Prime

Prime 50 Series Technical Summary

Revision 19.1

DOC6904-191L

Prime 50 Series Technical Summary

DOC6904-191

First Edition

by

Martha August and Sarah Lamb

This guide documents the software operation of the Prime Computer and its supporting systems and utilities as implemented at Master Disk Revision Level 19.1 (Rev. 19.1).

Prime Computer, Inc.
500 Old Connecticut Path
Framingham, Massachusetts 01701

CUSTOMER SUPPORT CENTER

Prime provides the following toll-free numbers for customers in the United States needing service:

1-800-322-2838 (within Massachusetts)	1-800-541-8888 (within Alaska)
1-800-343-2320 (within other states)	1-800-651-1313 (within Hawaii)

HOW TO ORDER TECHNICAL DOCUMENTS

Follow the instructions below to obtain a catalog, price list, and information on placing orders.

United States Only	International
Call Prime Telemarketing, toll free, at 800-343-2533, Monday through Friday, 8:30 a.m. to 8:00 p.m. (EST).	Contact your local Prime subsidiary or distributor.

PRINTING HISTORY — PRIME TECHNICAL SUMMARY

Edition	Date	Number	Documents Rev.
First Edition	May 1983	DOC6904-191	19.1

SUGGESTION BOX

All correspondence on suggested changes to this document should be directed to:

Sarah Lamb
Technical Publications Department
Prime Computer, Inc.
500 Old Connecticut Path
Framingham, Massachusetts 01701

Contents

1 Profile of the PRIME 50 Series Family

Prime's 50 Series family is a sophisticated group of totally compatible supermini computers. These systems are:

- The 2250, an entry-level office environment system
- The 250-II, a low priced end user system
- The 550-II, a midrange system
- The 750, a high end system
- The 850, a multistream system

Each member of the 50 Series embodies Prime's "total compatibility" and "software first" design philosophies. All members also implement advanced architecture concepts, making the 50 Series flexible enough to answer a number of diverse user needs while still preserving fast and efficient operation. In addition, Prime's full complement of data communications and networking capabilities make it possible to link any member of the 50 Series to a host of other machines, granting access to information and data distributed across great distances. These characteristics, coupled with a single operating system, PRIMOS, and a wide range of available software and hardware products, make each of the 50 Series systems a versatile solution to almost any commercial or scientific application.

COMPATIBILITY

The 50 Series family is designed to accommodate users who may have one set of system needs today and who plan to expand into a potentially different set of needs in the future. The process of converting from a smaller system to a larger one can be time-consuming and expensive. The 50 Series eliminates much of the expense and time spent on such a conversion, since all family members are totally upward- and downward-compatible.

Each 50 Series system supports a multiuser, multiprocess, interactive environment, making any of them an ideal choice for many fields and applications. Users upgrading to more powerful members of the 50 Series family can expect to use their existing application programs on the new system without change. This is because the Prime 50 Series software is compatible at object code level, so it does not have to be reassembled, recompiled, or relinked to run on any other 50 Series system. One operating system, PRIMOS, runs on all members of the 50 Series family, providing a consistent interface regardless of the system. In addition, any disk pack suitable for use on any family member can be transferred to and can run on any other member without change.

In general, all Prime hardware controllers, memory boards, peripherals, and other hardware components can be used without modification on any 50 Series system. This means that the user's hardware investment is protected as long as any 50 Series system is in use.

"SOFTWARE FIRST" DESIGN

This philosophy has always directed Prime's design efforts. It means that Prime is committed to producing systems that offer the user the maximum system resources possible whenever the user needs them. To make such resources available, Prime's hardware is designed to support the PRIMOS operating system instead of the other way around. In addition, user interfaces and software packages are designed to be as simple and as straightforward to use as possible, so that the user's time is spent productively rather than in learning the idiosyncrasies of the system.

ADVANCED ARCHITECTURE

The 50 Series systems embody an advanced 32-bit architecture that grants the user the ability to perform complex tasks efficiently and quickly. Increased performance and precision are two of the major benefits of this architecture. In addition, the virtual memory capabilities of the 50 Series architecture give each user a virtual address space large enough to run application and development programs without the use of overlays, easing program complexity and development time. To speed memory reference time, the 50 Series uses interleaved

MOS memory and a high speed bipolar cache memory. A large number of hardware- and firmware-implemented features further enhance operation.

NETWORKING AND DATA COMMUNICATIONS

A special feature of the 50 Series systems is their ability to link with other systems, both Prime and non-Prime, in data networks. PRIMENET, Prime's networking software, supports the CCITT X.25 standard protocol and allows any 50 Series system to connect into three types of networks. Depending on the user's need and application, PRIMENET can connect a 50 Series system into a local ring network suitable for any environment requiring closely coupled, secure systems, such as banking, finance, scientific research, or commercial recordkeeping. A user can also connect a system into a telecommunications network, useful to a business with several remote sales offices. It is also easy to link into one of the public data networks, such as TELENET, EURONET, and DATAPAC.

In addition to PRIMENET, the 50 Series also supports Remote Job Entry, which allows the user to communicate with systems manufactured by IBM, Honeywell, Univac, and other vendors, and the Distributed Processing Terminal Executive (DPTX), which allows the Prime user to communicate with IBM 3270 equipment. These two packages make the 50 Series compatible with many mainframe systems in widespread use, allowing users to offload work usually performed on a large system onto the Prime system. These capabilities, coupled with those provided by PRIMENET, ensure that the 50 Series systems can efficiently handle almost any communication and networking need.

WIDE RANGE OF SOFTWARE AND HARDWARE PRODUCTS

Prime offers many software and hardware products that allow the user to tailor a system to the exact needs of an application. Products include several database management systems, integrated development tools, and networking packages. Other types of software products offered are programming languages and language support utilities, CAD/CAM packages, communications systems, and Prime's Office Automation, as well as many third-party applications packages.

The hardware products for the 50 Series are equally versatile. Depending on requirements, the user may choose from three types of disks, four magnetic tape drives, and a large number of terminals, printers, plotters, and other peripherals. In general, these products are compatible with all members of the 50 Series family.

SUMMARY

This section has highlighted some of the general features shared by all members of the 50 Series. These are not the only exceptional aspects of the 50 Series systems; many more are described in detail throughout this book.

About This Book

All the 50 Series systems focus on the user's needs to provide a full spectrum of solutions. The descriptions in this book demonstrate the technical efficiency and sophistication of the 50 Series systems, and highlight the many practical benefits they offer. This book presents two closely related views of Prime's 50 Series systems:

- One view provides a technical description of the features of the 50 Series systems. This view shows the advanced architecture concepts built into the 50 Series systems, and lists the software and hardware products that they support.

- The second view demonstrates how 50 Series systems answer the needs of users in commercial and scientific environments. This view shows how the architecture and software/hardware products can be used to fit a variety of needs and applications.

The Technical Summary is organized as follows:

- Chapter 1 provides a high level view of the 50 Series systems, explaining the design features and philosophies they all embody.

- Chapters 2 through 10 describe the 50 Series architecture and the operating system, PRIMOS. These chapters show how the architecture and PRIMOS work together, focussing the design philosophies of the 50 Series on specific areas of machine implementation.

- Chapters 11 and 12 show the array of software and hardware products supported by the 50 Series, with brief descriptions of many of these products.

OTHER USEFUL BOOKS

For more information on some of the topics discussed in this book, you may find it useful to refer to:

- Prime User's Guide, (DOC4130-190), which introduces the new user to PRIMOS, and to Prime's file system, utilities, compilers, and subroutine libraries. This book also explains how to use the rest of Prime's user documentation.

- PRIMOS Commands Reference Guide, (FDR3108-190), which describes the format and usage of all PRIMOS user commands.

- System Administrator's Guide, (DOC5037-190), which contains information about system planning, resource allocation, and system security.

- PRIMENET Guide, (DOC3710-190), which explains Prime's networking system.

- The guides for users of the programming languages, software and hardware products in which you are interested.

2 The Central Processing Unit

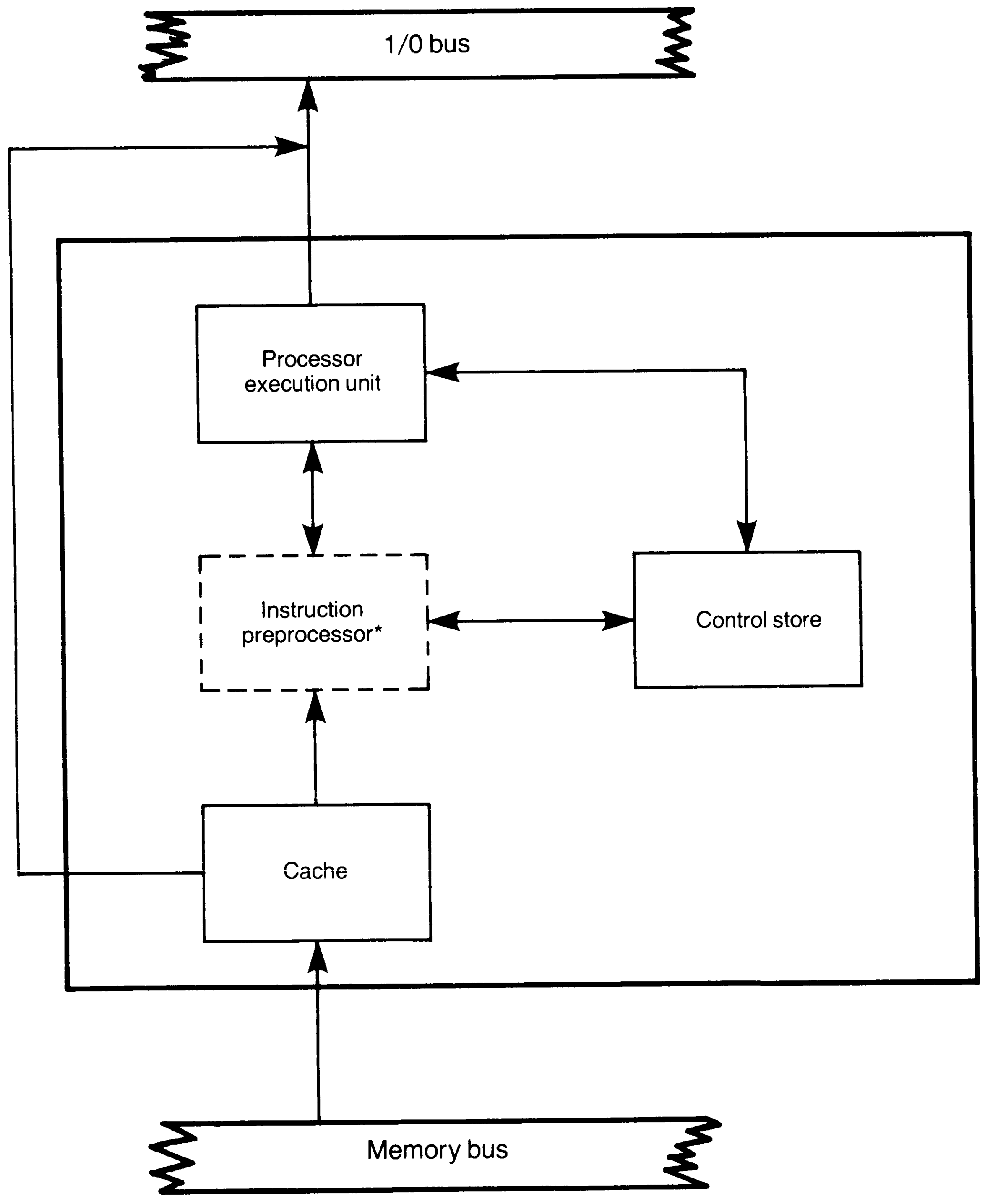
The central processing units (CPUs) of all 50 Series systems share a common architecture and one operating system, PRIMOS. This commonality is what makes the 50 Series a line of completely upward- and downward-compatible systems. The implementation of the common architecture, however, is slightly different for each member, allowing the 50 Series systems to address a wide variety of user needs as well as remain compatible. The first part of this chapter explores the single-stream CPU implemented on the 2250, 250-II, 550-II, and 750. The second part discusses the dual-stream 850 CPU.

SINGLE-STREAM ARCHITECTURE

The CPU can be divided into four major units. The first three of these are implemented on all single-stream members of the 50 Series family:

- Cache memory
- Control store
- Processor execution unit

The fourth, the instruction preprocessor unit, is a feature of the 750 (and dual-stream 850) systems only. It serves as a speedup mechanism to enhance the rate of throughput.



* = 750 and 850 only

Block Diagram of Single Processor Architecture
Figure 2-1

Cache and STLB

The 50 Series architecture incorporates a large virtual memory. This means that whenever a user specifies a virtual address, the system must translate it into the address of a physical location in memory, and then fetch the data from memory. The 50 Series systems contain a cache and a segmentation table lookaside buffer (STLB) to speed up the memory reference/address translation process.

The 50 Series uses a virtually addressed, write-through cache. Each of the cache entries contains the contents of and additional information about two consecutive bytes (2250, 250-II and 550-II) or four consecutive bytes (750 and 850) of recently accessed physical memory. If the contents of a specified location can be found in the cache, the system saves a great deal of time; it takes only 80 nanoseconds to access a cache entry, a vast improvement over the approximately 600 nanoseconds needed to access physical memory. The time saved can be spent performing other operations rather than waiting for a memory reference to complete.

To speed up the virtual-to-physical translation, the STLB contains the results of the last 64 address translations. Since programs tend to reference the same set of locations during their execution, the system can perform a translation once, store the result in the STLB, and then have it for reference the next time the user specifies the same location. Since the STLB has a much faster access time than physical memory does, referencing it saves translation time as well as access time.

See Chapter 4, Memory Management, for more information about accessing the cache and STLB, and about address translation.

The Control Store Unit

To speed up execution, the 50 Series systems implement many functions in hardware and firmware, such as procedure calls (see Chapter 8). The firmware that governs instruction execution is contained in the control store ROM. Each 50 Series system can support up to 128 Kbytes of firmware address space.

The Processor Execution Unit

This unit performs the computation required during instruction execution. Elements of the processor execution unit include:

- Integer arithmetic logic unit (ALU)
- Decimal ALU
- Floating point unit
- Register file
- Program counter

Figure 2-2 shows an expanded block diagram of the processor execution unit.

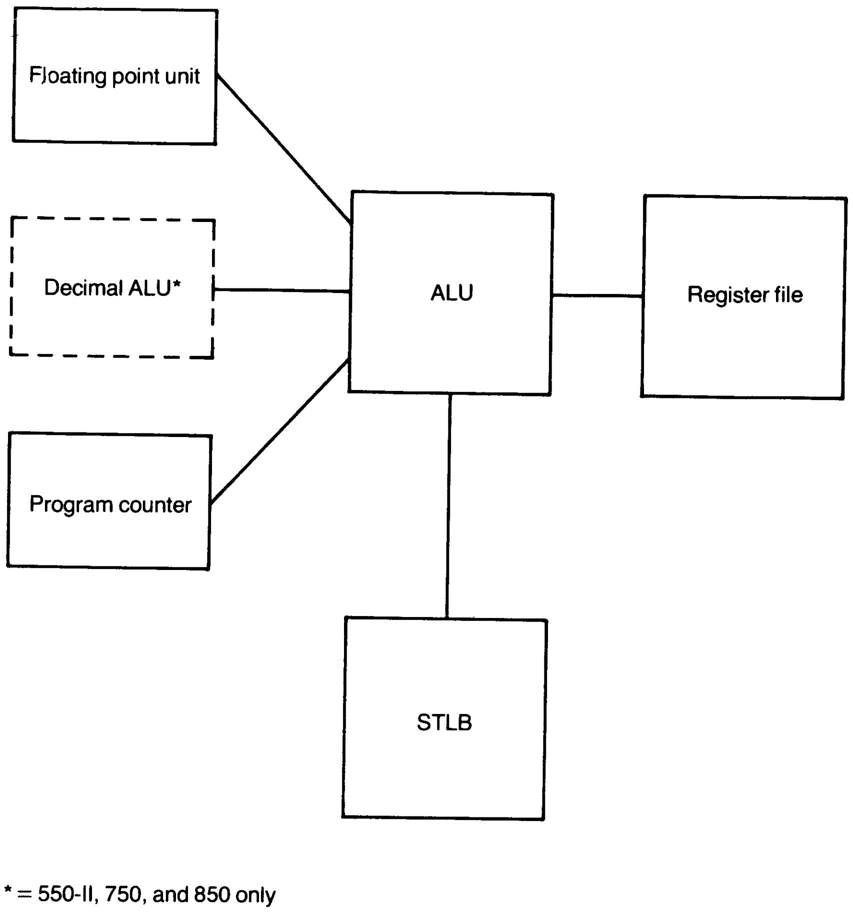
The integer arithmetic logic unit (ALU) performs the desired operation on the user's two's complement data. In a similar fashion, the decimal ALU and the floating point unit handle decimal and floating point operations, respectively. These units can perform tests and checks as well as arithmetic operations.

The register file contains four sets of registers, each containing 32 32-bit registers. Two of these are user register sets that contain information about a process and the system as the process sees it. These user register sets contain information about the general registers a process can use, addresses of fault handlers, contents of system registers, and other useful information.

As an example of the benefits two register sets provide, suppose one process is running, using one of the register sets, when an interrupt from a second process occurs. The system begins to run the second process, using the second set of registers to contain that process's state information. The second process completes, and the first process is to resume execution. Since the second process did not change the contents of the first process's register set, the system does not have to restore any data before resuming execution of the first process. The availability of two register sets avoids the need for saving or restoring complete copies of a process's state 98% of the time.

One of the remaining register sets contains microcode scratch and system status registers. The fourth set contains direct memory access (DMA) channels to speed I/O operations (see Chapter 6).

The program counter contains the address of the next instruction to be executed.



Processor Execution Unit
Figure 2-2

The Instruction Preprocessor Unit

A special Instruction Preprocessor Unit, on the 750 and 850, is designed to speed up execution. The unit does this by processing as much information about the next two instructions as possible before it is needed. While the processor execution unit is executing one instruction, the instruction preprocessor unit is decoding the next immediate instruction, calculating what the next address will be and determining what registers, if any, are to be accessed. The preprocessor unit is also fetching the second next instruction from the cache so that it is ready to be decoded when the next immediate instruction begins to execute. When the processor execution unit completes the current instruction, the instruction preprocessor has in most cases calculated enough to allow the processor execution unit to execute the next immediate instruction without delay.

DUAL-STREAM ARCHITECTURE

The 850 system implements a dual-stream version of the 50 Series architecture, which provides 60-80% more service than the 750. Figure 2-3 shows a block diagram of the 850 dual-stream architecture.

Instruction Stream Units

The 850 contains two instruction stream units (ISUs), each of which is similar in capabilities and power to a 750 CPU. Each ISU executes an independent stream of instructions simultaneously, synchronized by a Stream Synchronization Unit (SSU, see below). Each ISU is responsible for:

- Full instruction decode
- Effective address calculation
- Instruction execution
- Calculation of data for the next instruction

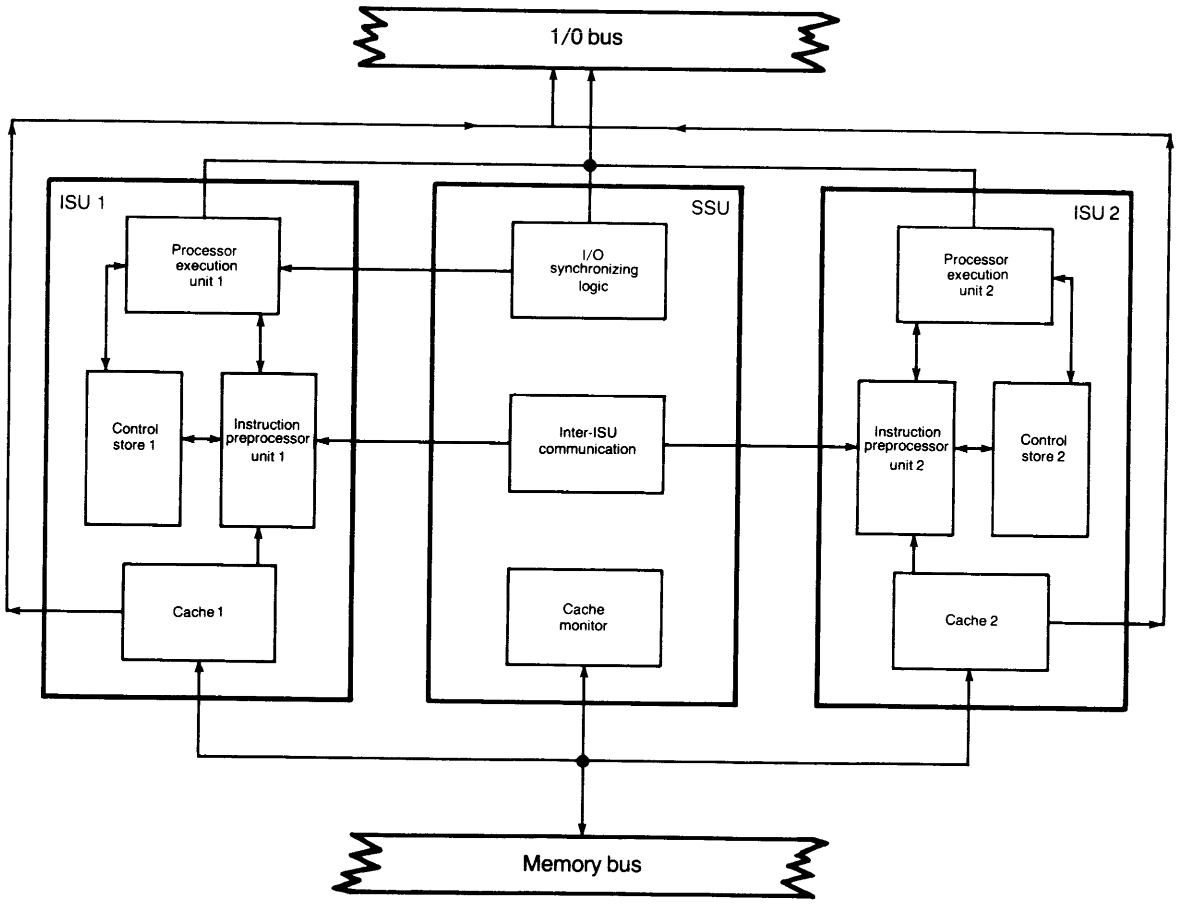
The four blocks shown in each ISU contain the same elements and perform the same functions as those described in the first part of this chapter.

The two ISUs share one copy of the operating system. PRIMOS is reentrant and can run on either ISU (as can any user program), so duplicate copies are not needed. System actions are also simplified, since there is no need to check for or handle discrepancies caused by different versions of the operating system.

The Stream Synchronization Unit

The primary task of the SSU is to prevent improper information from being loaded into the cache of either ISU. It does this by maintaining a list of the contents of both caches; when data is written into either cache, the SSU can detect it and invalidate the contents of the appropriate entry in its list of cache contents. This means that the SSU is always aware of which cache locations contain correct information and which do not.

When a cache location in one of the ISUs contains information that is out-of-date, the SSU notifies that ISU of the discrepancy. That ISU invalidates the stale entry, which forces a memory read to the current information the next time that location is referenced.



Dual-Stream Architecture
Figure 2-3

In addition to synchronizing cache references, the SSU also coordinates references to memory and system handlers. The two ISUs share one main memory, one operating system, and one copy of several system handlers. To ensure that these resources are used effectively and efficiently, the SSU contains four locks.

The process exchange lock aids the process exchange mechanism (see Chapter 4) so that control transfers smoothly between processes on both ISUs. The queue lock makes sure that simultaneously executing queue instructions (one on each ISU) are both given access to the specified queue. To guarantee that the one set of check handlers services all checks, the check lock allows only one ISU to signal a check at a time. The fourth lock, the mutual exclusion lock, can be used by software to prevent both ISUs from trying to access a particular procedure or piece of data at the same time.

Diagnostic operations and communications between ISUs are also handled through the SSU. The former feature aids in system monitoring and testing; the latter enhances the 850's ability to execute independent instruction streams without high system overhead.

SUMMARY

This chapter introduced the CPU as the heart of the 50 Series systems. It described how the common architecture implemented on each family member makes the 50 Series totally upward and downward compatible, yet applicable to many different needs. The rest of the chapters in this part explore various aspects of this common architecture. The next chapter, Process Management, begins the architecture discussion with an overview of processes and how control is transferred between them.

3 Process Management

The last chapter described the hardware elements of the system. It showed the various pieces that work together to provide a variety of resources. The means by which the user can invoke these resources is called a process.

A process is a dynamic entity that the system recognizes and can schedule. For example, when the user logs onto the system, the act of logging in commands the system to create a process for that user. The user process acts as a vehicle through which the user can request specific actions from the system. It also identifies a set of related actions for which the system must schedule resources.

Processes are managed by the process exchange mechanism. The mechanism embodies some of the 50 Series systems' most advanced architectural features to schedule and manage many processes as rapidly and efficiently as possible. This chapter describes the elements that make up the process exchange mechanism and shows their role in the transfer of control between processes. It also explores the PRIMOS scheduler and how it determines which processes to run.

PROCESS EXCHANGE MECHANISM

The process exchange mechanism transfers control from one process to another in a smooth, well ordered fashion. In addition to these characteristics, the mechanism is also very fast, because it is implemented in firmware. The elements of the mechanism that make the control transfer possible are:

- Process control blocks
- Process abort flags
- Ready list
- Wait lists
- WAIT and NOTIFY instructions
- Dispatcher

Process Control Blocks

A process control block specifies the state of one process. During process exchange, the block holds the contents of registers and timers used by the process as well as a variety of pointers to virtual memory and to fault, control, and processor information. It also specifies the level of priority of the process, a gauge of the process' importance relative to the other processes in the system. The process control block used on the 850 also specifies which of the two ISUs last ran the process and which one should run the process next.

Process Abort Flags

Each process control block contains a set of process abort flags. These flags are used most often to signal the occurrence of an asynchronous event, such as typing a BREAK character or logging off the system. If any of the abort flags are set when the associated process is selected to run, a process fault occurs. The selected process does not run; instead, the process fault handler takes control to clear the fault.

Ready List

The ready list is a list of processes that are ready to be run. A process is ready to be run when nothing needs to occur before it can begin to execute. For example, such a process is not awaiting the completion of an I/O operation.

The ready list organizes these processes according to their relative importance, or priority, so that the most crucial process is run before all others. The elements of the ready list are:

- Linked lists of process control blocks
- Ready list headers
- Two registers

Figure 3-1 shows the relationship between the elements of the ready list.

Each process control block contains the priority level of the process it represents. Each block also contains a pointer to another process control block that is on the same priority level (if there are any). This means that all processes on the same priority level are bound together in a linked list.

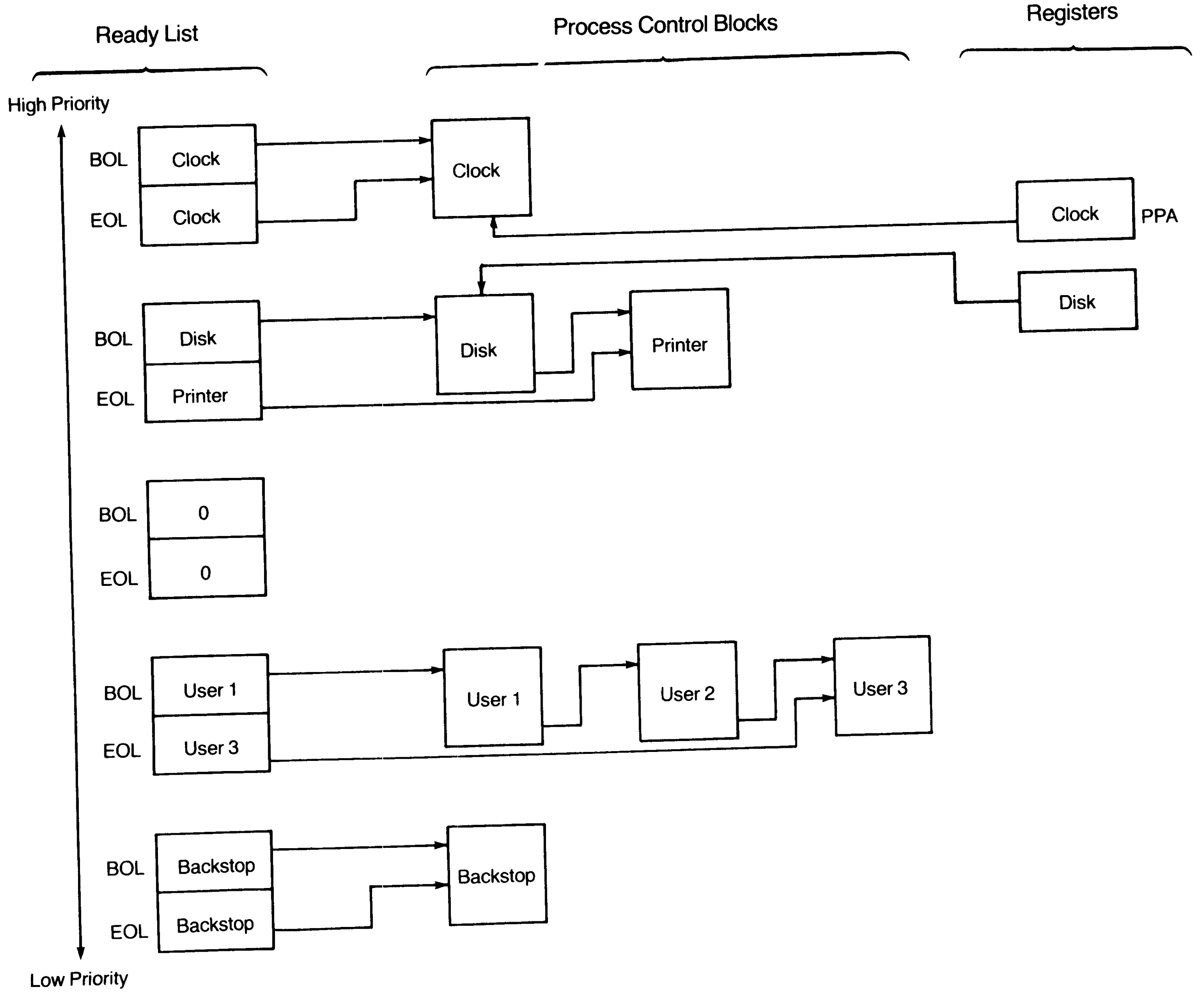
The ready list keeps track of the lists of process control blocks. The ready list is a sequential list of headers, one header for each priority level on the system. Each header contains two pointers that define the endpoints of the linked list for that level of priority. The beginning of list pointer references the first process control block in the list; the end of list pointer, the last.

The process exchange mechanism uses two registers to locate the next process to dispatch. The PPA register contains a pointer to the currently executing process. The PPB register contains a pointer to the next process to be run.

Wait Lists

Wait lists specify a group of processes that are waiting for an event to occur. The major elements of each wait list are:

- A semaphore
- A linked list of process control blocks



The Ready List
Figure 3-1

DOC6904-191

Semaphores are hardware implementations of Edsger Dijkstra's P and V operations. They define an event, such as process synchronization for access to a system resource. They insure that only a set number of users access certain system resources at a time, and that reallocation of the resource is orderly and controlled.

PRIMOS supports two types of user-accessible semaphores, explained at the end of this chapter.

Each wait list has associated with it a linked list of process control blocks. The processes represented by the process control blocks on a wait list all share the same semaphore.

Process Exchange Instructions

The process exchange mechanism uses two instructions to manipulate the wait lists. When WAIT executes, it checks the status of the event for which the process is waiting. If the event has occurred, the process keeps running and does not have to wait. If the event has not occurred, WAIT invokes the process exchange mechanism to move the process to the wait list and run a new process from the ready list.

When an event occurs, NOTIFY removes the first waiting process from the event's wait list (if there are any) and places it on the ready list. It then invokes the process exchange mechanism to scan the ready list and run a new process.

Dispatcher

The firmware routine called the dispatcher is responsible for smoothly transferring control from one process to another. When it is in control, it turns off the process interval timer, the clock that specifies how long a process has been running. The dispatcher then checks the ready list to choose the highest priority process to run next, and allocates a user register set for that process. Before dispatching the new process, it reactivates the process interval timer.

Summary

The process exchange mechanism determines which process to run next, saves the state of the old process (if necessary), locates a register set for the new process, restores the state for the new process, and transfers control to the new process. These tasks are all performed in hardware or firmware and can take as little time as 6 microseconds. They seldom require more than 24 microseconds.

THE PRIMOS SCHEDULER

The last section showed how processes wait for an event to occur. When the event takes place, the waiting process with the highest priority level moves to the ready list and may begin to run. The mechanism that controls the movement of processes between wait lists and the ready list is called the PRIMOS scheduler. The scheduler also controls the setting of default priority levels and the length of time processes can run. Elements of the scheduler are:

- Scheduler queues
- Two scheduling parameters
- Backstop process

Scheduler Queues

Each of the scheduler queues is a wait list that identifies the priority and type of process that is waiting to run. Processes with new terminal input are usually placed on the high priority queue. The processes whose execution the system temporarily suspends while it is servicing the high priority processes wait on the eligibility queue. The low priority queue is for background or CPU-bound processes.

Scheduler Parameters

Two parameters, MAXSCH and CHAP, set boundaries for process execution. The MAXSCH command specifies the number of processes that can be simultaneously active on the system. The limit MAXSCH sets prevents thrashing, which can occur if available memory resources are overcommitted. Prior to Rev 19.1, this value is initially set to 4. At Rev 19.1 and later, the value is initially determined by the formula:

$(m + 3) * x + y$

where

m is the number of megabytes of main memory.

x is 1 if the system is not using an alternate paging device or the alternate paging device and the paging device are both using the same controller, and *x* is 1.2 if the alternate paging device and the paging device are on different controllers.

y is 1 if the CPU is an 850, and *y* is 0 otherwise.

When a user process moves off one of the scheduler queues, the process exchange mechanism places the process on the ready level specified by its process control block. The CHAP command selects the default level of the ready list on which user processes are placed.

Backstop Process

The backstop process is responsible for all of the movement between the scheduler queues and the ready list. This process has the lowest priority on the system, which means that it runs only when there are no other processes to run.

Scheduler Operation

The backstop process, as mentioned above, is responsible for moving the processes from the queues to the ready list. Figure 3-2 illustrates how the backstop chooses a process from one of the three queues and moves it to the ready list.

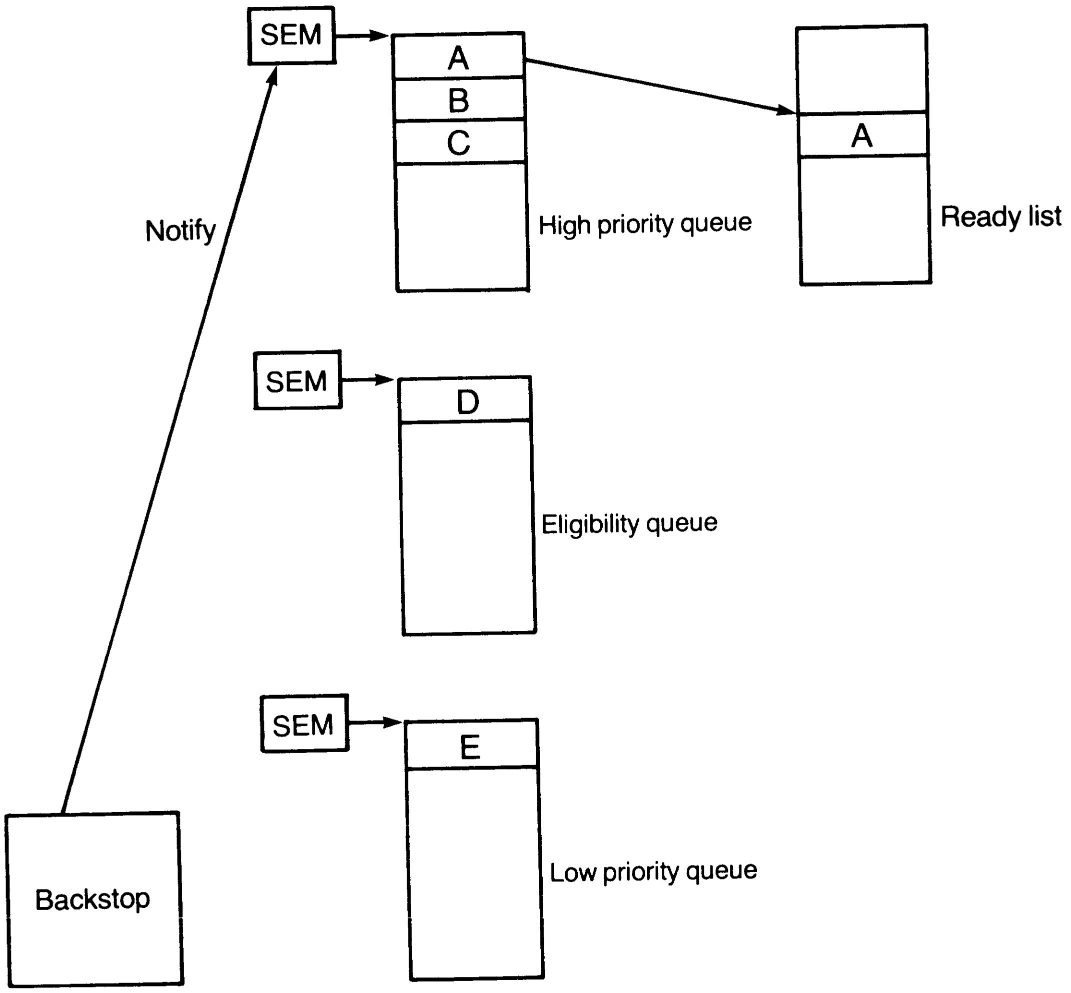
When there are no other processes in the system to run, the backstop process (which is always ready to run) begins to execute. It first checks the high priority queue to see if a process is waiting to run. If such a process exists, the backstop notifies the process, which moves it up to the ready list. The backstop is suspended immediately after the notify, since its priority is lower than that of any other process on the system.

If there are no processes on the high priority queue, the backstop checks the value of MAXSCH against the number of currently active processes. If the number of currently active processes is greater than or equal to MAXSCH, the backstop activates no new processes. If the number of currently active processes is less than MAXSCH, the backstop can activate a new process currently waiting on the eligibility queue or the low priority queue. It checks the eligibility queue first.

If there are processes waiting on the eligibility queue, the backstop notifies the eligibility queue's semaphore. This causes the first process in the queue to move up to the ready list. At the notify, the backstop is suspended as described above.

If there are no processes waiting on the eligibility queue, the backstop can activate a process waiting on the low priority queue. If there are no processes to run, the backstop idles, rechecking each queue every cycle.

When the backstop moves a process to the ready list, that process becomes the highest priority process in the system, so the backstop is suspended. Since backstop operation finishes as soon as it notifies one of the queues, its operation is never interrupted, and it never waits on any of the queues. It remains on the ready list and begins to



Scheduling Processes
Figure 3-2

execute from the beginning the next time there are no other processes on the system.

Time Slices

The scheduler uses two measures of process time to determine where to place a process on the scheduler queues.

The first measure of time is called the process's current time slice. This specifies the total length of time a process can spend on the high priority and eligibility queues before being relegated to the low priority queue. The default value is two seconds. This unit is divided into smaller units called quanta, or eligibility time slices. The default value for these smaller units is one third of a second. Figure 3-3 and the accompanying text show how the scheduler uses these values.

When a user types a carriage return, the scheduler places the user's process on the high priority queue. If it does not complete by the time the eligibility time slice has expired, the scheduler places it on the eligibility queue and will continue to do so for the rest of the process' eligibility time slices.

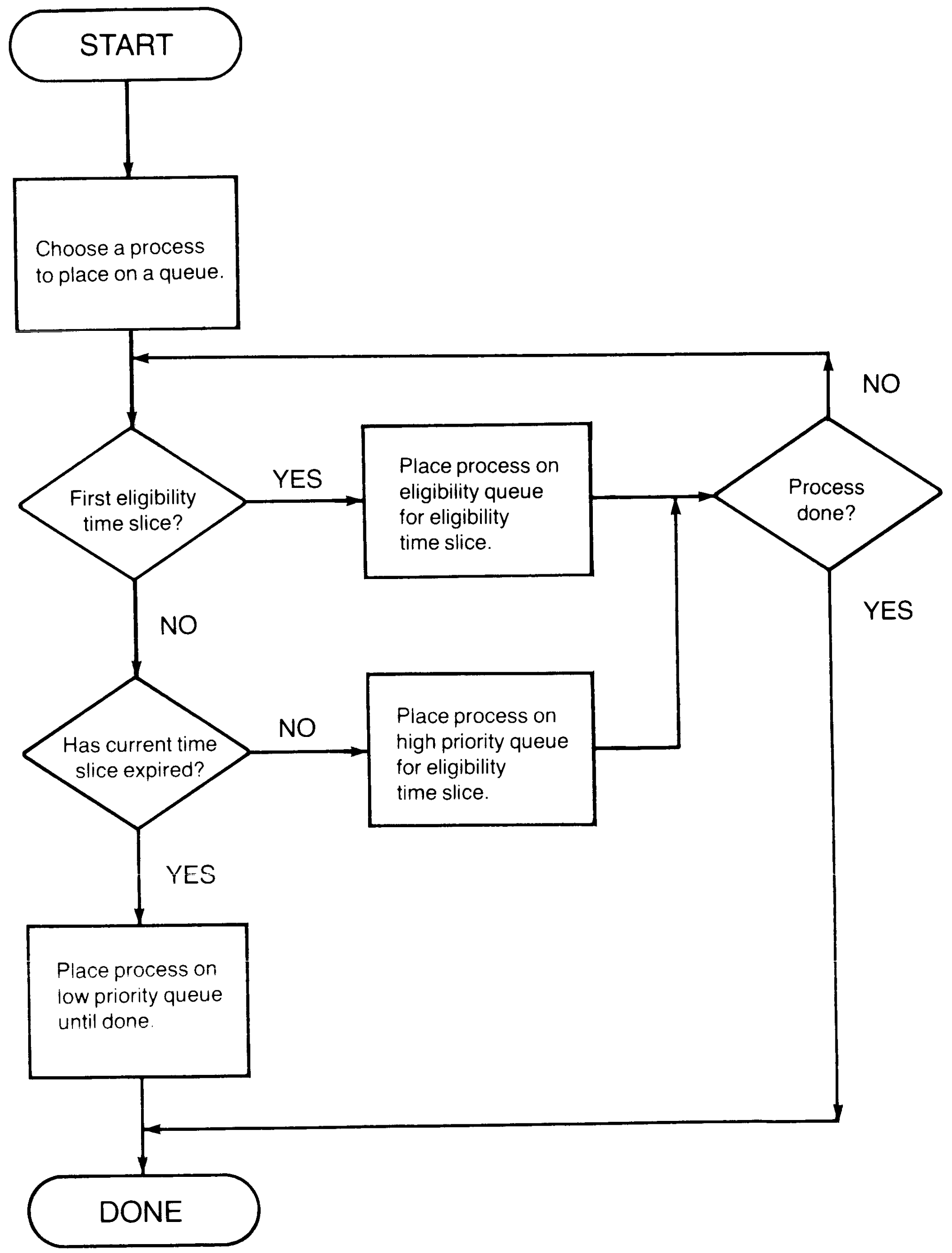
If the process does not complete by the time its entire current time slice runs out, the scheduler places it on the low priority queue where it is monitored. If it still has not completed after a predetermined length of time, the scheduler places it back on the eligibility queue. Processes requiring very long periods of CPU time to complete move back and forth between eligibility and low priority queues until they complete.

When the user types another carriage return, the user process is given a new current time slice, and the entire cycle repeats: one eligibility time slice on the high priority queue, then down to the eligibility queue for the rest of the current time slice, then down to the low priority queue.

Implementing this type of scheduler has many advantages. To the user, the most important one is provided by the high priority queue, because it ensures that a terminal command is given priority over less interactive processes.

850 Scheduler

The preceding discussion showed how the scheduler worked for all single stream 50 Series systems. The scheduler for the 850 works in the same way, with three differences. First, the PPB register is located in the SSU. Second, two backstop processes exist, one for each ISU. Third, because of the two backstops, the two highest priority processes can execute at the same time, one on either ISU. (Any process can run on either ISU). The two backstop processes use a single set of queues to move processes onto the ready list.


START
Choose a process to place on a queue.
First eligibility time slice?
YES
Place process on eligibility queue for eligibility time slice.
NO
Process done?
NO
YES
Has current time slice expired?
NO
Place process on high priority queue for eligibility time slice.
YES
Place process on low priority queue until done.
DONE

Placing a Process on the Scheduler Queues
Figure 3-3

USER-ACCESSIBLE SEMAPHORES

The actions of the scheduler and the process exchange mechanism occur transparent to the user. There are, however, two types of semaphores that the user can access if desired. These types are called numbered and named semaphores.

Numbered Semaphores

PRIMOS provides an array of 64 numbered semaphores. These semaphores, numbered from 1 to 64, allow the user to synchronize the execution of one process with that of one or more other processes. How the semaphores are allocated and used between processes is totally under the user's control, though PRIMOS does check specified semaphore numbers for validity.

The user can convert a numbered semaphore into a timer by having the system clock notify it periodically. Up to 15 of these timed semaphores can be used at once. Like regular numbered semaphores, timed semaphores are allocated and used strictly as the user defines.

Named Semaphores

The main difference between a numbered semaphore and a named semaphore is that the former is identified by number, while the latter uses a name. This name, however, allows PRIMOS to subject the named semaphore to the same access restrictions (access control lists, discussed in Chapter 6) that operate on files in the file system.

This restricted access makes named semaphores particularly useful with groups of related processes. Access can be granted to a group of cooperating processes and denied to all others. This ensures that the coordinated actions complete smoothly and use the correct information, and keeps outside events from upsetting the necessary timing.

SUMMARY

The process exchange mechanism provides a rapid, transparent, consistent way to allocate time and resources to many simultaneous user processes. This ensures that resources are not overloaded at any time, which protects the user against slow response time or system failure. It also gives the user a well-managed set of resources and the ability to access them easily without waiting. The next chapter, Memory Management, describes how the user also has easy access to physical and virtual memory resources.

4 Memory Management

The 50 Series members are virtual memory systems. This means that a very large virtual address space is available to each user logged onto the system. This virtual address space is supported by a much smaller physical address space invisible to the user.

Virtual memory has several advantages. To the user logged onto the system, there appears to be an address space of almost unlimited size which can support very large applications without using overlays. To the system owner, a virtual memory scheme provides the ease of use of a large memory at the cost of a much smaller amount of hardware.

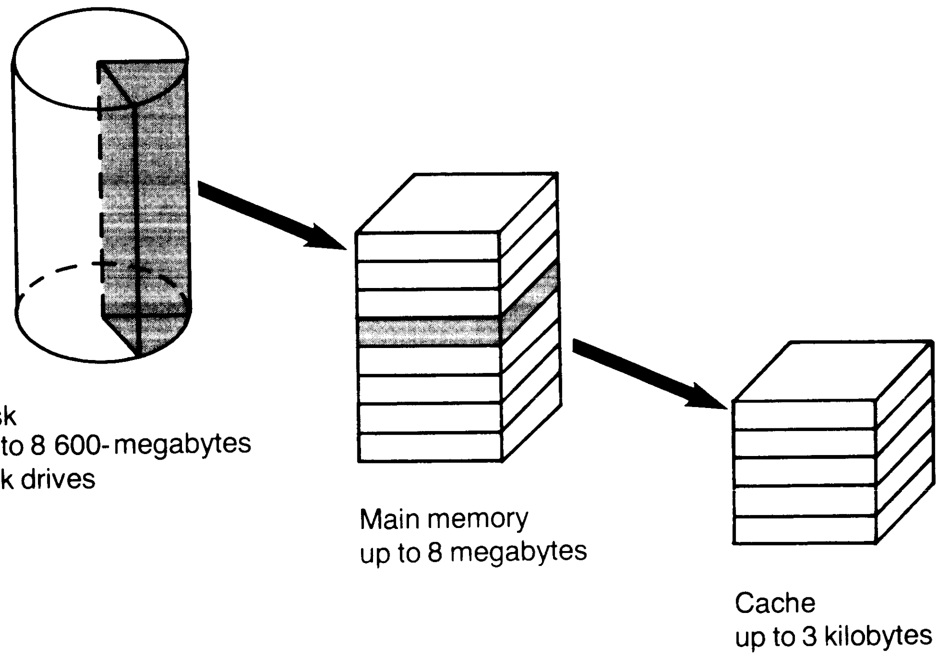
The three key parts to a virtual memory scheme are physical memory, virtual memory, and a manager to control the virtual memory scheme. In the 50 Series, this manager is PRIMOS, and its attendant hardware and firmware support. This chapter describes the characteristics of the 50 Series physical and virtual memory, and shows how PRIMOS coordinates the 50 Series virtual memory scheme. It also describes some of the hardware protection mechanisms implemented in the 50 Series virtual memory.

PHYSICAL MEMORY

Physical memory encompasses all hardware parts of the system used to store large blocks of information. The three types of physical memory are:

- Cache
- Main memory
- Disk

Figure 4-1 shows the relationship between the three elements of physical memory.



Elements of Physical Memory
Figure 4-1

Cache

The cache is a data buffer that stores copies of the information contained in the most frequently referenced memory locations. Its size varies from system to system (see Table 12-1). During program execution, this buffer is used to speed up memory references.

Since cache is a form of very high speed memory, it takes only 80 nanoseconds to access data stored there, whereas it takes about 600 nanoseconds to access data stored in main memory. This difference in access times makes it very advantageous to access cache whenever possible.

Three factors determine how often the cache contains the correct data (the cache hit rate):

- The size of the cache (2-32 Kbytes);
- The information fetch rate (16-64 bits, depending on the system and the amount of memory interleaving);
- Locality of reference (the tendency of a program to execute within a small part of itself at any time).

On the 50 Series, data can be found in the cache 85-95% of the time. See Table 12-1 for the exact figures for each 50 Series system.

Main Memory

The 50 Series main memory is high speed MOS with error checking and correction built in to correct single bit errors and detect double or multiple bit errors. The memory is packaged on boards in units of 512 Kbytes or 1 Mbyte.

All systems use two-way interleaving to speed up memory references and to make more efficient use of the memory bus. This means that consecutive physical locations are located on different memory boards; when a reference to memory is made, the system fetches the same location on each board. This doubles the amount of data that can be fetched with one memory operation. Systems with an odd number of memory boards use interleaving for all but the odd board.

Main memory is divided into units called pages. Each page is 2 Kbytes in size. The pages subdivide main memory into pieces that PRIMOS can conveniently and efficiently manage. One advantage of this subdivision is to reduce the amount of paging necessary to ensure that data is in main memory when the user needs it.

There are many other advantages to subdividing memory into pages. For example, since all pages are the same size, PRIMOS can reply to all requests for space in the same way regardless of who or what makes the request. In addition, disk records are the same 2 Kbytes in size, so transfers between main memory and disk are simplified. The section, Coordinating Physical and Virtual Memory, later in this chapter, describes many other advantages.

Disk

Disks provide storage for all of virtual memory. The system or the user can access any of this information at any time (given the proper access rights), at which time a copy of it is moved from disk to main memory. The Paging section in this chapter describes how the information is moved. The Disks section in Chapter 12 describes the physical characteristics of the disks supported on the 50 Series.

VIRTUAL MEMORY

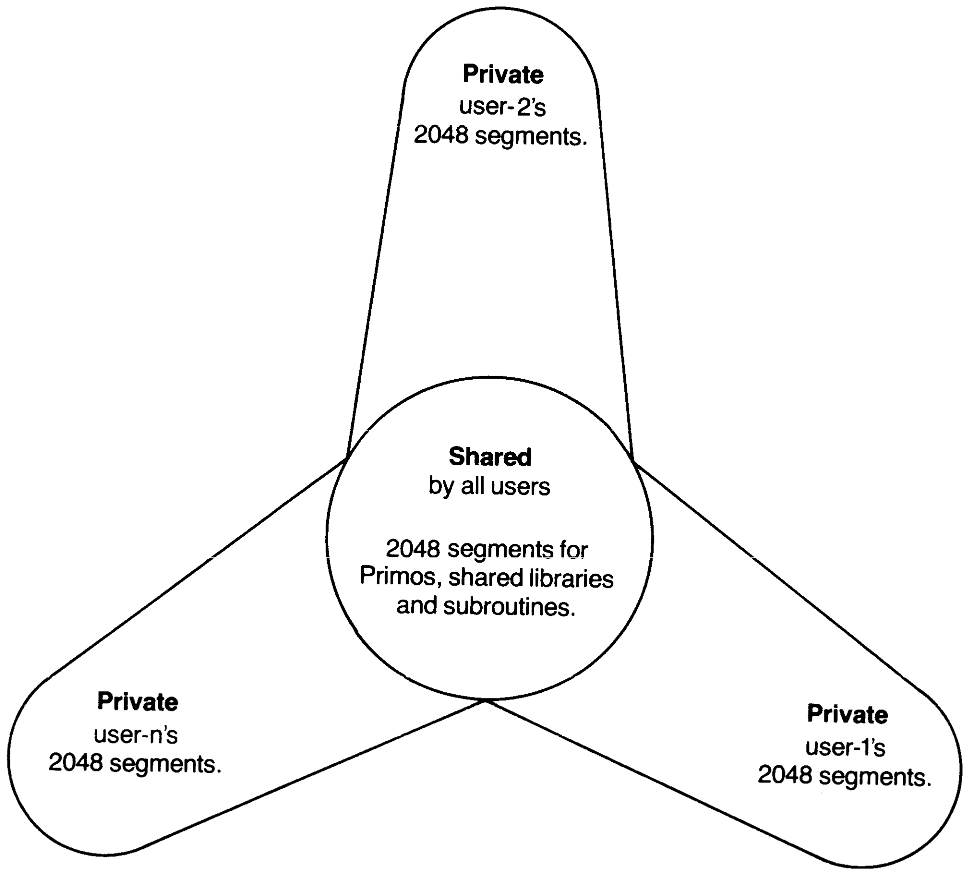
Virtual memory is divided into units called segments. Each segment can contain up to 128 Kbytes, or 64 virtual pages of 2 Kbytes each. Segments are virtual units, not physical ones, that aid both the user and the system in organizing their virtual address spaces and the information contained there. For example, the user can organize program code in one segment and program data in a second one. It is also possible to allow extra room in a program for variable length data structures, such as arrays whose dimensions can change each time the program runs. Segments also allow the user to build modular programs, one module to a segment. PRIMOS uses segments in a similar way to organize its own code into modules.

The virtual address space of each user contains 4096 segments. These are subdivided into four groups of 1024 segments each. The segments are subdivided to make address translation and segment sharing easier (see Address Translation and Shared and Unshared Segments, below).

Shared and Unshared Segments

In the Prime virtual memory scheme (see Figure 4-2), each user address space of 4096 segments is divided into shared and unshared space. The first 2048 segments are shared with all other users. This allows the operating system, shared libraries, and shared subsystems to be seen by all users. The second 2048 segments are private, containing information unique to unique to each user. This means that if two users reference Segment '4000, they are specifying completely different locations.

This arrangement of shared and unshared segments means that there is no possibility of one user's private space conflicting with that of another user. It also means that only one copy of PRIMOS and the shared system software need be maintained, which reduces memory use. Additionally, it means that PRIMOS is embedded in the virtual address space of each user and is directly accessible via a normal procedure call (see Chapter 8, Procedure Management). No interrupts, special supervisor calls, or system traps are necessary when the user accesses PRIMOS or any utility or library residing in shared space.



50 Series Virtual Memory Space
Figure 4-2

<u>Protection Rings</u>

Three hardware implemented <u>rings</u> provide a simple, unbreakable form of security that checks each memory reference for its right to access the specified part of memory.

The rings represent levels of protection. Ring 0 represents the highest level of protection and grants the greatest number of privileges. PRIMOS runs under Ring 0 protection, which means that its segments cannot be accessed by the user except through protected entrypoints, and that it has read, write, and execute privileges to all segments. PRIMOS can access any information in the system, can invoke special routines, and so on.

Users run under Ring 3 protection, which means that they cannot arbitrarily access Ring 0 routines. Each segment under Ring 3 protection may have a different combination of read, write, and execute access rights.

Ring 1 provides privileges less powerful than those of Ring 0, but more so than those of Ring 3.

To specify its degree of privilege and protection, each virtual address contains a ring field that specifies a ring number. The way ring fields guard against illegal memory accesses is explained under "Generating the Ring Number" later in this chapter.

Rings provide a simple, effective way to protect critical parts of the system. Without them, a Ring 3 procedure could directly access any Ring 0 procedure, which might corrupt system operation. Screening out such references protects the integrity of the entire system.

Segmentation Table Lookaside Buffer

Virtual memory has its counterpart of the cache, the STLB. The system uses this buffer with the cache to reduce the time needed to access information. Where a cache entry contains information about a recently accessed physical memory location, an STLB entry contains the information the system needed to find the physical location from the virtual address the user specified.

The section, Accessing the Cache and STLB, below, explains how the STLB entries are used. The section, Address Translation, shows how the STLB is loaded with updated information.

COORDINATING PHYSICAL AND VIRTUAL MEMORY

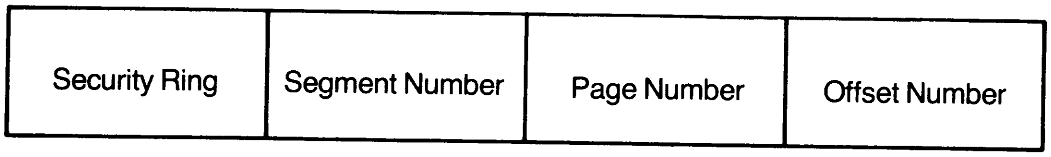
How does PRIMOS map the segmented virtual address space onto the pages of physical memory? The process starts when the user specifies a virtual address. This virtual address has the format shown in Figure 4-3.

Since this address only identifies a location within the virtual address space, not a physical location that can be referenced, PRIMOS must find the physical location of the user's information. The steps in this process are:

1. Check the STLB and cache. If both of these contain the correct information, the reference can be completed. If not, go on to the next step.

2. Translate the user's virtual address into a physical address. Once the translation is done, load the translation information into the STLB for future use, then check to see if the page containing the user's information is resident in memory. If the page is resident, the reference can be completed. If not, go on to the next step.

3. Find the correct page on disk and move it into main memory. The reference can be completed after the page is moved.



Virtual Address Format
Figure 4-3

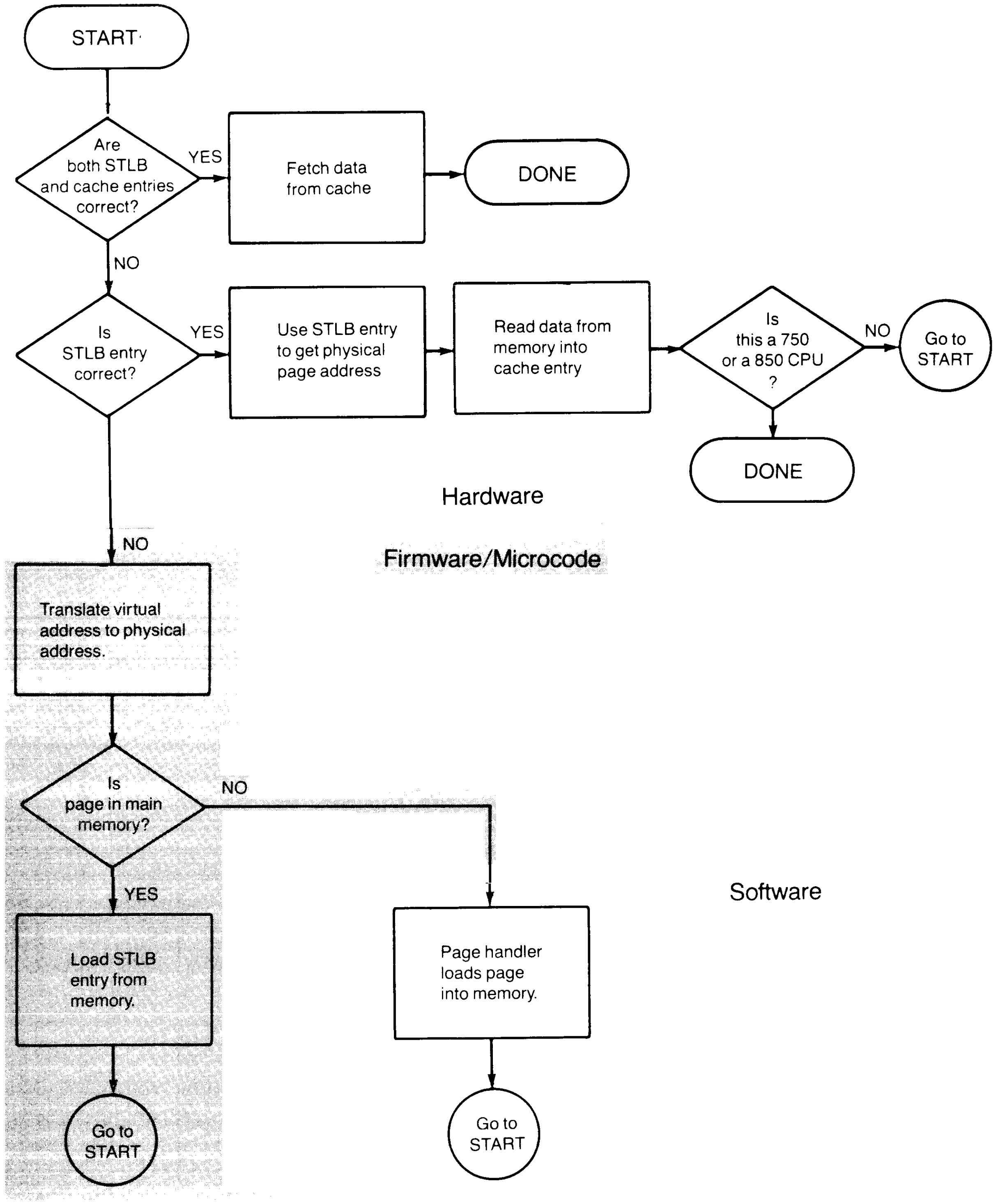
The first task is completely performed in hardware; the second, in firmware. A software page fault handler performs all aspects of paging.

Accessing the STLB and Cache

To find the user's information in the STLB and cache, the hardware accesses both buffers at the same time. The hardware uses the segment number-page number pair from the virtual address to choose an STLB entry. This entry specifies the number of the physical page containing the user's information.

At the same time, the hardware uses the contents of the offset field from the virtual address to choose a cache entry. The cache entry contains the contents of a location in main memory. If the hardware can validate the contents of both the STLB and cache entries, then the cache entry contains the user's information. In this case, the whole operation has been handled in hardware, and the correct information has been located in only 80 nanoseconds. Figure 4-4 illustrates this process.

Suppose the hardware could not validate the contents of both cache and STLB. When the STLB does not contain the correct translation, the hardware must translate the user's virtual address into the correct physical one (see Address Translation) and save the translation in the STLB. The hardware retries the reference from the beginning after the new translation is saved in the STLB.


START
Are both STLB and cache entries correct?
YES
Fetch data from cache
DONE
NO
Is STLB entry correct?
YES
Use STLB entry to get physical page address
Read data from memory into cache entry
Is this a 750 or a 850 CPU ?
NO
Go to START
DONE
Hardware
NO
Firmware/Microcode
Translate virtual address to physical address.
Is page in main memory?
NO
YES
Load STLB entry from memory.
Go to START
Software
Page handler loads page into memory.
Go to START

Accessing the STLB and Cache
Figure 4-4

If the cache does not contain the correct information, the hardware must reference memory. It takes the translation from the STLB to identify the correct physical page, then uses the offset in the user's virtual address to identify the correct address in the physical page. Depending on the amount of memory interleaving used on the system, PRIMOS loads 16 (unpaired memory boards), 32 (paired memory boards), or 64 (750/850 memory boards) bits of new data into the cache. After loading in the new data, the reference is retried from the beginning.

When the hardware needs to load information into the cache, the physical page containing that information may be on disk rather than in main memory. When this is the case, the page on disk must be moved into main memory. The section, Paging, below, describes how this is done.

Address Translation

Several data structures aid in address translation. These structures are:

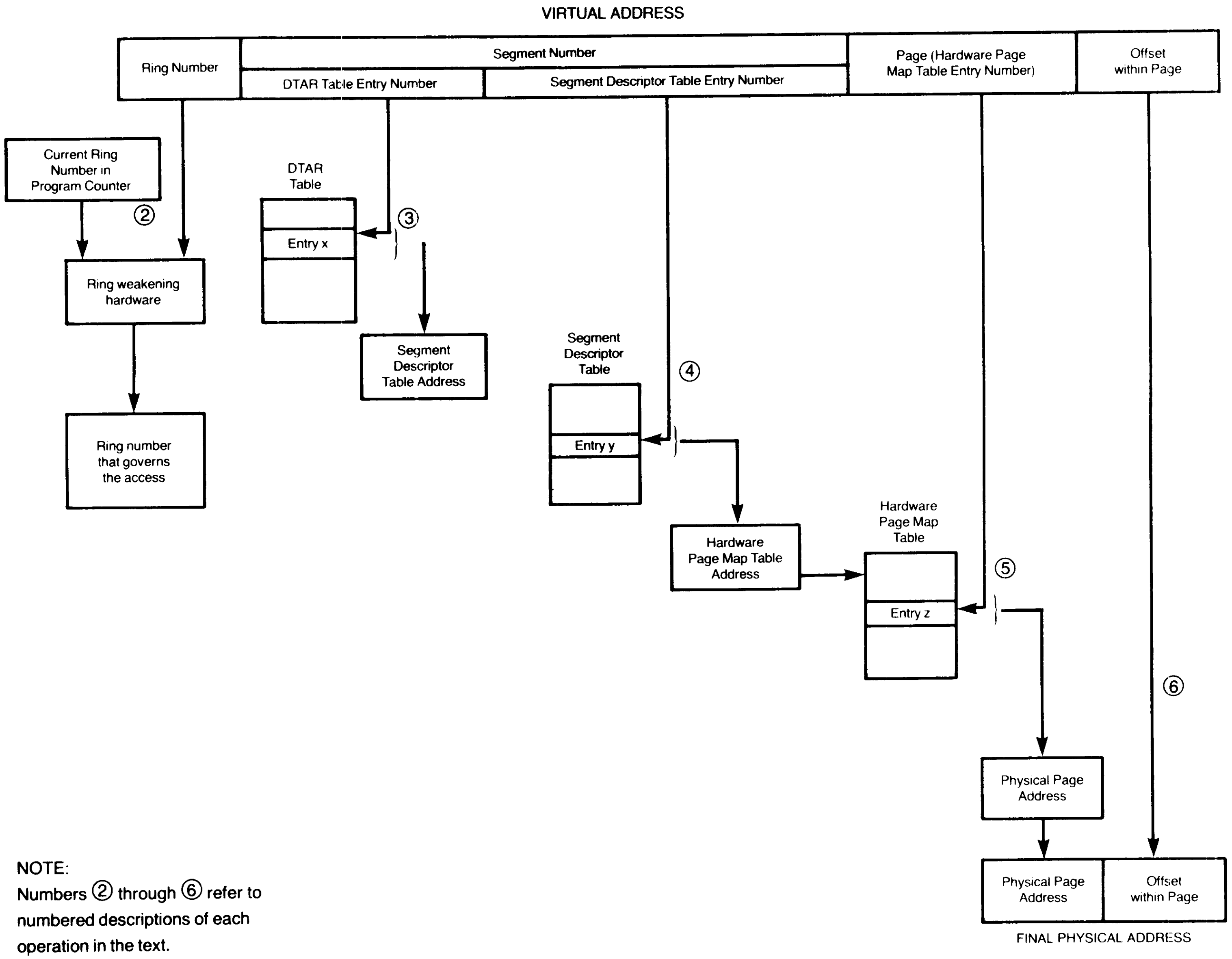
- The STLB described earlier
- DTARs, the descriptor table address registers
- SDTs, the segment descriptor tables
- HMAPs, the hardware page map tables

PRIMOS saves the 64 most recent virtual-to-physical address translations in the STLB for future use. When a translation is done, PRIMOS stores the identity of the process, the virtual address, and the physical address into the STLB as a triple. The next time that translation is needed, PRIMOS has only to read the physical address from the STLB, rather than to perform a calculation. In addition, since the STLB is a high speed memory buffer, it has a much shorter access time than does main memory. 97% of the time PRIMOS can find the necessary translation in the STLB, and thus avoid the longer main memory access time.

Each of the four DTARs describes one group of 1024 segments in the virtual address space. DTAR0 and DTAR1 specify information about the shared segments, while DTAR2 and DTAR3 describe the private segments. All four DTARs are located in the user's register file (see Chapter 2).

Each of the four DTARs contains a pointer to a segment descriptor table (SDT) in main memory. These SDTs contain from 0 to 1024 entries, each of which describes one segment. Contained in each entry is the address of a hardware page map table (HMAP), and segment access information.

Each of the 64 entries in an HMAP describes one virtual page. The entry specifies the location of the virtual page in physical memory, as well as access and control information.



NOTE:
Numbers ② through ⑥ refer to numbered descriptions of each operation in the text.

Figure 4-5
Address Translation

1. Interpreting the Virtual Address

   When the user specifies a virtual address, a firmware routine interprets it as shown in Figure 4-5.

2. Generating the Ring Number

   A section of hardware forms the ring number for the address translation firmware. It does this by logically ORing the current ring number (contained in the program counter) and the ring number contained in the virtual address. This is called weakening the ring number. Since higher numbered rings cannot access lower numbered ones, weakening ensures that the access is granted according to the highest ring number of the program counter/virtual address pair.

3. Referencing the DTAR

   Once the firmware has the weakened ring number, it checks the DTAR field. This value specifies one of the four DTARs. The contents of the selected DTAR specify the starting address of one of the SDTs.

4. Referencing the SDT

   The firmware now knows which SDT to reference. It uses the segment field of the virtual address to choose an entry in the table. If the entry is not in this variable-length table, a segment fault occurs. If entry is present, it contains a set of access rights which the firmware uses to determine if the access can be allowed. If the access is invalid, an access violation occurs; otherwise, the firmware uses the rest of the information in the SDT to reference an HMAP.

5. Referencing the HMAP

   The firmware now knows which HMAP to reference. It uses the page field in the virtual address to choose an entry in the HMAP. This entry contains status information about one page, such as whether it is currently in memory or on disk. This discussion assumes the page is loaded into physical memory (see the next section, Paging, for a discussion of actions taken when the page is on disk).

6. Referencing the Physical Memory Location

   Besides the status information, the HMAP entry specifies the physical page address of the page in main memory.

Knowing this, the firmware uses the offset field in the virtual address to identify the exact address in the page to reference.

The firmware saves the result of this translation process in the cache and the STLB in case the user wants to reference the same location again.

Figure 4-5 illustrates these data structures and shows how the address translation process uses them.

Paging

When a requested page is not in main memory, a page fault occurs. Often, a page must be moved out of main memory and onto disk so that the new page can be loaded in. Several data structures exist to assist the page fault handler with moving pages between main memory and disk. These structures are:

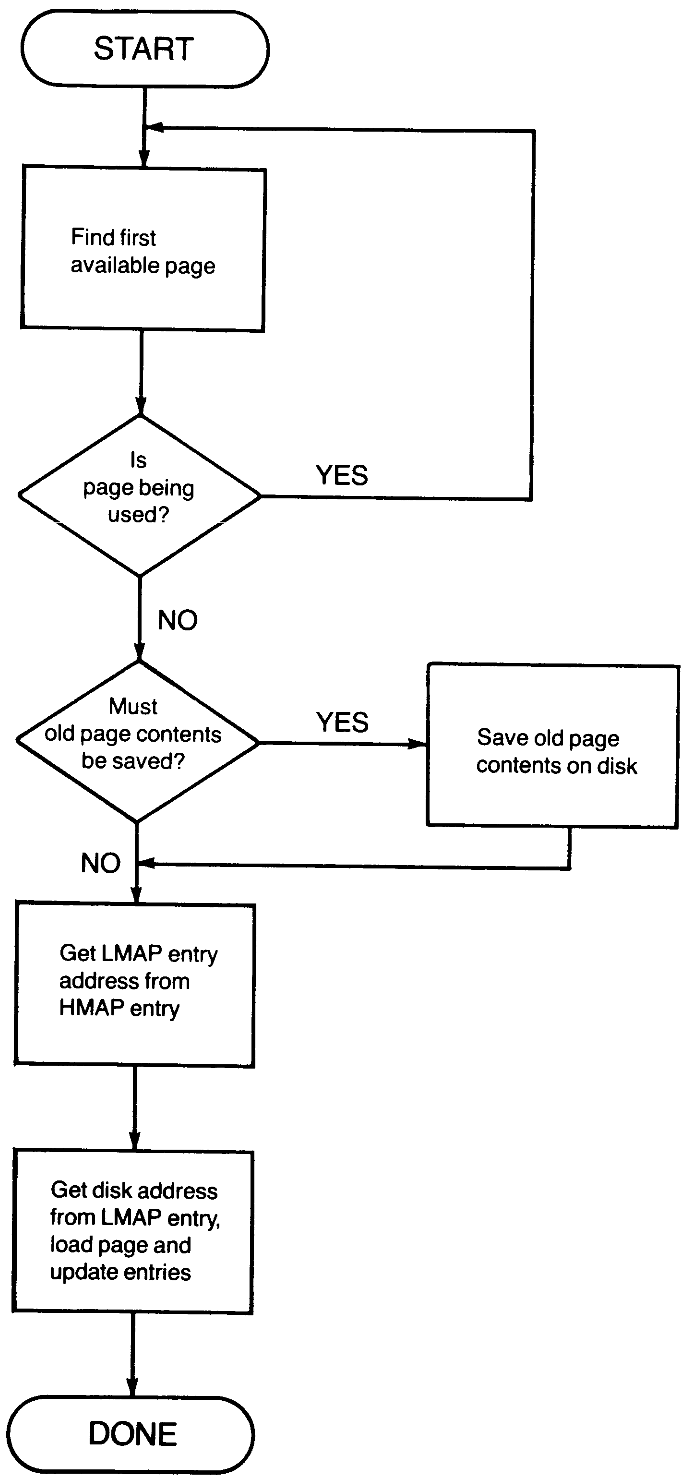
- LMAP, the logical, or disk record, address map
- HMAP, the hardware page map
- MMAP, the memory map

LMAP contains information about pages held on disk. Each LMAP entry specifies the disk location and various control information for one page. One LMAP entry is associated with each HMAP entry.

The HMAP table contains 64 entries, each of which contains information about one page. Besides control information, each entry specifies either the physical memory address of a page (if the page is in memory), or a pointer to an entry in LMAP (if the page is on disk). One HMAP exists for each segment.

Each entry of MMAP describes one physical page and whether it is already in use, available for use, or does not exist. The first two descriptions, page in use and page is available, are self explanatory. The last, page does not exist, indicates that the system is not currently accessing this page. This means that the system can still run even if part of physical memory has a problem or does not exist.

Figure 4-6 illustrates the stages in the paging process, including allocating a page, saving the old page contents, and loading the new page.


START
Find first available page
Is page being used?
YES
NO
Must old page contents be saved?
YES
Save old page contents on disk
NO
Get LMAP entry address from HMAP entry
Get disk address from LMAP entry, load page and update entries
DONE

Paging
Figure 4-6

Allocating a Page: Before the page fault handler can load the page into main memory, it must locate a place to put it. It checks MMAP for an available page. If there are no available pages, it uses a least recently used algorithm to choose a page to move out of main memory. The LRU algorithm assures the handler that the page to be moved out of memory is not one currently being used by another process.

The handler can be configured to check for more than one currently available page with the LRU algorithm. If this is the case, the handler identifies several least recently used pages and prepares to move the page out of main memory. This is called prepaging, and can speed up processor execution by paging out several pages during one page fault. When the next page fault occurs, the handler has only to load the new page in without having to clear a space for the page first.

Saving the Old Page Contents: After the handler identifies available physical pages, it must choose a page to page out, and determine if it must store the old page contents on disk before loading in the new information. The HMAP entry aids in this task.

Each HMAP entry specifies several things about its associated page. One of the things it specifies is whether its page has been used since this entry was last reset. If the page has not been used, then the handler can move it out without adversely affecting any running processes. If it has been used, the handler should locate another page to move out, since it is likely a process is using this page.

When the handler chooses an unused page to page out, it checks the HMAP entry again to determine if the page's contents have been modified since it was moved into main memory. If the old contents have not been modified, the handler can load in the new contents immediately. If the old contents have been modified, the handler must save a copy of them on disk first before loading in the new information.

Like the LRU algorithm, the HMAP entries save the system processing time by limiting the number of disk accesses necessary to page in new information. By checking the entries periodically and tracking how they change, the handler can determine the best page to swap out of main memory.

Some pages can be locked against being paged out. Part of the LMAP entry associated with each physical page can be set so that the associated page always remains in main memory and is not overwritten. This type of page usually contains system data such as mapping tables, I/O buffers, or some of the data structures described in this chapter. By locking these contents into main memory, the system can always be sure to access the correct information when it cannot stop to handle a page fault.

Loading the Available Page: Once the handler has a physical page available, it must find the disk location of the page to be moved into main memory. Associated with the HMAP entry is an LMAP entry, and this LMAP entry contains the page's disk address. The handler uses this address to fetch a copy of the information, then loads the copy into the available page. After the move, the handler updates the affected entries in the SDT, HMAP, LMAP, and MMAP.

SUMMARY

This chapter described the structure of physical and virtual memory and PRIMOS's interface between them. It showed what happens within the system when a user requests a piece of information. Through use of specialized data structures and algorithms, PRIMOS locates the user's information in the fastest and most efficient way possible. The next chapter, Input/Output, shows how the user's information, once located, moves between main memory and system peripheral devices.

5 Input/Output Management

Because the user's address space is much larger than physical memory, physical memory is used to store only information that will be used immediately. All other information is stored on peripheral devices such as disks, where it can be quickly accessed when needed. This chapter explains how information is transferred between these devices and main memory and how the devices request I/O service via interrupts. It also briefly describes the actions of the device interrupt managers that service interrupts.

TYPES OF I/O

Depending on the application, the quantity of information to transfer, and the speed required, I/O can take one of three forms:

- Programmed I/O (PIO)
- Direct memory I/O (DMx)
- Burst mode I/O

Programmed I/O

Programmed I/O transfers data between memory and a device, 16 bits at a time. It is invoked by executing one of the four programmed I/O instructions, each of which specifies a basic operation: move data in, move data out, initiate a control pulse, or test for a skip condition. One instruction must execute for each data word to be moved.

The operating system, PRIMOS, uses programmed I/O to initialize controllers and to specify parameters for DMx transfers.

DMx I/O

While programmed I/O is suitable to use when only small amounts of data need to be transferred, it is not practical for multiple word transfers. DMx operations allow devices to access all of memory without software intervention. This means that DMx operations transfer blocks of data very quickly compared to PIO.

There are four types of DMx transfers:

- DMA, or direct memory access
- DMC, or direct memory channel
- DMT, or direct memory transfer
- DMQ, or direct memory queue

All four types support data transfers to all parts of physical and virtual memory. A mapped I/O scheme allows data blocks that are virtually contiguous but physically disjoint to be automatically collected in the correct sequence for transfer.

DMA is useful for bulk data transfers when speed is important. Devices use one of the 32 DMA channels located in the system's register set to transfer blocks of up to 8 Kbytes directly to or from memory. Devices such as the MDLC, and the disks and some tape drives described in Chapter 12 use DMA.

To transfer data blocks larger than 8 Kbytes, or to transfer data between a device and several noncontiguous areas of memory, the system can chain DMA channels together. All the system needs to do is to specify the starting channel and the number of subsequent channels to transfer up to 128 Kbytes of information with one I/O operation.

DMC operates in much the same way as DMA does. The differences are that DMC provides up to 32,768 channels rather than 32, and that data blocks can contain up to 128 Kbytes. Also, the DMC transfer rate is slower than that for DMA, because DMC channels are contained in memory rather than in system registers. Typical devices using DMC to transfer information are printers and magnetic tape drives.

Unlike DMA and DMC transfers, DMT operations do not require register sets or memory channels to control the data flow, since the device controller itself governs the transfer. This type of controller is more sophisticated than those that govern devices using DMA or DMC transfers. Typical operations that use DMT transfers are downloading disk channel programs to control DMA I/O.

DMQ operations use a circular, double-ended buffer called a queue to hold data to be transferred. DMQ is particularly useful for serial line output, since the system loads characters into a queue as it generates them. No per-character interrupts are needed, so the system does not have to concern itself with how fast information is transferred from the queue. This results in low system overhead for asynchronous terminals.

Table 5-1
DMx Transfer Rates and Transfer Sizes

Transfer Type	Maximum Transfer Rates	Max. Transfer Size
DMA	2.4 Mbytes/sec (input) 2.0 Mbytes/sec (output)	8 Kbytes
DMC	1.2 Mbytes/sec (input) 1.1 Mbytes/sec (output)	128 Kbytes
DMT	2.8 Mbytes/sec (input) 2.2 Mbytes/sec (output)	determined by the controller
DMQ	*300 Kbytes/sec (input) *300 Kbytes/sec (output)	128 Kbytes
	*approximate values	

Burst Mode DMA I/O

The 750 and 850 systems provide burst mode DMA operations for very fast data transfers. Each of these I/O operations uses one of the 32 DMA channels to control the transfer. The difference between burst mode DMA and non-burst mode DMA operations is that burst mode transfers eight bytes of data at a time, rather than just two. This enables burst mode DMA to transfer data at a rate of 5 (output) to 8 (input) Mbytes per second. High speed disks and tapes typically use this type of I/O to transfer information.

REQUESTING I/O SERVICE

When a device wants to transfer information or to signal the system, it must request I/O service by generating an interrupt.

Disks and tapes are typical devices that request I/O service with an interrupt. When an interrupt occurs, the first action performed is a control transfer (via the process exchange mechanism) to the software that services the interrupt. The details of the control transfer vary with the state of the system and the type of controller, but the result is the same: a software routine called phantom interrupt code is designated to service the interrupt, and it begins to execute.

Phantom Interrupt Code and Device Interrupt Managers

Phantom interrupt code is usually a very brief routine that acts as a screen to identify and service all interrupts that require only minimal response. When an interrupt requires more than minimal handling, the phantom interrupt code notifies the device interrupt manager (DIM) for that particular device.

A device interrupt manager acts as the interface between the user and a device controller. It manages all aspects of a device's operation, such as signalling interrupts and buffering data. It also provides any specialized interrupt service the device may require. The phantom interrupt code ensures that the device interrupt manager is invoked only when it is really needed, and then with a minimum of delay.

Handling Interrupts

The system notifies the device interrupt handler of interrupts according to a device's level of priority. This level of priority is determined by the board slot that the device occupies in the system; the device occupying the slot nearest the bottom of the I/O chassis has the highest level of priority. The order in which the interrupts are serviced, however, depends on the process levels of priority specified on the ready list (see Chapter 3).

DISK I/O

I/O transfers to and from disks are a crucial part of system operation. Prime 50 Series systems have optimized these types of operations in three ways to enhance overall system performance: overlapped seeks, ordered seeks, and overlapped transfers.

A 50 Series system can contain up to 2 disk controllers, each of which can coordinate operation of four disks. These controllers are managed by two device interrupt managers (see above). Since these DIMs can be executing on the system at the same time, they can each be initiating disk operations at the same time. In addition, if there is more than one outstanding seek operation, each DIM can direct a number of its disks to begin seeking at the same time, so that the seek operations overlap.

In addition to being overlapped, the seek operations are directed to perform all seeks in one direction as a group, then all those in the other direction. The shorter seeks are generally done first, but since all the seeks are grouped by direction, even those involving very large amounts of data are guaranteed to be performed. This is called ordered seeking.

Since the two disk DIMs can execute at the same time, the disk controllers they govern can be sending information over the I/O bus at the same time. Since there are up to two controllers per system, up to two transfers can overlap at once.

ASSOCIATIVE BUFFERS

Many of a device's interrupt requests occur because the device needs additional information before it can continue operation. The file system, described fully in the next chapter, often has to fetch this information for the device from disk. To minimize the time it takes to provide this information, the 50 Series supports a group of associative buffers.

When the file system accesses a disk, PRIMOS loads the specified file record into one of the associative buffers. The buffers serve as a speed up mechanism, in much the same way the cache does. Where the cache avoids accesses to main memory, the associative buffers cut down the file system's accesses to disk. The initial disk reference loads the associative buffer, so all subsequent references to the same information can be made to the buffer rather than to the disk.

PRIMOS updates the associative buffer each time a file write is specified. PRIMOS will automatically rewrite the buffer, however, at least once each minute (or whenever requested) to maintain the file's integrity on disk.

The associative buffers are maintained on a least recently used (LRU) basis. This means that if all buffers have been used and new data is to be loaded, PRIMOS picks the buffer that has not been used for the longest length of time.

SUMMARY

Coordinating the smooth flow of information between the user and the system is a demanding task with two main requirements. One of these, an efficient method of transferring information, was described in this chapter. The other, an efficient method of organizing information within the system, can further speed up transfers by making information easy to find. The next chapter, File Management, illustrates how Prime's file system uses a hierarchical structure to organize information.

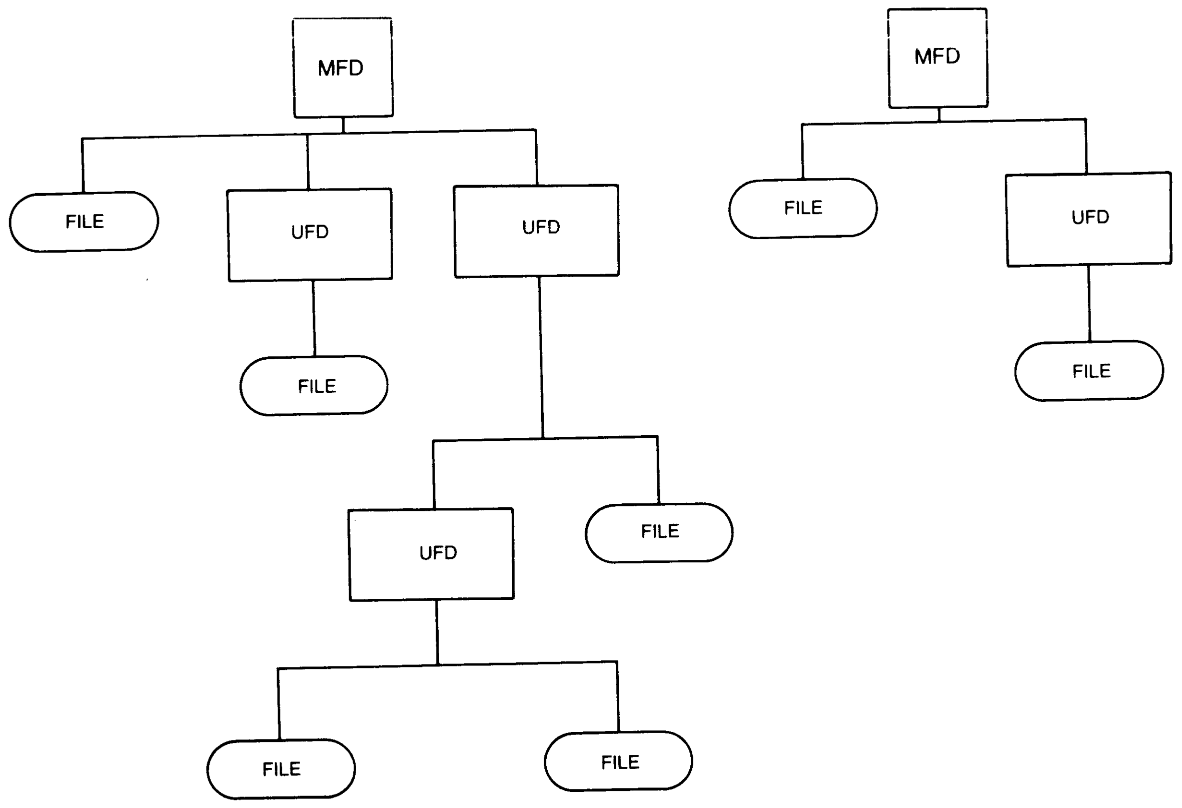
6 File Management

Prime's file management system allows easy access to information contained on disk. Three elements -- the file, the directory, and the partition -- form the basis of the system's hierarchical structure and allow any piece of information to be accessed. In addition, the file system has several security features built in to ensure the integrity of system information.

This chapter defines the basic elements of the file system and describes how they organize information. It also describes some of the security features.

FILES AND DIRECTORIES

The two simplest elements of Prime's hierarchical file system are the file and the directory.



The 50 Series File System
Figure 6-1

The File

A file is the smallest and most basic organizational unit recognized by the file system. It is a collection of related items and can take several forms, depending on what the user wants to represent. To distinguish between the various types of collection, the file system groups files into seven basic types:

- Sequential access method, or SAM, files
- Direct access method, or DAM, files
- Keyed-index direct access, or MIDASPLUS, files
- Directories
- Segment directories
- Access categories

The type identifies how the file system should handle the file, and indicates what type of information the file contains. SAM files contain a number of records, each of which contains a forward pointer to the next record in the file and a backward pointer to the previous record in the file. DAM files contain one multilevel index of pointers that reference all records in the file. MIDASPLUS files, directories, and segment directories are explained below. Access categories contain security information and are explained later in this chapter.

The Directory

A directory is also a file, but a special type to the system. The items contained in a directory are pointers to files. When the file system wants to locate a file, it references the appropriate directory for the file pointer, much as one looks up a library book in the card catalog. The section, Accessing Files and Directories, in this chapter explains more about how this reference process works.

Directories can contain pointers to other directories. Since directories are just special files, there is nothing to prevent any of a directory's pointers from pointing to a secondary directory rather than a simple file. These secondary directories are called subordinate directories, or simply subdirectories.

There are several types of directories in the Prime file system. The most common is identical to what has already been described; a file that contains pointers to other files. The user file directory, or UFD, holds all the files and subdirectories (subUFDs) of a particular user. The MIDASPLUS file is really a directory that contains an index and a series of files containing data. The segment directory is similar to to the UFD, but is generally referenced by programs rather than by the user.

Prime's file system imposes no limit on the number of levels of nested subdirectories. Each level of subdirectory represents information that is further down the file hierarchy. This hierarchical arrangement is called a tree structure, and it allows the user to layer information to the depth necessary for a particular application.

Disk Quotas

A disk quota specifies the maximum number of records a directory and all its subentries may contain. Quotas can be optionally applied to any directory on the system. A directory under quota control is called a quota directory. Each time a file is added to a quota directory, the system checks to make sure the quota will not be exceeded before allowing the addition.

Naming Files and Directories

Prime's file system encourages a set of standard naming conventions. In general, filenames are made up of one or more components; multiple components are separated by periods (.). Each component is made up of alphanumeric characters and selected punctuation symbols.

The last component of a filename is often called a suffix. For example, if a file has the name CHARGE.TEST, TEST is the suffix (and CHARGE is called the root of the filename). The file system recognizes some standard suffixes, such as F77, PL1G, BIN, and LIST. These suffixes readily identify how a file was generated. BIN, for example, shows that a Prime translator generated a binary file. The standard suffixes also save the user typing, since in many commands only the filename's root needs to be specified.

Partitions and MFDs

The hierarchical level above the UFDs is called a partition. Physically, a partition is a set of one or more disk heads, but the file system treats the whole partition as one logical entity. A partition contains one master file directory, or MFD, that lists all the UFDs that are part of the partition. The MFD can also contain files, including special ones for record allocation or badspot handling on the partition.

Partitions are divided into two groups relative to the system. Those physically connected to a system are called the local partitions of that system. The partitions visible to a system but physically connected to another are called the remote partitions of that system.

A system can activate many partitions at once, including some that may be physically part of other systems. The partitions list, the highest level in the file system hierarchy, makes this possible. This list is a set of active disk partitions. It contains the names and other information about all the partitions currently running on the system, regardless of where the partitions are. This means that a reference to a remote partition is just as easy as to a local partition; the user need never know the exact location of the information being accessed.

ACCESSING FILES AND DIRECTORIES

When logged onto the system, the user is always attached to an MFD, UFD, or subUFD. The directory to which the user is currently attached is called the current attach point. Accessing a single file in the current attach point is very simple; the user merely specifies the file's name as a parameter of the desired command.

To access a file in another directory, or to move from the current directory to another, is just as simple. The user specifies a pathname.

A pathname consists of a filename and the directory structure that contains the named file. Its purpose is to provide the file system with a path from a known starting point to the desired file.

Pathnames allow the user to access information with great flexibility. They allow access to objects throughout the file system or movement from one place to another. They can also be used in abbreviation files (see Chapter 9).

FILE SECURITY

There are times when access to a file or directory needs to be restricted. Perhaps the user wants to restrict access to particular information, such as programs essential to the system or sensitive data. Whatever the reason, Prime's file system provides four means of protection for critical data: access control lists, user profiles, passwords, and file access rights.

Access Control Lists

Access control lists provide passive protection for files and directories. They allow the user to specify access rights for:

- A single file
- A group of files
- A single directory
- A group of directories

Once the access control lists are in place, PRIMOS automatically checks every reference made to the protected information to make sure the reference is allowed. In fact, the user may be quite unaware of the protection because no user action is necessary to invoke it.

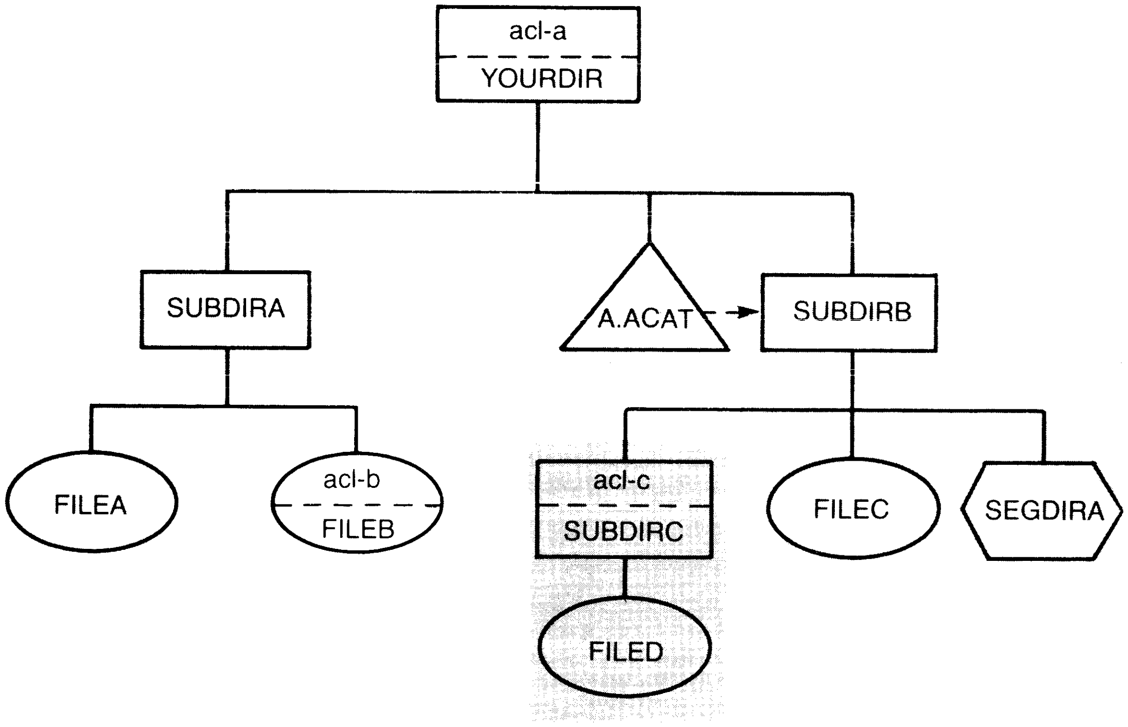
Each access control list is a series of ordered pairs that specify the access rights for a file or directory. The first element of each pair identifies a user or group of users; the second specifies the combination of access rights granted to that user or group of users. The access rights are listed in Table 6-1.

Table 6-1
Access Rights for Access Control Lists

Access Right	What It Protects	Rights Granted
Read	file	Open a file for reading.
Write	file	Open a file for writing. Includes truncation rights.
Add	directory	Create files and subUFDs.
Use	directory	Attach to a directory and use it in a pathname.
Delete	directory	Delete files and directories.
List	directory	List the directory.
Protect	directory	Create and edit access control lists, and protect files with ACLs.
ALL	file or directory	All access rights are granted.
NONE	file or directory	All access rights are denied.

Four kinds of protection can be specified using access control lists. They are:

- Default
- Specific
- Access category
- Priority

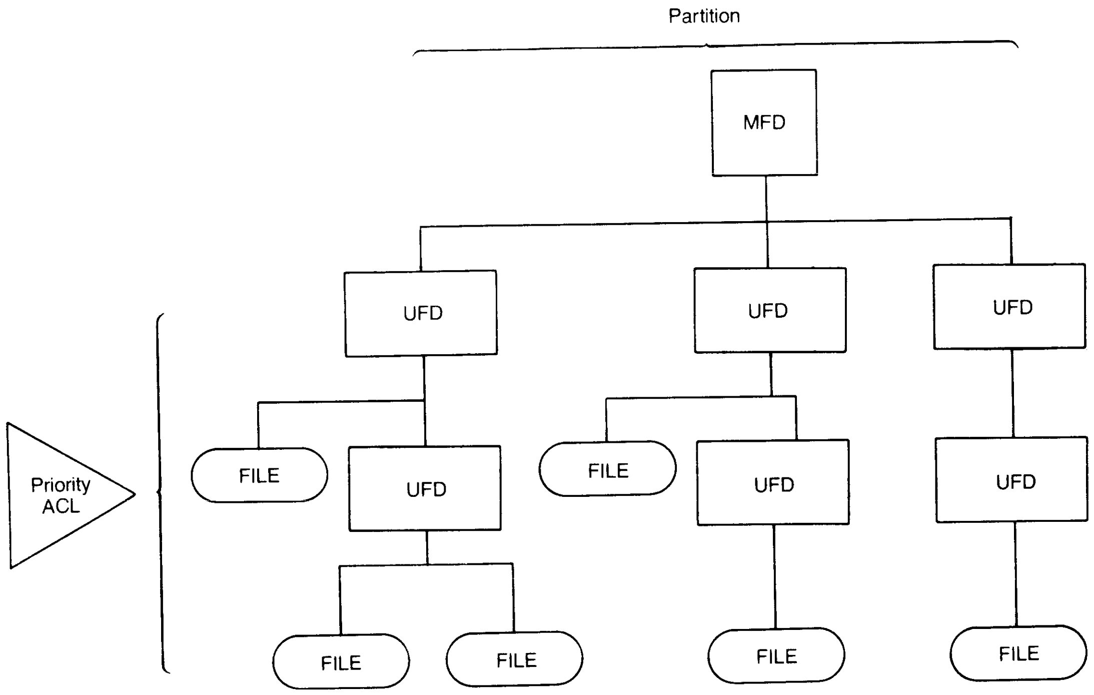


Types of ACL Protection
Figure 6-2

Default protection provides security for nested directories and entire tree structures. This means that the user can specify an access control list for a directory, and all subdirectories and files will be subject to the same protection unless the user specifies otherwise.

A file or directory with specific protection has an access control list directly associated with it. This control list can only be referenced through the name of the file or directory it protects.

An access category is a named file system object containing an access control list. The category provides collective protection for a group of files and/or directories. Changing the category's access control list changes the access rights to every file and directory in the group. This is particularly useful for protecting information scattered throughout the file system. See Figure 6-2 for illustration of the use of default, specific, and access category protection.



Priority Protection
Figure 6-3

Priority protection allows special case accesses to occur. For example, the system administrator can specify a special priority access when backing up the system. This type of access temporarily supercedes all user-specified protection, thus allowing backups to be made, but it does not alter or destroy any existing user accesses. It does not allow other users to make invalid accesses at any time, because it is available only to the system administrator or operator. Figure 6-3 illustrates the use of a priority ACL.

User Profiles

Each user process has a user profile associated with it. The profile identifies the access rights and privileges that are to govern execution of the user process. Among other things, the profile specifies where in the file system PRIMOS should initially attach the user upon login, and the type of access control that should be in effect.

Profiles allow the system administrator to tailor access protection to the system. For the system with many individual users and individual projects, profiles can be set up on a per user basis to associate each user's identity with a particular data base. If many users are working on a single project and require access to a single data base, a project profile can be set up to associate all members of the project with that data base. Combinations of individual and project profiles can be arranged to suit the application; there is no restriction on the number of ways a user profile identifies the user.

Passwords and File Access Rights

Passwords and file access rights are features of earlier Prime systems. They are supported on the 50 Series because of Prime's commitment to compatibility across its product line. This support allows applications designed on older Prime systems to run without reloading or recompiling. Although these types of security are supported, they are not described here. Refer to the Prime User's Guide, DOC4130-190, for information about this topic.

SUMMARY

Prime's file system is based on a hierarchical arrangement of files, directories, and partitions. This arrangement and the associated forms of protection ensure the integrity and security of all information on the system, and at the same time provide easy accessiblity for all proper accesses. The next chapter, PRIMENET, shows how Prime's networking facilities work with the file system to give the user easy access to files on other systems.

7 PRIMENET

The previous chapter discussed the file system and how its hierarchical structure grants the user easy access to information throughout the system. The structure of the file system makes references to information on other systems just as easy as those to information contained locally. This chapter describes how PRIMENET, Prime's networking software supported on all 50 Series systems, provides a reliable, standardized medium through which remote accesses can be made.

INTRODUCTION

With a variety of services and transmission methods at its disposal, PRIMENET supports communications between linked systems. This software operates in a fashion that is completely transparent to the user. This transparency eliminates the need to learn new commands, details about the link between systems, or details about the physical location of information. Instead, PRIMENET makes referencing remote information identical to referencing local information.

In addition to its transparency and ease of use, PRIMENET software meets the International Telegraph and Telephone Consultative Committee's (CCITT) X.25 standard for packet switching networks. This is advantageous in situations requiring domestic and international public data networks, since PRIMENET's X.25 support allows it to communicate with any other system that also supports X.25. Two other types of networks, ring and point-to-point synchronous, are also supported; the combination of the three types gives the user several options that can be exercized based on need and application.

BASIC ARCHITECTURE

PRIMENET is made up of several layers, as shown in Figure 7-1. The levels are based on the International Standards Organization's open systems interconnection (ISO OSI) model to make the 50 Series systems able to communicate with all other systems that support X.25 protocols.

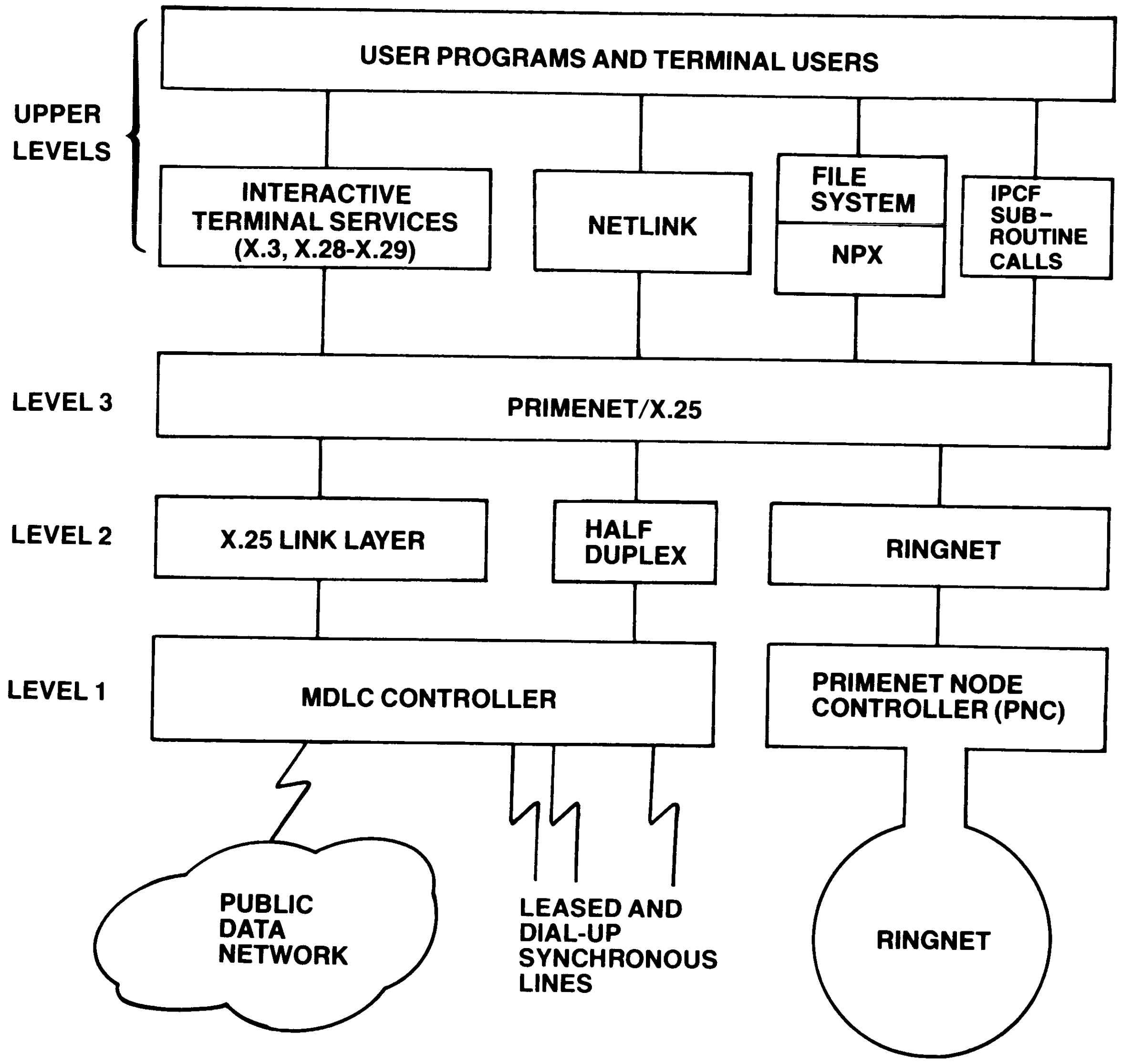
Levels of Protocol

PRIMENET consists of several functional layers. Each one embodies a standard interface to the adjacent layers, and each implements a different level of protocol.

Level 3, the packet interface, creates and controls virtual circuits across the network, handles error recovery, and controls the flow of information. It also keeps track of the process to which each packet is being transferred. It is this layer with which the user interfaces (see PRIMENET Subroutines, below); its X.25 support gives the user a standard interface to the upper levels no matter what kind of link makes up Levels 1 and 2.

Level 2, the link protocol level, corresponds to the International Standards Organization's open systems interconnection (ISO OSI) data link layer. It describes a protocol that two linked nodes must adhere to when transferring information between them. This protocol dictates the format of the data, how the nodes should request, transfer, receive, and acknowledge the data, and how to signal faulty transmissions should any occur.

Level 1 is the hardware interface. It is the equivalent of the ISO OSI's physical layer. This layer acts as an intermediary between the physical transmission medium (twin-axial cable or transmission line) and the rest of PRIMENET and the system. Depending on the type of network, one of two controllers govern action at this level. These controllers, the PRIMENET node controller (PNC), and the multi-line data link controller (MDLC), are described in the section, Network Types, below.



Levels of PRIMENET Architecture
Figure 7-1

Other layers of PRIMENET exist in addition to the three described above. The PRIMENET internals operate on these layers to perform an action the user directly specifies, or one that facilitates completion of a user task. The internals also use PRIMENET subroutines.

Advantages of a Layered Architecture

This layered approach has several benefits for the user. The simplicity and structured design is the same no matter how many systems are linked in a network. In addition, since PRIMENET supports internationally recognized standards such as X.25, X.3, X.28, and X.29, the user can easily link to any other network, both Prime and non-Prime, that supports the same standards. This allows the 50 Series systems to be easily integrated with already existing equipment.

Users interface only with the top layers of PRIMENET. Since these top layers perform all interaction with lower levels themselves, the user does not have to know anything about the physical connection between two linked systems, how the data is formatted and checked as it is transferred, or even if a network connection exists. This makes PRIMENET transparent to the user, so accesses to information on remote and local systems are identical.

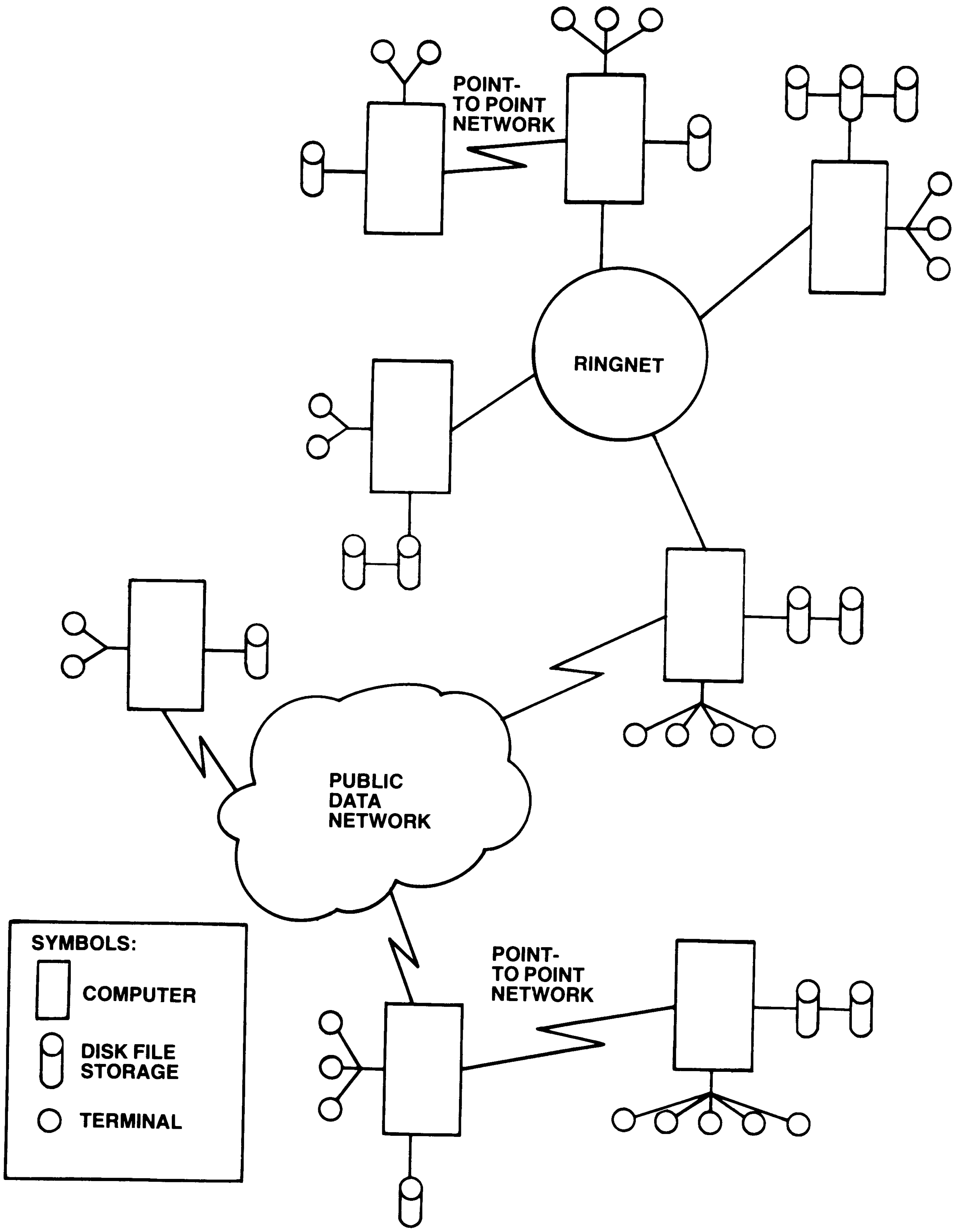
Another advantage to the layered architecture of PRIMENET is that any changes made to the lower levels are transparent to the user. Because the user sees the top levels of PRIMENET, changes made to lower levels do not change the way the user invokes or uses PRIMENET.

NETWORK TYPES

PRIMENET's levels of protocol and support of international standards allow it to support three types of network links: RINGNET, point-to-point, and Public Data Network.

These types differ in their ranges of effectiveness and how they link systems, but they all provide the user with the means to access information held on other Prime and non-Prime systems. This means that a user can log remotely into any system in the network from any terminal. It is also possible to log onto the usual local system, and then remotely access information contained on other systems. Figure 7-2 shows typical examples of the three types of network links.

For the rest of this discussion, the word node represents any system that is linked to others in a network.



Examples of Networks
Figure 7-2

RINGNET

Prime's ring network is called RINGNET. Its protocol supports many nodes connected through a high speed, one way, serial synchronous coaxial cable. (See Figure 1-3.) A junction box and a PRIMENET node controller (PNC) within each node allow information to pass between the node and the rest of the network.

RINGNET is a token-based network. A special bit pattern circulates around the ring, and a node cannot transmit information until it detects the token. When it can transmit, the node issues a packet containing a 4-byte header and from 4 to 508 bytes of information through its PNC. The packet circulates around the ring at a speed of 1 Mbyte/second, traveling through all intervening PNCs until it reaches the destination node. There the destination PNC receives the packet and sets a flag in the packet to acknowledge the receipt. The packet travels around the rest of the ring to the source, which:

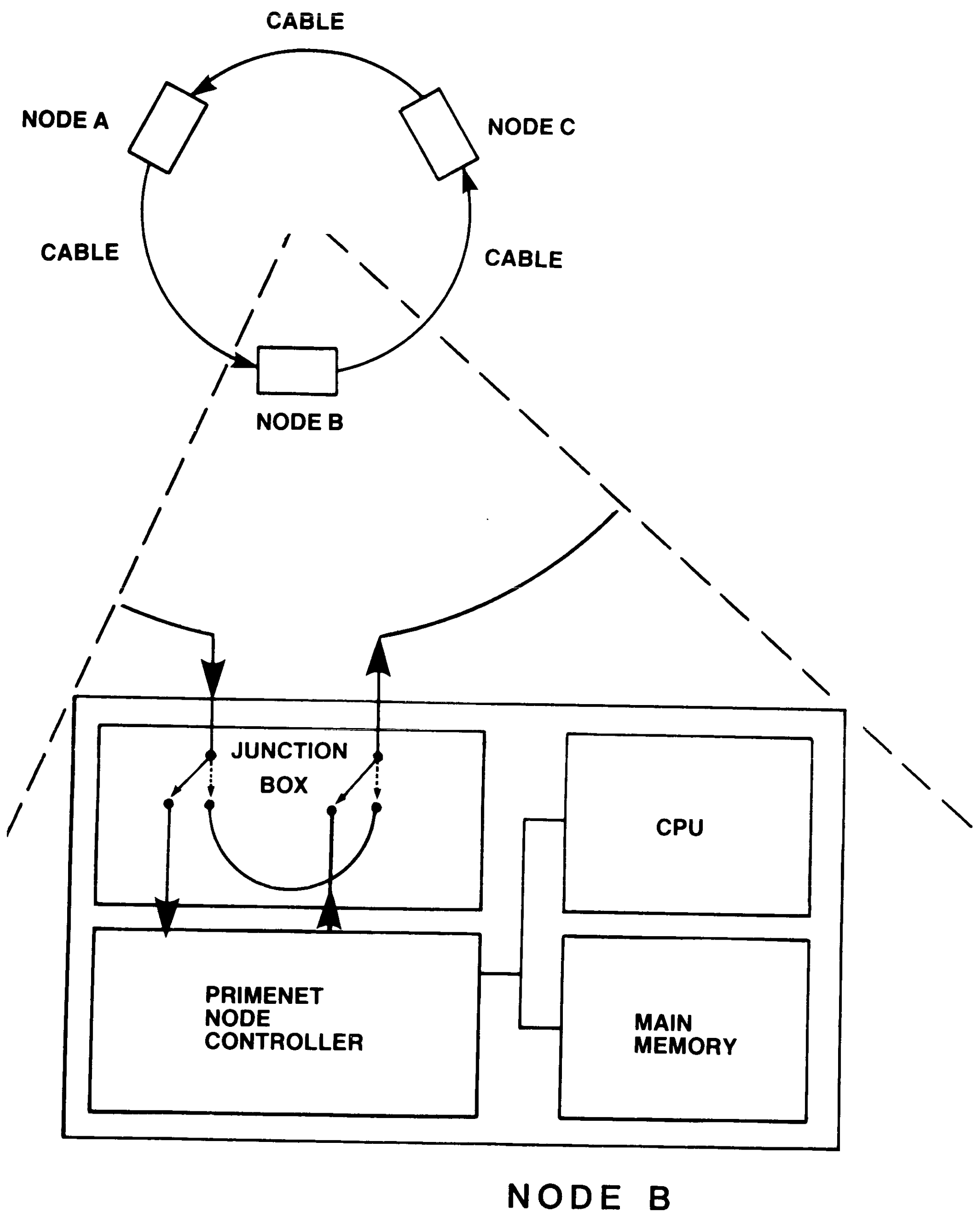
- Removes the packet from the ring
- Passes on the token
- Checks the acknowledgement flag
- Interrupts the host node and returns the transmit status

RINGNET offers several advantages to the user. Each PNC acts as a data repeater for packets between other ring nodes. When the PNC in Node B (see Figure 7-3) repeats data from Node A to Node C (i.e., Node B is neither the transmitter nor the receiver) no software intervention is necessary; the operation is handled completely in hardware, transparent to Node B and to Node B's users. When Node B is transmitting or receiving data, the PNC handles the ring protocols in firmware (except for error-caused retransmissions).

Another advantage is the PNC's use of DMA I/O to transfer up to 2 Kbytes of data per block. Only one interrupt per data block is needed to indicate the node's success or failure to receive or transmit a packet.

RINGNET serves all nodes in the ring equally, so that one system cannot monopolize the network. It automatically checks all packets for integrity, requiring no user intervention or separate acknowledgement messages. In addition, RINGNET offers the user the ability to expand into larger networks without suffering any degradation of performance.

If any number of the nodes in a ring network are powered down or broken, the rest of the nodes are still able to send data around the ring. The junction box in a disconnected node allows messages to pass through without interruption to the next node in the ring, as long as the distance between adjacent connected nodes does not exceed 750 feet.



RINGNET Configuration
Figure 7-3

Point-to-Point Networks

Nodes in point-to-point networks communicate via dial-up or leased telephone transmission lines. Each node can contain up to two multi-line data link controllers (MDLCs); each MDLC can support two or four lines at standard modem speeds. Information travels from the processor of one node, through the MDLC and a modem, across the transmission lines to the modem and MDLC of another node.

For leased-line communications, the MDLC supports bisynchronous and X.25 high-level data link control (HDLC) protocols. It also supports HDX, a modified HDLC protocol, to allow half duplex communications across dial-up lines. These protocols, and multiline support, allow the MDLC to be used simultaneously by several processes, both PRIMENET- and non-PRIMENET-related. Like the PNC, the MDLC uses DMA I/O to transfer data to and from main memory, and issues an interrupt to the host node only after an entire packet of data has been received or transmitted. It handles all error checking and frame formation in firmware.

Public Data Networks

The 50 Series systems can subscribe to all public data networks (PDNs) that support the CCITT X.25 protocol standard. Supported PDNs include TELENET and TYMNET in the United States, DATAPAC in Canada, IPSS in Great Britain, TRANSPAC in France, and EURONET in Europe. All of these networks transfer and process information in packets, charging the user according to the amount of information sent rather than to the time of connection. This can provide service at substantial savings over networks requiring dedicated transmission lines or dial up circuits.

The user with a Prime 50 Series system linked in a PDN has access to all other members of the PDN. This means that any Prime terminal user can access all other member systems, both Prime and non-Prime, and that all PDN terminal users can access the 50 Series system.

PRIMENET INTERNALS

PRIMENET's internal elements of interest to the user include:

- Network process extension (NPX)
- PRIMENET subroutines (IPCF)
- Loopback facility
- Port mechanism
- Virtual circuits
- Network configuration facility (NETCFG)

Network Process Extension

All file accesses to remote PRIMENET-linked systems go through the network process extension (NPX) mechanism. MIDASPLUS and the electronic mail feature of Prime's Office Automation also use NPX to perform remote functions. This mechanism allows local processes to make procedure calls (see Chapter 8) to any remote system. The calls are made transparent to the user.

To make the calls, NPX has an associated set of server, or slave, processes on each node. These processes lie dormant, using no system resources, until a remote call is received. When a user or system process (the master) makes a remote call, one of the slaves on the remote system is activated. The slave receives the procedure name and any arguments, builds an entry name and argument list referencing the called procedure, and makes the call. When the remote procedure completes, it transfers any return arguments to the slave, which in turn transfers them to the master process.

A slave process remains dedicated to the master that activated it until it is released. A master process, however, can have many slave processes serving it on many remote systems.

What determines the access rights of a slave on a remote system? Depending on how stringent the security requirements are, the slave is given access rights in one of two ways. It can take on the master's login identity; this means that the slave is given whatever access rights the master has without any further validation. For the second method, the master sets up an appropriate slave identity before making the remote call. When a call is made in this case, the slave identity is subject to full validation.

PRIMENET Subroutines

The set of PRIMENET subroutines are called the interprocess communication facility (IPCF). These subroutines allow a user process to set up communication links with other processes (target processes) within the network. Through these links the user process can exchange data with any of its target processes. The user process executes the appropriate subroutines to initiate communications with a target process, to transmit or receive data, to wait for a reply, and so on. In addition, NPX uses these subroutines to make its remote calls.

Procedures written in any high-level language can call any of the PRIMENET subroutines. This is especially useful when the user is developing distributed applications programs.

Loopback Facility

To help develop, test, and debug an application that uses the IPCF subroutines, PRIMENET supports a loopback facility. It allows the user to run and check the application's calls to remote processes on the local system alone. The user sets up both source and destination processes on the local system; when the source makes the call, the facility loops it back to the local system rather than allowing it to go over the network. This allows the application's use of the network to be fully tested and any errors corrected before installation and preserves the integrity of network resources.

Ports

When one process communicates with a remote system, the process must have some way of identifying the node and the destination process. A standard X.25 node address can identify the correct node, but not which of 128 possible processes is the destination. To accomplish process identification, each node has a list of ports that act as subaddresses within the node.

Each 50 Series system supports 256 ports. Ports 1-99 are reserved for the user making calls through the IPCF subroutines; ports 100-255, for system use. (See Figure 7-4.) A process that expects a call assigns itself an appropriate port number and waits for the call. The process making the call must specify this port number for the call to be made. Processes that do not specify a port number make their calls to the default port, the remote login port (Port 0).

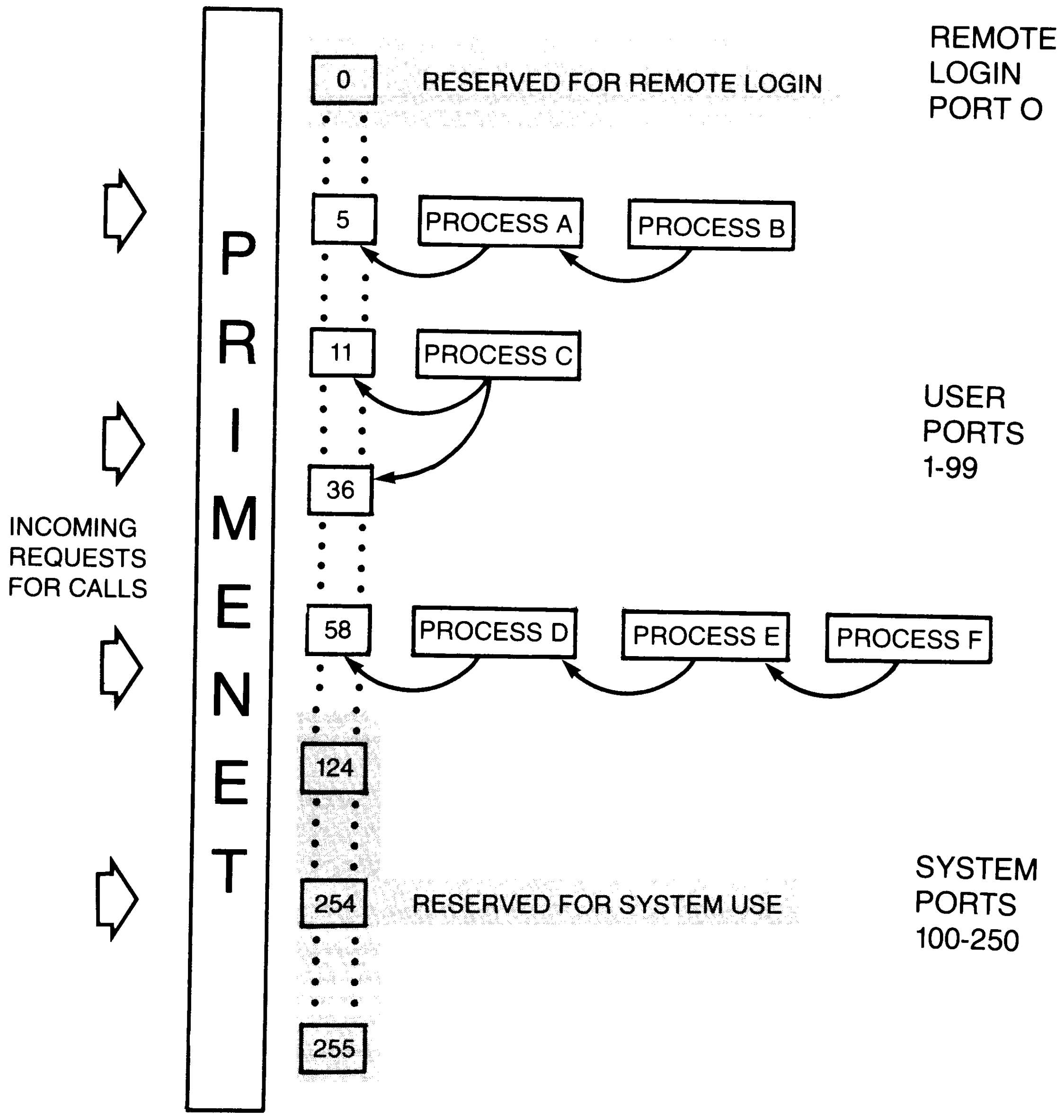
Virtual Circuits

When one process specifies the node and port of another process, PRIMENET establishes a bidirectional link between the two through a virtual circuit. This is a logical path or channel that traverses the network from one process to another via several physical, point-to-point links.

Each virtual circuit has an identifying number to distinguish it from all others. Up to 63 virtual circuits can be supported per system.

Network Configuration Facility

The 50 Series provides an interactive user facility that guides the user through the process of network configuration. It is invoked by specifying the PRIMOS NETCFG command. This command asks the user a series of simple questions to determine what the network configuration should be. No complex network generation is needed, since NETCFG handles all aspects based on the user's responses to its questions.



50 Series Port Mechanism
Figure 7-4

USER FACILITIES

PRIMENET provides the user with many facilities for referencing remote information. The IPCF subroutines were described in the previous section. In addition to these, PRIMENET offers:

- Remote login
- Remote file access
- NETLINK
- File transfer service (FTS)

Remote Login

The standard command that enables the user to log onto the local system is:

```
    LOGIN user-id
```

By adding the option -ON <remote_system_name> to the LOGIN command, the user can log onto any remote system that is linked via PRIMENET to the local system. The remote system may be connected to the local one via any of the three types of links supported by PRIMENET. Once remote login is established, tasks are specified as if the user were logged onto the system locally.

Remote File Access

PRIMENET also provides immediate access to any remote file within the network, even if the user does not know the file's location. This means that the user does not have to learn any new commands to specify a remote file, since PRIMENET works with the file system to access the file in a transparent fashion. In fact, the user may not even know that the file is not contained within the local system. In addition, programs accessing remote information do not have to be changed or recompiled if the remote information is moved.

NETLINK

The NETLINK software allows a terminal user to communicate over any X.25 network to which the local system is linked. This software does this by emulating a PDN packet assembler/disassembler (PAD). It converts the asynchronous terminal output into X.25-formatted packets of information that can be transmitted over the network.

For the user with a PDN link, NETLINK allows access to any system in the network, both Prime and non-Prime. The user does not have to log out of the local system to invoke NETLINK; in fact, NETLINK supports simultaneous links with up to six remote systems and allows the user to move between them and the local system at will. This capability puts the wide variety of PDN facilities within quick reach of any Prime user in the network.

File Transfer Service

To complement the capabilities of PRIMENET, the 50 Series systems support the file transfer service (FTS) subsystem. With FTS, the user can transfer files between the local system and any PRIMENET-linked remote system. The two systems involved in the transfer can be either Prime or non-Prime systems, but both must support the X.25 protocol and the same file transfer protocols that FTS uses.

FTS provides facilities for users, operators, and administrators. The user can set up, monitor, and control file transfers between Prime systems. FTS operators take care of day-to-day control and monitoring of FTS internals, such as the file transfer manager and queues of user file transfer requests. A system administrator can tailor FTS to the particular local system. Users can send or fetch any file (given the proper access rights) in the system; if a node is currently disconnected, FTS will automatically retry a transfer later. In addition, FTS can notify both the source and the destination users via the PRIMOS MESSAGE command whenever a transfer takes place.

SUMMARY

Prime's network and communication capabilities make it easy for the user to access information on several systems, both those made by Prime and those made by many other manufacturers. PRIMENET operates transparently to the user, which means that accesses from one system to remote resources are identical to those made to local resources. Since local and remote accesses are handled identically, the networking software is more efficient, which in turn increases performance.

In addition, Prime's network facilities handle many of the transmission tasks internally. These features mean that the communication portions of many application programs are easy to write, since they need not worry about the exact format of the links between systems. The next chapter, Procedure Management, discusses more of the ways the 50 Series enhances the program environment.

8 Procedure Management

The two previous chapters, The File System and PRIMENET, showed how the 50 Series systems could provide the user with a rich variety of services on both local and remote systems. To access these resources, the user invokes the procedure call mechanism. This mechanism acts as a universal method of control transfer for all parts of a 50 Series system.

A procedure call is an orderly transfer of control. In its simplest form one procedure invokes, or calls, a second; the second executes, then control is transferred back to the first. In more complex situations, the procedure call allows the user to invoke part of PRIMOS, reentrant or recursive procedures, or library subroutines.

This chapter describes many aspects of the procedure call mechanism:

- How standard procedure calls occur
- How procedure calls to PRIMOS services in Ring 0 (direct entrance calls) occur
- How the condition mechanism handles errors

PROCEDURE, LINKAGE, AND STACK AREAS

To run a program, the system requires:

- The main procedure and any subprocedures (pure procedure code)
- Static information, such as linkage information and common areas
- Dynamic information needed upon execution, such as arguments from one procedure to another or dynamic data.

To simplify and control access to this information, separate areas in virtual memory are designated for each type of information. A dedicated, user-accessible base register is assigned to each area so that references can be made relative to the area itself. The procedure area contains the main procedure and any subprocedures that are to be executed; PB is the dedicated procedure base register. Static linkage data, such as local variables, is contained in the linkage area and is accessible via LB, the linkage base register. The stack area provides dynamic storage and uses SB, the stack base register. Each area may be in a separate segment with its own access protection.

The procedure call mechanism must coordinate the transfer of control from one procedure to another. This means that it must control and monitor the changes that occur in the procedure, linkage, and stack areas and registers for both procedures. The next section describes the three elements of the procedure call mechanism that make this possible.

ELEMENTS OF THE PROCEDURE CALL

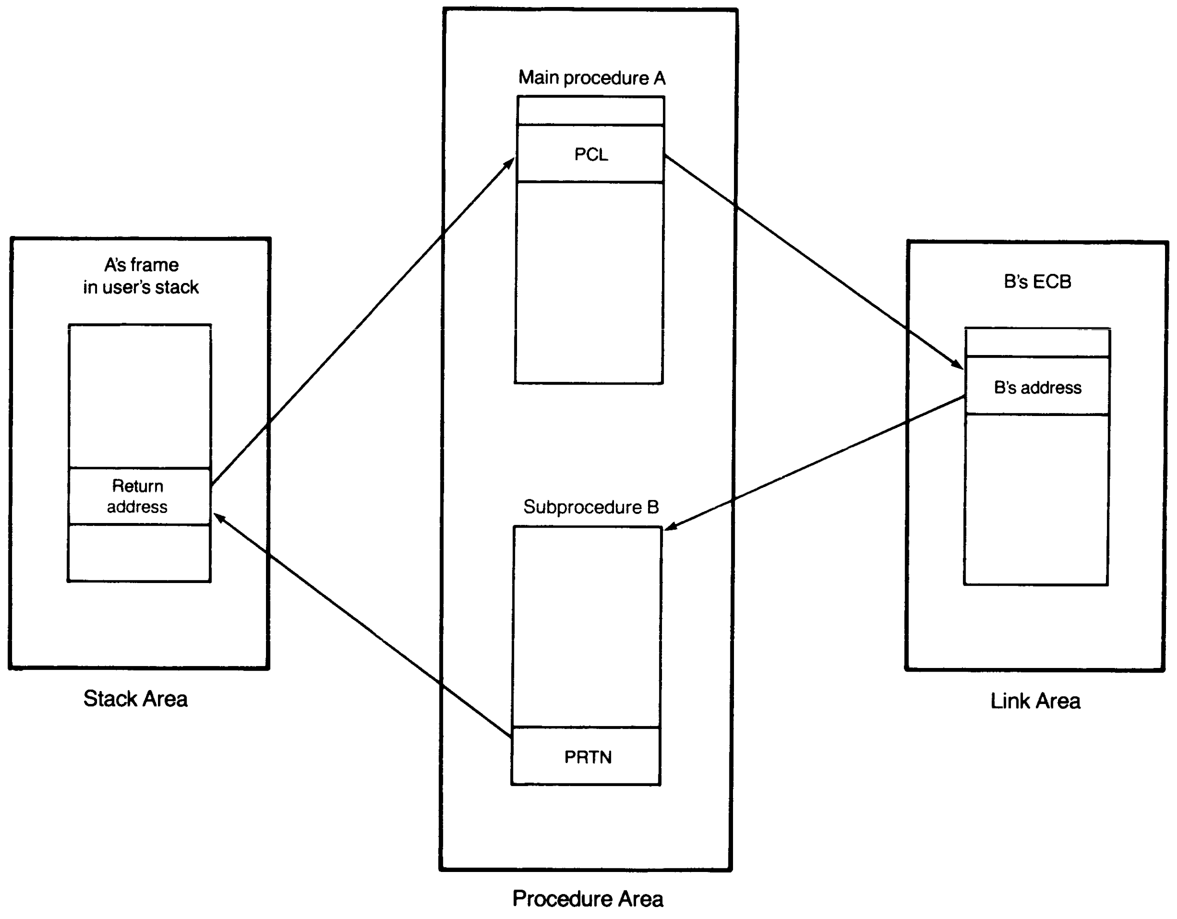
To perform the control transfer from one procedure to another requires:

- A stack
- An entry control block
- The procedure call instructions: PCL and PRTN

The stack area described above contains the stack. The entry control block resides in the linkage area. The PCL instruction is contained in the main procedure in the procedure area; the PRTN instruction, in the subprocedure(s) in the procedure area.

The Stack

When one procedure calls another, the procedure call must save the current state of the caller before transferring control to the callee. This is because the callee's execution may change the contents of some of the system registers; when control transfers back to the caller,



Procedure Environment
Figure 8-1

the registers may not contain what the caller expects. The procedure uses a stack to save the state for each calling procedure, as well as any dynamic variables each may use.

A stack is one or more segments in the user's address space. The first segment is called the stack root and uniquely identifies the stack. Any other segments in the stack are called stack extension segments. The number of extension segments is limited only by the space available in the user's address space.

The system supports several stacks at once. It supports stacks for its own use as well as those for each user logged onto the system. The user, however, can access only one stack at a time via SB (usually only the current user stack).

Each time one procedure calls another, a block of information about the caller is stored onto the stack. The block of information, called a stack frame, describes the state of the caller in effect when it made the call. It also contains indirect pointers to whatever arguments the callee expects, and a return address. The return address allows control to return directly to the caller, rather than requiring a reference to the caller's stack frame first.

Entry Control Block

The entry control block in the linkage area defines a called procedure. It contains information such as how many arguments the called procedure expects, the starting address of the called procedure, the number of the stack root segment, the expected size of the stack frame, and so on. When one procedure calls another, it references the called procedure's entry control block, rather than the procedure itself, so that the procedure call mechanism has access to the called procedure's identifying linkage information.

The Procedure Call Instructions

To make a procedure call, a calling procedure executes a PCL. This instruction handles all tasks that must be done before control can transfer to the callee. These tasks include using SB to set up a stack frame, and keeping track of the stack information needed by both procedures. PCL also ensures that the called procedure can reference any parameters from the caller that it might need. In the procedure area, PCL is followed by a series of argument templates if the callee expects arguments. These templates are used to form indirect pointers that are passed to the callee so that it can reference the parameters. During its execution PCL calculates the indirect pointers and passes them to the callee.

The templates allow the user to specify different indirect pointers within the same PCL instruction. For example, suppose a PCL instruction is followed by an argument template that specifies a particular base register. Also suppose that the contents of that base register change prior to each execution of the PCL. This means that PCL calculates a different indirect pointer each time it executes. With this technique, the user can call procedures recursively without any extra programming or special procedures.

Once PCL completes execution, control transfers to the callee and it executes. When it is time to transfer control back to the caller, the procedure return instruction, PRTN, executes. This instruction removes the stack frame PCL created in the stack area, restores the caller's system state via the linkage information contained in the entry control block, and transfers control in a smooth and consistent manner back to the caller in the procedure area.

PCL and PRTN are implemented in firmware to speed up argument transfers, stack operations, and the control transfers. The 750 and 850 also incorporate hardware support.

MAKING A STANDARD PROCEDURE CALL

When PCL executes, it:

1. Verifies the caller's right to access the callee's entry control block

2. Creates a new stack frame for the callee

3. Saves the caller's state, and then loads the callee's state

4. Calculates and stores indirect pointers for the callee's use

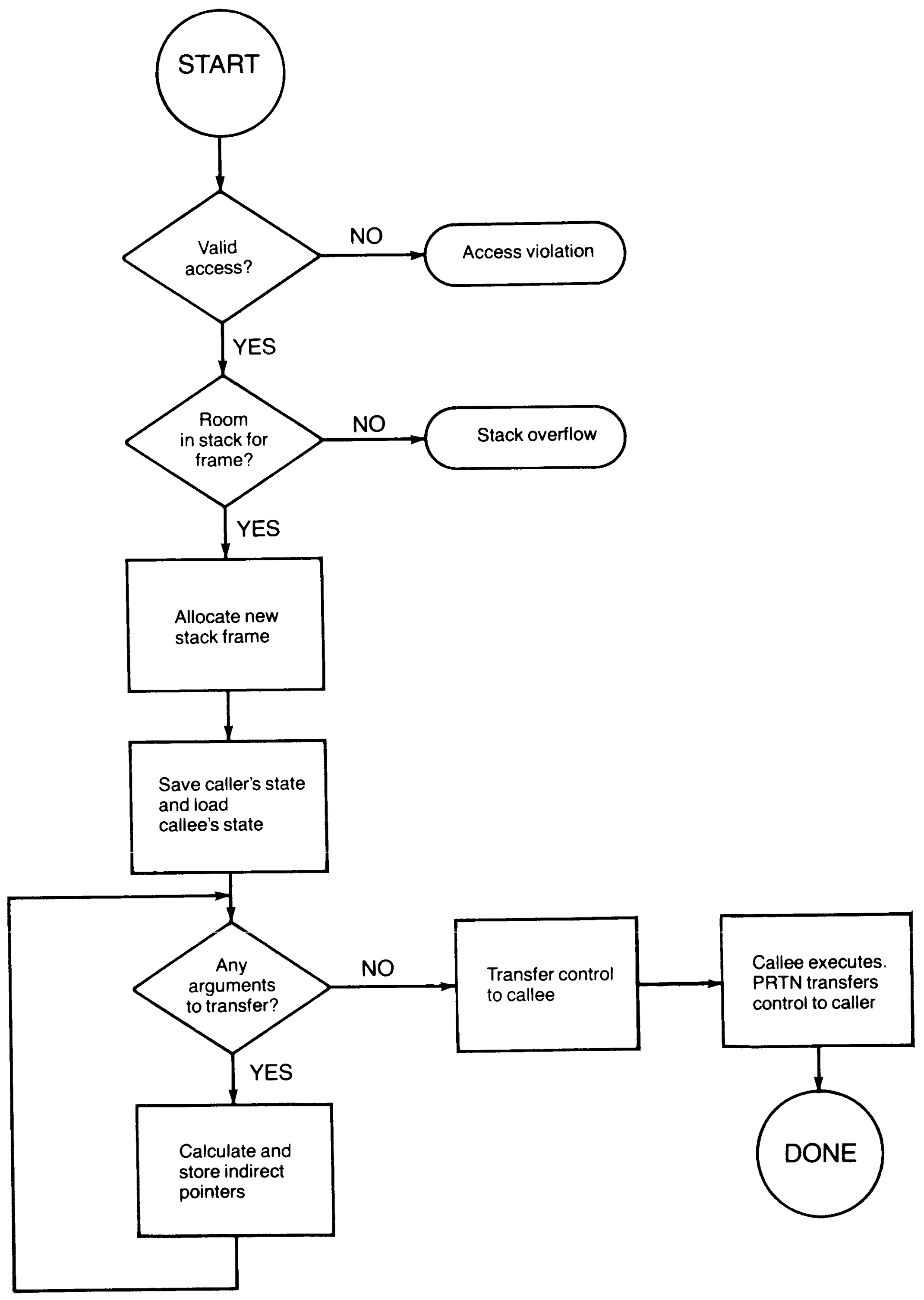
Figure 8-2 summarizes this sequence of events.

1. Verifying Access Rights

When PCL begins execution, it must make sure the caller has the proper access rights to reference the called procedure's entry control block. It does this in the same way any memory reference is checked. (See the section, "Protection Rings", in Chapter 4, Memory Management, and "Making a Direct Entrance Call", below.) If the access is not allowed, PCL causes an access fault and does not continue. If the access is allowed, PCL continues.

2. Allocating a Stack Frame

PCL references the callee's entry control block to determine which segment contains the stack root. After identifying the stack root, PCL checks it to see if there are enough free locations to contain the new frame. If there are, the new frame begins at the first free location. If there are not, PCL extends the stack, if possible, and begins the new frame in the stack extension segment. If there is no room in the extension segment for the frame, a stack overflow occurs.


START
Valid access?
NO
Access violation
YES
Room in stack for frame?
NO
Stack overflow
YES
Allocate new stack frame
Save caller's state and load callee's state
Any arguments to transfer?
NO
Transfer control to callee
Callee executes. PRTN transfers control to caller
DONE
YES
Calculate and store indirect pointers

Actions of PCL and PRTN
Figure 8-2

3. Saving the Caller's State and Loading the Callee's State

   PCL stores the caller's state into the new stack frame. Once this is done, it loads the callee's state (contained in the callee's entry control block) into the system registers.

4. Calculating Indirect Pointers

   If the called procedure expects parameters, the indirect pointers to the parameters are calculated now. PCL uses the argument templates that follow it in the calling procedure to create the pointers. After they are calculated, these pointers are stored in the caller's stack frame.

5. The PRTN Instruction

   Once all indirect pointers are calculated and stored, control transfers to the called procedure. When it completes, PRTN ensures that the control transfer back to the calling procedure is smooth. This instruction deallocates the caller's stack frame, then restores the state of the caller. Once the state is restored, the caller resumes execution at the instruction that follows the calling PCL and its argument templates, if any.

DIRECT ENTRANCE CALLS

The procedure call actions described above take place whenever both caller and callee are within the same ring of protection. When Ring 3 procedures want to call Ring 0 procedures, however, the actions taken change. (Ring 0 calling Ring 3 should never occur, so this case is not considered here.) Granting user procedures access to Ring 0 procedures must be closely monitored to guard against any abuse of privileges. Accesses of this type are called direct entrance calls.

The Gate Access Segment

All user accessible Ring 0 procedures must be set apart from those that are not user accessible. To accomplish this, the entry control blocks of all Ring 0 user accessible procedures are contained in a gate access segment. Users can make direct entrance calls only to those Ring 0 procedures whose entry control blocks are present in the gate access segment and to no others.

When a user procedure specifies a procedure call to a Ring 0 procedure, the procedure call must specify an entry control block in the gate access segment. If the Ring 0 procedure's block is not in the gate access segment, the call cannot be made. If the Ring 0 procedure's block is contained in the gate access segment, the call proceeds as described in the previous section.

Making a Direct Entrance Call

When one procedure calls another whose entry control block is located in the gate access segment, how does the system link the two? The 50 Series systems use dynamic linking to accomplish this. One of the indirect pointers formed by the calling procedure's PCL points to the name of the gate access procedure. When this PCL is executed for the first time a pointer fault occurs, because the system knows only the callee's name, not the location of its entry control block. Control transfers to the PRIMOS pointer fault handler, which examines the name of the callee.

If the callee's name is not that of a valid gate access procedure, no access occurs. If it is, the fault handler locates the entry control block of that procedure, and replaces the PCL's indirect pointer to the callee's name with a pointer to the callee's entry control block. Control transfers back to the PCL, which re-executes. This time, the PCL instruction executes correctly and control transfers from caller to callee.

The pointer fault happens only the first time the gate access procedure is called in a procedure; the system automatically saves the address of that procedure's entry control block for use on any subsequent calls.

Dynamic linking offers several advantages to the user. It allows the contents of the gate access segment to be rearranged without requiring any changes to user procedures. It also allows new procedures to be added to the system easily, as well as allowing old ones to be removed. All calls to shared library routines are also made dynamically.

CONDITION MECHANISM

The condition mechanism is used to handle errors, conditions, and exceptions. Each time one of these unexpected events occurs, the condition mechanism is automatically invoked as if it were an unscheduled procedure call. This invocation suspends the currently executing procedure and transfers control to a condition handler.

This feature of the 50 Series provides orderly and consistent error handling without terminating procedure execution whenever possible. It is invoked whenever:

- Software cannot handle a condition, such as an illegal address
- A hardware or arithmetic exception occurs
- A user procedure calls SIGNL$, the PRIMOS condition signalling routine
- An external interrupt occurs, such as a user-generated break command

When any of these situations arises, the condition mechanism identifies the condition type, then transfers control to a condition handler. These handlers are called _on-units_.

On-Units

The basic element of the condition mechanism is the _on-unit_. Each on-unit is a procedure that can handle one or more specific conditions. Typical actions of an on-unit can include:

- Servicing a condition and returning control to the executing procedure
- Interrupting procedure execution and returning control to PRIMOS
- Altering the normal flow of procedure execution
- Performing some task, such as opening or closing a file
- Signaling another condition
- Running diagnostic routines
- Printing messages at the user terminal

An on-unit generally contains a series of procedure calls to one or more of the condition mechanism subroutines. Some of these subroutines can clear conditions, then restart or end procedure execution. Others can signal the occurence of a condition, scan for more on-units, create or revert an on-unit, transfer control from one part of the user's procedure to another, and perform other similar tasks.

Using On-Units

The user can define on-units in any CPL, FORTRAN 77, PL/1-G, PASCAL, or PMA procedure. These user-defined on-units can be tailored to handle a condition in whatever fashion the user wants. The system default on-unit, ANY$, is also available to the user; ANY$ intercepts any condition that may arise.

Once a user defines an on-unit, it remains in effect until the user procedure ends, the user defines a new on-unit for the same condition, or the user reverts the on-unit.

SUMMARY

The 50 Series system provides a standard method of invoking one procedure from another. Through this method, the procedure call, the system supports calls to any procedure within the system or network, and provides condition handling and access validation for every call. The system also provides the user a command environment with many built-in security and checking features. Additionally, this environment can be completely customized to meet the user's needs and applications. These topics are covered in the next chapter, The Command Environment.

9 The Command Environment

The 50 Series systems present a standard command environment suitable to a number of widespread applications. This environment, however, can be tailored in many ways to meet differing individual needs. The user can, for example, opt to use the standard environment enhanced with a few personal commands, or can choose to implement a complete menu-driven interface that governs all user actions. A spectrum of options between these two applications can also be implemented.

This chapter discusses the user command environment from several angles:

- How the system initializes it
- How the site initializes it
- How the user initializes it
- Other ways the user can use and customize it
- How to clean up and exit it at logout

SYSTEM INITIALIZATION TASKS

System initialization tasks are performed each time the user logs onto the system. These tasks center around identifying the login system and the user, and assigning access rights.

Identifying the Login System

When the user types the login command, the login system is identified either by explicit specification or by physical connection. In the case of the command

```
LOGIN user-id [password]
```

the login system is not explicitly named and the system physically connected to the user's terminal is assumed to be the login system. The user can explicitly specify the login system with the login command's -ON node option, such as:

```
LOGIN STEPHEN -ON SYSGB
```

This is a remote login, discussed in Chapter 7, PRIMENET.

Identifying the User

Once the system identifies the login system, it checks to see if the user name is valid for that system. User profiles provide this validation.

Chapter 6 mentioned user profiles and how they work with the 50 Series access control lists to ensure protection of system and user information. Among other things, a user profile may include:

- A user-id
- A user login password
- A project-id

The user-id uniquely identifies the user to the system. Associated with it is a password to ensure that access to information is restricted to the appropriate user or users. The password is encrypted into the validation files which the system uses to govern accesses, and cannot be decoded from its encrypted form. The password can be changed by the appropriate user as desired.

A project identifies a group of users. It provides similar file system access rights and validation for each member of the project. It can also be used for accounting purposes.

Assigning Access Rights

After the system identifies the user, it must assign access rights to the user. The user's profile specifies access rights; in addition, the system uses ACL groups to further determine the user's access rights.

As described in Chapter 6, access control lists specify the user's rights to access information. An ACL group is a general label that identifies a group of users. The names of these groups begin with a period (.) to distinguish them from user-ids, but otherwise use the same structure as the user-ids. Examples of ACL group names are .ADMIN, .TOOLS, .ACCOUNTING, and .DEPT_4042.

An ACL group identifies a set of users and a single set of attributes to be used for all those users. This set of attributes is specified in just one access pair, rather than one pair per user, thus simplifying the validation and checking procedures needed with each reference to the file system. This means that users with different needs can be granted different levels of access. ACL groups can be associated with projects, and adding users to or removing users from the system is easily done according to the user's membership in these projects.

ACL groups can be associated with a project-id specified in the user profiles to set up levels of access privilege. For example, an architectural firm could set up an ACL group to allow architects access to the building plans, but deny access to the financial records. In addition, the level of access privileges can be changed as time passes to meet changing demands. By adding, deleting, or changing the ACL groups, as many hierarchical levels of access can be created as the user wants.

SITE-SPECIFIC TASKS

After identification and validation are completed, any system may specify an optional set of site-specific tasks. These tasks may be as simple as displaying system notices or reminders, or may be much more complex. For example, a site may choose to invoke additional validation tasks to insulate the system from potential security breaches. Accounting functions to monitor system usage might also be invoked, or menu-driven user interfaces for data base management. The user's needs and applications will determine whether a set of site-specific tasks are needed, and what form they should take. Chapter 11, Software Products, describes the products that can aid the user in developing site-specific tasks.

INITIAL ATTACH POINT

Once the system has identified the login system, the user, and the user's access rights, and has performed any optional site-specific tasks, it must determine where to initially attach the user. In other words, what part of the file system is the user to see upon login? The user profile described above specifies the initial attach point. The system checks the user's right to access the UFD at the initial attach point, and if access is allowed, attaches the user. If access is not allowed, or if the system does not recognize the UFD name, the user is not attached.

USER TASKS

Now that the system has processed, validated, and set up the initial command environment, the user can specify a number of optional user tasks in a command file. For example, the user can:

- Invoke a command, login, or CPL file
- Activate an abbreviation file
- Change the prompt characters
- Change the terminal characteristics

This is not an exhaustive list of tasks the user can specify, but lists only some of the more commonly performed ones. System application and user needs will determine additional ones.

Command, Login, and CPL Files

The most common way for the user to specify a group of initialization tasks is with a login file. This file can be a command input file, a CPL file, or a runfile. Any of these specifies a series of tasks to be performed when the file is executed. Some typical tasks might be to print a message or reminder, set a clock, open a command output file to keep a record of the user terminal session, or invoke one of the 50 Series software products described in Chapter 11.

Login files can invoke other runfile, command or CPL files. They can also specify any of the other tasks described in this section. Login files and procedures are discussed in more detail in the System Administrator's Guide.

Abbreviation Files

PRIMOS and the 50 Series systems allow the user to define and use abbreviations for standard commands, command sequences, or command arguments. These abbreviations are contained in an abbreviation file that the system references each time the user specifies a command. To use the abbreviation file the user can place an activation command in the login file:

```
ABBREV pathname
```

where pathname identifies the name and location of the abbreviation file. The user can deactivate and reactivate the abbreviation file at any time with the -OFF and -ON options to the ABBREV command.

Using the abbreviation file has several advantages. By designating an abbreviation for a long, commonly used command, the user can save typing. It also allows the user to tailor the command environment by designating more familiar command names, for example. Abbreviations can also represent an argument in a command. The examples of typical abbreviations shown below demonstrate these advantages.

```
    EMACS      emacs %1% -terminal_type pt45 -ulib wrap.em
    CLEAN      close -all; delseg all; rls -all
    REMIND     slist my_ufd>$reminder
    AR         accounts_receivable_data_base
```

The word EMACS in the first sample abbreviation designates the name of the abbreviation. The rest of the line specifies the command and arguments that EMACS abbreviates. Note that the single word EMACS invokes the EMACS screen editor for a user working on a PT45 terminal, and specifies the library WRAP.EM to be used during editing. All the user needs to type is the abbreviation name and the name of the file to be edited. (See Chapter 11, Software Products, for a description of the EMACS screen editor.)

The second sample abbreviates three commands; it closes any open files, releases any segments, and empties the user stack. The third sample prints the contents of a user reminder file. If the fourth abbreviation, AR, were used in the ATTACH command,

```
    ATTACH AR
```

the command would expand to

```
    ATTACH ACCOUNTS_RECEIVABLE_DATA_BASE
```

This allows the user to establish simple personal commands without eliminating the descriptive advantages of the the longer directory name, ACCOUNTS_RECEIVABLE_DATA_BASE.

Prompt Characters

The standard brief prompts PRIMOS uses are OK, and ER!. The user can change these to other values with the PRIMOS RDY command. Options to the RDY command allow the user to change either prompt to some other value, to suppress all prompts, or to display CPU and I/O information along with the prompt. For example, to change the OK, prompt to System A obeys!, the command is:

```
    RDY -READY_BRIEF 'System A obeys!'
```

The RDY command can be useful to the user with access to several systems. By changing the prompts on each system to a different value, as in the example shown, it is easy to remember on which system the user is currently working.

Terminal Characteristics

The user can specify terminal characteristics other than the default values with the PRIMOS command TERM. Such characteristics are the choice of erase and kill characters, enabling or disabling of break characters, and terminal display modes. For example, the user can designate the asterisk (*) as the erase character with the command:

```
    TERM -ERASE *
```

USING THE COMMAND ENVIRONMENT

After logging in and customizing the command environment as desired, the user can begin working. Several features of the command environment, such as the abbreviation file, that can simplify the user's contact with the system, have already been discussed. The command environment offers many additional features that enhance system use. Some of the more commonly used features are:

- Status checks
- Wildcards and treewalking
- Name generation
- Iteration
- PRIMOS commands
- User application programs

The user's needs and application will indicate other features that are of particular use.

Status Checks

The STATUS command specified in a user's command file automatically checks status of the system users, networks, units, and disks. For example, to check the status of the other users on the system, the command is:

```
    STATUS USERS
```

and the system would respond by displaying something similar to:

```
User                 No  Line Devices

SYSTEM                1   asr <SYSTMA> AL057
ANNABEL               3     1 <SYSTMA> (TO OPERA )
BIZET                 6     4 <CARMEN>
MARCUS               13    13 <CARMEN> <SYSTMA>
MATT                 16    16 <CARMEN>
NETMAN               76   nsp <SYSTMA>
BATCH_SERVICE        92 phant <SYSTMA> (2)
```

This lists the other user processes currently active on the system, the process numbers, the terminal lines through which they are connected to PRIMOS, and the disk partitions they are currently accessing. See FDR3108-190, the PRIMOS Commands Reference Guide, for a complete explanation of this command and its output.

Wildcards and Treewalking

In the cases where a user wants to specify more than one file at a time, such as in a list or search command, a convenient way to do so is with wildcards. A wildcard replaces an explicit part of a filename. The symbol + designates a wildcard to replace a single character in a filename; the symbol @, a single component. The symbol ^ implies negation of an entire string. To specify a wildcard to replace one or more components, use @@.

When the file system encounters a wildcard, it performs the specified command on all files that match the wildcard. A match occurs if a filename and the wildcard have the same number of components, and if both contain the same literal characters in the same relative positions.

For example, suppose the current directory contains the six files:

```
TEST1  TEST4   SYMBOLTABLE   TEST4.RUNOFF   ALPHA   BETA
```

Suppose the user specifies @A in a command. The system would perform the desired command on files ALPHA and BETA. The wildcard TEST+ would match TEST1 and TEST4. The wildcard TEST@@ would match TEST1, TEST4, and TEST4.RUNOFF, since TEST@@ specifies all file names that have one or more components separated by periods (.) and begin with the letters TEST. Finally, the wildcard ^TEST@@ would match SYMBOLTABLE, ALPHA, and BETA, since they are the only file names that do not begin with the letters TEST.

The user can insert wildcards into pathnames to search several subdirectories in one operation. For example, if a user command specifies the pathname <CARMEN>TESTS>@@>FIGURES, the file system would seach each subUFD of the UFD TESTS for all files named FIGURES and perform the command on each file. This capability is called treewalking.

Both wildcards and treewalking have two main advantages to the user. They allow the user to perform more work with a single command. They also reduce typing and make the system easier to use, since the user needs to type only the minimum number of identifying characters to specify a pathname or file name.

Generated Names

Selected commands allow the user to generate names implicitly. The generated names are based upon the value of the first argument to the command. The symbols =, ==, +, and ^ appear in the second or greater numbered argument to indicate the format of the generated name.

A single equal sign (=) represents one component; two (==), any number of components. The symbol (+) indicates that one component should be added to the generated name, while the symbol (^) indicates that one component should be deleted.

For example, the command:

```
MRGF ALLEN>SAMPLE.ALPHA ALLEN>=.BETA ALLEN>=.GAMMA
```

expands to:

```
MRGF ALLEN>SAMPLE.ALPHA ALLEN>SAMPLE.BETA ALLEN>SAMPLE.GAMMA
```

and the command:

```
CNAME test.sample.1 =.^=.=
```

expands to:

```
CNAME test.sample.1 test.1
```

Like wildcards and iteration, name generation eliminates the need for typing. The user has to type a common component or file name only once in a command.

Iteration

Many tasks can be simplified by using iteration. A set of values is enclosed in parentheses and inserted in a regular command. The system executes the command once for each value within the parentheses. In other words, a command containing an iteration set of four values is equivalent to four separate commands. For example, the command

```
DELETE (TEST PROGRAM MYFILE)
```

is equivalent to the three commands

```
DELETE TEST
DELETE PROGRAM
DELETE MYFILE
```

and the command

```
CNAME (TEST PROGRAM MYFILE)  =.SAMPLE
```

expands to the three commands

```
CNAME   TEST      TEST.SAMPLE
CNAME   PROGRAM   PROGRAM.SAMPLE
CNAME   MYFILE    MYFILE.SAMPLE
```

Multiple iteration sets can be inserted into some commands, such as

```
CNAME (TEST PROGRAM MYFILE)  (ALPHA BETA GAMMA).SAMPLE
```

expands to the three commands

```
CNAME   TEST      ALPHA.SAMPLE
CNAME   PROGRAM   BETA.SAMPLE
CNAME   MYFILE    GAMMA.SAMPLE
```

This feature can be applied in any situation where the user must issue a number of redundant commands that differ only in the arguments specified.

PRIMOS Commands

This chapter has already mentioned some PRIMOS commands, such as RDY, CNAME, and STATUS. The user can specify a number of other commands, both internal and external, in a command file. Internal commands, such as the three mentioned in this paragraph, are executed in the address space that PRIMOS occupies. The majority of external commands are used to invoke programming and debugging facilities and execute in the user's address space. At Rev 19.0 and later versions of PRIMOS, however, there are some external commands which do not execute in the user's address space.

Tables 9-1 lists the internal PRIMOS commands. Table 9-2 lists those external commands that execute in the user's address space, and Table 9-3 lists other commands, which do not overwrite the user's address space. For more detailed information on all PRIMOS commands, see the PRIMOS Commands Reference Guide.

Table 9-1
Internal PRIMOS Commands

*	ABBREV	ADDISK	ADD_REMOTE_ID
AMLC	ASRCWD	ASSIGN	ATTACH
BINARY	CHANGE_PASSWORD	CHAP	CLOSE
CNAME	COMINPUT	COMOUTPUT	CPL
CREATE	DATE	DEFINE_GVAR	DELAY
DELSEG	DISKS	DMSTK	DROPDTR
EDIT_ACCESS	ELIGTS	INPUT	LISTING
LIST_ACCESS	LIST_GROUP	LIST_PRIORITY_ACCESS	
LIST_QUOTA	LIST_REMOTE_ID	LOGIN	LOGOUT
LOOK	MAXSCH	MAXUSR	MESSAGE
NET	OPEN	OPRPRI	ORIGIN
PASSWD	PHANTOM	PM	PRERR
RDY	REMOTE	REMOVE_PRIORITY_ACCESS	
REN	REPLY	RESUME	RLS
SAVE	SETIME	SETMOD	SET_ACCESS
SET_PRIORITY_ACCESS		SET_QUOTA	SHARE
SHUTDN	START	STARTUP	STATUS
TIME	UNASSIGN	USAGE	USERS
USRASR			

Table 9-2
External PRIMOS Commands

AVAIL	BASIC	BASICV	BASINP	BATCH	BATGEN
BATGEN	CMDL	CLUP	CMPF	COBOL	CONCAT
COPY_DISK	CPMPC	CRMPC	CSUBS	DBACP	DBASIC
DBG	DBUTL	DPTCFG	DPTX	ED	EDB
EDIT_PROFILE		F77	FAP	FDL	FDML
FILMEM	FILVER	FIX_DISK	FSUBS	FTGEN	FTN
FTOP	FTR	HDXSTAT	HPSD	JOB	KBUILD
KIDDEL	LABEL	LATE	LOAD	LOGPRT	MAGNET
MAGRST	MAGSAV	MAKE	MCLUP	MDL	MRGF
NCOBOL	NETCFG	NETPRT	NSED	NUMBER	OWLDSC
PL1G	PMA	POWER	PRMPC	PROP	PRSER
PRTDSC	PSD	PSD20	REMAKE	RJOP	RJQ
RPG	RUNOFF	SCHDEC	SCHED	SCHEMA	SEG
SIZE	SLIST	SORT	SPOOL	SPSS	TAP
TCF	TERM	TRAMLC	UPCASE	VPSD	VPSD16
$$					

Table 9-3
Other PRIMOS Commands

COPY	DELETE	HELP	LD	PROTECT
REVERT_PASSWORD		RWLOCK	SET_DELETE	

User Application Programs

In addition to the PRIMOS commands listed above, the user may choose to invoke an applications program via a command file. This is a convenient way to invoke a simple user clock program, for example, or a comprehensive menu-type user interface.

The command file can invoke any of the software products described in Chapter 11 to provide data base management, CAD/CAM capabilities, or network access, to list some examples. The command file can also invoke one of the languages supported on the 50 Series, or a development tool such as a loader, spooler, or editor. See Chapter 11 for a description of these and the many other software products supported on the 50 Series systems.

A CPL file can also be used to initiate an update mechanism, for example, ensuring that any change the user makes is propagated throughout all relevant channels (database, logbook, error fix file, and CAD data). The user's needs and scope of use will determine many additional ways these structures can be used.

```
&args treename
como -ntty
&if [exists %treename%] &then &do
    &data ed product>installation_log
          bottom
          insert [date -full] Installing [pathname %treename%].
          file
          &end
   &if [exists product>source>[entryname %treename%]] &then &do
       copy product>source>[entryname %treename%]
            product>arc>[entryname %treename%].[date -ftag] -dtm
       &data ed product>installation_log
             bottom
             insert [date -full] Archived [entryname %treename%]
             file
             &end
       &end
   copy %treename% product>source>[entryname %treename%] -dtm -nq
   &data ed product>installation_log
         bottom
         insert [date -full] Installed new [entryname %treename%]
         file
         &end
   como -tty
   type Installation of [pathname %treename%] completed.
   &end
&else &do
   como -tty
   type Cannot find [pathname %treename%].
   &return 1
   &end
```

Sample CPL Program
Figure 9-1

Figure 9-1 shows a simple CPL program to install a new source file, and monitor the installation. The program archives the previous version of the source file if necessary, copies the new file, maintains a log of these activities, and notifies the user whether the file is successfully installed or does not exist.

LOGGING OUT

When work on the system is complete, the user logs out. As upon login, user, site, and system tasks can occur. The user may choose to invoke some application, such as printing a message, and these tasks are performed first. Upon their completion, any site-specific tasks are performed. These, like the user tasks, are optional. Upon their completion, the system closes any files the user may have left open and releases control of any system resources the user held. It then removes the user process from the system.

SUMMARY

This chapter has briefly described the 50 Series command environment and how it can be customized to suit the user's needs and applications. The next chapter describes integrity features of the 50 Series systems and how they provide the user with a stable and secure system environment.

10 Integrity

The preceding chapters discussed major functional aspects of the 50 Series architecture and how they allow a system to be tailored to fit a variety of applications. The last aspect to be discussed ties together several points made in these sections. This aspect, integrity, represents the features of the 50 Series that help to maintain the proper operation of the system. Integrity encompasses both software and hardware elements.

SOFTWARE INTEGRITY ASSURANCES

Some of these integrity assurances have been mentioned in earlier chapters. The major elements are:

- Embedded operating system
- Access control lists
- User profiles
- Error logging mechanisms
- FIX_DISK

Embedded Operating System

As described in Chapter 4, Prime's software implementation of virtual addressing allows each user an address space of 512 megabytes. 256 Mbytes of this address space is reserved for user programs and data. The other half of the user address space contains PRIMOS and shared subsystems. In other words, each user working on the system shares a single copy of PRIMOS and shared subsystems, embedded in the user virtual address space.

There are several advantages to this embedded operating system. It provides reduced system overhead, since there is no need to make calls outside the user's address space when PRIMOS must perform some task on behalf of the user. Sharing subsystems is easy as well, since each user has access to the single copy of each shared subsystem. This embedded operating system also maintains separate copies of the private sections in each user address space, so there is no chance of one user accidentally destroying the contents of another's private space.

For security purposes, the operating system includes hardware to validate the access rights of all memory references. The gate accesses described in Chapter 8 also allow the user to access the operating system directly, in a controlled and secure fashion.

Access Control Lists

Prime provides access control list protection for the user who desires passive, secure, automatic file protection. This protection can be applied to any file or directory in the system. For more information about ACLs, see Chapter 6, File Management, and Chapter 9, The Command Environment.

User Profiles

User profiles are lists of information with which PRIMOS identifies the user logging into the system. This information includes details about the forms of protection to which the user is subject, what parts of the data base the user can legally access, and other such specifics. PRIMOS uses the user profiles in tandem with the access control lists to guard against invalid accesses to protected parts of the file system. See Chapters 6 and 9 for more information.

Event Logging Mechanisms

The 50 Series systems have an event logging mechanism to record the occurrence of events such as disk errors, cold and warm starts, machine checks, and single and double bit memory data errors. There is also a network event logging mechanism that records network events. These mechanisms store a record about each event in internal buffers; PRIMOS dumps the contents of these buffers to disk files or the system terminal. When the mechanisms are invoked, they automatically format the contents of these files and show when each event occurred.

FIX_DISK

FIX_DISK is a file system consistency check program. It checks that the disk record headers and the UFD containing the header both reflect the same state of the file system. When it encounters an inconsistency, it displays a message and tries to resolve the problem. FIX_DISK can fix many of the errors it encounters, such as mismatched pointers and inconsistent quota information, without user intervention.

HARDWARE INTEGRITY ASSURANCES

Prime 50 series computers have several built-in hardware features that ensure integrity. The major hardware assurances are:

- Rings
- Interrupts
- Faults
- Checks
- Traps
- A diagnostic status word (DSW)
- Error checking and correction for all disk records
- Parity checking of the cache, all registers, and all busses
- Microverification capability
- The virtual control panel (VCP)

Rings

ACLs are not the only means of data protection available on Prime 50 Series systems. ACLs can provide protection for information located on disk, but they do not provide for permanent protection of programs and information vital to the system, such as PRIMOS itself. Three hardware implemented rings and the level of privileges associated with them provide security for crucial information in virtual memory. Chapter 4, Memory Management, describes rings and how they govern memory references.

Interrupts

Peripheral devices cause an interrupt to signal their need for system service, as described in Chapter 5, Input/Output Management. Once the interrupt is acknowledged, control transfers to phantom interrupt code for simple handling. If more complex service is necessary, a device interrupt manager is called to complete service. See Chapter 5 for more details.

Faults

A fault is a break in software execution that occurs synchronous to system operation. Examples of faults are page faults, where a reference is made to a page not currently loaded in physical memory, and stack overflow or underflow.

When a fault occurs, an unscheduled procedure call is made to the appropriate PRIMOS fault handler. This handler may set system registers to indicate the type of fault that occurred, and performs the actions necessary to clear the fault condition. In the case of user applications the handler may invoke the condition mechanism so that user-defined actions can be performed. Once the fault is cleared, the fault handler transfers control back to the instruction that was executing when the fault occurred.

Checks

When an uncorrectable system hardware problem arises, a check occurs. Problems of this sort are usually serious enough to halt all system operations.

Four types of checks can occur. When AC power fails, the power supply initiates a power fail check. This check indicates that 20 milliseconds of DC power remain before all power is gone. The memory error checking logic issues a memory parity error check when it detects a memory parity error or an uncorrectable memory error. The CPU issues a machine check when it detects an internal parity error. Finally, the memory control unit initiates a missing memory module check when the system tries to access nonexistent physical memory.

When a check occurs, the system references one of the four check vectors. The check vector points to the firmware-implemented check handling routine. Information about the check is stored in the diagnostic status word (described below).

Traps

Traps are breaks in firmware execution and may take one of two general forms. The first form is fully processed by the firmware and is followed by a return to normal execution. Examples of this form are write address trap and DMQ. The second form generates a fault or check and does not return. Examples of this form of trap are access violation, and internal parity error.

All traps are serviced, as are all other exception conditions, on a priority basis. They are not visible to the user.

Diagnostic Status Word

The diagnostic status word contains 96 bits (250-II and 550-II) or 128 bits (750 and 850) of information about the state of the system at the time of a check. This information identifies the type of check, where it occurred during execution, and what part of the system (byte, module, etc.) was affected.

Error Checking and Correcting (ECC) Code

Error correcting codes on each word of physical memory note when single and double bit errors occur. Single bit errors are automatically corrected in memory with no break in execution. Double bit errors are reported.

Parity Checking

To ensure data integrity, hardware checks the parity of data traveling over internal or external busses, between main memory and the processor, and between register set locations, as well as checking all cache data. If a parity error is found, a machine check occurs.

Microverification Capability

All Prime 50 Series computers have microverification programs as a standard feature. These microprograms execute each time the system is brought up from a cold start, testing almost all parts of the system (except for clocks). Any failures are reported to the operator for further actions.

Virtual Control Panel (VCP)

The VCP, on all 50 series systems except the 2250, allows the user to perform typical supervisor terminal operations, such as bringing up the system and sending messages. It also allows control panel operations such as initiating diagnostics, bootstrapping PRIMOS, master clears, and system halts. The VCP also allows system diagnosis or control to be performed from a remote location.

The diagnostic processor on the 2250 performs all the functions of the VCP. Further, it provides an automatic boot for PRIMOS, and also allows use of the supervisor terminal as a user terminal.

SUMMARY

The 50 Series has several software and hardware integrity features built in to eliminate system downtime to every extent possible. These features enhance the already efficient functions of the architecture by offering contingencies for all possible exception conditions. The result is greater reliability for the system as a whole, and orderly processing in the event an exception does occur.

11 Software Products

Prime's software products are divided into eight groups:

- Languages
- Language support facilities
- Development tools
- Data management systems
- Computer aided design/computer aided manufacturing (CAD/CAM)
- Communication services
- Backup facilities
- Office automation

LANGUAGES

Prime supports many industry-standard programming languages on its 50 Series systems to offer the user the ability to tailor applications closely to specific needs. Most of them support the debugging and monitor functions of the source level debugger (see Language Support Utilities and Development Tools, below). All of them support the standard Prime procedure call (see Chapter 8) when making references from one procedure to another. This control transfer standard means that the user can easily combine procedures written in different languages.

Prime's <u>interlanguage interface</u> further enhances the user's ability to tailor applications programs to specific needs. With this interface, the user can call procedures written in one language from those written in another. This feature means one basic routine can be used in many different situations without communications problems. This interface also allows the user to produce procedures in the language most appropriate to the application, since complex interfaces between mixed-language procedures are unnecessary.

FORTRAN 77

FORTRAN 77 (F77) is supported on all Prime 50 Series systems. It is an extended implementation of an ANSI standard, ANSI X.39-1978. F77 uses a three pass compiler to perform a high degree of optimization. It also supports many mainframe extensions such as local and block code optimization to produce efficient object programs, embedded comments, 32-character names, IBM-compatible direct access and namelist I/O, extended range DO loops, the character datatype, IF-THEN-ELSE blocks, and enhancements to I/O operations. The user can access DBMS, MIDASPLUS, and FORMS from F77. F77 is fully supported by DBG and EMACS.

BASIC/VM

Prime's BASIC/VM (BASICV) is a multiple-user implementation of the BASIC language. It is supported on all 50 Series systems and allows the user to perform character string and matrix operations, structured programming, and output formatting. In addition, the user can choose one of three modes of execution supported by BASICV: conversational, for terminal operations; batch mode, for previously prepared programs; or immediate, for calculator-type use. BASICV has its own debugger for error detection and can be used in conjunction with MIDASPLUS, described below.

Pascal

The Pascal language implemented by Prime is based on the preliminary standard issued jointly by the IEEE and ANSI. It uses a multipass compiler to generate highly optimized object files. Like all other languages supported on the 50 Series, Pascal supports Prime's standard procedure call conventions to provide access to files regardless of the format type of both data and calling program. DBG supports Pascal, as described in the <u>Language Support</u> section of this chapter.

PL/I Subset G

Prime's PL/I-G (PL1G) is a structured language appropriate for general purpose, modular programming. It fully implements the ANSI Standard X3.74-1980 for PL/I-G. In addition, PL1G supports Prime's subroutine calling conventions, and its object and data files are compatible with those of other languages. It directly supports FORMS and MIDAS, two of Prime's data management packages (see sections later in this chapter). DBG also supports PL1G (see the Language Support section in this chapter) to provide efficient error tracing and correction.

CBL

Prime's COBOL, CBL, is an interactive, business-oriented language based on the 1974 ANSI COBOL standard. The user writing CBL programs can choose between two data management systems, MIDAS and DBMS, to control the information the CBL programs manipulate (see the Data Management section in this chapter for more information about MIDAS and DBMS). The user can also combine CBL with FORMS, Prime's forms management product, to create and use screen formats. CBL supports decimal arithmetic and character operations in addition to Prime's standard procedure call conventions. It is fully supported by DBG and EMACS.

VRPG

This business-oriented language is compatible with IBM System/3 Model 10 RPG II. Initially designed to produce reports, VRPG allows the user to use RPG applications written for other systems on a 50 Series system. It can use the MIDASPLUS data management package to manipulate files, and is supported by EMACS and DBG. With VRPG the user can invoke the FORMS forms management system to design and create business forms and charts.

CPL

Prime's command procedure language is called CPL. This powerful programming tool embodies many high level language features such as branching, argument transfer, and global variables, and also supports the condition mechanism (see Chapter 8) to provide orderly error handling. It allows the user to store sequences of PRIMOS commands and CPL statements in a command procedure file (see Chapter 9) to be executed whenever the name of the file is specified.

CPL can be used to simplify complex command strings, to reduce typing, or to write applications programs from simple login procedures to complex menu-driven user interfaces. CPL is also useful for unattended control of processes, compilation tasks, and standard procedures that are frequently performed.

Prime Macro Assembler Assembly Language

Prime's PMA assembly language is of interest to assembly language programmers and to Prime systems internals specialists. It provides over 600 different instructions applicable to almost any circumstance. Supported by DBG, SEG, and many other Prime system utilities, it allows the user to develop a wide range of assembly language applications that can be easily integrated with the rest of the system. PMA routines can also be effectively combined with those written in high level languages to enhance the overall performance of large applications.

LANGUAGE SUPPORT UTILITIES AND DEVELOPMENT TOOLS

To complement the array of languages supported on the 50 Series systems, Prime supplies several utilities to assist program development. In addition, many development tools are available to aid the design and programming processes, report writing, documentation, and training. A group of output facilities is also available.

Source Level Debugger (DBG)

The source level debugger is a language-independent programming tool for use with high-level languages. It allows the user to interactively control and monitor F77, PL1G, Pascal, CBL, and VRPG programs at the source code level. The debugger is designed for all levels of user expertise, and no knowledge of assembly language is necessary to control debugger actions.

A conversational user interface is easy to use and does not detract from the debugger's speed of execution. Since the debugger is language independent, it provides the user with a consistent set of commands and procedures to use. Features include the ability to trace variables as well as examine, execute, monitor, and test source code interactively.

SEG

SEG is a linkage editor that allows the user to take full advantange of the 50 Series systems' virtual memory capabilities. It converts the user's object file into a segmented runfile that can be up to 32 Mbytes in size. It can also load a single or multiple programs, optimize the user's program in some areas, load shared procedures, and perform dynamic linking.

HELP Facility

Prime's software products are equipped with a HELP facility to aid the user. When the user types HELP, the system displays a list of available commands and their arguments. A further request for help with a particular command produces more detailed information, such as command actions, optional parameters, and so on. This facility can be tailored to offer information about any topics the user chooses.

EMACS

EMACS is a interactive screen editor. It is programmable, allowing the user to set up and define personal commands. Also provided is a HELP facility to help the user identify typing mistakes, available commands, and specifics of command use. With EMACS the user can enter text and modify it, move blocks from one part of the text to another, save the text, delete text, and perform many other tasks. A status line continuously displays the status of the editor and reports on the actions being performed. In addition, three programming modes automatically aid the user in formatting programs written in F77, CBL, and RPG.

ED

ED is a line-by-line text editor that is most useful for entering and modifying line-oriented items such as programs. The user has the choice of displaying input one line at a time or several at once. It also allows the user to move information between files; add, change, and delete information; and perform many other tasks.

RUNOFF

RUNOFF is Prime's text formatting program. While creating or editing a file with one of Prime's text editors, the user inserts runoff commands into the file that specify how the text is to be processed. When the user invokes RUNOFF to process the file, it automatically formats the file based on the inserted commands. Commands are available to specify tab stops, page size, headers, footers, revision bars, degree of indentation, and decimalized headings, among others.

SPOOL

The user invokes SPOOL to produce a printed copy of a file. This utility automatically queues the user's file on one of the system spool queues to await printing. If the printer on the user's local system is very busy, SPOOL checks for an available printer with less activity to evenly distribute the load. In addition, several options allow the user to check the status of spool queues, delete a file from a queue, print multiple copies of a file, and queue files for printing at a later time.

BATCH

The BATCH facility allows the user to submit programs for execution when it is most appropriate for the system to do so. The facility is a series of batch queues, each of which may represent a different set of parameters, such as CPU time or process priority. When the system is very busy, almost no jobs waiting on the batch queues are run; when the system is otherwise idle (at night, for example), batch jobs can be executed.

This allows the user to submit time- and resource-consuming jobs for execution when the system does not have to service many other processes. The BATCH facility also allows the user to monitor a job from a terminal, cancel jobs, change a job's parameters while in the queue, and list available batch queues and their characteristics.

DATA MANAGEMENT PACKAGES

The 50 Series supports several different data management packages so that the user can pick the one that best suits the application. For those frequently generating and using business forms, the FORMS package is available. Creating and managing data bases is accomplished with the Data Base Management System, DBMS. The PRIME/POWERPLUS package provides report generation, information gathering, and query capabilities. The user who needs keyed-index file capability can achieve this with MIDASPLUS, the Multiple Indexed Data Access System. To accompany the line of Information systems, Prime offers Information software.

FORMS

The forms management system, FORMS, provides both the end user and the programmer with an easy-to-use facility for generating and using business forms. The system handles input and output internally, so programmers can quickly create portable applications using FORTRAN, CBL, VRPG, and PMA programs. The FORMS Editor, FED, provides the user with an interactive, screen-oriented interface for easy form creation. The FORMS Definition Language, FDL, allows the user to create a file that describes the format of a form. The FORMS Administrative Processor, FAP, aids the user in maintaining a catalog of forms for future use.

DBMS

Prime's data base management system is called DBMS. It is a sophisticated implementation of the CODASYL data base standard and allows the user to create, modify, and maintain data for a wide variety of applications. Some additional key features of DBMS include a transaction-based, interactive user interface; data independence; and minimum data redundancy.

With the DBMS data definition language (DDL), the user can create structures called schemas and subschemas within the DBMS data base for FORTRAN or CBL (Prime's COBOL). These structures allow the user to organize the DBMS information in a variety of ways. Once a schema or subschema has been created, the user can use the data manipulation language (DML) to access the information.

To protect the user's data, DBMS contains extensive validation and recovery features. These include validation of DML commands before the data base is altered, and commands that can in many cases restore schemas and subschemas when a transaction is interrupted. In addition, when a faulty transaction occurs, the data base can be restored to its state before that transaction (rolled back).

DBMS/QUERY

Prime's DBMS/QUERY is a query language and report writer for use with DBMS data bases. With a simple set of commands, the user can direct QUERY to search for information according to a variety of criteria and present it in a suitable fashion. No knowledge of DBMS or programming is needed to use QUERY.

PRIME/POWERPLUS

PRIME/POWERPLUS is a data management system designed to aid the decision-making process. It allows the user to make ad hoc queries or reports, collect data into various forms, and process the data. Rather than a device for creating and maintaining a data management structure, PRIME/POWERPLUS is a tool to help the user collect and analyze data.

Parts of the PRIME/POWERPLUS system include a data dictionary for defining, processing, and linking data; a query processor for specifying search criteria; and an interactive report generator that allows the user to set up individualized formats. PRIME/POWERPLUS can also perform simple file maintenance, text processing, and system administration. In addition, a comprehensive help facility is available to the user, as are built-in security procedures for data protection.

MIDASPLUS

MIDASPLUS, Prime's multiple index data access system, offers the user an interactive means to create and maintain keyed-index files. Through MIDASPLUS, the user can create and use MIDASPLUS files, check file status, add data to and delete data from MIDASPLUS files, and index information with up to 18 keys. Application programs written in FORTRAN, CBL, PL/I-G, Pascal, BASIC/VM, VRPG, and PMA can access files created with MIDASPLUS. Other features of MIDASPLUS are:

- Multiuser support
- Network support
- PRIME/POWERPLUS support
- Dynamic file allocation
- Dynamic maintenance of index structures
- Interactive environment

Prime INFORMATION Software

The INFORMATION software provides user-oriented data management for the line of INFORMATION systems (see the section, "CPUs", in Chapter 12). The software consists of a data base management system, INFO/DMS, that supports a variable-length, hierarchical file system similar to that supported by PRIMOS. Complementing INFO/DMS are four modules:

- INFORM
- INFO/BASIC
- EDITOR
- PERFORM

INFORM is used to search the INFORMATION file system for data and to format the retrieved data. Its English-based interface allows easy file access and simplifies report generation. This module includes the ENTRO processor, a structured update facility that creates and modifies system files according to user command.

A structured procedural language, INFO/BASIC, combines features of COBOL, Pascal, PL/1 and BASIC to aid the user with business programming tasks, such as inventory control and sales analysis, as well as with arithmetic or computational tasks. INFO/BASIC is based on the standard Dartmouth BASIC language.

For modifying and maintenance tasks, EDITOR is a powerful tool that comes with its own HELP facility. This line-oriented text editor allows the user to create and modify files containing text, data, or programs. It also can be used to create EDITOR command files that function in the same fashion as CPL and command files (see Chapter 9).

A multi-user command and control facility, PERFORM is based on the PRIMOS operating system and has a simple user interface. It accepts the user's command and directs it to the INFORMATION Processor that can perform the specified command.

CAD/CAM - MEDUSA

MEDUSA is a sophisticated CAD/CAM package available for all 50 Series systems. It has been tailored for multiuser, interactive environments. Features of MEDUSA include many state-of-the-art capabilities that can speed product development in many fields. With MEDUSA's abilities the user can draft civil and mechanical engineering diagrams; produce electrical schematics, floor plans, and flow diagrams; and model two- and three-dimensional solids.

The user has the choice of two versions of MEDUSA. The two-dimensional (2-D) version is primarily a schematic and drafting system with powerful ANSI, ISO, and BS dimensioning abilities. It also features a versatile annotation/editing function for record keeping, and a macro facility. Extensive symbol libraries make this version of MEDUSA applicable to many different tasks. There is also a choice of output devices (drum, flatbed, or electrostatic plotters).

With the 2-D version of MEDUSA, the user has available for use:

- Automatic cross hatching
- A macro language
- Multiple type fonts
- Any number of views of one drawing
- Semi-automatic dimensioning to any accuracy
- Mirror capability
- Line and geometric array capability
- Parametric macro definition of objects
- Conic curves
- Automatic filleting
- Curve fitting by quadratic splines or by least squares

The 2-D version also has documentation capabilities that the user can invoke to generate reports, parts lists, schedules, bills of material, or other necessary documents.

The second version of MEDUSA is three dimensional (3-D). This version allows the user to design a complex three-dimensional device and produce an engineering drawing of it. Standard features of this version are:

- A simple and easy-to-use 2-D drawing convention through which the user creates 3-D models
- A simple language
- Ability to generate arbitrary, orthogonal, oblique, axonometric, and perspective projections
- Hidden lines in dashed, visible, or invisible formats
- Automatic construction of 3-D objects (the 3-D modeller scans 2-D drawings)
- A 3-D viewer that can section any solid
- An interface to GNC for numerical control of machine tools

MEDUSA is a CAD product designed to streamline the drafting and product development process in fields ranging from mechanical engineering to architecture. It provides a user interface that is both easy to use and powerful in its degree of widespread application. It integrates input from many users into one data base to provide information that is always up to date. In addition, MEDUSA's modular design makes it useful for both beginning and sophisticated users.

COMMUNICATION PACKAGES

Prime's communication packages are summarized below. Refer to Chapter 7 for more details about PRIMENET, NETLINK, and FTS.

PRIMENET

PRIMENET supports communications between linked systems in a fashion transparent to the user. This transparency eliminates the need to learn new commands, details about the link between systems, or details about the physical location of information. Instead, PRIMENET makes referencing remote information identical to referencing local information.

In addition to its transparency and ease of use, PRIMENET software meets the CCITT X.25 standard for packet switching networks. This feature is advantageous in situations requiring domestic and international networks, since PRIMENET's X.25 support allows it to communicate with any other system that also supports X.25. Two other types of networks, ring and point-to-point synchronous, are also supported; the combination of the three types gives the user several options that can be exercised according to needs and applications.

NETLINK

When logged onto a system connected in any PRIMENET-supported network, the user can invoke NETLINK to remotely log onto any other system in the network. Rather than accessing a remote resource, performing a task, and then returning information to the user, NETLINK allows the user to actively work on the remote system as a local user would. Once logged onto the remote system, the user can invoke NETLINK to log onto another remote system, and so on. See Chapter 7, PRIMENET, for more information about NETLINK.

File Transfer Service

To complement the capabilities of PRIMENET, the 50 Series systems support the file transfer service (FTS) subsystem. This facility transfers files between the local system and any PRIMENET-linked remote system. See Chapter 7, PRIMENET, for more information about FTS.

DPTX

The distributed processing terminal executive (DPTX) software allows the 50 Series to exchange information with the IBM 3270 family of block mode CRT devices and controllers. With this mainframe compatibility, the 50 Series meets the needs of the user who has already invested in large amounts of IBM equipment.

DPTX supports three types of communication with the IBM 3270 equipment:

- 3270 data stream compatibility (DPTX/DSC)
- 3270 terminal support facility (DPTX/TSF)
- 3270 transparent connect facility (DPTX/TCF)

DPTX/DSC allows the 50 Series to interact with an IBM mainframe that supports 3270 devices. DPTX/DSC emulates the actions of a 3270 device so the mainframe sees the Prime system as a CRT device rather than another complete system. Through this type of communication users can interact with programs on the IBM mainframe as if they were directly connected to it.

DPTX uses DPTX/TSF to control a network of 3270 devices. This means that users active on 3270 devices can interact with PRIMOS and applications that run under PRIMOS.

DPTX/TCF makes the 50 Series act as a transparent link between an IBM mainframe and a 3270 device connected to the Prime system. Those using the 3270 device are able to interact with programs on the mainframe as if they were directly connected to it.

DPTX provides the means for a 50 Series system to communicate with IBM equipment. It also allows the IBM system to delegate some of its processing tasks to the 50 Series. In addition, supporting and emulating IBM devices gives the 50 Series the opportunity to extend its accessing power to a vast number of systems. IBM systems, however, are not the only systems with which the 50 Series can communicate. The next section describes Prime's RJE facility, which supports communications with the equipment of several other manufacturers.

Remote Job Entry

Prime's remote job entry facility allows the 50 Series to interface with systems made by a number of other companies. Users can produce jobs on a Prime machine, then use RJE to submit it to another system for execution.

The facility consists of several emulators that allow a Prime system to be connected to a host computer via telephone lines or some other means of communication. These emulators make the Prime system appear to be a RJE terminal of the type normally supported by the host.

Through RJE, a 50 Series system is compatible with a variety of popular mainframe systems. This is especially an advantage to the mainframe user, because it protects previous hardware investments, and preserves mainframe communications protocols. RJE also allows a 50 Series system to serve as an offload system, which frees the mainframe for other tasks.

The RJE terminals that the Prime system can emulate are:

- IBM 2780, 3780, and HASP
- Honeywell GRTS
- Univac 1004

- CDC 200UT
- ICL 7020 and XBM

These emulators allow a user to submit remote jobs from any terminal in the Prime system. They also allow the user to submit a remote job to more than one remote system at the same time.

Each emulator has three parts. The send facility translates and formats the file into the form expected by the host computer and places it in a queue for transmission to the host. After the user submits a file via a send facility command, the symbiont removes the file from the transmission queue and sends it to or receives it from the host. The workstation allows the transmission, code conversion, and reception of files between the Prime and the host systems.

BACKUP UTILITIES

There are a variety of backup utilities available to the user. MAGNET transfers information from one tape to another. MAGSAV, MAGRST, PHYSAV, and PHYRST perform disk-to-tape or tape-to-disk transfers. COPY_DISK is available for disk-to-disk transfers.

MAGNET

MAGNET transfers file contents in a standard format between tape and disk, or between two tapes. It is also useful for translation of EBCDIC-formatted data into BCD- or ASCII-formatted data during the data transfer, making it useful for transferring file contents between a Prime and a non-Prime system.

MAGSAV and MAGRST

Like MAGNET, MAGSAV and MAGRST allow the user to back up and archive information. These two utilities, however, transfer file data as well as the logical file system structure that governs them between tape and disk. Both of the utilities move SAM and DAM files, segment directories, UFDs, ACLs, quotas, and partitions so that the file structure on source and destination devices is identical. MAGSAV saves the contents of a disk on tape. MAGRST loads the contents of a tape onto a disk.

PHYSAV and PHYRST

To create a disk image backup on magnetic tape, the user specifies PHYSAV. PHYRST will copy the contents of the tape back into the disk partition. The smallest unit of information the user can save or restore with these two utilities is one partition.

COPY_DISK

To copy the contents of one disk onto another, the user specifies the COPY_DISK utility. This utility copies the disk contents and optionally verifies them to ensure physical correctness. The copying process for a full 300 MB disk takes approximately one hour. This utility is particularly useful on a large system, since it takes less time than any of the disk-to-tape backup routines.

OFFICE AUTOMATION

The Prime Office Automation package integrates word processing, management communications, advanced text processing, and data processing for use with all 50 Series systems. It is designed to improve productivity in the workplace and to speed the transfer of information for managerial, professional, and administrative personnel.

Office Automation is implemented in software modules that run under Prime's operating system, PRIMOS. These modules are:

- Word processing
- Management communications and support
- Advanced text management

Word Processing

With Office Automation's word processing, one or more users can easily work with documents of any size. This module includes a screen editor, a menu-driven user interface, and several powerful text editing capabilities. Some of these capabilities are:

- Editing
- Filing and retrieval
- Abbreviation storage in a boilerplate library

- List processing
- Word processing/data processing conversion

Management Communications and Support

This module of Office Automation consists of three parts. Electronic mail helps the user create and distribute documents. It also provides filing, annotation, and acknowledgement features so the user can act upon mail as is appropriate. Correspondence management further aids in managing the flow of information between users with a filing capability, retrieval service, and report generator. Management support keeps users abreast of appointments, pending deadlines, and other activities. Additional support features include an electronic intray, a two-month calendar, a scheduler for noting appointments, and a prompter that keeps track of pending affairs.

Advanced Text Management

Advanced Text Management provides the user with spelling and single-word translation dictionaries in several languages. It also performs automatic hyphenation. This module is most useful for proofreading and translation tasks.

In summary, the Office Automation system fulfills a multitude of office functions in an efficient and easily understood fashion.

SUMMARY

This chapter has given a thumbnail sketch of the software products available for the 50 Series systems. Products exist for users in almost any field of application. In addition to Prime's own products, many other packages tailored specifically for individual fields, such as banking, finance, and administration, are available. For more details about these and Prime's own software products, see your sales representative.

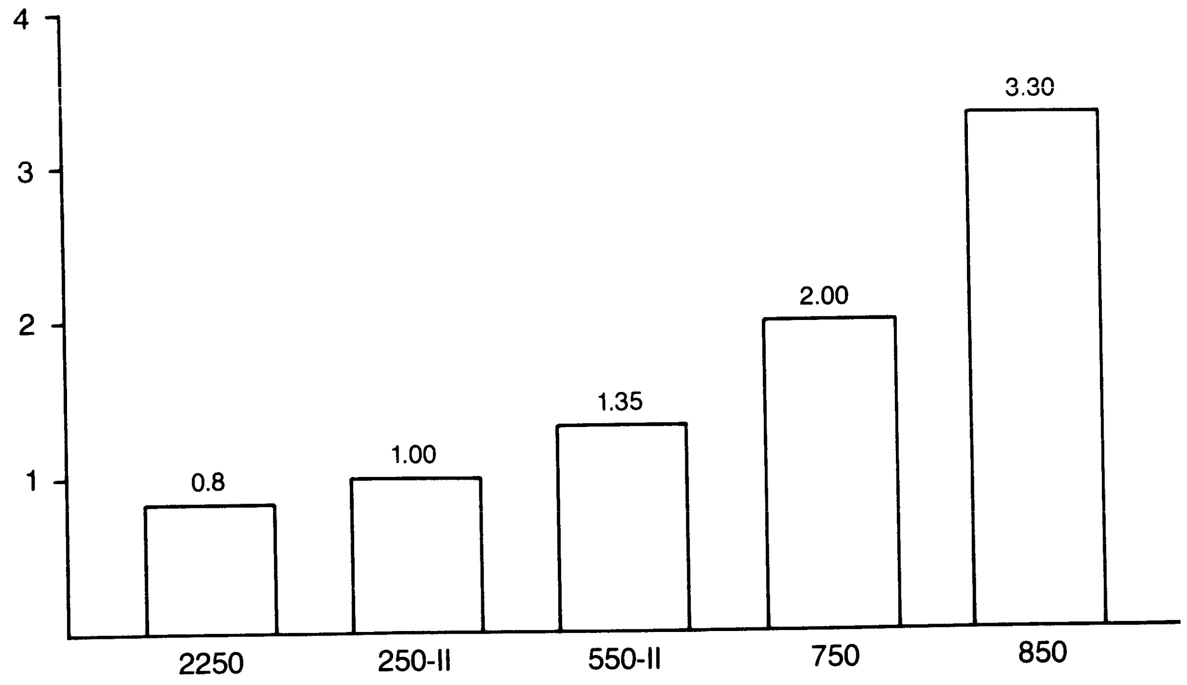
12 Hardware Products

Hardware products are divided into six groups:

- Central processing units (CPUs)
- Memory expansion units
- Terminals
- Magnetic tapes
- Disks
- Unit record devices

CPUs

The CPU is the heart of the 50 Series systems. Prime offers a group of direct-sale CPUs of varying capabilities and power to address the needs of end-users in many fields and applications. Figure 12-1 contrasts the relative performances of the 50 Series systems in a multi-user, computational environment.



Relative Performance of 50 Series Systems
Figure 12-1

Prime 2250

The 2250, a compact entry-level system designed to fit easily into any office environment, is an economical high-performance system which is ideal for distributed processing network nodes or compact, multiuser system applications. Standard components of the 2250 include:

- 32-bit 2250 CPU
- 512 Kbytes of error correcting memory (expandable to 4 Mbyte)
- One 68-Megabyte disk and one cartridge tape unit, with a multifunctional disk/tape controller
- A communications controller with 8 asynchronous and 1 synchronous communications lines
- A diagnostic processor that also acts as a supervisor terminal interface
- Easy-to-use operator interface that allows one-step system initialization
- Firmware floating point instructions
- The PRIMOS operating system

The 2250 is contained in a 30-inch cabinet that includes space for two 68- or 158-Megabyte non-removable disks and two 1/4-inch cartridge tapes, and an eight-board chassis. Five of the chassis slots are reserved for the CPU and other system components; the remaining three are available to support additional memory boards or any standard peripheral subsystem such as the PRIMENET node controller, a printer, or a disk.

Table 12-1 lists some of the additional hardware features of the 2250. For a complete list of the software products supported by the 2250, refer to Chapter 11. Your Prime sales representative can provide more detailed information.

Prime 250-II

The 250-II is a high performance, low cost system that supports up to 32 users in a distributed processing network environment. This CPU also provides a complete system useful in a number of multiuser applications. Standard components of the 250-II system include:

- 32-bit 250-II CPU
- 512 Kbytes of error correcting memory (expandable to 2 Mbyte)
- 8- or 16-line asynchronous terminal controller
- Virtual control panel for local/remote accesses and CPU control
- Supervisor terminal
- Firmware floating point instructions
- The PRIMOS operating system

The 250-II is contained in a standard ten-board chassis. Seven of the chassis slots are reserved for the CPU and other system components; the remaining three are available to support additional memory boards or any standard peripheral subsystem such as the PRIMENET node controller, a printer, or a disk.

Table 12-1 lists some of the additional hardware features of the 250-II. For a complete list of the software products supported by the 250-II, see Chapter 11. Your Prime sales representative can provide more detailed information.

Prime 550-II

The 550-II is a high performance system that supports up to 128 interactive processes, 64 of which may be user terminals, in a scientific, commercial, or Office Automation environment. This CPU is also suitable for use in distributed processing or networking applications. Standard components of the 550-II system include:

- 32-bit 550-II CPU
- 512 Kbytes of error correcting memory (expandable to 4 Mbytes)
- 16-line asynchronous terminal controller
- Virtual control panel for local/remote accesses and CPU control
- Supervisor terminal
- Hardware floating point, decimal, and character instructions
- The PRIMOS operating system

The 550-II is contained in a standard 27-board chassis. Fourteen of the chassis slots are reserved for the CPU and other system components; the remaining thirteen are available to support additional memory boards or any standard peripheral subsystem such as the PRIMENET node controller, a printer, or a disk.

Table 12-1 lists some of the additional hardware features of the 550-II. For a complete list of the software products supported by the 550-II, see Chapter 11, and contact your sales representative.

Prime 750

The 750 is a high performance system that supports up to 128 simultaneously active processes, of which up to 96 may be user terminals. It provides speed, low overhead, and great flexibility in scientific, commercial, interactive, and/or timesharing environments. Standard features of the 750 system include:

- 32-bit 750 CPU
- 1 Mbyte of error correcting memory (expandable to 8 Mbytes)
- Instruction preprocessor unit
- Burst mode I/O with a bandwidth of 8 Mbytes per second
- 64-bit interleaved memory data transfers
- Hardware floating point, decimal, and character instructions

- 16-line asynchronous terminal controller
- Virtual control panel for local/remote accesses and CPU control
- Supervisor terminal
- The PRIMOS operating system

The 750 is contained in a standard 38-board chassis. Nine of the slots are reserved for memory, 10 for the CPU, and 19 to support up to 13 I/O controllers. Slots not occupied by the basic system boards are available to support additional features.

Table 12-1 lists some of the additional hardware features of the 750. For a complete list of the software products supported by the 750, see Chapter 11, and contact your sales representative.

Prime 850

The 850 embodies a multistream architecture that can support up to 128 active user processes. Its integrated hardware, software, and firmware features make it an excellent choice for scientific, commercial, interactive, and/or timesharing applications. Standard features of the 850 system include:

- Two 32-bit instruction stream units
- One stream synchronization unit
- 2 Mbytes of error correcting memory (expandable to 8 Mbytes)
- Two instruction preprocessor units
- Burst mode I/O with a bandwidth of 8 Mbytes per second
- 64-bit interleaved memory data transfers
- Hardware floating point, decimal, and character instructions
- 16-line asynchronous terminal controller
- Virtual control panel for remote diagnostics
- Supervisor terminal
- The PRIMOS operating system

The 850 is contained in a standard 54-board chassis. 19 of the slots are reserved for the instruction stream units; 16 are reserved for the stream synchronization unit and memory; and the remaining 19 are reserved to support up to 13 I/O controllers.

Table 12-1 lists some of the additional hardware features of the 850. For a complete list of the software products supported by the 850, see Chapter 11, and contact your sales representative.

Table 12-1
Summary of 50 Series Characteristics

Feature	2250	250-II	550-II	750	850
32-bit architecture	yes	yes	yes	yes	yes
Simultaneous active processes	64	128	128	128	128
Direct connect terminal users	32	32	64	96	128
Virtual address space per system	512 Mb	512 Mb	512 Mb	512 Mb	512 Mb
Maximum physical memory size	4 Mb	2 Mb	4 Mb	8 Mb	8 Mb
Size of cache	2 Kb	2 Kb	8 Kb	16 Kb	32 Kb
Average cache hit rate	85%	85%	90%	95%	95%
I/O bandwidth (Mb/sec)	2.5	2.5	2.5	8.0	8.0
Input I/O transfer rate (Mb/seç)	2.5	2.5	2.5	8.0	8.0
Output I/O transfer rate (Mb/sec)	2.5	2.5	2.5	5.0	5.0
Burst mode I/O	no	no	no	yes	yes
Hardware integer arithmetic	yes	yes	yes	yes	yes
Character and decimal ops.	firmware	firmware	hardware	hardware	hardware
Floating point arithmetic	firmware	firmware	hardware	hardware	hardware
Instruction preprocessor	no	no	no	yes	yes
Microprocessor control unit with process exchange	yes	yes	yes	yes	yes
Parity checking	yes	yes	yes	yes	yes

Prime INFORMATION Systems

As an option for the systems house customer, Prime offers the INFORMATION line of CPUs. These CPUs are designed to support the INFORMATION operating system and the INFORMATION software (see Chapter 11).

The I450-II is identical to the 50 Series 250-II, except that it supports an 8-Kbyte cache and a microsecond timer. It also includes microcode assist capabilities for the INFORMATION software.

The I250-II, I750, and I850 are identical to the 50 Series 250-II, 750, and 850 CPUs, respectively.

MEMORY EXPANSION UNITS

There are two memory expansion packages available for the members of the 50 Series family. These packages are available in 512-Kbyte and 1-Mbyte sizes and can be used on any 50 Series system to extend memory capacity to the maximum allowed per system.

TERMINALS

Several terminals are available to address the user's need for interactive capabilities. Both the PT25 and PST 100 terminals are designed to function as a system console or as a user terminal. Four other terminals are provided to address the needs of users in many fields, including the office environment and graphics applications. For information about a hardcopy device suitable for use as a system console, refer to the 3115 Low Speed Serial Matrix Printer in the section, Unit Record Devices, below.

PT25 Character Mode Terminal

The PT25 is a general purpose, character mode terminal suitable for any interactive application. It displays characters in 24 lines of 80 characters each, with an additional line at the bottom of the screen that displays status, self-test, and control information. The attached keyboard has a main typewriter-like keypad, as well as a 14-key numeric pad, eight user-defined function keys, and five cursor control keys. The keyboard also allows users to specify a full set of visual screen attributes (reverse video; blink; underscore; and full half, or zero intensity).

This terminal is compatible with EIA RS232C (CCITT V.24), and interfaces with Prime's AMLC or ICS1 at speeds between 100 and 9600 bits per second. It also offers the users a choice of even, odd, mark, or space parity. PRIMOS supports the PT25 in full duplex mode.

PT45 Block Mode Terminal

This microprocessor-controlled terminal addresses the needs of all applications requiring block mode, character mode, and alphanumeric operations. The screen format is 24 lines by 80 characters, plus an additional status line at the bottom of the screen. The keyboard is detached for user comfort.

In addition to the PT25 features described above, the PT45 also contains two-page display memory with scrolling and paging controls. It, too, is equipped with a bi-directional, serial auxiliary port that is compatible with the EIA RS232C standard. This terminal interfaces with Prime's AMLC or ICS1 at speeds between 110 to 19200 bits per second. PRIMOS fully supports the PT45 as a character mode terminal for all alphanumeric, buffered mode, and block mode operations. FORMS, DPTX, and Office Automation software also support this terminal.

PT65 Intelligent Terminal

The PT65, like the PT45, is microprocessor controlled. It is designed for use with Prime's Office Automation software, where it is used as the administrative workstation. It is equipped with such editing and programming capabilities as 32 Kbytes of program or display memory, many Prime-defined function keys, and built-in word processing functions. The screen format of the PT65 is larger than that of the PT25 and PT45 for easier viewing. This terminal is also available with a Selectric-style keyboard.

Like other Prime terminals, the PT65 supports a standard asynchronous interface that allows the terminal to communicate at speeds between 300 and 9600 bits per second. The communication is by EIA RS232C.

PST 100 Block Mode Terminal

This microprocessor-controlled terminal, the first made by Prime, addresses the needs of all applications requiring block mode, character mode, and alphanumeric operations. The screen format is 24 lines by 80 characters, plus an additional status line at the bottom of the screen.

The ergonomic features include a detachable keyboard and a display screen that swivels to the left and right and tilts up and down. The keyboard module has a typewriter keypad, a numeric keypad, and a row of function keys.

Built-in terminal menus let users change terminal characteristics. For example, the terminal can be put in reverse video or the form of the cursor can be changed.

The PST 100 has a two-page display memory with smooth and jump scrolling controls. It uses a bi-directional, serial auxiliary port that is compatible with the EIA RS232C standard. The terminal connects to an AMLC or ICS1 line at speeds ranging from 50 to 19200 bits per second. PRIMOS fully supports the PST 100 as a character mode terminal for all alphanumeric, buffered mode, and block mode operations. FORMS, FED, and Office Automation software also support this terminal.

PW93 and PW95 Graphics Workstations

The Prime graphics workstations are hardware products designed for use with MEDUSA. Both systems are made up of a 19-inch diagonal display, a joystick, a data tablet, a PT25 terminal, and a graphic controller. PW93 includes a monochromatic display; PW95, an eight-color display with over 4,000 colors from which to choose. The graphics display, with raster scan technology and resolution of 1280 x 1024, is coupled with high speed hardware for efficient, versatile use.

Either workstation is connected to the host system via two serial ports. This means that workstations can be located some distance from the host without difficulty. In addition, much of the processing power is built into the workstations so that the time spent requesting service from the host is kept to a minimum.

Some of the features built into the workstations are conics, graphic primitives, selective erase and update, selectable cursor types, and variable line types. The PW95 also has local pan and zoom capabilities.

MAGNETIC TAPES

The user wanting magnetic tape capability has four drives from which to choose. These drives offer a choice of speeds and storage capacity.

4550 GCR Tape Drive

This 9-track tape drive is supported on all members of the 50 Series family. It records either 1600 (phase encoded) or 6250 (group code recording) bits of encoded information per inch of tape at a speed of 75 inches per second. The high speed and the density of information represents significant saving of resources for the user who must manipulate large volumes of information. In addition, this drive supports burst mode I/O to further speed up transfers on the 750 and 850 systems.

Features include automatic thread and load, high speed rewind, high reliability, and the ability to select the density of information either manually or by program. Built-in integrity monitors automatically correct single and double track errors, and internal exercizers can quickly identify problems if and when they occur. Finally, since the drive is supported on all 50 Series systems, it can easily work with an upgraded system in the future should the user's needs grow in that direction.

Up to eight 4550 tape subsystems can be supported on one system (four per controller). This total is subject to system configuration rules.

4522 Tape Drive

This 9-track tape drive records information at a speed of 75 inches per second. Users can select data density of either 800 (NRZI) or 1600 (PE) bits of encoded data. These industry standard formats allow the 4522 tape to be highly applicable to Prime systems that exchange data other non-Prime systems. The choice of data density also represents a significant saving of resources for the user who must manipulate large volumes of information. In addition, this drive supports burst mode I/O to further speed up transfers on the 750 and 850 systems.

4520 Tape Drive

The 9-track 4520 drive embodies all of the same features of the 4522 except that it operates at 45 inches per second. This makes the 4520 a good choice for the user who performs small numbers of tape operations. Note that the user can mix both 4520 and 4522 tape drives on the same controller for a total of eight drives.

Cartridge Tape Drive

The 4580/4651 Cartridge Tape Drive provides up to 15 Megabytes formatted capacity per cartridge. This 4-track drive records information at a speed of 30 inches per second, using a data density of 6400 bpi. Cartridge tapes provide economical backup, program load, and software distribution facilities, offer high reliability and data integrity, and are available on all Prime systems.

One Cartridge Tape Drive is included with each 2250 system, and a second drive (Model 4651) can be added. On other 50 series systems, 4580 Cartridge Tape Drives are supported, so users can transfer tapes between the 2250 and these systems.

DISKS

Prime offers three types of disk drives:

- Storage module disks (SMDs)
- Cartridge module disks (CMDs)
- Fixed media disks (FMDs)

All Prime systems support up to eight of these devices, four per controller. Any combination of disks can be used, except that the 675Mb FMD cannot be configured with any CMDs. The Prime 2250 supports all three types of disk, and also supports 68Mb and 158Mb FMDs, described below.

The user can thus to choose the storage combination that best suits an application. Nearly all the disks can be moved to a new Prime 50 Series system without difficulty if the user decides to upgrade the computer facility.

Table 12-2 summarizes information about the three types of disks.

Storage Module Disks

For the user who prefers removable disk storage, Prime offers two storage module disks. Available in 80 and 300 Mbyte capacities, storage module disks allow the user to change disk packs at will. Backing up data is a simple matter of copying data from one disk pack to another, then removing one pack to storage. These disks are also multipurpose to make them useful for many applications.

Table 12-2
Summary of Disk Characteristics

Disk Characteristic	Storage Module		Cartridge Module			Fixed Media	
	80Mb	300Mb	32Mb	64Mb	96Mb	160Mb	675Mb
Mbytes/Disk	77.0	292.7	30.8	61.6	92.4	154	630
Bytes/Sector	2080	2080	2080	2080	2080	2080	2080
Sectors/Track	9	9	9	9	9	9	9
Track/Drive	4115	15637	1646	3292	4938	8210	33640
Cylin./Drive	823	823	823	823	823	821	841
Av. Latency (Ms)	8.3	8.3	8.3	8.3	8.3	8.3	8.3
Min. Seek (Ms)	6	6	6	6	6	7	10
Av. Seek (Ms)	30	30	30	30	30	30	25
Max. Seek (Ms)	55	55	55	55	55	55	50
Transfer Rate (Mb)	1.2	1.2	1.2	1.2	1.2	1.2	1.2

Cartridge Module Disks

Prime's three cartridge disks combine fixed and removable storage in a family of three moderate capacity units. The 32 Mbyte disk has 16 Mbytes of fixed storage; the 64 Mbyte disk, 48 Mbytes; and the 96 Mbyte disk, 80 Mbytes. All three have 16 Mbytes of removable storage and use error correcting data encoding.

The user who can benefit most from using this type of disk typically needs only moderate amounts of storage. Systems that have only a limited amount of physical space in which to put equipment can also benefit from including these disks, since all three are rack mountable.

Fixed Media Disks

The two fixed media disks are the most cost effective, reliable storage units available from Prime. The larger 4490 is a free standing unit offering 675 Mbytes of storage. The 4480 has 160 Mbytes of storage and mounts either in a standard Prime peripheral cabinet (550-II, 750, and 850) or in a central system cabinet (250-II). Both embody state-of-the-art sealed Winchester technology.

Fixed Media Disks on the 2250

The 2250 supports both 68- and 158-Megabyte Fixed Media Disks. One 68-Megabyte disk and drive is included with each of these systems, and two can be mounted in the system cabinet. Table 12-3 summarizes the characteristics of both these FMDs.

Table 12-3
Characteristics of Fixed Disks on the 2250

Disk Characteristic	Fixed Media 68Mb	158Mb
Mbytes/Disk	63.0	146.9
Bytes/Sector	2080	2080
Sectors/Track	9	9
Track/Drive	3363	7847
Cylin./Drive	1121	1121
Av. Latency (Ms)	9.7	9.7
Min. Seek (Ms)	8	8
Av. Seek (Ms)	45	40
Max. Seek (Ms)	85	75
Transfer Rate (Mb)	1.04	1.04

UNIT RECORD DEVICES

Prime's offerings in the unit record device field are varied, giving the user a choice of products. The general types of devices are:

- Chain driven line printer
- Matrix line printer/plotter
- Band printer
- Matrix character printer
- Card reader

3166 and 3167 Chain Driven Line Printers

For the user requiring a heavy duty printer, Prime offers a chain driven line printer available in two speeds. The 3166 prints 1000 lines per minute; the 3167, 750 lines per minute. The 3167 uses a printing set of 96 characters to provide upper and lower case alphabetic, numeric, and punctuation characters. The 1000 lpm printer has a printing set of 64 characters.

Matrix Line Printer/Plotters

The matrix printer/plotter is available in two models: the 3126 uses a serial asynchronous interface, while the 3174 uses a parallel interface. Both print upper and lower case characters, as well as underlines, at a rate of 300 lines per minute. Both models can print up to five copies plus original at once, accepting a variety of standard forms and labels up to 16 inches wide. To generate graphs, maps, bar codes, curves, and block characters, the matrix printer/plotter can operate in plot mode. A static eliminator, paper guide, and paper basket are standard features. Users can also include special character sets or international AC power configuration as options to this type of printer.

This type of printer is suitable for the user with low or medium printing requirements. It is compatible with all members of the 50 Series family, so it can move to a new system if the user desires to upgrade a facility.

3323, 3327, 3333, and 3337 Band Printers

Prime's band printer is available in four models. The 3323 is a pedestal model printer and prints 300 lines per minute using a standard set of 64 ASCII characters. The 3333 cabinet printer also uses a set of 64 ASCII characters and prints 600 lines per minute. Both the 3327 and the 3337 use a set of 96 ASCII characters; the former is a pedestal model and prints 200 lines per minute, while the latter is a cabinet model printing 450 lines per minute.

3350 and 3351 Serial Matrix Printers

The 3350 serial character printer is for users with light duty printing needs. Three types of these devices are offered:

- A 160 character per second, keyboard send/receive (KSR) device
- A 30 character per second, keyboard send/receive (KSR) device
- A 160 character per second, receive only (RO) device

The KSR devices are primarily for use as hard copy terminals. Users can use the RO device with the SPOOL program as an output device. Either type of device is an excellent choice for use in a transaction based environment, since either can easily print a variety of business forms and labels.

Additional features of the serial character printers include a standard upper/lower case ASCII character set, 42 programmable functions, a character buffer, and bi-directional printing. Built in self-test routines and heavy duty construction ensure reliable operation.

3115 Low Speed Serial Matrix Printer

This printer is for use as a supervisor terminal or hard copy device. It is supported on all 50 Series systems and acts as a send/receive (KSR) unit. All features are totally programmable and can be controlled by application or system software. It prints at a bidirectional, nominal rate of 30 characters per second, with bursts of 60 characters per second to catch up when the buffer is full. It is easily maintained due to its self-test feature, and to its heavy duty construction.

3175 Letter-Quality Printer

This printer is a serial impact printer for letter-quality business applications. It is compatible with all 50 Series systems equipped with an AMLC or ISC1 controller. It prints up to 55 characters per second and can handle a number of standard business forms. The printer uses interchangeable print thimbles, and has the capability to print many international print characters.

3159 Card Reader

For the user with moderate duty card reading requirements, Prime offers the 3159 card reader. This device has a 550-standard-size-card hopper capacity and reads 300 cards per minute. Its heavy duty construction and tabletop size make it suitable for a variety of applications. Other features, such as a vacuum system to reduce dust, data resynchronizing logic, and a straight-through card track keep the reader trouble free.

SUMMARY

This chapter has summarized the 50 Series hardware products. The wide variety of components provide the user with many options and allow user applications to be addressed in several fashions. For more details about Prime's hardware products, contact your Prime sales representative.

INDEX

Index

READER RESPONSE FORM

DOC6904-191 Prime 50 Series Technical Summary

Your feedback will help us continue to improve the quality, accuracy, and organization of our user publications.

1. How do you rate the document for overall usefulness?

___excellent ___very good ___good ___fair ___poor

2. Please rate the document in the following areas:

Readability: ___hard to understand ___average ___very clear

Technical level: ___too simple ___about right ___too technical

Technical accuracy: ___poor ___average ___very good

Examples: ___too many ___about right ___too few

Illustrations: ___too many ___about right ___too few

3. What features did you find most useful? ______________________________

____

____

____

4. What faults or errors gave you problems? ______________________________

____

____

____

Would you like to be on a mailing list for Prime's current documentation catalog and ordering information? ___yes ___no

Name: ______________________________ Position: ____________________

Company: ____

Address: ____

____Zip: ____________

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

First Class Permit #531 Natick, Massachusetts 01760

BUSINESS REPLY MAIL

Postage will be paid by:

PRIME

Attention: Technical Publications
Bldg 10B
Prime Park, Natick, Ma. 01760