of statements from one or more source code files for conversion into a pseudo-machine code which is executed interpretively using the run command. The HOST compiler will run on any HP 64000 System with a memory expansion module installed.

For complete information on the HOST compiler, refer to the HP 64000 HOST Pascal Manual.

Linker

Introduction

Another table-driven system module, referred to as the linker, combines relocatable object modules into one absolute file that is stored by the 64000 for execution in an emulation system or for programming PROMs (see figure 5-5).

The table-driven architecture allows the linker to support a variety of processors, and the assembler directive in each relocatable file is used to identify the required processor tables. Each supported processor has a linker table that is used by the linker for configuration.

The linker tables contain two types of information: general information such as word width and addressing space, and tables or sequences of instructions for the linker. The different instruction types and addressing modes allowed in the target processor correspond to entry points in the linker table.

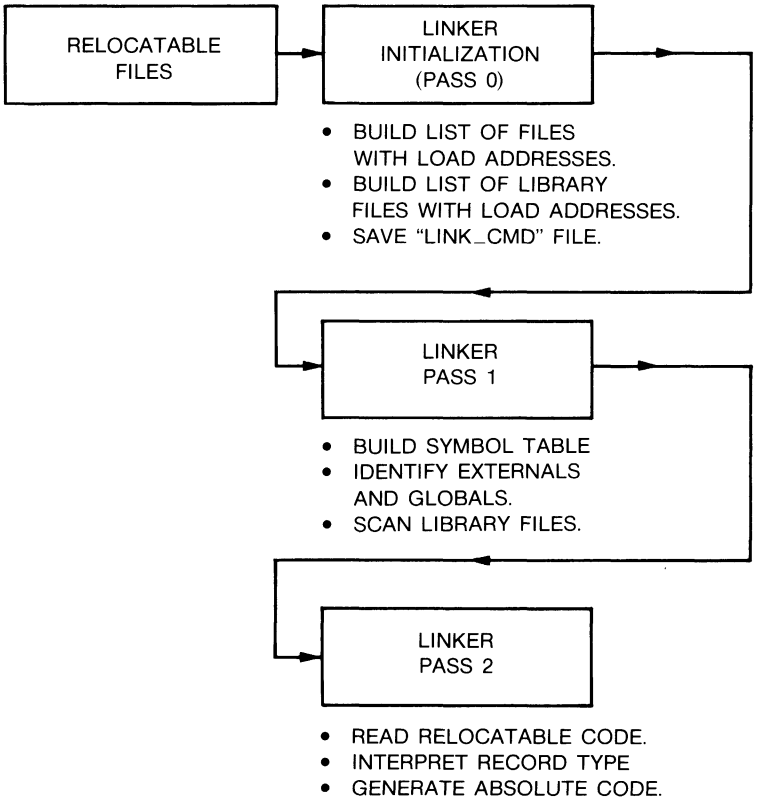


Figure 5-5. Linker Module Functions

Functional Description

To prepare object code modules for the 64000 load program, the linker performs two functions: relocation and linking. These two functions are discussed in the following paragraphs.

Relocation. The 64000 assembler and linker provides the user with several relocatable areas. The assembler directives ORG, PROG, DATA, and COMN define the relocatability of code. ORG defines code to be absolute or nonrelocatable. PROG and DATA are general-purpose relocatable counters that allow the user to partition code to be loaded at different memory locations. For example, all program in ROM and all data in RAM. COMN specifies that the data be relocated to the same starting address as the COMN data from all other relocatable modules. When the relocatable modules are linked, the user provides the starting addresses for the PROG, DATA, and COMN relocatable code.

Linking. The linker links the relocatable modules and generates symbol files for use in emulation. This allows the user, during emulation, to debug the program using symbols from the source program. This is useful when dealing with the relocated code, since the user does not have to know where in memory the linker stored the code. Any location in memory may be referred to by its symbolic name or its absolute address.

The linker also creates a global symbol file for every link operation. This file is used by the emulator, along with assembler symbol tables, to provide symbolic debugging. It may also be used in subsequent links to preload the linker symbol table. This feature has uses in overlays and in reducing linking and download time in large systems.

"No-load" Files

Any subset of the relocatable files may be declared to be "no-load". This subset is linked and relocated with the files that are not "no-load" files. However, the absolute file generated by the linker contains no code from the "no-load" relocatable file. "No-load" files may be useful in linking to existing ROM code or in the design of software systems requiring memory overlays.

Linker Output

The user can optionally select an output listing of the program load map and a cross-reference (xref) table. The linker also generates a listing that contains all errors that were noted. These error messages will contain a description of the error along with the file name and relocation/address information when applicable.

For additional information concerning the linker, refer to the Assembler/Linker Reference Manual.

Emulation

Introduction

Emulation is accomplished by using a two-processor system. One of the processors, located in the development station, is used by the 64000 host system. The other processor, located in the emulation pod, is used to emulate the target system processor.

The emulation and analysis system hardware may consist of the following components.

a. Emulation Control Card

b. Emulation Bus Logic Analyzer Card

c. Memory Control Card

d. Memory Card(s)

e. Emulation Pod with Probe

The emulator system is partitioned into three interfaces:

a. The user interface - defined by the specifications of the processor being emulated.

b. The emulation bus - a high-speed bus that connects the processor emulator, the emulation memory, and the logic analyzer.

c. The 64000 mainframe bus - provides for control and communication between the host processor and the emulation system.

This architecture provides complete separation of the host processor and host memory from the emulation system. It permits host processing and emulation to run simultaneously without resource contention.

Hardware Description

The processor specific portion of the emulation system is separated into two subassemblies: a pod external to the 64000 mainframe and a control card in the 64000 mainframe (see figure 5-6).

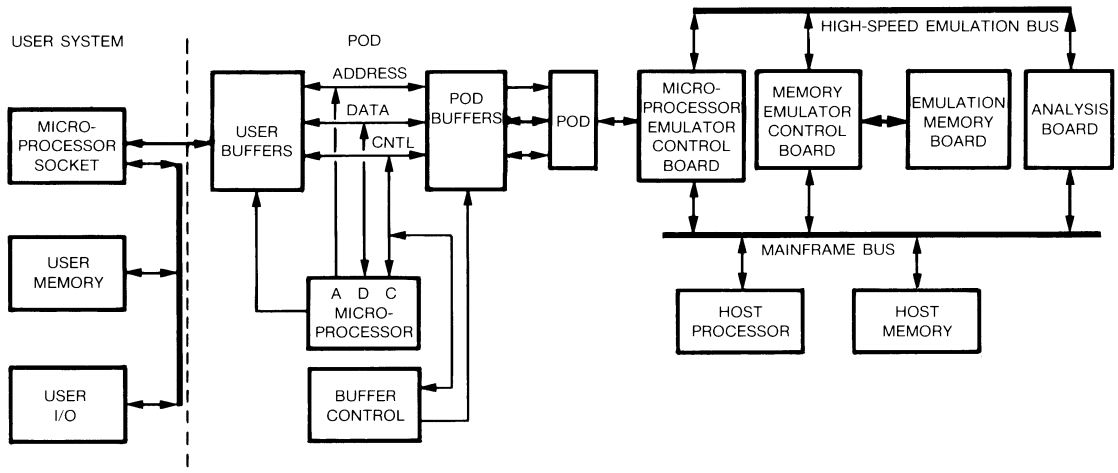


Figure 5-6. 64000 Emulator Subsystem

The emulator contains the microprocessor being emulated, interface buffers, buffer control circuitry, and an internal clock source. Some of the advantages of this buffered configuration are the minimization of possible damage from the user's system and the ability of the 64000 System to gain control of the emulation processor and continue to function even though an electrical fault may exist in the user system. The pod is connected to the user's microprocessor socket by a cable and emulation probe. Each signal wire in the cable is isolated from adjacent signals by alternating ground wires with the signal wires to minimize coupling. The pod is connected to the emulator control card by flat-ribbon cables.

The emulation control card controls the interaction between the 64000 operating system software and the emulation hardware. This board is the primary interface between the emulation pod and the emulation bus.

Operational Description

In the 64000 System, there are two basically different types of emulator systems. The implementation of a background monitor is used for emulating some microprocessors while a foreground monitor is used for emulating others. Operation of these two types is briefly described in the paragraphs that follow.

Emulators Using Background Monitor

When the emulator is operating, it is in one of four states: foreground, jam background, idle background, or exit background (see figure 5-7). In the foreground state, the emulator appears to the user system as a standard microprecessor and executes user-written code which may be physically resident in either user memory emulation memory, or a combination of both, depending on the user's memory map. Even though physical memory, such as ROM, may exist in a given address space in a user's system, it is possible to substitute that memory with 64000 emulation memory for code debugging purposes.

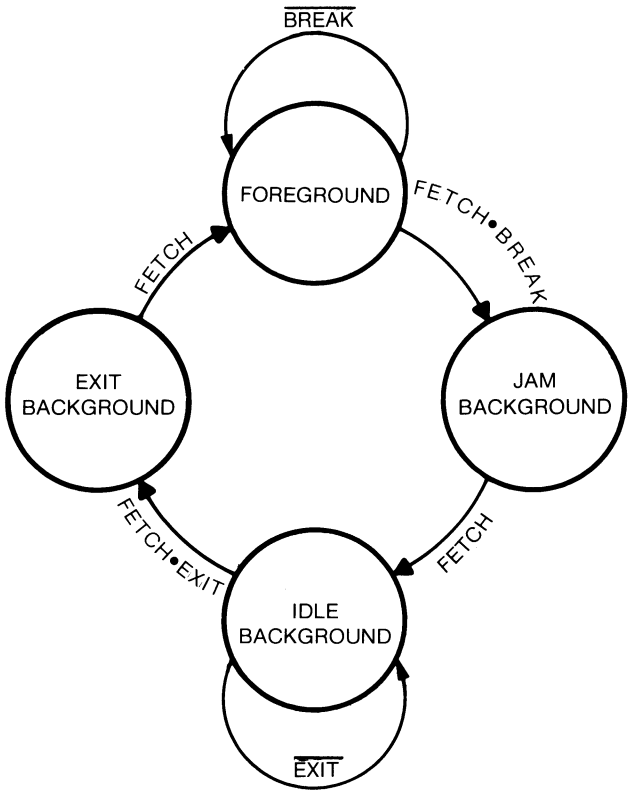


Figure 5-7. Emulator Four-state Operation

In the background state, execution in the user system is suspended and the processor appears halted to the user system. The apparent halted state at the user interface is accomplished by manipulation of the pod buffers while the processor is actually running under 64000 System control in background memory. While in background, all inputs from the user system are inhibited to prevent possible user system interference with the execution of emulator background tasks.

Background memory is a 256-byte RAM resident on the emulator control card. This memory is physically distinct from any memory in the user system or on the emulation memory card. The background memory is accessible to both the emulation processor and the 64000 host processor and serves as the primary communication link between the two. The 64000 host processor loads various register and memory read/modify routines into background memory. These routines are then executed by the emulation processor when it is transferred from foreground to background.

Transferring control from the emulation processor is accomplished by jamming addresses, independent of the addresses being generated by the processor, onto the emulation background memory address bus. This causes the next opcode to be fetched from background memory.

The jamming process is synchronized by the background controller to the first opcode fetch cycle following the occurrence of a break condition. This process simultaneously inhibits the user interface buffers and the address buffers from the processor to the background memory. Typically, a call instruction is used in the background code as the first instruction (see figure 5-8). The use of this type of instruction serves two purposes. First, the processor responds by placing the program counter on the stack. The stack is always in the same two locations in background memory regardless of where the processor stack pointer is set. This information is later used to determine where to send the processor when the emulator is returned to the foreground state. Second, the program counter is changed to the starting address of the background program. This results in transferring program control to the background memory when the jam cycle is terminated on the next opcode fetch.

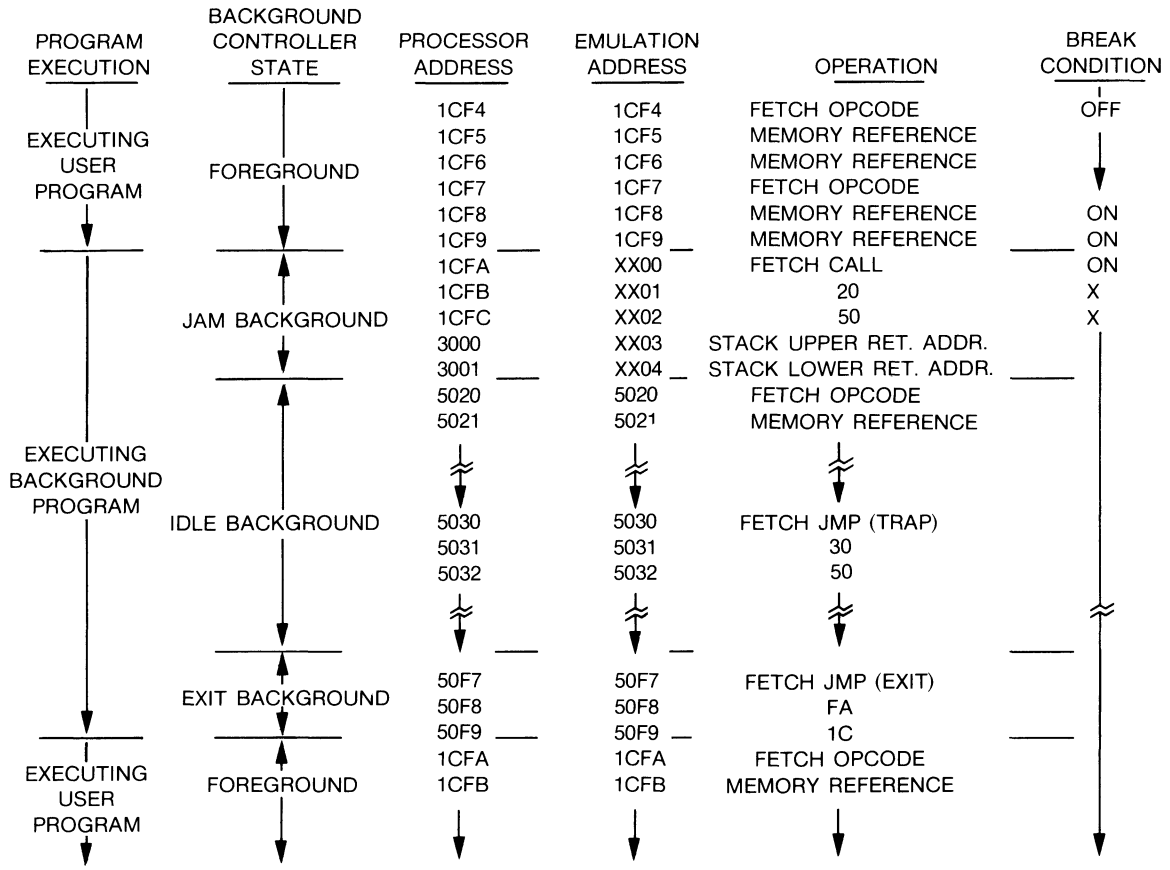


Figure 5-8. Typical Foreground/Background Operation

The host processor communicates with and controls the emulator processor indirectly through the background memory. This is possible because the memory is designed so that the host processor can read or modify background memory at the same time the emulator processor is executing code in that memory. The method of control involves the host processor loading a program or programs into background memory and then changing the jump address of the TRAP function to coincide with the starting address of the desired background program. The emulator processor reads the new jump address and transfers to that point.

The exit background state is initiated when the host processor causes the emulator processor to make an opcode fetch from a dedicated background address called EXIT. The background controller recognizes the fetch from EXIT and makes the state transition. The opcode loaded into location EXIT is a jump instruction and the following bytes contain the address of the desired foreground entry point.

The process of entering or exiting background described here is employed in all cases where it is necessary for the host system to control the emulator processor. An example of this is single-stepping, where the emulator is returned to foreground for a single instruction cycle and then immediately jammed into background. Continuous stepping and non-real-time analysis (see figure 5-9) are done in a similar manner.

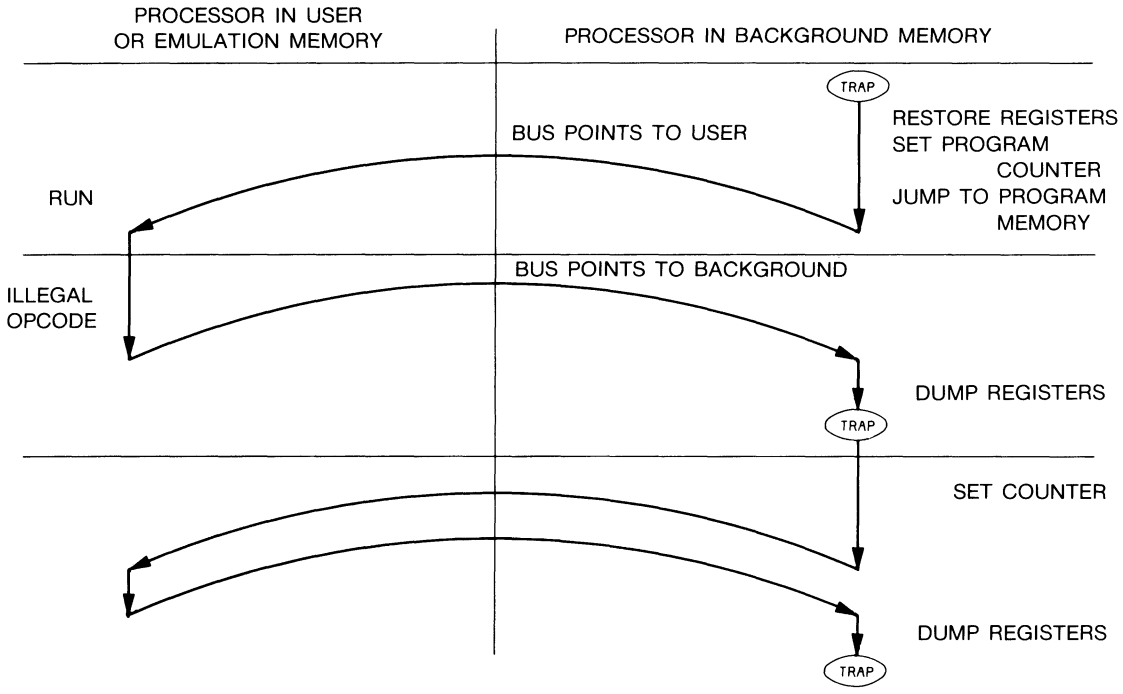


Figure 5-9. Non-real-time Analysis Operation

Emulators Using Foreground Monitor

In emulators that implement the foreground monitor, the emulation process is controlled by a software program referred to as the Emulation Monitor. This "monitor" is unique for each processor (see figure 5-10).

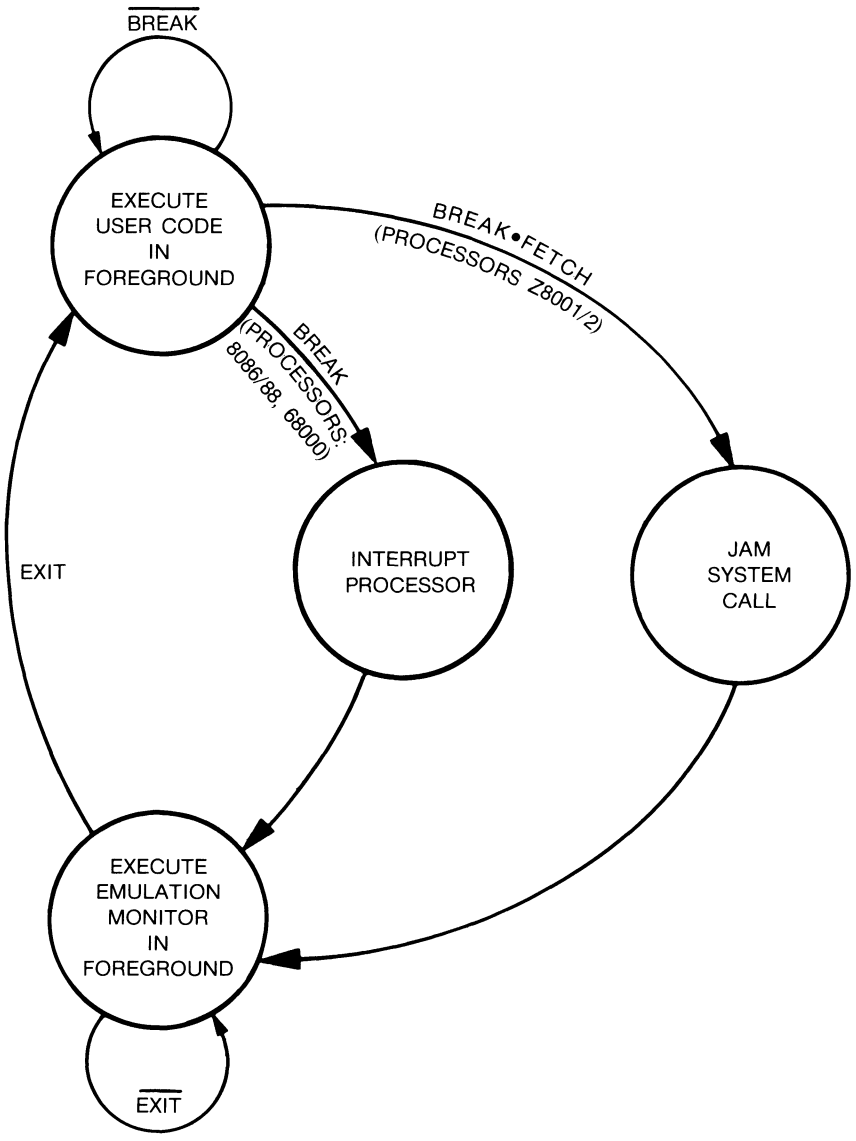


Figure 5-10. Foreground Monitor Emulator Operation

Many features of the foreground monitor emulator, such as display or modification of user memory, the display of registers, and the initialization of program execution, require communication between the host system and the emulation processor located in the emulation pod. This is accomplished through the Emulation Monitor. This program is supplied to the user in source form and may be modified by the user to accommodate specific requirements of the target system. The monitor should be assembled and linked with the relocatable program modules which form the software for the target system.

The absolute file developed by the linker can be loaded into emulation memory using the emulation software. Although the Monitor portion of the absolute module can be loaded anywhere in the processor address range, the particular range must be mapped to emulation memory.

The 64000 host system communicates with the emulation processor by transferring data to and from emulation memory. This is accomplished through the memory controller card. In addition to providing an access port for the host system, the memory controller contains a hardware mapper which is programmed to map the processor address space.

The monitor program contains several unique entry points, each reserved for a particular type of entry. The most common and useful entry into the Monitor is by means of the Break function. The Break function can be evoked from any of three separate emulation sources: emulation memory, emulation internal analysis, or the "break" softkey. A Break causes a controlled transfer from the program currently being executed to the Monitor program. All pertinent information concerning the processors status prior to the transfer is saved and all the Monitor control resources become available to the operator.

Memory Mapping

The memory controller provides mapping of the target processor address space. This is accomplished by using a mapper RAM with the address lines from the emulator. Each block can contain either 256 bytes, 1k bytes, or 4k bytes depending on the particular processor being emulated. The mapping feature allows emulation memory to be placed anywhere in the address space of the emulated processor. The memory mapper partitions the processor address space into emulation RAM or ROM, user RAM or ROM, or guarded space. Status bits are provided for each block of memory.

For a typical memory map, the CRT displays information describing the amount of emulation memory available to be mapped, the amount of memory already mapped, and the memory block size (see figure 5-11). The remainder of the CRT screen (above the STATUS line) displays up to 32 entries arranged in two columns of 16 entries each. Each entry is displayed as an entry number, the address range covered, the memory type, and then (if the type was emulation memory) the physical block number actually used.

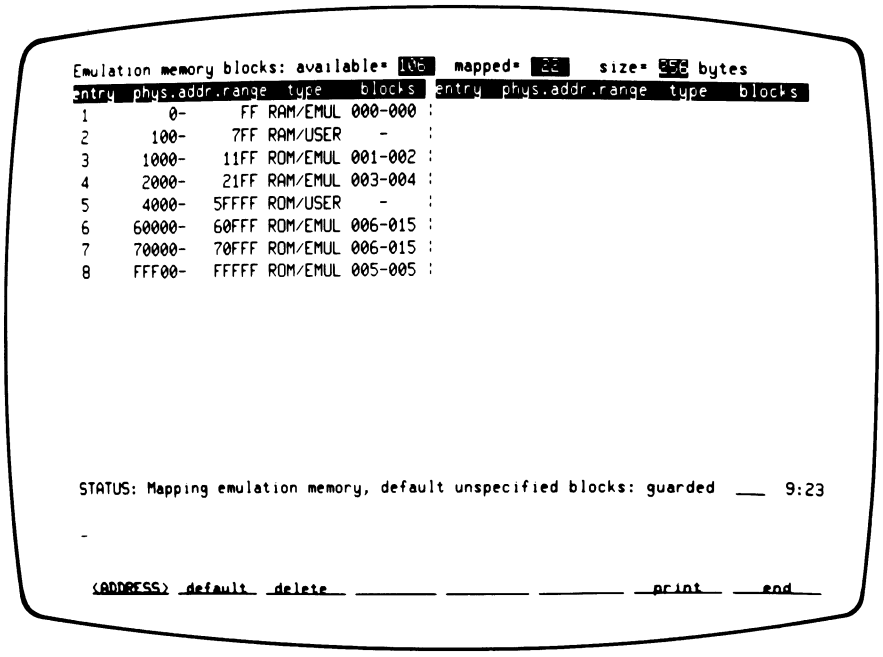


Figure 5-11. Typical Memory Map Display

Any address ranges which are unmapped when the mapping session is terminated are assigned the memory descriptor specified as the default. The default descriptor can be set up to be user RAM, user ROM, or guarded by using the "default" command. If no default descriptor is specified, all unmapped memory blocks will be defined as guarded memory.

For additional information concerning emulation, refer to the Emulator/Internal Analysis 8-bit Reference Manual and the Emulator/Internal Analysis 16-bit Reference Manual.

Preparing Software for Emulation

Code may be supplied in one of three ways:

a. By linking relocatables created by the assembler or compiler.

b. By transferring absolute code from another system.

c. By ROMs contained in the target system.

Combinations of these methods may be used. However, methods (a) and (b) have certain advantages. When relocatable files are linked in the 64000, files of type asmb_sym (assembler symbols) and link_sym (linker symbols) are produced. These files contain information describing local and global symbols used in the programs and are passed to the emulation software system when a file is loaded into emulation memory. The emulation system uses the symbol table to allow the user to specify items of data by the symbol name defined in the source code. If a program is generated externally to the 64000 and loaded into ROMs, the emulation processor will execute the code in the ROMs but will not be able to access addresses within the address range of the ROMs by symbol names. However, these addresses may be accessed using a link_sym and asmb_sym type files if the ROM code was generated on the 64000.

However the user generates the program code used in emulation, it is to his advantage to generate a detailed listing of the code being executed so he can follow the program trace. The listing includes the addresses of the instructions, the instructions in mnemonics, the resultant absolute code, and a cross-reference table. If a high-level language is being used, the original source text, as well as the generated assembly language, will be available in the listing.

Emulation Analysis

The Emulation Bus Logic Analyzer Card provides logic state analysis for the Emulation System. Once a trace has been requested by the operating system, the internal analysis module will monitor the emulation bus for "states" that satisfy the trace specification. It will continue monitoring the bus until stopped by a command from the operating system or until the trace specification is met.

Trace specifications are entered into the operating system by the user. They are loaded into the analysis card trigger hardware by the operating system by way of the mainframe bus. When the required trigger occurs on the emulation bus, the analysis card stores 256 contiguous events in a dedicated analysis memory referred to as the trigger buffer. The 256 events captured can be specified as being before, around, or after the trigger event.

When the display trace command is initiated from the development station console, the operating system accesses the trigger buffer by way of the mainframe bus. The content of the trigger buffer is retrieved, dissasembled, formatted, and displayed on the system CRT.

Simulated Input/Output

The simulated Input/Output (I/O) feature of the 64000 System allows the user to simulate various system peripherals prior to their being available in the target sytem. After programs are written, assembled (or compiled), and linked, they may be incorporated into an emulation configuration, executed, and tested.

The following 64000 System hardware may be used to simulate target system hardware during user-program development:

a. Printer

b. Display

c. Keyboard

d. Disc files

e. RS-232-C Communications Channel

Each simulated I/O interface requires a unique location to which all I/O handshaking codes are sent by both the user and the 64000 programs. The address for this location is generically referred to as the control address or CA. The location of CA must be initially defined in the user's program. If more than one simulated I/O interface is to be implemented, the user must ensure that each I/O program has a unique address for location CA. Additionally, the user program must allow for contiguous buffer spaces following location CA. The exact amount of, and the use of this buffer space, is determined by the type of I/O interface (refer to the Emulator/Internal Analysis 8-bit Reference Manual for a complete description of simulated I/O).

Addresses for the different control addresses are entered into the 64000 program during emulation configuration. The CA locations must be located in memory space assigned as either user or emulator RAM.

Logic Analysis

General

The 64000 Logic Analysis capabilities offer the user an extended list of options not available in previous analyzers. Its modular construction allows for the following:

a. Analyzer width is available in 20- to 120-channel configurations in either 20- or 40-channel increments.

b. Different analysis modules may be installed in one mainframe for interactive measurements. Available types of modules are:

   1. Emulation analysis for different microprocessors.

   2. Logic state analysis.

   3. Logic timing analysis.

c. Mainframes may operate as stand-alone units or in clusters of up to six units.

d. Mass storage available either as internal flexible discs for stand-alone units or external hard disc drives.

e. Multiple disassemblers can be stored on the disc and called as needed.

Up to four independent analyzers of any type may be installed in one large mainframe. Synchronization of these analyzers in interactive measurement applications is accomplished by way of an asynchronous intermodule control bus. The bus transfers five kinds of control signals: master enable, trigger enable, trigger, storage enable, and delay clock. When it receives a trigger, enable, or store enable signal from an analysis module, it channels this signal to the appropriate function within the state analyzer.

Logic State/Software Analyzer

The 64000 state analyzer is composed of two distinctly different measurement systems: the trace measurement system and the overview measurement system. Both of these measurement systems are controlled by a sequencer. In addition, the overview measurement system can trigger data acquisition in the trace measurement system (see figure 5-12).

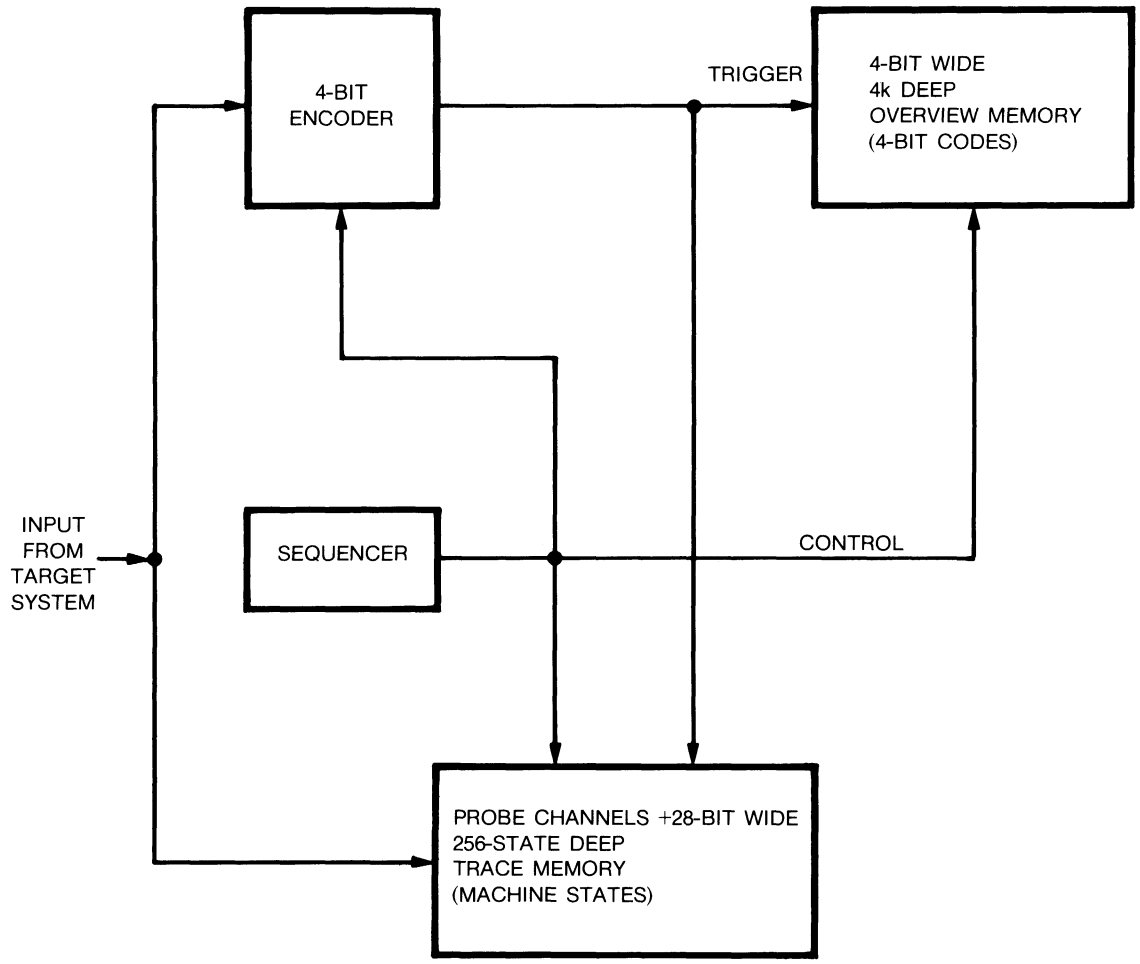


Figure 5-12. Logic State/Software Analyzer Memory

Overview Measurement System. Memory for the overview measurement system is four bits wide and can store up to 16 different 4-bit codes. Each time a label is assigned to a range or point in the state flow, an encoder assigns one of the 4-bit codes to that label. As each pattern arrives from the input, the encoder generates the appropriate code for a specified pattern and sends that code to the overview memory. When the state analyzer prepares a histogram, it totals each of the 4-bit codes in memory and then places representations of these totals on the display.

Trace Measurement System. The trace memory stores up to 256 states from the data acquisition circuitry. The width of each state is the same as the number of probe channels supplying information to the analyzer, plus an extra 28 bits. The additional 28 bits are divided between the count memory and the sequence memory (see figure 5-13).

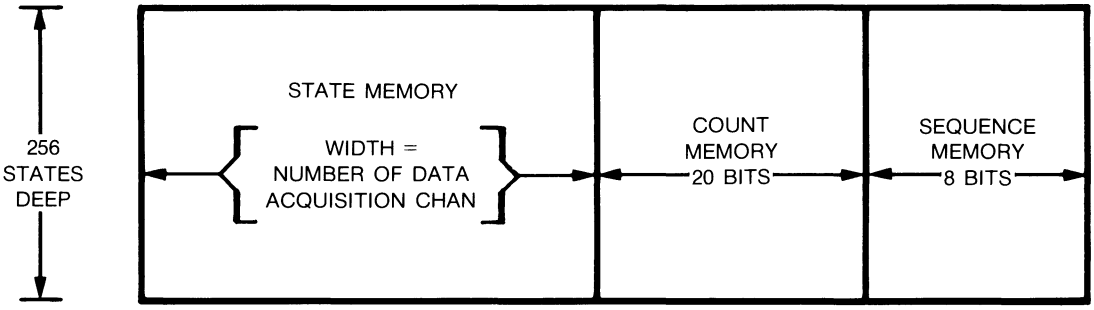


Figure 5-13. Trace Memory

The count memory stores the output of the count generator for each counted state. The count generator measures time or counts occurrences of states in the data flow according to the count specification. The function performs the task of recognizing which states to count and recognizing those states that enable or disable the count generator.

The sequence memory stores the status of the sequencer and windows so that the user can call up a list display that includes the status of each sequence specification. The user may establish up to three separate sequences for control of data acquisition in the overview memory and the trace memory.

State Analysis Channels

There are two types of probe channels used with the Logic State/Software Analyzer: the 20-channel ranging probe, and the 20-channel data acquisition probe. The ranging probe pod not only supplies logical activity to the state analyzer just as other probe pods, but it also supplies overview information to the overview memory. Figure 5-14 shows how the overview events are detected in the ranging pod.

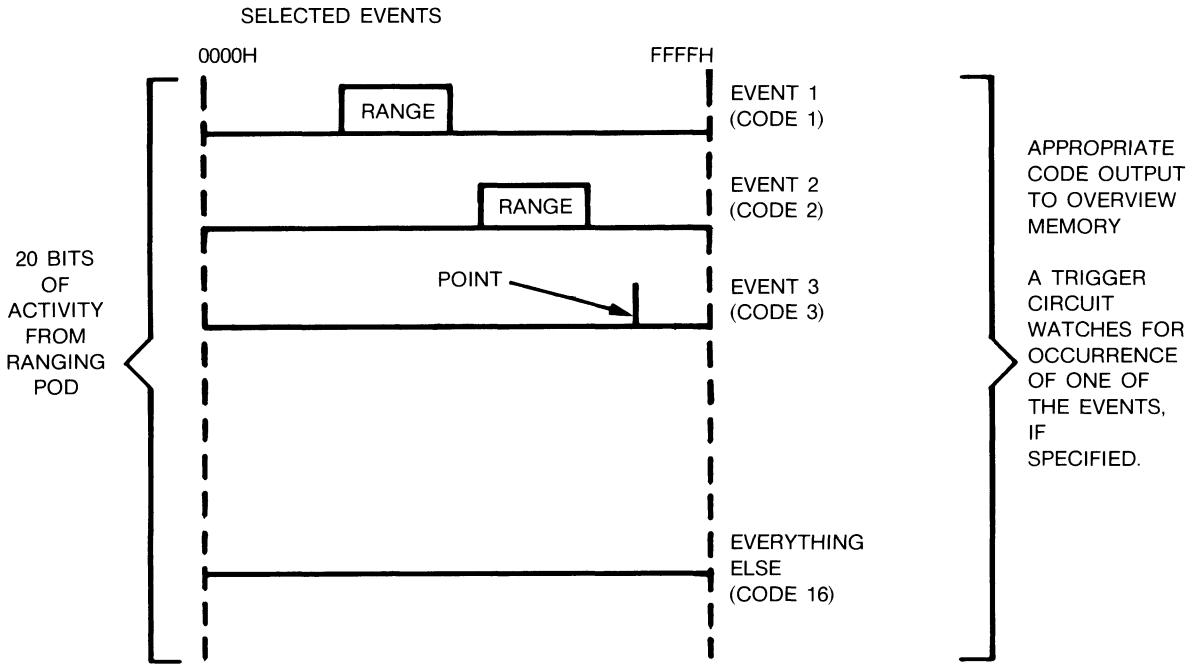


Figure 5-14. Detecting the Overview Event

Each time a 20-bit state event arrives in the ranging circuit, it is compared with events that were specified before the run. If the incoming event matches any of the previously specified events, then the ranging circuit sends an assigned code to that event in the overview memory. If the incoming state does not match any of the specified events, then it is classified as an "everything else" event. If the "everything else" events have been selected to be saved, they will be sent to the overview memory; otherwise, nothing will be sent during that clock period.

Preprocessors

The preprocessor provides the user an easy-to-use interface for most of the popular processors. It contains its own input signal conditioning electronics and totally replaces the general-purpose probes.

The preprocessor is connected to the user system through a personality or customizing module. Latching circuits have been incorporated in this unit to allow bus signal demultiplexing of up to 32 of the 60 input channels.

Stimulus and halt lines allow the user to control the target system. These lines can be activated after the occurrence of certain events have been detected by the analyzer or upon measurement complete or any trigger condition.

Also available is a General-purpose Personality Module that allows the user to design and implement more complex stimulus signals which can be routed to the target hardware via the nine general-purpose input/output lines provided in the general-purpose preprocessor.

For additional information concerning logic state analysis, refer to the Logic State/Software Analyzer Reference Manual.

Logic Timing Analysis

The Logic Timing Analyzer options add powerful real-time analysis to the 64000 System. One option provides eight input channels while another option can add a second data acquisition for a total of 16 channels (see figure 5-15). The Timing Aanlyzer can be integrated into the system or used as a stand-alone timing analyzer with either the Model 64100A or the Model 64110A Development Station.

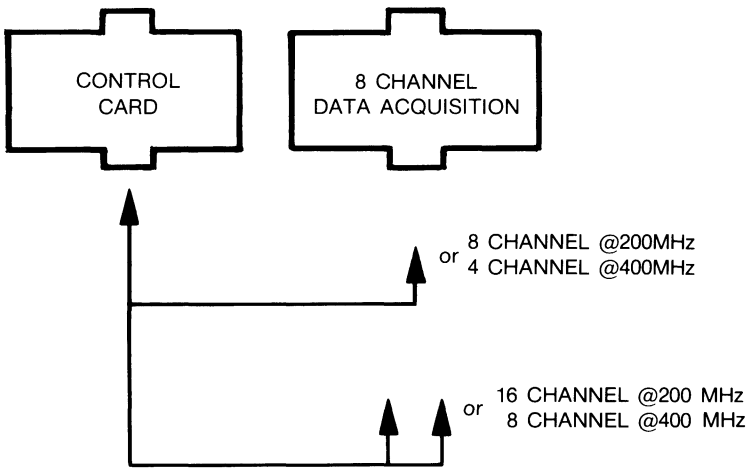


Figure 5-15. Timing Analysis

Main Features

Modular architecture for best-fit systems.

Memory depth of 4060 states in the wide-sample modes with 5 ns sample resolution.

Glitch capture for 3 ns glitches with separate 4060-word memory.

Fast-sample mode at 400 MHz for sample resolution of 2.5 ns with 8140-word memory.

Low inter-channel skew within the pod and from pod to pod.

Dual-threshold operating mode at 200 MHz with 4060-word memory.

Sophisticated triggering on:

- Specific patterns.
- Patterns persisting for a specified range of time.
- Patterns which exceed or fall short of time limit.
- Transition into, or out of, a specified condition.
- Glitch
- OR trigger
- NOT trigger
- Middle level
- Combinational level and transition triggering for the 16-channel option.

Friendly user interface.

Extensive symbolic tracing.

Adjustable thresholds for 4-channel groups.

Interactive measurements with state analyzer or emulator.

Post-processing of collected data for performance evaluations.

Versatile displays:

- 8 or 16 channels
- Magnifications
- Dual threshold.

Convenient, versatile, high-quality, oscilloscope-type, compensated probing.

Architecture

Any Model 64100A or 64110A with two open card slots can be configured as an 8-channel timing analyzer. Depending on the availability of slots, timing channels can be added in multiples of eight to a maximum of 48 channels (3 Timing Modules) for a single 64100A or 24 channels (one 8-channel and one 16-channel) for a Model 64110A. Multiple modules can also be installed for combined state and timing analysis configurations.

Format Specification

Setting up timing analysis formats is simple with the softkeys. A softkey is used to call up the format specification menu. The format display automatically shows the current data acquisition mode and the number of channels available. For example, figure 5-16 contains a format specification menu for a fast sample mode with a 16-channel system. When the speed is 400 MHz, the number of channels is halved, so the display shows eight channels, four under each of the two pods. Thresholds are defined on sets of four channels and displayed directly above the channel numbers.

For more information, refer to the Logic Timing Analyzer Reference Manual.

Format Specification Timing 5, 16 channels, 200/400 MHz module

fast_sample

	Pod_1								polarity
Threshold:	ttl	+1.4	V		ttl	+1.4	V		
Label	7	6	5	4	3	2	1	0	
POD1	*	*	*	*	*	*	*	*	pos_true
POD1upper	*	*	*	*					pos_true
POD1lower					*	*	*	*	pos_true
HOLD								*	pos_true
ALE							*		pos_true
SOD						*			pos_true
SID					*				pos_true
WR				*					pos_true
RD_				*					neg_true

STATUS: Awaiting timing command-userid N_DIAG 9:55

Figure 5-16. Format Specification---
Fast Sample Mode with 16-Channel System

Interactive Measurements

Simultaneous, interactive timing and state measurements are sophisticated techniques for logic analysis in a microprocessor-based system. These measurements are made with a development station configured with both a Timing Analyzer and a State Analyzer connected by an Intermodule Bus (IMB). The IMB has five lines to interface the timing and state analysis modules. Either module can arm or trigger the other module in several functional combinations. For example:

State arms timing, timing triggers both.

Timing triggers state, state triggers both.

State triggers state, state arms timing.

Timing triggers timing, timing arms state.

Other forms of interactive measurements with external instruments are driven by the timing through a BNC in the rear panel of the station. Commonly, this mode can be used to trigger an oscilloscope, trigger a serial data analyzer, or provide a stimulus to user system hardware.

For more information, refer to the Measurement System Reference Manual.

Terminal Mode Operation

The terminal mode allows the 64000 station to function in an RS-232-C environment as either a Data Communication Equipment (modem) device or as a Data Terminal Equipment (data processor device). The signal interface between the 64000 station and a remote device is accomplished through two asynchronous RS-232-C ports on the rear panel of the station.

Full duplex RS-232-C communication to and from the 64000 station is initialized with the system terminal module program. When initialized, file transfers between the 64000 disc and the remote device is supported using a 7-bit ASCII character code in one of three absolute file formats: Intel Absolute, Motorola Absolute, and Tektronix Absolute. If no file type is specified during uploading or downloading, source type files are assumed.

Actually, two submodes are used while in the terminal mode: the terminal submode and the command submode. The submode of operation is determined by the position of the cursor on the CRT display. When the cursor is within the first 18 lines of the display, the terminal submode of operation is in effect, and the keyboard functions as a communications link with the remote device. When the cursor is in the command area of the display, the command submode of operation is in effect and the keyboard functions normally (part of the 64000 System).

The terminal is capable of running at a 1200-baud rate without requiring the remote device to understand a protocol. Since many devices may be overrun by a 1200-baud rate during uploading, the terminal supports two common protocols: XON/XOFF and ENQ/ACK. In addition, extensions to these protocols are provided to permit operation with slow time-shared machines.

For more information concerning terminal mode operation, refer to the System Software Reference Manual.

PROM Programming

Introduction

The PROM Programmer option provides the capability to program, verify, and read most popular bipolar and MOS (fusible link, UV, or electrically eraseable) PROMs. Hardware includes a control card and a choice of unique personality PROM modules. The PROM Programmer option always includes circuitry for both the operator self-test and the service self-test.

The PROM Programmer can use two types of files for programming or verifying PROM contents: absolute files and listing files. These two file types are described below.

Absolute files are created by the system linker. They contain information about the processor, the addresses where data is located in memory, and who generated the file. All of this information will be transparent to the user. The programmer is compatible with any microprocessor, regardless of word size.

Listing files provide several features within the programmer. For instance, they allow the user to see what is in a PROM. By listing a PROM program to a file, it can be reviewed for changes or corrections via the system editor and returned to the programmer for downloading into a PROM.

The PROM programming system consists of a control card occupying one slot in the 64100A station and a socket module that resides in the 64100A panel insert. The control card contains adjustable power supplies and general input/outout driver circuits, as well as a 64000 mainframe interface. The individual socket modules match PROM pinouts and tailor the card's general signals to meet specific PROM programming specifications.

To ensure the widest support of PROMs, the hardware requirements of the controller and the socket module have been minimized. All sequence timing and pulse width control are done by software in the PROM driver.

FOLD HERE

HEWLETT PACKARD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 1303 COLORADO SPRINGS, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD
Attn: Logic Publications Dept.
Centennial Annex - D2
P.O. Box 617
Colorado Springs, Colorado 80901-0617

FOLD HERE

Your cooperation in completing and returning this form will be greatly appreciated. Thank you.

READER COMMENT SHEET

Part Number: 64980-90912

Your comments are important to us. Please answer this questionaire and return it to us. Circle the number that best describes your answer in questions 1 through 7. Thank you.

1. The information in this book is complete:

Doesn't cover enough (what more do you need?)	1	2	3	4	5	Covers everything

2. The information in this book is accurate:

Too many errors	1	2	3	4	5	Exactly right

3. The information in this book is easy to find:

I can't find things I need	1	2	3	4	5	I can find info quickly

4. The Index and Table of Contents are useful:

Helpful	1	2	3	4	5	Missing or inadequate

5. What about the "how-to" procedures and examples:

No help	1	2	3	4	5	Very helpful
Too many now	1	2	3	4	5	I'd like more

6. What about the writing style:

Confusing	1	2	3	4	5	Clear

7. What about organization of the book:

Poor order	1	2	3	4	5	Good order

8. What about the size of the book:

too big/small	1	2	3	4	5	Right size

Comments: _____

_____

_____

_____

Particular pages with errors?

_____

Name (optional): _____

Job title: _____

Company: _____

Address: _____

Note: If mailed outside U.S.A., place card in envelope. Use address shown on other side of this card.

FOLD HERE

HEWLETT PACKARD

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 1303 COLORADO SPRINGS, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD
Attn: Logic Publications Dept.
Centennial Annex - D2
P.O. Box 617
Colorado Springs, Colorado 80901-0617

FOLD HERE

Your cooperation in completing and returning this form will be greatly appreciated. Thank you.

READER COMMENT SHEET

Part Number: 64980-90912

Your comments are important to us. Please answer this questionaire and return it to us. Circle the number that best describes your answer in questions 1 through 7. Thank you.

1. The information in this book is complete:

Doesn't cover enough (what more do you need?)	1	2	3	4	5	Covers everything

2. The information in this book is accurate:

Too many errors	1	2	3	4	5	Exactly right

3. The information in this book is easy to find:

I can't find things I need	1	2	3	4	5	I can find info quickly

4. The Index and Table of Contents are useful:

Helpful	1	2	3	4	5	Missing or inadequate

5. What about the "how-to" procedures and examples:

No help	1	2	3	4	5	Very helpful
Too many now	1	2	3	4	5	I'd like more

6. What about the writing style:

Confusing	1	2	3	4	5	Clear

7. What about organization of the book:

Poor order	1	2	3	4	5	Good order

8. What about the size of the book:

too big/small	1	2	3	4	5	Right size

Comments: _____

_____

_____

_____

Particular pages with errors?

_____

Name (optional): _____

Job title: _____

Company: _____

Address: _____

Note: If mailed outside U.S.A., place card in envelope. Use address shown on other side of this card.

64980-90912, FEBRUARY 1982

PRINTED IN U.S.A.